МИКРОПРОЦЕССОРЫ

Р. Токхайм

МИКРО-ПРОЦЕССОРЫ

Курс и упражнения

Р. Токхайм

МИКРО-ПРОЦЕССОРЫ

Курс и упражнения

Перевод с английского
В.Н. Грасевича и Л.А. Ильяшенко

МОСКВА
ЭНЕРГОАТОМИЗДАТ
1988

ББК 32.973
Т 51
УДК 681.31—181.48

Рецензенты: В. Н. Грасевич, Б. М. Каган, Л. А. Ильяшенко

R. L. TOKHEIM
THEORY AND PROBLEMS OF MICROPROCESSOR FUNDAMENTALS
Mc Graw-Hill, 1983

Токхайм Р.
Т 51 Микропроцессоры: Курс и упражнения/Пер. с англ., под ред. В. Н. Грасевича. М.: Энергоатомиздат, 1988. — 336 с.: ил.
ISBN 5-283-02487-3

В книге представлены в элементарном изложении основные вопросы, касающиеся архитектуры, функционирования и программирования микропроцессоров. Материал прекрасно иллюстрирован. Наличие большого числа решенных задач способствует лучшему усвоению и закреплению материала.

Для инженерно-технических работников, не имеющих специальной подготовки по вычислительной технике, студентов вузов.

Т $\frac{2405000000\text{-}338}{051(01)\text{-}88}$ 283-87 ББК 32.973

ISBN 5-283-02487-3 (русс.)
ISBN 0-07-064958-8 (англ.)

ПРЕДИСЛОВИЕ К РУССКОМУ ИЗДАНИЮ

Широкое внедрение микропроцессорной техники во все сферы человеческой деятельности, эффективность этого процесса неразрывно связаны как с развитием многочисленных сложных технических разработок, так и с уровнем подготовки в этой области специалистов самого различного профиля. Соответствие функциональных возможностей микропроцессорных систем и технологического назначения связанных с ними объектов обусловливает необходимость соответствующей подготовки специалистов-прикладников в ранее далекой от их профессиональных интересов области.

Решение этой задачи связано как с организацией учебного процесса на всех уровнях, включая и систему повышения квалификации специалистов, так и с его методическим обеспечением. В последние годы в нашей стране издано довольно много интересных и глубоких книг советских и зарубежных авторов, посвященных различным аспектам архитектуры, функционирования и применения микропроцессоров и микропроцессорных систем и предназначенных как для специалистов в области разработки этой аппаратуры и ее программного обеспечения, так и для огромной армии пользователей, прямые интересы которых в первую очередь замыкаются на решении специальных технических проблем: математического моделирования технических объектов, их эксплуатации и управления, разработки различных видов технологии и оборудования и т. д. Круг читателей, заинтересованных в появлении литературы, освещающей на доступном для неспециалистов в области вычислительной техники уровне сложные вопросы структуры, функционирования, принципов построения аппаратных и программных средств микропроцессорных систем, постоянно расширяется.

На фоне многообразия литературы по микропроцессорной тематике книга Р. Токхайма выгодно отличается глубиной методической проработки, тщательным отбором и

композицией материала, рассчитанными на самый широкий круг читателей. Перед начинающими изучение новой области знаний всегда возникает вопрос — с чего начать? Если речь идет об основах микропроцессорной техники и первых практических шагах, особенно в области программирования микропроцессорных систем, предлагаемая читателю книга станет хорошим началом.

В своем предисловии автор указывает, что целевым назначением книги является второй уровень среднего технического образования или первый уровень высшего образования. Мы разделяем такую точку зрения. Вместе с тем следует отметить, что круг заинтересованных читателей гораздо шире. К нему можно отнести всех специалистов-прикладников, начинающих изучение микропроцессоров и микропроцессорных систем. Большой интерес представляет она для преподавателей средних школ, техникумов и вузов. Это связано не только с методическими достоинствами книги, но и с тем, что содержание ее ориентировано на широко распространенный в нашей стране микропроцессорный комплект серии КР580 (зарубежный аналог — Intel 8080).

Успехи в изучении этой сложной и интересной тематики — микропроцессоров, аппаратной и программной составляющих микропроцессорных систем, их применение будут зависеть, разумеется, от глубины проработки материала книги. Настоятельно рекомендуем читателю делать это с карандашом в руках и бумагой, особенно при выполнении многочисленных, весьма полезных упражнений. Еще лучше делать это, используя реальные микропроцессорные системы, такие, например, как «Электроника-580».

При работе над книгой мы постарались, насколько это возможно, некоторые узкие места ее расширить своими примечаниями. Для читателей, заинтересованных в углублении знаний, в конце книги мы приводим список рекомендуемой литературы, изданной в нашей стране.

В. Н. Грасевич
Л. А. Ильяшенко

ПРЕДИСЛОВИЕ АВТОРА

Современные специалисты в области электронной техники и те, кто только еще ее изучает, должны обладать знаниями в области микропроцессоров и микропроцессорных систем, т. е. сведениями как об ее аппаратной, так и о программной частях. Микропроцессоры являются основой совершенно нового поколения интеллектуальных машин. Они встречаются повсюду: в детских игрушках, карманных калькуляторах для бытовых целей, промышленных роботах, бытовых электронагревательных приборах и т. д. Вследствие создания программируемого элемента, называемого микропроцессором, теперь можно рассчитывать на ускорение разработок искусственного интеллекта.

В предлагаемой читателю книге излагаются основные вопросы, составляющие обычно введение в микропроцессорную технику. Философия серии Schaum [1], внимание которой сконцентрировано на типичных задачах, возникающих в ходе изучения той или иной темы, здесь находит свое отражение в большом количестве упражнений по микропроцессорам и микропроцессорным системам, большая часть которых сопровождается подробными решениями.

Обсуждаемые вопросы подбирались с особой тщательностью в соответствии со вторым циклом среднего технического образования и первым циклом высшего образования. Были изучены многие учебники по этой тематике, а также монографии, посвященные микропроцессорам и микропроцессорным системам, поэтому вопросы и упражнения, приводимые здесь, идентичны часто встречаемым в подобных традиционных курсах. Фрагменты программ и вся приводимая здесь программная часть тщательно готовились на серийных микро-ЭВМ.

Предлагаемую книгу открывает короткое введение. Затем приводится основная информация, касающаяся вопро-

[1] Серия учебной литературы издательства Mc Graw-Hill. — *Прим. ред.*

сов информационной арифметики, машинного языка, а также основных элементов аппаратной части систем двоичной информации. Поставляемые на рынок микропроцессорные системы очень сложны, но чтобы ввести основные понятия, касающиеся микро-ЭВМ, микропроцессора, их программирования и интерфейса, мы прибегли также к понятию типового упрощенного микропроцессора. Затем подробно изучаем хорошо известный микропроцессор Intel 8080/8085, посвящая одну главу его программированию. Отметим, что типовой упрощенный микропроцессор обладает некоторыми свойствами развитых универсальных микропроцессоров.

Наш материал в значительной мере опирается на микропроцессоры Intel 8080/8085, поскольку они широко распространены как в промышленности, так и в системе образования. По своим свойствам микропроцессоры Intel 8080/8085 близки к популярному 8-разрядному микропроцессору Z 80 фирмы Zilog и к мощным 16-разрядным микропроцессорам Intel 8086/8088. В настоящее время микропроцессор Intel 8088 является основой персональных компьютеров IBM.

Я благодарен представителю фирмы Intel Дж. Альяно и моим ученикам из института Генри Сиблея за их помощь и советы. Добавлю мою признательность моей семье — Каролине, Даниэлю и Маршаллу за их терпение, доброе настроение и поддержку, которые меня вдохновляли.

Роджер Токхайм

Глава 1
ВВЕДЕНИЕ

Электронные вычислительные машины широко используются с 50-х годов. Вначале это были ламповые и дорогие машины, предназначенные для административно-управленческих целей, доступные крупным предприятиям. В последние годы структура и формы вычислительных машин изменились из-за появления нового элемента — микропроцессора. Микропроцессор — это интегральная схема (ИС), обладающая такой же производительностью при переработке информации, что и большая ЭВМ. Более точно — это очень сложное программируемое[1] устройство малых размеров, представляющее собой большую интегральную схему (БИС). Электронные вычислительные машины работают в соответствии с загружаемой в них программой, микро-ЭВМ действует по такому же принципу, она содержит микропроцессор и, по меньшей мере, один какой-либо тип полупроводниковой памяти.

1.1. СТРУКТУРА ЭВМ

На рис. 1.1 представлен традиционный состав ЭВМ. Такая структура функциональных элементов очень часто называется *архитектурой ЭВМ*. Самая простая система содержит пять устройств: *ввода* исходной информации, *управления* и *арифметических действий*, составляющих центральный процессор (ЦП) или центральное устройство, а также *память* и *устройство вывода* результатов обработки информации.

Физические устройства, приведенные на рис. 1.1, являются *аппаратными средствами*. Для их полезного использования занесенная в память программа предписывает цен-

[1] Согласно терминологии, принятой в СССР [1], микропроцессор (МП) — программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и управления им, построенное, как правило, на одной или нескольких БИС. — *Прим. ред.*

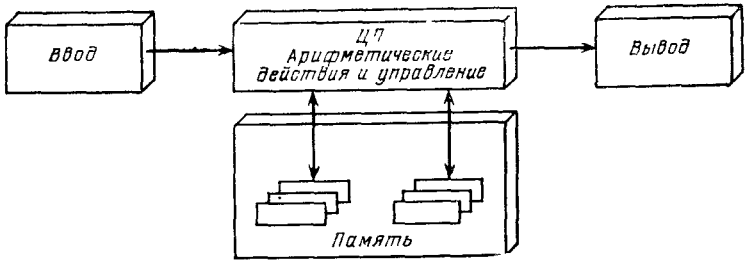


Рис. 1.1. Общая структура ЭВМ

тральному процессору то, что он должен выполнять. Подготовка списка команд называется *программированием.* Подготовленный список команд представляет собой *программу,* которая временно или постоянно размещается в программной памяти. Эти программы перерабатывают информацию, называемую общим термином — *данные.* Программные средства — это также общий термин, охватывающий все программы. Помещенные постоянно в программную память, они иногда называются *микропрограммными средствами.*

Таким образом, микро-ЭВМ функционирует в следующем порядке: программа и данные загружаются в центральный процессор и размещаются в предназначенной для них области памяти. Центральный процессор считывает из памяти первую команду и выполняет ее. Команды могут быть простыми, как, например: ADD (СЛОЖИТЬ) два числа, MOVE (ПЕРЕДАТЬ) данные, INPUT (ВВЕСТИ) или OUTPUT (ВЫВЕСТИ) данные, JUMP (ПЕРЕЙТИ) в другую область программы. Как только обработка данных закончится, результат передается на выход ЭВМ. Подчеркнем, что большинство действий ЦП подчинено командам, размещенным в программной памяти.

Упражнения

1.1. Назовите пять функциональных устройств цифровой ЭВМ.

1.2. Структура приведенной на рис. 1.1 системы иногда называется _____ ЭВМ.

1.3. Электронные устройства, приведенные на рис. 1.1, являются _____ средствами, тогда как программы, указывающие ЭВМ то, что она должна выполнять, являются программными средствами.

1.4. Программист пишет список _____, называемый программой.

1.5. Какие два типа информации вводятся в ЭВМ и помещаются в память?

1.6. Большинство действий ЦП подчинено командам, помещенным в _____ памяти.

Решения

1.1. Пять функциональных устройств цифровой ЭВМ приведены на рис. 1.1. Центральный процессор — это термин, относящийся к части, содержащей одновременно устройства управления и арифметических действий. **1.2.** Архитектурой. **1.3.** Аппаратными. **1.4.** Команд. **1.5.** Программы и данные. **1.6.** Команды располагаются в программной памяти. Обрабатываемые данные помещаются в памяти данных. В некоторых ЭВМ физического различия между памятью данных и программной памятью не существует.

1.2. АРХИТЕКТУРА МИКРО-ЭВМ

Микро-ЭВМ — это цифровая ЭВМ. Она классифицируется как микро вследствие своих малых размеров и низкой стоимости. Обычно роль центрального процессора в таких системах выполняет микропроцессор. На рис. 1.2 приведена архитектура обычной микро-ЭВМ. Она состоит из пяти основных элементов ЭВМ: устройства *ввода,* устройств *управления* и *арифметических действий* (оба входят в микропроцессор), *памяти* и, наконец, устройства *вывода.*

Микропроцессор контролирует все системы и управляет ими посредством *линий управления* и (или) *контроля,* приведенных на рис. 1.2 слева, параллельно линиям управления расположена *адресная шина* (16 параллельных проводников), которая выбирает ячейку памяти данных, порты ввода или вывода данных. *Шина данных* (8 параллельных проводников) на рис. 1.2 справа является *двунаправленной* и служит для передачи данных в центральное устройство обработки информации или из него. Важно отметить, что ЦП может пересылать данные в память или получать их из нее посредством шины данных.

Когда программа помещается в память постоянно, она обычно располагается в устройстве, называемом *постоян-*

ным запоминающим устройством (ПЗУ) [1]. Постоянное запоминающее устройство является обычно кристаллом программируемой интегральной микросхемы с неизменяемой программой. Память изменяющихся данных, называемая *оперативным запоминающим устройством* (ОЗУ) [2] или памятью с произвольным доступом (первое название используется более часто), составляется также обычно из ИС.

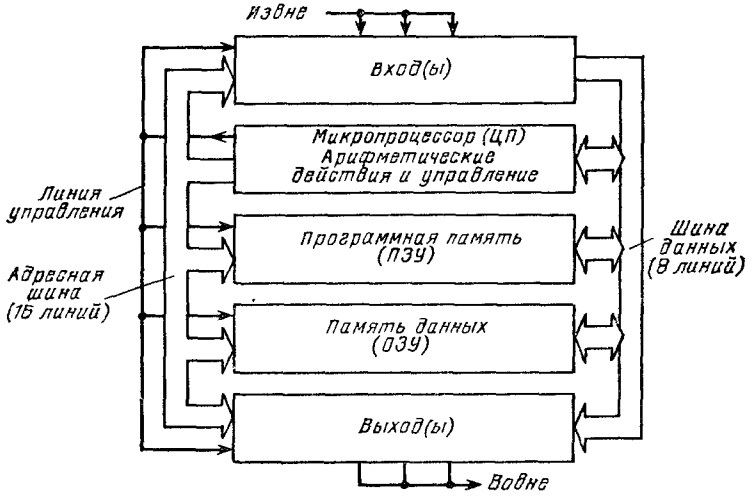


Рис. 1.2. Типовая структура микро-ЭВМ

Программы пользователя, которые по своей природе изменяемы, также помещаются в оперативной части памяти вместе с данными. На рис. 1.2 ОЗУ и ПЗУ показаны раздельно, потому что они обычно составляются различными ИС.

Приведенная на рис. 1.2 система может рассматриваться как обобщенная архитектура микро-ЭВМ. Большая часть микро-ЭВМ содержит, по меньшей мере, эти характерные устройства, а также и некоторые другие. Для упрощения таких схем обычно не показывают питание, источник тактовых импульсов (часы) и некоторые входящие в микро-ЭВМ линии, необходимые для ее работы.

[1] На английском языке используется запись ROM (*Read Only Memory*) — память только для считывания. — *Прим. пер.*

[2] На английском языке используется запись RAM (*Read/Write Memory*) — память для записи/считывания. — *Прим. пер.*

Упражнения

1.7. Центральный процессор представляет собой интегральную схему, называемую _____.

1.8. Какой блок на рис. 1.2 должен рассматриваться как ЦП?

1.9. Перечислить три типа связей в микро-ЭВМ, приведенной на рис. 1.2.

1.10. Адресная шина на рис. 1.2 для кодированной информации является однонаправленной, шина _____, напротив, является двунаправленной.

1.11. Обычно постоянные программы располагаются в БИС, называемой _____.

1.12. Какой тип памяти сокращенно называется ПЗУ?

1.13. Какой тип памяти сокращенно называется ОЗУ?

1.14. Временные данные и программы располагаются в БИС, называемой _____ (ОЗУ, ПЗУ).

1.15. Размещение данных в микро-ЭВМ выполняется _____ (временно, постоянно).

1.16. Размещение программ в ПЗУ выполняется (временно, постоянно).

1.17. Ввод или вывод информации в(из) микро-ЭВМ выполняется с использованием _____ (порта, датчика времени).

Решения

1.7. Микропроцессором. В некоторых архитектурах микро-ЭВМ задачи ЦП выполняют несколько ИС. **1.8.** Микропроцессор (на рис. 1.2 обозначен как ЦП) контролирует все прочие устройства микро-ЭВМ (или управляет ими). **1.9.** Адресная шина, шина данных и линии управления. В действительности линий может быть больше, чем представлено на рис. 1.2. **1.10.** Данных. **1.11.** ПЗУ. **1.12.** Постоянное запоминающее устройство. **1.13.** Оперативное запоминающее устройство. **1.14.** ОЗУ. **1.15.** Временно. **1.16.** Постоянно. **1.17.** Порта.

1.3. РАБОТА МИКРО-ЭВМ

Обратимся к рис. 1.3, иллюстрирующему работу микро-ЭВМ. На этом примере показаны следующие процедуры: *1* — нажатие клавиши *A* клавишного устройства; *2* — размещение буквы *A* в памяти; *3* — воспроизведение буквы *A* на экране дисплея.

Процедура ввод — размещение — вывод, показанная на рис. 1.3, является типичной. Аппаратные электронные сред-

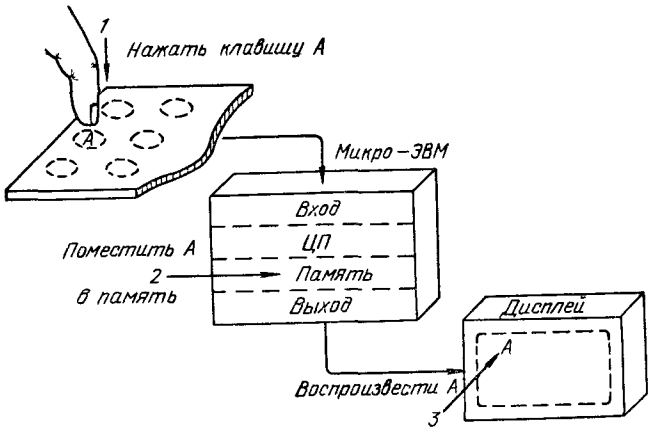


Рис. 1.3. Действия микро-ЭВМ

ства, используемые в такой схеме, довольно сложны. Однако анализ процесса передачи данных поможет понять роль различных устройств, составляющих систему. Более подробная схема на рис. 1.4 позволяет осмыслить типичную процедуру в микро-ЭВМ — ввод—размещение—вывод. Прежде всего внимательно рассмотрим содержимое программной памяти на рис. 1.4. Заметим, что команды предварительно были загружены в шесть первых ячеек памяти. Согласно рисунку текущими командами в программной памяти являются:

1. ВВЕСТИ (INPUT) данные через порт ввода *1*.
2. РАЗМЕСТИТЬ (STORE) данные, поступающие с порта *1*, в ячейке памяти данных 200.
3. ВЫВЕСТИ (OUTPUT) данные через порт вывода 10.

Заметим, что приведенная выше программа содержит только три команды. Однако в программной памяти на рис. 1.4 имеется шесть команд. Это обусловлено тем, что обычно команды делятся на две части. Первая часть команды 1 была ВВЕСТИ данные, вторая указывает нам ее происхождение (порт *1*). Первая часть представляет собой действие и называется *операцией*, а вторая — *операндом*. Операция и операнд помещены в ячейки памяти, показанные на рис. 1.4. Что касается первой команды на этом рисунке, операция ВВЕСТИ содержится в ячейке памяти *100*, а в ячейке памяти *101* — операнд (порт *1*), который нам указывает, откуда поступит информация.

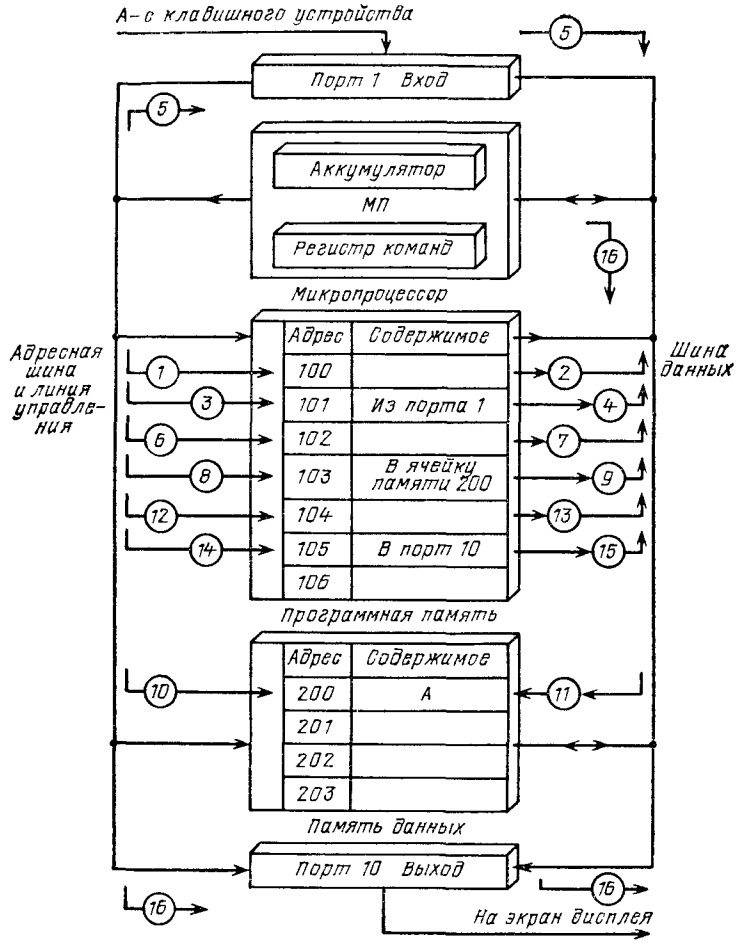


Рис. 1.4. Поэтапные действия микро-ЭВМ в ходе выполнения команды программной памяти

На рис. 1.4 введены два новых элемента — *регистры*. Этими двумя специальными регистрами являются *аккумулятор* и *регистр команд*.

Последовательность событий, происходящих в микро-ЭВМ в ходе выполнения приведенного на рис. 1.3 примера, подробно показана на рис. 1.4. Не забудем, что МП является центром всех преобразований данных и операций.

Обратимся к рис. 1.4, чтобы проследить все этапы.

Этап *1.* Микропроцессор выставляет адрес *100* на адресную шину. Линия управления *признает* (или активизирует) ввод *считывания* из ИС программной памяти (считывать данные означает копировать информацию из ячейки памяти). Этот этап на рис. 1.4 обозначен *1* в кружке

Этап *2.* Программная память выставляет первую команду на шину данных (ВВЕСТИ данные), а МП принимает эту кодированную информацию. Это послание помещается в специальное пространство памяти МП, называемое регистром команд. Оно *декодируется* МП (интерпретируется), этой команде нужен операнд.

Этап *3.* Микропроцессор выставляет на адресную шину адрес *101*. Линией управления активизируется вход считывания из программной памяти.

Этап *4.* Программная память помещает операнд (изъятый из порта *1*) на шину данных. Этот операнд находился в ячейке памяти 101. Кодированное послание (адрес порта *1*) взято на шине данных и помещено в регистр команд. Теперь МП декодирует полную команду (ВВЕСТИ данные, поступающие из порта *1*).

Этап *5.* Микропроцессор побуждает открыть порт *1* посредством адресной шины и линии управления устройством ввода. Кодированная форма *A* передается в аккумулятор, где и размещается.

Важно заметить, что МП все время действует в последовательности — *извлечение — декодирование — выполнение*. Он извлекает сначала из программной памяти команду, затем расшифровывает ее и, наконец, выполняет. Продолжим теперь рассмотрение программы.

Этап *6.* Микропроцессор выставляет на адресную шину адрес ячейки памяти *102* и активизирует вход считывания из программной памяти посредством управляющих линий.

Этап *7.* Код команды ПОМЕСТИТЬ данные считывается с шины данных, принимается МП и помещается в регистр команд.

Этап *8.* Микропроцессор декодирует эту команду и определяет, что нужен операнд. Он выставляет на шину данных следующий адрес *103* и активизирует вход считывания из программной памяти.

Этап *9.* Код операнда «В ячейку памяти *200*» из памяти (программы) помещен на шину данных, МП принимает операнд и помещает его в регистр команд. Команда ПОМЕСТИТЬ данные, расположенные в ячейке памяти *200*, полностью извлечена и декодирована.

Этап *10.* Теперь начинается процесс выполнения. Микропроцессор выставляет на адресную шину адрес *200* и активизирует вход *записи* в память (запись означает, что данные введутся в память).

Этап *11.* Микропроцессор выдает помещенную в аккумуляторе информацию на шину данных (кодированная форма *A*). Это *A* записано в ячейке памяти *200* и таким образом теперь выполнена вторая команда.

Этап *12.* Теперь МП должен извлечь следующую команду. Он адресует ячейку памяти *104* и активизирует вход считывания из памяти.

Этап *13.* Команда ВЫВЕСТИ данные помещена на шину данных, МП принимает ее и помещает в регистр команд. Микропроцессор декодирует послание и устанавливает, что нужен операнд.

Этап *14.* Микропроцессор помещает адрес *105* на адресную шину и активизирует вход считывания из памяти

Этап *15.* Память помещает код операнда в порт *10* на шину данных. Этот код принимается МП, который помещает его в регистр команд.

Этап *16.* Микропроцессор декодирует команду ВЫВЕСТИ данные в порт *10* полностью, активизирует порт *10* посредством адресной шины и управляющей линии устройства вывода. Он помещает код *A* (постоянно находящийся в аккумуляторе) на шину данных. Наконец *A* передается портом *10* на видеотерминал.

Большинство микро-ЭВМ передают информацию описанным сейчас и показанным на рис. 1.4 способом. Самые большие различия сосредоточены в элементах ввода и вывода. Может быть так, что потребуется больше этапов для осуществления этих операций.

Важно отметить, что МП является центром всех операций и полностью ими управляет. Он следует последовательности — извлечение—декодирование—выполнение. Выполняемые операции, напротив, диктуются командами, помещенными в памяти.

Упражнения

1.18. Список команд для использования в микро-ЭВМ составляет _____.

1.19. Программа помещается внутри микро-ЭВМ в памяти _____.

1.20. Большинство команд микро-ЭВМ состоит из двух частей — операции и _____.

1.21. Команды, составляющие программу микро-ЭВМ, обычно выполняются _____ (последовательно, случайно).

1.22. После выполнения команды ВЫВЕСТИ данные в порт *10* МП на рис. 1.4 обратится по адресу ячейки памяти _____ за следующей командой.

1.23. Сокращение ЦП означает _____.

1.24. Следствием такой команды, как ПОМЕСТИТЬ данные в ячейку памяти *201*, была бы передача содержащихся в ЦП данных в ячейку памяти *201*, расположенной в памяти _____.

1.25. Обратимся к рис. 1.4. Результатом команды ПОМЕСТИТЬ данные в ячейку памяти *202* была бы передача данных из МП, расположенных в _____ (аккумуляторе, регистре команд), в ячейку памяти _____ ОЗУ.

1.26. Процесс _____ (считывания, записи) выполняется, когда данные извлекаются из ячейки памяти.

1.27. Помещение данных в ячейку памяти является операцией _____ (считывания, записи).

1.28. Для выполнения каждой команды МП действует в последовательности: _____, _____, _____.

1.29. После этапа *16* (рис. 1.4) каким будет содержимое аккумулятора МП после выполнения команды вывода данных в порт *10*?

1.30. Обратиться к рис. 1.4. Сохраняются ли команды, расположенные в ячейках памяти от *100* до *105*, после этапа *16*?

Решения

1.18. Программу. **1.19.** Программы. **1.20.** Операнда. **1.21.** Последовательно. Однако позже мы введем комаиды ветвления, которые позволяют МП перейти к другой команде, обычно не следующей последовательно за данной. Во всяком случае, команды никогда не выполняются случайно. **1.22.** *106*. На рис. 1.4 она пока не идентифицирована, результат непредсказуем. Действительно, эта ячейка памяти должна бы содержать команду ЖДАТЬ (WAIT) или СТОП (HALT), как в случае, когда бы действия микро-ЭВМ были предсказуемы. **1.23.** Центральный процессор. **1.24.** Результатом выполнения этой команды будет передача данных из МП в ячейку памяти *201* в памяти данных. **1.25.** Аккумуляторе; в ячейку памяти *202* ОЗУ. В конце операции эти данные будут одновременно содержаться и в ОЗУ, и в аккумуляторе. Содержимое аккумулятора не разрушается после передачи данных. **1.26.** Считывания. **1.27.** Записи. **1.28.** Извлечение, декодирование, выполнение. **1.29.** Всегда кодированная форма *A*. Считывание из регистра или ячейки памяти не разрушает их содержимое. **1.30.** Да.

Дополнительные упражнения к гл. 1

1.31. Сокращение ИС означает _____.

1.32. Сокращение БИС означает _____.

1.33. Сокращение ЦП означает _____.

1.34. Интегральная схема, обладающая большинством свойств ЭВМ, называется _____.

1.35. В микро-ЭВМ сокращение МП означает _____

1.36. Микро-ЭВМ, выполняющая только одну задачу, является _____ (универсальной, специальной).

1.37. Какие, по меньшей мере, пять основных устройств входят в типовую микро-ЭВМ?

1.38. Список команд составляет _____ (программные, аппаратные) средства микро-ЭВМ.

1.39. Информация, перерабатываемая ЭВМ, является _____ (данными, числовой информацией).

1.40. Человек, пишущий команды для введения их в ЭВМ, называется _____.

1.41. Помещение данных на постоянное хранение в памяти ЭВМ делается обычно в _____.

1.42. Интегральные схемы, широко применяемые в ЭВМ для организации памяти со сменяющимися данными, называются _____.

1.43. См. рис. 1.2. Команды в программной памяти _____ (постоянны, сменяемы).

1.44. Аппаратные средства, составляющие систему на рис. 1.3, _____ (просты, сложны).

1.45. См. рис. 1.4. Окончив этап *16*, МП приступит к _____ (извлечению, декодированию, выполнению).

1.46. См. рис. 1.4 Центральный процессор получает доступ к ячейке памяти посредством шины _____.

1.47. См. рис. 1.4. Кодированная информация передается из аккумулятора МП в ячейку памяти данных посредством шины _____.

1.48. Если МП извлек и декодировал такую команду как ПОМЕСТИТЬ данные в ячейку памяти 205, откуда они будут затребованы?

1.49. Микро-ЭВМ содержит, по меньшей мере, устройства ввода, вывода, центральной процессор и _____ программ и данных.

Решения

1.31. Интегральная схема. **1.32.** Большая интегральная схема. **1.33.** Центральный процессор. **1.34.** Микропроцессор. **1.35.** Микропроцессор. **1.36.** Специальной. **1.37.** Устройства ввода, вывода, памяти, управления и арифметических действий. **1.38.** Программные. **1.39.** Данными. **1.40.** Программистом. **1.41.** ПЗУ. **1.42.** ОЗУ. **1.43.** Постоянны. **1.44.** Сложны. **1.45.** Выполнению. **1.46.** Адреса. **1.47.** Данных. **1.48.** Из аккумулятора МП. **1.49.** Памяти.

Глава 2
ЧИСЛА, КОДИРОВАНИЕ И АРИФМЕТИЧЕСКАЯ ИНФОРМАЦИЯ

2.1. ДВОИЧНЫЕ ЧИСЛА

Цифровые вычислительные машины работают с двоичными числами. Двоичная система счисления или система с основанием 2 использует только цифры 0 и 1. Эти двоичные числа названы битами (от *bi*nary digi*t*). Физически в цифровых электронных системах бит 0 представлен напряжением LOW (низким), а бит 1 — напряжением HIGH (высоким) [1].

Человеческая деятельность предполагает использование десятичной системы счисления. Десятичная система, или система с основанием 10, содержит 10 цифр (от 0 до 9). Она также характеризуется значением позиции (или весом). В табл. 2.1 показано, например, что десятичное число 1327 равно одной тысяче, плюс три сотни, плюс два десятка, плюс семь единиц (1000+300+20+7=1327).

Двоичная система обладает также свойством уравновешивания. В табл. 2.2 приведены десятичные значения первых чегырех двоичных позиций. Двоичное число 1001 (произносится: один, нуль, нуль, один) преобразовано, таким образом, в свой десятичный эквивалент 9. Бит единицы двоичного числа в табл. 2.2 называется младшим битом (МБ), бит восьмерки — старшим битом (СБ).

[1] Хотя LOW и HIGH переводятся соответственно как низкое и высокое, здесь принята международная терминология, использующая английские термины К тому же сокращения L и H будут использованы и в дальнейшем для обозначения соответствующих уровней сигналов. — *Прим. ред.*

Таблица 2.1. *Значения позиций десятичных чисел*

Степень основания	10^3	10^2	10^1	10^0
Значения позиций	1000	100	10	1
Десятичные	1	3	2	7
	1000 +	300 +	20 +	7=1327

Таблица 2.2. *Значения позиций двоичных чисел*

Степень основания	2^3	2^2	2^1	2^0
Значения позиций	8	4	2	1
	СБ			МБ
Двоичное	1	0	0	1
Десятичное	8 +	0 +	0 +	1=9

В табл. 2.3 приведены десятичные числа от 0 до 15, а также их двоичные эквиваленты. Те, кто работает в области использования ЭВМ, должны, по меньшей мере, запомнить эти двоичные числа.

Таблица 2.3. *Десятичные числа и их двоичные эквиваленты*

Десятичные	Двоичные	Десятичные	Двоичные
10 1	8 4 2 1	10 1	8 4 2 1
0	0	8	1 0 0 0
1	1	9	1 0 0 1
2	1 0	1 0	1 0 1 0
3	1 1	1 1	1 0 1 1
4	1 0 0	1 2	1 1 0 0
5	1 0 1	1 3	1 1 0 1
6	1 1 0	1 4	1 1 1 0
7	1 1 1	1 5	1 1 1 1

Как преобразовать двоичное число 1011 0110 (т. е. один, нуль, один, один, нуль, один, один, нуль) в его десятичный эквивалент? Процедура преобразования выполняется в соответствии с табл. 2.4. Десятичные значения каждой позиции записаны под каждым битом, затем десятичные числа суммируются (128+32+16+4+2=182), что дает 182.

Таблица 2.4. *Двоично-десятичные преобразования*

Степень основания	2^7		2^6	2^5		2^4	2^3	2^2		2^1	2^0	
Значение позиций	128		64	32		16	8	4		2	1	
Двоичное	1			1		1	0	1		1	0	
Десятичное	128	+		32	+	16	+	4	+	2	=	182

Обычно основание системы счисления указывается индексами. Таким образом, число $1011\ 0110_2$ является двоичным (или основания 2), а число 182_{10} — десятичным: $1011\ 0110_2 = 182_{10}$.

Как преобразовать десятичное 155 в его двоичный эквивалент? Процедура преобразования приведена на рис. 2.1.

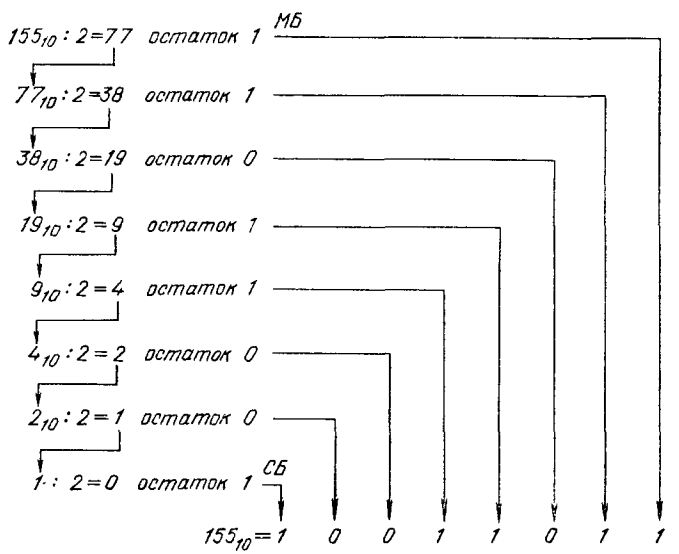


Рис. 2.1. Двоично-десятичные преобразования

Десятичное 155 сначала делится на 2, что дает нам частное 77 и остаток *1*. Этот остаток становится МБ двоичного числа и помещается в эту позицию (см. рис. 2.1). Затем частное (77) перемещается, как показывает стрелка, и становится следующим делимым. Затем каждое частное последовательно делится на 2 до тех пор, пока не получится частное, равное 0, и остаток, равный 1 (см. предпоследнюю строку на рис. 2.1). Последняя строка на рис. 2.1 дает нам результат $155_{10} = 1001\ 1011_2$.

Упражнения

2.1. Большинство людей в своей практической деятельности использует десятичную систему, цифровая ЭВМ использует _____ систему.

2.2. В двоичной системе бит означает _____.

2.3. Число 100_{10} является _____ числом.

2.4. Записать двоичное число один, один, нуль, нуль в цифровой форме.

2.5. Что означает сокращение МБ?

2.6. Преобразовать в десятичный код следующие двоичные числа: а) 0001; б) 0101; в) 1000; г) 1011; д) 1111; е) 0111.

2.7. Преобразовать в десятичный код следующие двоичные числа: а) 1000 0000; б) 0001 0000; в) 0011 0011; г) 0110 0100; д) 0001 1111; е) 1111 1111.

2.8. Преобразовать в двоичный код следующие десятичные числа: а) 23; б) 39; в) 55; г) 48.

2.9. 204_{10}=_____$_2$.

2.10. $1110\ 1110_2$=_____$_{10}$.

Решения

2.1. Двоичную или основания 2. **2.2.** Двоичную цифру (*binary digit*). **2.3.** Десятичным (основания 10). **2.4.** 1100_2. **2.5.** Младший бит. **2.6.** Обратимся к табл. 2.3: а) $0001_2 = 1_{10}$; б) $0101_2 = 5_{10}$; в) $1000_2 = 8_{10}$; г) $1011_2 = 11_{10}$; д) $1111_2 = 15_{10}$; е) $0111_2 = 7_{10}$. **2.7.** Следуем процедуре, приведенной в табл. 2.2: а) $1000\ 0000_2 = 128_{10}$; б) $0001\ 0000_2 = 16_{10}$; в) $0011\ 0011_2 = 51_{10}$; г) $0110\ 0100_2 = 100_{10}$; д) $0001\ 1111_2 = 31_{10}$; е) $1111\ 1111_2 = 255_{10}$. **2.8.** См. рис. 2.1:

а) 23_{10} : 2=11, остаток 1 (МБ)
11 : 2= 5, остаток 1;
5 : 2= 2, остаток 1;
2 : 2= 1, остаток 0;
1 : 2= 0, остаток 1 (СБ);

б) 39_{10} : 2=19, остаток 1 (МБ)
19 : 2= 9, остаток 1;
9 : 2= 4, остаток 1;
4 : 2= 2, остаток 0;
2 : 2= 1, остаток 0;
1 : 2= 0, остаток 1 (СБ);

в) 55_{10} : 2=27, остаток 1 (МБ)
27 : 2=13, остаток 1;
13 : 2= 6, остаток 1;
6 : 2= 3, остаток 0;
3 : 2= 1, остаток 1;
1 : 2= 0, остаток 1 (СБ);

г) 48_{10} : 2=24, остаток 0 (МБ);
24 . 2=12, остаток 0;
12 : 2= 6, остаток 0;
6 : 2= 3, остаток 0;
3 : 2= 1, остаток 1;
1 : 2= 0, остаток 1 (СБ).

2.9. Обратиться к рис. 2.1: $204_{10}=1100\ 1100_2$ **2.10.** Обратиться к табл. 2.2: $1110\ 1110_2=238_{10}$.

2.2. ШЕСТНАДЦАТЕРИЧНЫЕ ЧИСЛА

Ячейка памяти типичной микро-ЭВМ может содержать двоичное число 1001 1110. Такая длинная цепь нулей и единиц сложна для запоминания и неудобна для ввода с клавиатуры. Число 1001 1110 могло бы быть преобразовано в десятичное, что дало бы 158_{10}, но процесс преобразований занял бы много времени. Большая часть систем микроинформатики использует шестнадцатеричную форму записи, чтобы упростить запоминание и использование таких двоичных чисел, как 1001 1110.

Шестнадцатеричная система счисления (*hexadecimal*) [1], или система с основанием 16, использует 16 символов от 0 до 9 и A, B, C, D, E, F. В табл. 2.5 приведены эквиваленты десятичных, двоичных и шестнадцатеричных чисел.

Заметим из табл. 2.5, что каждый шестнадцатеричный символ может быть представлен единственным сочетанием четырех бит. Таким образом, представлением двоичного числа 1001 1110 в шестнадцатеричном коде является число 9E. Это значит, что часть 1001 двоичного числа равна 9, а часть 1110 равна E (конечно, в шестнадцатеричном коде). Следовательно, $1001\ 1110_2=9E_{16}$. (Не следует забывать, что индексы означают основание системы счисления.)

Как преобразовать двоичное число 111010 в шестнадцатеричное? Надо начать с МБ и разделить двоичное число на группы из 4 бит. Затем надо заменить каждую группу из 4 бит эквивалентной шестнадцатеричной цифрой: $1010_2=A$, $0011_2=3$, следовательно, $111010_2=3A_{16}$.

Как преобразовать шестнадцатеричное число 7F в двоичное? В этом случае каждая шестнадцатеричная цифра должна быть заменена своим двоичным эквивалентом из

[1] Отсюда происходит использование в конце шестнадцатеричных чисел латинской буквы H наряду с индексом 16 для обозначения шестнадцатеричной системы счисления. Мы будем использовать также термин H-код. — *Прим. ред.*

Таблица 2.5. *Десятичные, шестнадцатеричные и двоичные эквиваленты*

Десятичные	Шестнадцатеричные	Двоичные			
		8	4	2	1
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
10	A	1	0	1	0
11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	F	1	1	1	1

4 бит. В примере двоичное число 0111 заменено шестнадцатеричной цифрой 7, а 1111_2 заменяет F_{16}, откуда $7F_{16}=1111\ 0111_2$.

Шестнадцатеричная запись широко используется для представления двоичных чисел, поэтому необходимо табл. 2.5 также запомнить.

Таблица 2.6. *Преобразование шестнадцатеричного числа в десятичное*

Степень шестнадцати	16^3		16^2		16^1		16^0	
Значение позиции	4096		256		16		1	
Шестнадцатеричное	2		C		6		E	
Десятичное	4096 × 2 = 8192	+	256 × 12 = 3072	+	16 × 6 = 96	+	1 × 14 = 14	$=11374_{10}$

Преобразуем шестнадцатеричное число 2C6E в десятичное. Процедура действий соответствует табл. 2.6. Значениями позиций первых четырех шестнадцатеричных цифр

являются соответственно слева направо 4096, 256, 16 и 1. Десятичное число содержит 14 (E_{16}) единиц, 6 чисел 16, 12 (C_{16}) чисел 256 и 2 числа 4096. Каждая цифра умножается на соответствующий ей вес, получается сумма, которая и дает нам десятичное число 11374.

Преобразуем десятичное число 15797 в шестнадцатеричное. На рис. 2.2 показана процедура действий. В первой

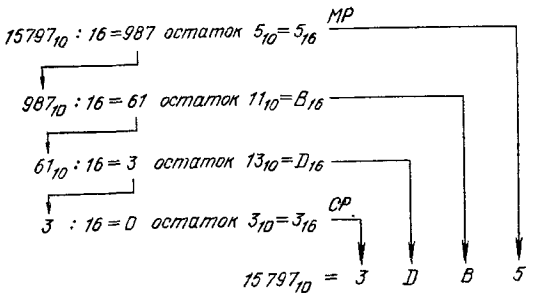


Рис. 2.2. Десятично-шестнадцатеричное преобразование

строке 15797_{10} разделено на 16, что дает частное 987_{10} и остаток 5_{10}, который преобразуется затем в свой шестнадцатеричный эквивалент ($5_{10}=5_{16}$) и становится цифрой младшего разряда (МР) шестнадцатеричного числа. Первое частное (987) становится делимым во второй строке и снова делится на 16, что дает частное 61 и остаток 11_{10} или шестнадцатеричное B. В третьей строке 61 делится на 16, дает частное 3 и остаток 13_{10} или D_{16}, а в четвертой строке делимое 3 делится на 16, дает частное 0 и остаток 3_{10} или 3_{16}. Когда частное равно 0, как в четвертой строке, преобразование заканчивается. 3_{16} становится цифрой старшего разряда (СР) результата, т. е. $3DB5_{16}$.

Упражнения

2.11. Шестнадцатеричная запись широко используется как сокращенная форма записи _____ (двоичных, десятичных) чисел.

2.12. Шестнадцатеричная система имеет основание _____.

2.13. Записать следующие шестнадцатеричные числа в двоичной форме: а) C; б) 6; в) F; г) E; д) 1A; е) 3D; ж) AO; з) 8B; и) 45; к) D7.

2.14. Преобразовать следующие двоичные числа в шестнадцатеричный код: а) 1001; б) 1100; в) 1101; г) 1111; д) 1000 0000; е) 0111 1110; ж) 001 0101; з) 1101 1011.

2.15. Преобразовать следующие шестнадцатеричные числа в десятичный код: а) 7E; б) DB; в) 12A3; г) 34CF.

2.16. 217_{10} = _____ 16.

2.17. 48373_{10} = _____ 16.

Решения

2.11. Двоичных **2.12.** 16. **2.13.** Используя табл. 2.5 и процедуру, приведенную в § 2.2, получаем: а) $CH=1102_2$; б) $6H=0110_2$; в) $PH=1111_2$; г) $E2H=1110\ 0010_2$; д) $1AH=0001\ 1010_2$; е) $3DH=0011\ 1101_2$; ж) $AOH=1010\ 0000_2$; з) $8BH=1000\ 1011_2$; и) $45H=0100\ 0101_2$; к) $D7H=1101\ 0111_2$.

2.14. а) $1101_2=9H$; б) $1100_2=CH$; в) $1101_2=DH$; г) $1111_2=FH$; д) $1000\ 0000_2=80H$; е) $0111\ 1110_2=7EH$; ж) $001\ 0101_2=15H$; з) $1101\ 1011_2=DBH$.

2.15. а) $7EH=(16\times7)+(1\times14)=126_{10}$; б) $DBH=(16\times13)+(1\times14)=219_{10}$; в) $12A3H=(4096\times1)+(256\times2)+(16\times10)+(1\times3)=47710_{10}$; г) $34CFH=(4096\times3)+(256\times4)+(16\times12)+(1\times15)=13519_{10}$.

2.16. $217_{10}=D9H$ — действия следующие:

$217_{10} : 16 = 13$, остаток $9_{10}=9H$ (МР);

$13_{10} : 16 = 0$, остаток $13_{10}=DH$ (СР).

2.17. $48373_{10} : 16 = 3023$, остаток $5_{10}=5H$ (МР);

$3023 : 16 = 188$, остаток $15_{10}=FH$;

$188 : 16 = 11$, остаток $12_{10}=CH$;

$11 : 16 = 0$, остаток $11_{10}=BH$ (СР).

Таким образом, $48373_{10}=BCF5H$.

2.3. ВОСЬМЕРИЧНЫЕ ЧИСЛА

Восьмеричная запись, как и шестнадцатеричная, используется для представления двоичных чисел. Восьмеричная система содержит 8 цифр от 0 до 7 и является соответственно системой с основанием 8. В табл. 2.7 представлено несколько десятичных, восьмеричных и двоичных чисел.

Преобразуем двоичное число 11111000100 в его восьмеричный эквивалент. Процедура действий в этом случае следующая. Начиная с МБ двоичного числа, делим его на группы из 3 бит. Затем, используя табл. 2.7, преобразуем каждую триаду (группу из 3 бит) в эквивалентную восьмеричную цифру. Таким образом, мы заменим двоичное число 11111000100 его восьмеричным эквивалентом 3704_8:

Двоичное число	011	111	000	100
Восьмеричное число	3	7	0	4

Таблица 2.7. *Десятичные, восьмеричные и двоичные эквиваленты*

Десятичные	Восьмеричные	Двоичные		
		4	2	1
0	0	0	0	0
1	1	0	0	1
2	2	0	1	0
3	3	0	1	1
4	4	1	0	0
5	5	1	0	1
6	6	1	1	0
7	7	1	1	1

Преобразуем теперь восьмеричное число 6521 в его двоичный эквивалент. Каждая восьмеричная цифра заменяется двоичной триадой и получится, что $6521_8 = 110101010001_2$:

Восьмеричное число	6	5	2	1
Двоичное число	110	101	010	001

Запишем восьмеричное число 2357 в десятичной форме Классическая процедура выполняется согласно табл. 2 8. Здесь 512, 64, 8 и 1 есть веса чегырех первых восьмеричных позиций. Заметим, что в этом примере содержится 7 единиц, 5 восьмерок, 4 числа 64 и два числа 521. Мы их складываем и получаем результат: $1024+192+40+7=1263_{10}$.

Таблица 2.8. *Восьмерично-десятичное преобразование*

Степень восьми	8^3		8^2		8^1		8^0	
Значения позиций	512		64		8		1	
Восьмеричное число	2		3		5		7	
	512		64		8		1	
	×		×		×		×	
	2		3		5		7	
Десятичное число	1024	+	192	+	40	+	7	$=1263_{10}$

Наконец, преобразуем десятичное число 3336 в его восьмеричный эквивалент. Процедура показана на рис. 2.3. В первую очередь 3336 разделено на 8, что дает частное 417 и остаток 0_{10}, причем $0_{10}=0_8$, восьмеричный 0 становится значением МР восьмеричного числа. Первое частное (417) становится делимым и снова делится на 8 (вторая строка), что дает частное 52 и остаток $1_{10}=1_8$, который становится второй цифрой восьмеричного числа. В третьей строке частное (52) становится делимым и деление его на 8 дает частное 6 и остаток $4_{10}=4_8$. В четвертой строке последнее частное 6 разделено на 8 с частным 0 и остатком $6_{10}=6_8$.

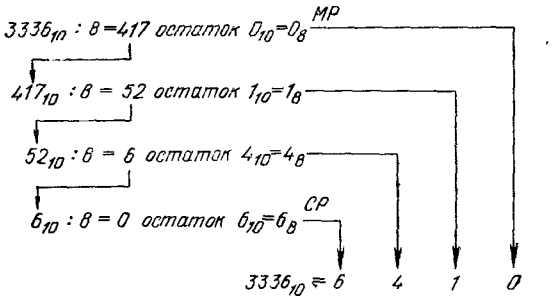


Рис 2.3. Десятично-восьмеричное преобразование

Теперь счет закончен последним частным 0. Цифра 6_8 становится значением СР восьмеричного числа, и мы можем видеть на рис. 2.3, что $3336_{10}=6410_8$.

Большинство микропроцессоров и микро-ЭВМ обрабатывают группы из 4, 8 или 16 бит. Отсюда следует, что обычно чаще используется шестнадцатеричная запись, чем восьмеричная. Однако восьмеричная запись более удобна, когда группы бит делятся на 3 (например, группы из 12 бит).

Упражнения

2.18. Для представления двоичных чисел текст документации 8-разрядного микропроцессора использует _____ (шестнадцатеричную, восьмеричную) систему.

2.19. Другим названием восьмеричной системы является _____.

2.20. Записать следующие восьмеричные числа в двоичном коде: а) 3; б) 7; в) 0; г) 7642; д) 1036; е) 2105.

2.21. Записать следующие двоичные числа в восьмеричном коде: а) 101; б) 110; в) 010; г) 111000101010; д) 1011000111; е) 100110100101.

2.22. $6724_8=$_____ 10.

2.23. $2648_{10}=$_____ 8.

Решения

2.18. Шестнадцатеричную, при которой удобно представить двоичное число двумя 4-разрядными группами. **2.19.** Система с основанием 8. **2.20.** а) $3_8=011_2$; б) $7_8=111_2$; в) $0_8=000_2$; г) $7642_8=111110100010_2$; д) $1036_8=1000011110_2$; е) $2105_8=10001000101_2$. **2.21.** а) $101_2=5_8$; б) $110_2=6_8$; в) $010_2=2_8$; г) $111000101010_2=7052_8$; д) $1011000111_2=1307_8$; е) $100110100101_2=4645_8$. **2.22.** Согласно процедуре табл. 2.8: $6724_8=(512\times6)+(64\times7)+(8\times2)+(1\times4)=3540_{10}$. **2.23.** Согласно процедуре рис. 2.3:

$2648_{10}:8=331$, остаток 0 (МР);
$331:8=41$, остаток 3;
$41:8=5$, остаток 1;
$5:8=0$, остаток 5 (СР);
$2648_{10}=5130_8$.

2.4. ДВОИЧНО-ДЕСЯТИЧНЫЕ ЧИСЛА

С целью удобства преобразования чистые двоичные числа представляются десятичными либо шестнадцатеричными. Однако двоично-десятичное преобразование — операция не простая. В калькуляторах, магистралях и числовых приборах, когда на доступных пользователю выходах и входах широко распространены десятичные числа, для их представления используют специальный двоично-десятичный код (ДДК). В табл. 2.9 приведено несколько десятичных чисел и соответствующих им двоично-десятичных эквивалентов (система 8421). Этим определяются веса позиций каждого из четырех бит ДДК (используют другие ДДК, например 5421 и плюс 3).

Таблица 2.9. *Двоично-десятичный код 8421*

Десятичные числа	Двоично-десятичные числа			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Запишем десятичное число 3691 в ДДК 8421. Каждая десятичная цифра преобразуется прямо в свой двоично-десятичный эквивалент из 4 бит, и преобразования дают $3691_{10}=0011\ 0110\ 1001\ 0001_{ДДК}$:

Десятичное число	3	6	9	1
Двоично-десятичное число	0011	0110	1001	0001

Преобразуем теперь двоично-десятичное число 1000 0000 0111 0010 в его десятичный эквивалент. Каждая группа из 4 бит прямо преобразуется в ее десятичный эквивалент, и тогда получаем $1000\ 0000\ 0111\ 0010_{ДДК}=8072_{10}$:

Двоично-десятичное число	1000	0000	0111	0010
Десятичное число	8	0	7	2

Микропроцессоры складывают чистые двоичные числа, но они обладают, однако, командами для преобразования результата своих сложений в двоично-десятичную запись. Полученные двоично-десятичные числа легко затем представить в десятичной записи, используя выше описанные простые процедуры.

Упражнения

2.24. Запись ДДК является сокращением _____.
2.25. Наиболее общей записью двоично-десятичного кода является ДДК _____ (5421, 8421).
2.26. Записать следующие дсятичные числа в ДДК 8421: а) 39; б) 65; в) 40; г) 17; д) 82; е) 99.
2.27. Записать следующие двоично-десятичные числа в десятичном коде: а) 1000 0000; б) 0000 0001; в) 1001 0010; г) 0111 0110; д) 0100 0011; е) 0101 0101.

Решения

2.24. Двоично-десятичного кода. **2.25.** 8421. **2.26.** Следуя процедуре, приведенной в § 2.4, получаем: а) $39_{10}=0011\ 1001_{ДДК}$; б) $65_{10}=0110\ 0101_{ДДК}$; в) $40_{10}=0100\ 0000_{ДДК}$; г) $17_{10}=0001\ 0111_{ДДК}$; д) $82_{10}=1000\ 0010_{ДДК}$; е) $99_{10}=1001\ 1001_{ДДК}$. **2.27.** Следуя процедуре, приведенной в § 2.4, получаем: а) $1000\ 0000_{ДДК}=80_{10}$; б) $0000\ 0001_{ДДК}=1_{10}$; в) $1001\ 0010_{ДДК}=92_{10}$; г) $0111\ 0110_{ДДК}=76_{10}$; д) $0100\ 0011_{ДДК}=43_{10}$; е) $0101\ 0101_{ДДК}=55_{10}$.

2.5. ДВОИЧНАЯ АРИФМЕТИКА

Сложение, вычитание или умножение двоичных чисел выполняются так же, как и в арифметике десятичных чисел. Большинство микропроцессоров владеет командами сложения и вычитания двоичных чисел, однако некоторые, менее многочисленные выполняют команды умножения и деления (например, микропроцессоры Intel 8086 и Intel 8088).

На рис. 2.4, *а* представлены простые правила двоичного сложения. Два первых (слева) правила очевидны, третье

Рис. 2.4. Двоичное сложение:
а — правила; *б* — пример

показывает, что 1+1=10, т. е. наиболее значимая 1 переносится в ближайший старший разряд. Четвертое правило, наконец, показывает, что 1+1+1=11. В этом случае первое, второе слагаемые и запоминаемое в результате сложения в младшем разряде число — все 1. Результатом является сумма —1 с переносом 1.

Сложим двоичные числа 0011 1011 и 0010 1010 (операция показана на рис. 2.4, *б*). Для большей ясности действия с десятичными эквивалентами обрабатываемых чисел показаны на рисунке справа. Суммой двух чисел 0011 1011 и 0010 1010 будет 0110 0101_2.

На рис. 2.5, *а* приведены правила двоичного вычитания. Первые три аналогичны десятичному вычитанию. Последнее требует заема из более значимого предшествующего разряда (в этом случае вес 2). Уменьшаемым является двоичное число 10, вычитаемым *1*, разностью — 1.

Вычтем двоичное число 0011 1001 из 0101 0101. Этот пример приведен на рис. 2.5, *б*. Разряды весов 1, 2 и 4 этого двоичного вычитания просты для выполнения и относятся к первым трем правилам на рис. 2.5, *а*. В колонке веса 8 имеет место вычитание 1 из 0. Тогда 1 занимается из ко-

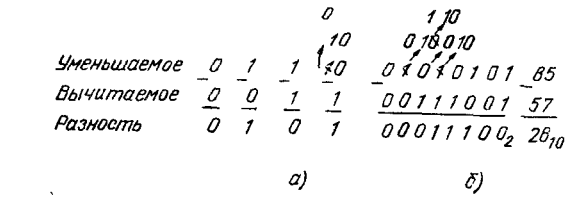


Рис. 2.5. Двоичное вычитание:
а — правила; *б* — пример

лонки веса 16. Единица вычитается из 10_2, что дает разность 1 согласно четвертому правилу на рис. 2.5, *а*. После этого заема в колонке веса 16 имеет место вычитание 1 из нового вычитаемого 0. Согласно четвертому правилу 1 должна быть занята из следующей, более значимой позиции (колонка веса 32), но в колонке 32 имеем 0; поэтому колонка 32 должна сделать заем из колонки веса 64, что и выполнено. Окончательно колонка 16 делает заем из колонки 32, уменьшаемым в колонке 16 становится 10_2, вычитаемым 1, разностью 1. В колонке 32 имеем 1—1=0, в колонке 64 — 0—0=0, в колонке 128 — 0—0=0. Таким образом, рис. 2.5, *б* иллюстрирует операцию вычитания 0011 1001_2 из 0101 0101_2 (справа эта задача решена в десятичной записи).

Приведем правила десятичного умножения:

Множимые	0	1	0	1
	×	×	×	×
Множители	0	0	1	1
Произведения	0	0	0	1

Два первых правила не требуют никаких пояснений. В двух следующих множителем является 1: когда множителем является 1 при двоичном умножении, *множимое становится результатом и представляет собой произведение*. Когда множитель 0, произведение всегда 0.

Выполним умножение 1101 на 101. Как и в случае умножения десятичных чисел, множимое сначала умножается на

число, стоящее в младшем разряде (в рассматриваемом случае — бит в колонке веса 1).

Множимое	1101	13
	×	×
Множитель	101	5
1-е частичное произведение	1101	65_{10}
2-е частичное произведение	0000	
3-е частичное произведение	1101	
Конечное произведение	1000001_2	

Поскольку бит множителя в разряде веса 1 является 1, множимое копируется и составляет первое частичное произведение. Вторым битом множителя является 0, тогда второе частичное произведение есть 0000 (заметим, что оно сдвинуто на одну позицию влево). Битом разряда веса 4 множителя является 1, тогда для получения третьего частичного произведения снова следует копирование множимого (заметим, что копирование завершается новым сдвигом на одну позицию влево). После этого выполняем сложение трех частичных произведений, что дает результат 1000001_2. Полученный результат $1101_2 \times 101_2 = 1000001_2$ соответствует произведению десятичных чисел $13_{10} \times 5_{10} = 65_{10}$.

Упражнения

2.28. Выполнить следующие сложения двоичных чисел:

а) 1010 + 0101; б) 1101 + 0101; в) 0101 1011 + 0000 1111; г) 0011 1111 + 0001 1111.

2.29. Выполнить следующие вычитания двоичных чисел:

а) 1110 − 1000; б) 1010 − 0101; в) 0110 0110 − 0001 1010; г) 0111 1000 − 0011 1111.

2.30. Первое число при умножении называется _____, второе — множителем, а результат составляет _____.

2.31. Выполнить следующие умножения двоичных чисел:

а) 1001 × 11; б) 1101 × 1001; в) 1111 × 101; г) 1110 × 1110.

Решения

2.28. См. рис. 2.4: а) 1111; б) 10010; в) 0110 1010; г) 0101 1110. **2.29.** См. рис. 2.5: а) 0110; б) 0101; в) 0100 1100; г) 0011 1001. **2.30.** Множимое. Произведение. **2.31.** Согласно § 2.5: а) 11011; б) 111 0101; в) 100 1011; г) 1100 0100.

2.6. ДОПОЛНИТЕЛЬНЫЙ КОД

Сама ЭВМ обрабатывает информацию обычно в двоичном коде. Однако если нужно использовать числа со знаком, используется специальный *дополнительный код*, что упрощает аппаратные средства ЭВМ.

На рис. 2.6, *а* приведено обычное изображение регистра МП или ячейки памяти вне МП. Такой регистр представля-

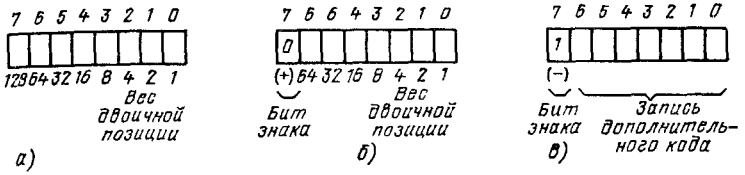


Рис. 2.6. Изображение регистра МП или ячейки памяти:
а — расположение двоичных позиций; *б* — идентификация положительных чисел нулем в знаковом бите; *в* — идентификация отрицательных чисел единицей в знаковом бите

ют пространством из 8 бит данных. Позиции бит пронумерованы от 7 до 0, а веса двоичных позиций указаны в основании регистра, бит 7 имеет вес 128, бит 6 — 64 и т. д.

На рис. 2.6, *б* и *в* показаны типовые структуры 8-разрядных регистров для размещения *чисел со знаком*. В обоих случаях бит 7 является знаковым. Он указывает, является ли число положительным (+) или отрицательным (−). При 0 в знаковом бите число положительно, при 1 — отрицательно.

Если, как показано на рис. 2.6, *б*, число положительно, оставшиеся ячейки памяти (6—0) содержат двоичное 7-разрядное число. Например, если регистр на рис. 2.6, *б* содержит 0100 0001, это соответствует числу $+65_{10}$ (64+1, знаковый бит положителен). Если в него записано 0111 1111, содержимым будет $+127_{10}$ (знаковый бит положителен: +64+32+16+8+4+2+1), что является наибольшим по-

ложительным числом, которое может содержать 7-разрядный регистр.

Если, как это показано на рис. 2.6, *в*, регистр содержит то же число со знаком, но отрицательное, он будет содержать дополнительный код этого числа. В табл. 2.10 приведена

Таблица 2.10. *Десятичные числа со знаком и их представление в дополнительном коде*

Десятичные	Представление чисел со знаком	Примечания
+127	0111 1111	
.	.	
.	.	
.	.	
+8	0000 1111	Положительные числа представлены в той же форме, что и прямые двоичные числа
+7	0000 0111	
+6	0000 0110	
+5	0000 0101	
+6	0000 0100	
+3	0000 0011	
+2	0000 0010	
+1	0000 0001	
+0	0000 0000	
—1	1111 1111	Отрицательные числа представлены в форме дополнительного кода
—2	1111 1110	
—3	1111 1101	
—4	1111 1100	
—5	1111 1011	
—6	1111 1010	
—7	1111 1001	
—8	1111 1000	
.	.	
.	.	
.	.	
—128	1000 0000	

запись в дополнительном коде положительных и отрицательных чисел. Заметим, что все положительные числа имеют 0 в старшем бите, остальные биты составляют двоичное число. Все отрицательные числа имеют 1 в старшем разряде. Рассмотрим строку +0 в табл. 2.10: запись в дополнительном коде +0 будет 0000 0000. В ближайшей нижней строке видим, что запись в дополнительном коде —1 следующая: 1111 1111. Рассмотрим пошаговое перемещение в обратном направлении от 0000 0000 до 1111 1111.

Какой будет запись в дополнительном коде числа —9? Рассмотрим этапы преобразования. Они следующие:

Десятичное числ	9	Этап *1*.	Запись десятичного числа без знака (9)
Двоичное число	0000 1001	Этап *2*.	Преобразование десятичного числа в двоичный код (0000 1001)
Дополнение до 1 (обратный или инверсный код)	1111 0110	Этап *3*.	Получить обратный код двоичного числа заменой нулей единицами, а единиц — нулями (1111 0110)
Дополнение до 2 (дополнительный код)	+1 1111 0111	Этап *4*.	Прибавить единицу к обратному коду. Здесь прибавить 1 к 1111 0110, что дает 1111 0111

Полученный результат является дополнительным кодом положительного десятичного числа. В приведенном примере дополнительным кодом числа 9 является 1111 0111. Заметим, что знаковый бит —1, это означает, что рассматриваемое число (1111 0111) отрицательно.

Каким будет десятичный эквивалент числа 1111 0000, записанного в форме дополнительного кода? Процедура преобразований в этом случае следующая:

Дополнительный код	1111 0000	Этап *1*.	Запись дополнительного кода (1111 0000).
Дополнение до 1	0000 1111	Этап *2*.	Получается обратный код дополнительного кода заменой нулей единицами, а единиц — нулями (0000 1111)
Двоичное число	+1 0001 0000=16	Этап *3*.	Добавить 1

Таким образом, формирование обратного кода и добавление 1 являются теми же процедурами, которые мы проводили при преобразовании двоичного числа в дополнительный код. Однако следует отметить, что, хотя мы получили двоичное число 0001 0000=16_{10}, исходная запись дополнительного кода 1111 0000=—16, т. е. имеем отрицательное число, поскольку старший бит в дополнительном коде является 1.

Упражнения

2.32. Когда числа со знаком помещаются в 8-разрядный регистр микропроцессора, старший (7-й) бит называется _____.

2.33. Установить, являются ли следующие числа в дополнительном коде положительными или отрицательными:

а) 0111 0000; б) 1100 1111; в) 1000 1111; г) 0101 0101.

2.34. Используя табл. 2.10, дать дополнительный код следующих десятичных чисел со знаком: а) +1; б) +5; в) +127; г) —1; д) —2; е) —128.

2.35. Используя процедуру, приведенную в § 2.6, дать дополнительный код следующих десятичных чисел со знаком: а) —10; б) —21; в) —34; г) —96.

2.36. Расположение бит в дополнительном коде _____ (в ДДК, в двоичном коде) одинаково для положительных двоичных чисел.

2.37. Используя процедуру, приведенную в § 2.6, дать десятичные эквиваленты следующих чисел в дополнительном коде: а) 1111 1011; б) 0000 1111; в) 1000 1111; г) 0111 0111.

Решения

2.32. Знаковым. **2.33.** а) 0111 0000 в дополнительном коде положительно, так как знаковый бит — 0; б) 1100 1111 в дополнительном коде отрицательно, так как знаковый бит — 1; в) 1000 1111 в дополнительном коде отрицательно, так как знаковый бит — 1; г) 0101 0101 в дополнительном коде положительно, так как знаковый бит — 0. **2.34.** См. табл. 2.10: а) $+1=0000\ 0001$; б) $+5=0000\ 0101$; в) $+127=0111\ 1111$; г) $-1=1111\ 1111$; д) $-5=1111\ 1110$; е) $-128=1000\ 0000$. **2.35.** В соответствии с рекомендациями § 2.6 имеем следующие результаты: а) этап 1 (преобразование десятичного числа в двоичный код): $10_{10}=0000\ 1010_2$; этап 2 (получение обратного кода): $0000\ 1010_2\rightarrow 1111\ 0101$; этап 3 (добавление 1): $1111\ 0101+1=1111\ 0110$ (доп. код); б) этап 1: $21_{10}=0001\ 0101_2$; этап 2: $0001\ 0101_2\rightarrow 1110\ 1010$; этап 3: $1110\ 1010+1=1110\ 1011$ (доп. код); в) этап 1: $34_{10}=0010\ 0010_2$; этап 2: $0010\ 0010_2\rightarrow 1101\ 1101$; этап 3: $1101\ 1101+1=1101\ 1110$ (доп. код); г) этап 1: $96_{10}=01100\ 0000_2$; этап 2: $0110\ 0000_2\rightarrow 1001\ 1111$; этап 3: $1001\ 1111+1=1010\ 0000$ (доп. код). **2.36.** В двоичном коде. **2.37.** а) 1111 1011 (доп. код) $=-5_{10}$

$$1111\ 1011 \xrightarrow{\text{обратный код}} 0000\ 0100$$
$$+\quad 1$$
$$0000\ 0101 = 5;$$

б) 0000 1111 (доп. код) $=+15$;

в) 1000 1111 (доп. код) $=-113$

$$1000\ 1111 \xrightarrow{\text{обратный код}} 0111\ 0000$$
$$+\quad 1$$
$$0111\ 0001 = 113;$$

г) 0111 0111 (доп. код) $=+119$.

2.7. АРИФМЕТИКА В ДОПОЛНИТЕЛЬНОМ КОДЕ

Микропроцессор может использовать числа в форме дополнительного кода, потому что он в состоянии выполнять операции *дополнения (инверсии)*, *инкрементирования* (добавления 1 к числу) и *сложения* двоичных чисел. Микропроцессор не приспособлен для прямого вычитания. Он использует сумматоры и для выполнения вычитания оперирует над дополнительным кодом.

Сложим десятичные числа +5 и +3. Рассмотрим процедуру действий в случае одновременного сложения чисел в десятичном и в дополнительном кодах:

1-е число	(+5)	0000 0101
	+	+
2-е число	(+3)	0000 0011
	(+8)	0000 1000

Согласно табл. 2.10 $+5=0000\ 0101$ в дополнительном коде аналогично $+3=0000\ 0011$. Тогда числа в дополнительном коде 0000 0101 и 0000 0011 складываются, как обычные двоичные числа, давая сумму 0000 1000 в дополнительном коде, т. е. $0000\ 1000=+8_{10}$.

Пусть надо сложить десятичные числа +7 и —3. Согласно табл. 2.10 $+7=0000\ 0111$ и $-3=1111\ 1101$ соответственно в дополнительном коде. Они затем складываются, как обычные двоичные числа, и результат 1 0000 0100 получается в дополнительном коде:

1-е число	(+7)	0000 0111
	+	+
2-е число	(—3)	1111 1101
	(+4)	1 0000 0100

Пренебречь переполнением.

Старший бит является переполнением 8-разрядного регистра, и им можно пренебречь. Получаем сумму 0000 0100 или $+4_{10}$.

Сложим десятичные числа +3 и —8. Согласно все той же табл. 2.10 $+3=0000\ 0011$ и $-8=1111\ 1000$. Их дополнительные коды 0000 0011 и 1111 1000 складываются, как обычные двоичные числа, что дает $1111\ 1011=-5_{10}$:

1-е число	(+3)	0000 0011
	+	+
2-е число	(—8)	1111 1000
	(—5)	1111 1011

Сложим десятичные числа —2 и —5. В дополнительном коде согласно табл. 2.10 —2=1111 1110 и —5=1111 1011. Два числа 1111 1110 и 1111 1011 складываются, как обычные десятичные числа, что дает 1 1111 1001:

1-е число (—2) 1111 1110
+ +
2-е число (—5) 1111 1011
(—7) 1 1111 1001

Пренебречь переполнением.

Старший бит результата является переполнением 8-разрядного регистра, и им пренебрегаем. Таким образом, суммой двух чисел 1111 1110 и 1111 1011 в дополнительном коде будет 1111 1001. Согласно табл. 2.10 сумма 1111 1001=-7_{10}.

Вычтем теперь десятичное число +5 из десятичного числа +8. Первое число +8=0000 1000, второе +5=0000 0101. В дополнительный код (инвертировать и добавить 1) должно быть преобразовано число 0000 0101, что дает 1111 1011. Затем первое число 0000 1000 складывается с дополнительным кодом второго 1111 1011, как с обычным двоичным числом, что дает 1 0000 0011:

1-е число (+8) 0000 1000
— Дополнительный код
2-е число (+5) =0000 0101 → 1111 1011
(+3) 1 0000 0011

Пренебречь переполнением.

Старший бит является переполнением регистра, им пренебрегаем, что дает результат 0000 0011=$+3_{10}$. Заметим, что второе число было представлено в дополнительном коде, затем сложено с первым. *Используя дополнительный код и сумматор, микропроцессор выполняет вычитание.*

Вычтем теперь большее десятичное число +6 из десятичного числа +2:

1-е число (+2) 0000 0010
— Дополнительный код +
2-е число (+6) =0000 0110 → 1111 1010
(—4) 1111 1100

Дополнительный код первого числа +2=0000 0010, второе число +6=0000 0110, его дополнительный код (инверсия и добавление 1) — 1111 1010. Оба эти кода сложены затем, как обычные двоичные числа, что дает 1111 1100, а согласно табл. 2.10 1111 1100=-4_{10}.

Упражнения

2.38. Сложить следующие десятичные числа со знаком, используя метод дополнительного кода:

а) (+7) + (+1) б) (+31) + (+26).

2.39. Сложить следующие десятичные числа со знаком, используя метод дополнительного кода:

а) (+8) + (—5) б) (+89) + (—46).

2.40. Сложить следующие десятичные числа со знаком, используя метод дополнительного кода:

а) (+1) + (—6) б) (+20) + (—60).

2.41. Сложить следующие десятичные числа со знаком, используя метод дополнительного кода:

а) (—3) + (—4) б) (—13) + (—41).

2.42. Вычесть следующие десятичные числа со знаком, используя метод дополнительного кода:

а) (+7) — (+2) б) (+113) — (+50).

2.43. Вычесть следующие десятичные числа со знаком, используя метод дополнительного кода:

а) (+3) — (+8) б) (+12) — (+63).

Решения

2.38. Следовать процедуре, приведенной в § 2.7:

а) (+7) 0000 0111
+ +
(+1) 0000 0001
(+8) 0000 1000 (доп. код);

б) (+31) 0001 1111
+ +
(+26) 0001 1010
(+57) 0011 1001 (доп. код).

2.39. Следовать процедуре, приведенной в § 2.7:

а) (+8) 0000 1000 + (−5) 1111 1011 = (+3) 10000 0011 (доп. код);

б) (+89) 0101 1001 + (−46) 1101 0010 = (+43) 10010 1011 (доп. код).

Пренебречь переполнением. Пренебречь переполнением.

2.40. Следовать процедуре, приведенной в § 2.7:

а) (+1) 0000 0001 + (−6) 1111 1010 = (−5) 1111 1011 (доп. код);

б) (+20) 0001 0100 + (−60) 1100 0100 = (−40) 1101 1000 (доп. код).

2.41. Следовать процедуре, приведенной в § 2.7:

а) (−3) 1111 1101 + (−4) 1111 1100 = (−7) 11111 1001 (доп. код);

б) (−13) 1111 0011 + (−41) 1101 0111 = (−54) 11100 1010 (доп. код).

Пренебречь переполнением. Пренебречь переполнением.

2.42. Следовать процедуре, приведенной в § 2.7:

б) (+7) 0000 0111

(+2) = 0000 0010 $\xrightarrow{\text{дополнительный код и сложить}}$ + 1111 1110

(+5) 10000 0101 (доп. код).

Пренебречь переполнением.

б) (+113) 0111 0001

(+50) = 0011 0010 $\xrightarrow{\text{дополнительный код и сложить}}$ + 1100 1110

10011 1110 (доп код)

Пренебречь переполнением.

2.43. Следовать процедуре, приведенной в § 2.7:

а) (+3) 0000 0011

(+8) = 0000 1000 $\xrightarrow{\text{дополнительный код и сложить}}$ + 1111 1000

(−5) 1111 1011 (доп. код);

б) (+12) 0000 1100

(+63) = 0011 1111 $\xrightarrow{\text{дополнительный код и сложить}}$ + 1100 0001

(−51) 1100 1101 (доп. код).

2.8. ГРУППИРОВКИ БИТ

Одна отдельная двоичная цифра называется битом, сгруппированных 4 бит составляют тетраду, 8 бит — *байт*.

Входящий в состав МП аккумулятор является очень важной частью всего МП. Обычно МП содержат 8-разрядный аккумулятор. *Длина слова* МП составляет тогда 8 бит, т. е. в этом случае 1 байт составляет слово. Микропроцессор может иметь длину слова в 4, 8, 16 и даже 32 бит. Таким образом, 16-разрядный МП имеет длину слова 2 байт или 16 бит. *Слово* — это одна группа обрабатываемых бит, единое выражение или одна команда микропроцессора. Восьмиразрядный микропроцессор переносит и помещает все данные группами из 8 бит, которые передаются во-

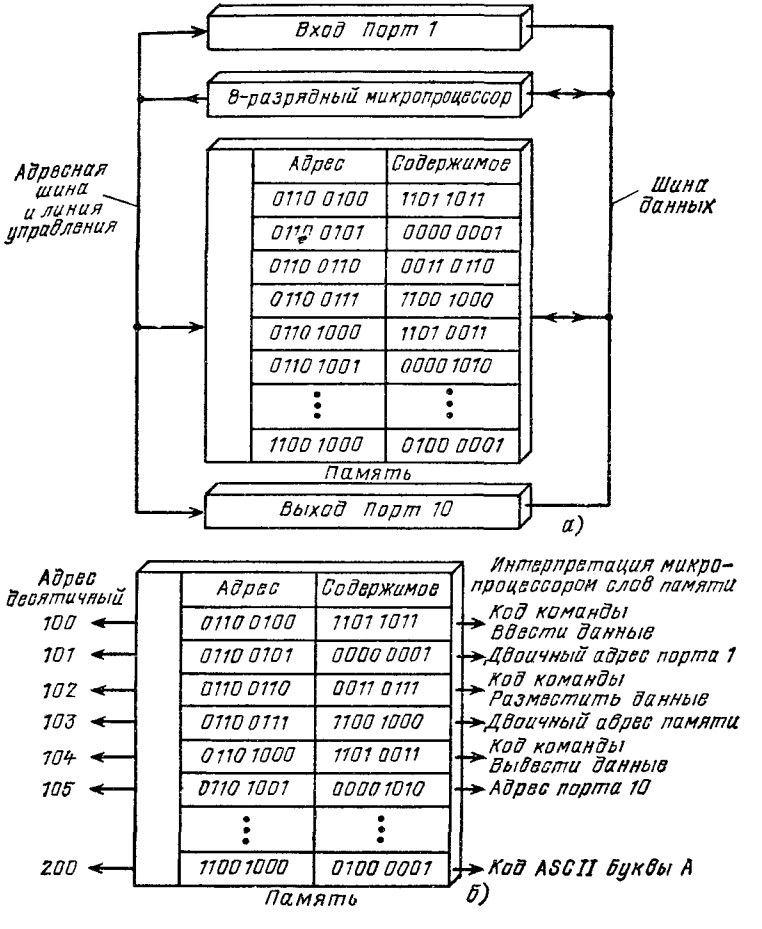


Рис. 2.7. Представленне памяти микро-ЭВМ:
а — типичное содержание; *б* — интерпретация содержимого МП

семью параллельными проводниками, составляющими шину данных. На рис. 2.7, *а* приведено состояние, которое могла бы иметь память 8-разрядной микро-ЭВМ. Заметим, что каждая адресуемая позиция (названная содержимым) составляет группу из 8 бит информации. Каждый байт называется запоминаемым словом, поскольку МП является 8-разрядным устройством. Каждое запоминаемое слово имеет особое значение, когда оно извлечено и декодировано микропроцессором. Содержимое памяти на рис. 2.7, *а* может иметь один из следующих смыслов: 1) двоичное число; 2) двоичное число со знаком; 3) двоично-десятичное число; 4) буква алфавита; 5) команда; 6) адрес памяти; 7) адрес порта ввода или вывода.

Рассмотрим верхнюю ячейку памяти на рис. 2.7, *а*, т. е. ячейку 0110 0100_2. Ее содержимым является 1101 1011. Это двоичное слово могло бы быть интерпретировано как

1) двоичное число 1101 $1011_2 = 219_{10}$;

2) двоичное число со знаком 1101 1011 $= -37_{10}$ (здесь подразумевается дополнительный код);

3) двоично-десятичное число — это невозможно, так как ни 1101, ни 1011 не представляет двоично-десятичный код;

4) буква алфавита — это не соответствует ни одной букве кода ASCII (ASCII — широко распространенный буквенно-цифровой код, см. § 2.9);

5) командой, 1101 1011 — команда INPUT (ВВЕСТИ) для хорошо известного процессора Intel 8080/8085;

6) адрес ячейки памяти 1101 $1011_2 = DB_{16} = 219_{10}$;

7) адрес порта ввода или вывода 1101 $1011_2 = 219_{10}$.

Оператор МП Intel 8085 включит счетчик команд с адреса 100_{10} (0110 0100_2), МП извлечет, затем декодирует слово в памяти 1101 1011 как команду ВВЕСТИ (INPUT) данные. Микропроцессор обратится затем к следующему адресу 101_{10} (0110 0101_2). Содержимое памяти на рис. 2.7, *а* то же, что и на рис. 1.4. Программа на рис. 1.4 выполняет следующие команды:

1) ВВЕСТИ (INPUT) данные, приходящие из порта 1;

2) ПОМЕСТИТЬ (STORE) эти данные в ячейку памяти 200;

3) ВЫВЕСТИ (OUTPUT) эти данные в порт 10.

Способ, по которому МП интерпретирует содержимое ячеек памяти, детализирован на рис. 2.7, *б*. Команды программы помещены в шесть верхних ячеек (100—105). Нижняя ячейка памяти (200_{10}) является местом размещения данных. В этом случае код ASCII для буквы *A* помещен в эту ячейку памяти.

В итоге важно отметить, что биты сгруппированы в слова внутри микро-ЭВМ. Эти слова памяти программы интерпретируются микро-ЭВМ одно за другим последовательно. Программисту очень важно знать, как микро-ЭВМ располагает и интерпретирует данные. У каждого МП есть свой состав команд, но у всех микропроцессоров доступ к ячейкам памяти осуществляется одинаково.

Упражнения

2.44. Группа из 4 бит составляет тетраду, а группа из 8 бит — _____.

2.45. Длина _____ является важной особенностью МП. Она соответствует числу передаваемых, обрабатываемых бит одной сущности.

2.46. Обратиться к рис. 2.7, *а*. Байт данных, помещенный в какую-либо ячейку памяти, называется словом _____.

2.47. Дать список семи возможных толкований 8-разрядного слова в памяти.

2.48. Как на рис. 2.7, *б* МП интерпретирует слово 0000 0001 по адресу 101_{10}?

2.49. Как на рис. 2.7, *б* МП интерпретирует слово 0011 0111 по адресу 102_{10}?

Решения

2.44. Байт. В некоторых случаях эта группа может составлять также слово. **2.45.** Слова. **2.46.** Данных. **2.47.** Двоичное число, двоичное число со знаком (записанное в дополнительном коде), двоично-десятичное число, буква алфавита, команда, адрес памяти, адрес порта ввода/вывода. **2.48.** Он извлекает заполненное слово, ожидая, что оно ему укажет, из какого порта он должен взять данные. Это слово укажет ему, что речь идет о порте 1. **2.49.** Он извлекает слово 0011 0111, ожидая новой команды. Это слово декодируется микропроцессором как команда ПЕРЕДАТЬ (MOVE) данные из аккумулятора в ячейку памяти, адрес которой приводится в следующей ячейке памяти.

2.9. БУКВЕННО-ЦИФРОВОЙ КОД

Когда микро-ЭВМ взаимодействует с телетайпом или видеотерминалом, необходимо прибегать к коду, который одновременно включает в себя числовые и алфа-

витные знаки. Такие коды называются буквенно-цифровыми.

Наиболее распространен буквенно-цифровой код ASCII (произносится АСКИ) — стандартный американский код обмена информации.

В табл. 2.11 приведена выдержка 7-разрядного кода ASCII. В этот список входят 7-разрядные коды цифр, прописных букв и знаков пунктуации. Полный код ASCII включает кодирование строчных букв и признаков команд.

Таблица 2.11. *Выдержка из алфавитно-цифрового кода* ASCII

Символ	Код ASCII	Символ	Код ASCII	Символ	Код ASCII
Массив	010 0000	0	011 0000	I	100 1001
!	010 0001	1	011 0001	J	100 1010
»	010 0010	2	011 0010	K	100 1011
#	010 0011	3	011 0011	L	100 1100
$	010 0100	4	011 0100	M	100 1101
%	010 0101	5	011 0101	N	100 1110
&	010 0110	6	011 0110	O	100 1111
,	010 0111	7	011 0111	P	101 0000
(	010 1000	8	011 1000	Q	101 0001
)	010 1001	9	011 1001	R	101 0010
*	010 1010	A	100 0001	S	101 0011
+	010 1011	B	100 0010	T	101 0100
,	010 1100	C	100 0011	U	101 0101
—	010 1101	D	100 0100	V	101 0110
.	010 1110	E	100 0101	W	101 0111
/	010 1111	F	100 0110	X	101 1000
		G	100 0111	Y	101 1001
		H	100 1000	Z	101 1010

Упражнения

2.50. Двоичный код, используемый обычно для кодирования цифр и букв, называется _____ кодом.

2.51. Нуль в коде ASCII представляется как 011 0000, 9 — как _____.

2.52. Если маскировать три старших разряда цифр от 0 до 9 в коде ASCII, что оставит маска в итоге?

Решения

2.50. Буквенно-цифровым. **2.51.** См. табл. 2.11: 011 1001. **2.52.** Останется 4 немаскированных бита, составляющих двоичный или двоично-десятичный эквивалент каждого числа (см. табл. 2.11).

Дополнительные упражнения к гл. 2

2.53. Дать на память десятичные эквиваленты каждого из следующих двоичных чисел: а) 0000; б) 0010; в) 0011; г) 0111; д) 1001; е) 1100.

2.54. $0110\ 1001_2$=_____.

2.55. 60_{10}=_____.

2.56. Двоичное число 1001 1100 представляет собой число 9С в _____ записи.

2.57. 8_{16}=_____.

2.58. $0101\ 1111_2$=_____.

2.59. AE_{16}=_____.

2.60. $1011\ 1100_2$=_____.

2.61. $3C_{16}$=_____.

2.62. 90_{10}=_____.

2.63. 7130_8=_____.

2.64. $1001\ 0111\ 0010_2$=_____.

2.65. 57_8=_____.

2.66. 63_{10}=_____.

2.67. 92_{10}=_____ ДДК.

2.68. $1000\ 0110_{ДДК}$ =_____.

2.69. Выполнить сложение следующих двоичных чисел: а) 1100 0011+0011 1100; б) 0110 1110+0011 1101.

2.70. $1101\ 1000_2$—$0011\ 0011_2$=_____.

2.71. $1001_2 \times 1101_2$=_____.

2.72. Когда числа со знаком помещаются в регистр микропроцессора, 1 в старшем бите означает, что число _____ (положительное, отрицательное).

2.73. Запись в дополнительном коде 0111 1110 представляет собой _____ (отрицательное, положительное) число.

2.74. Записать следующие десятичные числа со знаком в дополнительном коде: а) +12; б) —12.

2.75. Записать следующие числа (в дополнительном коде) десятичными числами со знаком: а) 0111 0100; б) 1101 1101.

2.76. Сложить следующие десятичные числа со знаком, используя метод дополнительного кода:

а)	б)	в)
(+13)	(+17)	(—6)
+	+	+
(+8)	(—8)	(—14).

2.77. Вычесть следующие десятичные числа со знаком, используя метод дополнительного кода:

а) (+13) б) (+19)

(+5) (+29).

2.78. Байт — это группа из _____ бит.

2.79. Тетрада состоит из _____ бит.

2.80. Наиболее широко распространена длина слова микропроцессора _____ (8, 32) бит.

2.81. См. рис. 2.7, *а*. На этом рисунке приведена функциональная схема _____ (микропроцессора, микро-ЭВМ).

2.82. См. рис. 2.7, *б*. Как интерпретирует микропроцессор слово 1101 0011 по адресу 104_{10}?

2.83. Сокращение ASCII читается по русски _____ .

2.84. Для вывода на видеотерминал используется специальный код _____ .

Решения

2.53. а) 0; б) 2; в) 3; г) 7; д) 9; е) 12. **2.54.** 105_{10}. **2.55.** 111100_2. **2.56.** Шестнадцатеричной. **2.57.** $1000\ 1101_2$. **2.58.** $5F_{16}$. **2.59.** 1010 1110. **2.60.** BC_{16}. **2.61.** 60_{10}. **2.62.** $5A_{16}$. **2.63.** $111001\ 011000_2$. **2.64.** 4562_8. **2.65.** 47_{10}. **2.66.** 77_8. **2.67.** $1001\ 0010_{ДДК}$. **2.68.** 86_{10}. **2.69.** а) $1111\ 1111_2$; б) $1010\ 1011_2$. **2.70.** $1010\ 0101_2$. **2.71.** 1110101_2. **2.72.** Отрицательное. **2.73.** Положительное. **2.74.** а) 0000 1100; б) 1111 0100. **2.75.** а) $+116_{10}$; б) -35_{10}. **2.76.** а) 0001 0101 (доп. код); б) 1111 0110 (доп. код); в) 1110 1100 (доп. код). **2.77.** а) 0000 1000 (доп. код); б) 1111 0110 (доп. код). **2.78.** 8. **2.79.** 4. **2.80.** 8. **2.81.** Микро-ЭВМ. **2.82.** Он извлекает из памяти слово 1101 0011, полагая, что это новая команда Слово 1101 0011 декодируется микропроцессором как команда ВЫВЕСТИ данные, содержащиеся в аккумуляторе, в порт, адрес которого будет находиться в следующей ячейке памяти **2.83.** Американский стандартный код обмена информацией. **2.84.** ASCII или буквенно-цифровой.

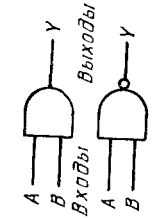
Глава 3

ОСНОВНЫЕ ЭЛЕМЕНТЫ ЦИФРОВОЙ ТЕХНИКИ

3.1. ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

Используемые для обработки цифровых сигналов устройства называются логическими элементами, и для их идентификации используют *логические символы*. В табл. 3.1 приведены семь основных логических элементов цифро-

Таблица 3.1. Семь логических функций

Логические функции	Обозначения логических элементов	Таблицы истинности	Булева функция
Инвертор НЕ (NOT)		Вход / Выход A / $\overline{A}$ 0 / 1 1 / 0	$A=\overline{A}$
И (AND) НЕ-И (NAND)		Входы: A B / Выходы: И, НЕ-И 0 0 / 0 1 0 1 / 0 1 1 0 / 0 1 1 1 / 1 0	$A \cdot B = Y$ $\overline{A \cdot B} = Y$

Продолжение табл. 3.1

Логические функции	Обозначения логических элементов	Таблицы истинности	Булева функция
ИЛИ (OR) НЕ-ИЛИ (NOR)		Входы B A / Выходы ИЛИ НЕ-ИЛИ 0 0 / 0 1 0 1 / 1 0 1 0 / 1 0 1 1 / 1 0	$A+B=Y$ $\overline{A+B}=Y$
ИЛИ ИСКЛЮЧАЮЩЕЕ (XOR) НЕ-ИЛИ ИСКЛЮЧАЮЩЕЕ (XNOT-OR)		Входы B A / Выходы XOR НЕ-XOR 0 0 / 0 1 0 1 / 1 0 1 0 / 1 0 1 1 / 0 1	$A\oplus B=Y$ $\overline{A\oplus B}=Y$

вых систем. В таблице истинности 0 означает низкий уровень напряжения (LOW), а 1 — высокий (HIGH). В правой колонке приведены булевы функции[1], выполняемые каждым из логических элементов.

Приведенный на рис. 3.1 пример несколько поясняет способ преобразования информации логическими элементами. Каким будет сигнал на выходе инвертора (часто называется элементом отрицания НЕ) на рис. 3.1, когда на его вход поступает импульс *a*? Согласно второй строке таблицы истинности (табл. 3.1) на выходе должен быть 0, т. е. значение, противоположное входному сигналу. Когда на вход инвертора подается импульс *b* (0 или LOW), выход перейдет в состояние 1 (H-состояние). Импульс *c* вызовет LOW (L-состояние) на выходе, тогда как импульс *d* вызовет на выходе H-состояние. Процесс инверсии называется также *дополнением* или *отрицанием*. Булевой функцией дополнения является $A=\overline{A}$ (говорят НЕ—*A*). Черта сверху *A* читается как НЕ и означает, что надо инвертировать или дополнить (до 1) переменную, над которой она стоит.

Рис. 3.1. Пример инверсии

Рис. 3.2. Выполнение логических операций: *а* — И; *б* — ИЛИ; *в* — ИЛИ ИСКЛЮЧАЮЩЕЕ

На рис. 3.2, *а* приведен другой пример — элемент И с двумя входами. Импульсами *a* на его входах являются 0 и 1. Согласно таблице истинности (табл. 3.1) это должно вызвать 0 (LOW) на выходе. Импульсы *a*, *b* и *c* вызовут на выходе L-уровень. Когда же оба

[1] Дж. Буль (1815—1864) — английский математик и логик, им заложены основы математической логики. Более подробно см. [2]. — *Прим. ред.*

входа элемента И станут HIGH (см. импульсы *d* на рис. 3.2, *а*), выход становится равным 1 или HIGH.

Рассмотрим задачу, приведенную на рис. 3.2, *б*. В этом случае тетрады 1010 (на входе *A*) и 1001 (на входе *B*) совместно поступают на вход логического элемента ИЛИ. Тетрада на выходе может быть определена по таблице истинности (табл. 3.1).

В результате логической операции ИЛИ над 1010 и 1001 получим на выходе 1011. Отметим, что функция ИЛИ сначала выполняется с импульсами *a*, затем *b* и т. д.

Какой будет тетрада на выходе, если 1010 и 1001 будут подвергнуты операции ИЛИ ИСКЛЮЧАЮЩЕЕ (XOR), как показано на рис. 3.2, *в*? Согласно таблице истинности XOR видим, что результатом операции XOR с тетрадами 1010 и 1001 будет 0011.

Таким образом, микропроцессор может выполнять логические операции. Обычно микропроцессоры наделены способностью выполнять команды логических операций НЕ (дополнение или отрицание), И, ИЛИ и ИЛИ ИСКЛЮЧАЮЩЕЕ. Эти команды выполняются побитно аналогично приведенным в табл. 3.1 и на рис. 3.1, 3.2.

Упражнения

3.1. Перечислить семь логических функций.

3.2. Перечислить четыре логические функции, которые могут быть выполнены, как правило, одной командой.

3.3. Если МП выполняет функцию 1100 И 1011, тетрадой выхода будет_____.

3.4 Если МП выполняет функцию 0011 ИЛИ 1000, тетрадой выхода будет_____.

3.5. Если МП инвертирует (операция НЕ) тетраду 1001, результатом будет_____.

3.6. Если МП выполняет функцию 0011 ИЛИ ИСКЛЮЧАЮЩЕЕ 0110, результатом будет_____.

3.7. Записать выходы элемента НЕ-И на рис. 3.3.

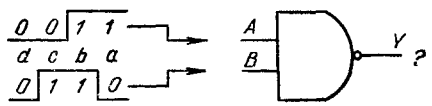


Рис. 3.3. К упражнению 3.7

3.8. Записать выходы элемента НЕ-ИЛИ на рис. 3.4.

3.9. Записать выходы элемента НЕ-ИЛИ ИСКЛЮЧАЮЩЕЕ на рис. 3.5.

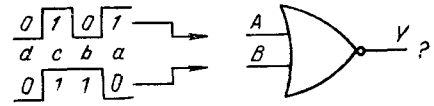


Рис 3.4. К упражнению 3.8

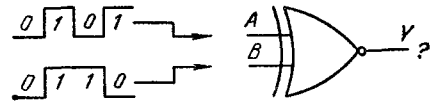


Рис. 3.5. К упражнению 3.9

Решения

3.1. См. табл. 3.1. Семь логических функций: НЕ (инверсия), И, НЕ-И, ИЛИ, НЕ-ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ, НЕ-ИЛИ ИСКЛЮЧАЮЩЕЕ. **3.2.** НЕ, И, ИЛИ и ИЛИ ИСКЛЮЧАЮЩЕЕ. **3.3.** См. табл. 3.1. Результат 1000. **3.4.** См. табл. 3.1 — таблица истинности ИЛИ. 1011. **3.5.** 0110. **3.6.** 0101. **3.7.** См. таблицу истинности в табл. 3.1. Импульсы на выходе будут иметь значения: при импульсах *a* — 1; при импульсах *b* — 0; при импульсах *c* — 1; при импульсах *d* — 1; **3.8.** При импульсах *a* — 0; при импульсах *b* — 0; при импульсах *c* — 0; при импульсах *d* — 1. **3.9.** При импульсах *a* — 0; при импульсах *b* — 0; при импульсах *c* — 1; при импульсах *d* — 1.

3.2. КОМБИНАЦИИ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

Цифровые системы строятся на основе комбинаций логических элементов. Такие комбинации могут быть описаны таблицей истинности, булевой функцией или логической схемой.

Рассмотрим таблицу истинности (табл. 3.2). Эта таблица описывает все возможные комбинации четырех входов (*A*, *B*, *C*, *D*). Заметим, что только комбинация 1010 даст 1 или Н-сигнал на выходе. Эквивалентная этой таблице истинности булева функция приведена в заголовке табл. 3.2. Входы подчинены операции И, что дает булеву функцию $D \cdot \overline{C} \cdot B \cdot \overline{A} = Y$ (читается как *D* и НЕ-*C* и *B* и НЕ-*A* дают на выходе *Y*).

Таблица 3.2. *Получение булевой функции из таблицы истинности* $D \cdot \bar{C} \cdot B \cdot A = Y$

Входы				Выход	Входы				Выход
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	1
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	0

Логическая схема построена исходя из булевой функции (см. рис. 3.6, *а*). Входы *A* и *C* должны быть инвертированы инверторами. На выходе использован элемент И с четырьмя входами.

Явно более простой вариант такой логической схемы приведен на рис. 3.6, *б*. На этой схеме инверторы обозначены маленькими кружками, называемыми *кружками инверсии*, которые могут рассматриваться как LOW-*активные входы*. Другими словами, чтобы активизировать элемент И, на рис. 3.6, *б* входы *A* и *C* должны быть LOW, а входы *B* и *D* — HIGH. Так как входы *D* и *B* должны быть HIGH для активизации элемента И, их называют HIGH-*активными входами*.

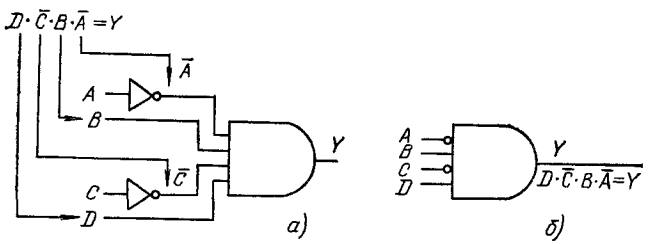


Рис. 3.6. Построение логических схем:
а — на основании булевой функции; *б* — упрощенный вариант

Рассмотрим другую таблицу истинности (табл. 3.3). Здесь две комбинации входов вызовут 1 или HIGH на выходе. Булевой функцией, соответствующей этой таблице истинности, становится тогда такая: $D \cdot C \cdot \bar{B} \cdot \bar{A} + D \cdot C \times \times \bar{B} \cdot A = Y$ (читается: *D* и НЕ-*C* и НЕ-*B* и НЕ-*A* или *D* и *C* и НЕ-*B* и *A* равно *Y* на выходе).

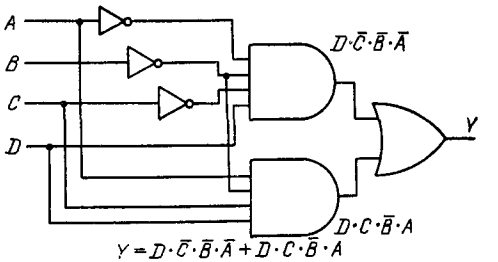


Рис. 3.7. Построение логической схемы по булевой функции

Затем на основании булевой функции получаем логическую схему (см. рис. 3.7). Заметим, что такая булева функция обусловлена сетью логических элементов И-ИЛИ, ближайшим к выходу является элемент ИЛИ. Такая специальная схема называется *формой суммы произведений* или *реализованной формой* булевой функции. Таблица 3.3 показывает состояние составляющих реализуемой булевой функции в колонке выхода таблицы истинности, где на выходе появляется 1.

Таблица 3.3. *Таблица истинности, соответствующая эквивалентному выражению* $D \cdot \bar{C} \cdot \bar{B} \cdot \bar{A} + D \cdot C \cdot \bar{B} \cdot A = Y$

Входы				Выход	Входы				Выход
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	0

Упражнения

3.10. Записать выражение булевой функции таблицы истинности (табл. 3.4).

3.11. Начертить логическую схему системы, соответствующую таблице истинности (табл. 3.4).

3.12. Обратимся к рис. 3.8, *б*. Входы *A*, *B* и *C* _____ (H-L-) активные, тогда как вход *D* _____ (H-, L-) активный.

Таблица 3.4. *Таблица истинности*

Входы				Выход	Входы				Выход
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	1	1	1	1	1	0

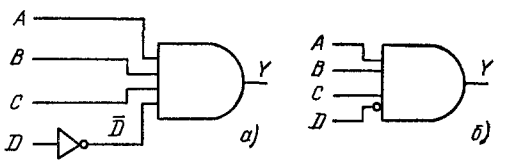


Рис. 3 8. Логическая схема (*а*) и ее упрощенный вариант (*б*)

3.13. Записать выражение булевой функции таблицы истинности (табл. 3.5).

Таблица 3.5. *Таблица истинности*

Входы				Выход	Входы				Выход
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

3.14. Начертить логическую схему системы, соответствующую таблице истинности (табл. 3.5).

Решения

3.10. Единицу на выходе дает единственное сочетание входов. Выражение булевой функции получается в виде $\overline{D}\cdot C\cdot B\cdot A=Y$. **3.11.** Выражением реализуемой функции для табл. 3.4 будет $D\cdot C\cdot B\cdot A=Y$. Его реализация осуществляется схемой, представленной на рис. 3.8, *а*. На рис. 3.8, *б* приведено другое возможное решение. **3.12.** На рис. 3.8, *б* входы *A*, *B* и *C* являются Н-активными, а вход *D*—*L*-активный. Действительно, элемент И с четырьмя входами может быть активизирован (1 на выходе) только тогда, когда входы *A*, *B* и *C* будут Н-активными, а вход *D*—L-активный. **3.13.** Н-уровень на выходе получается при двух сочетаниях входов согласно таблице истинности (табл. 3.5). Для этих двух случаев и получено выражение булевой функции $D\cdot\overline{C}\cdot\overline{B}\cdot A+D\cdot C\cdot B\cdot A=Y$. **3.14.** Выражением булевой функции для этой таблицы истинности будет $D\cdot\overline{C}\cdot\overline{B}\cdot A+D\cdot C\cdot B\cdot A=Y$, на основании которого строим логическую схему рис. 3.9.

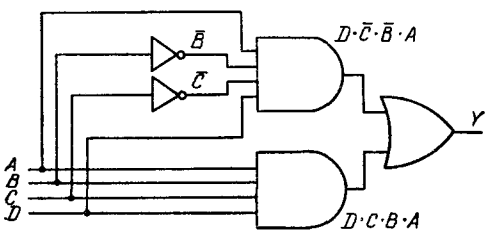


Рис. 3.9. Логическая схема, соответствующая табл. 3.5

3.3. ТРИГГЕРЫ И ЗАЩЕЛКИ

Логические цепи могут быть разделены на две большие группы. Первая—*цепи комбинационной логики,* составленные из логических элементов, вторая — *последовательные логические цепи,* состоящие из элементов, называемых триггерами. Триггеры объединяют в системы с целью образования последовательных логических цепей, предназначенных для размещения данных, обеспечения нужной временной задержки, вычислений и формирования требуемых последовательностей сигналов. Триггеры обладают важной способностью запоминания. Триггер запомнит свои входные сигналы даже тогда, когда эти сигналы бу-

дут сняты. Логический элемент, напротив, не сможет запомнить свое состояние на выходе, если будут сняты входные сигналы.

На рис. 3.10 приведена очень широко используемая схема *D-триггера* (называемого также *триггером данных*). Отметим здесь два входа, обозначенных D (для данных) и CK (для сигналов синхронизации или тактовый вход).

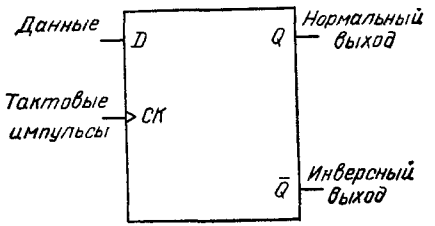


Рис. 3.10. *D*-триггер

Триггер имеет обычно два взаимоинверсных выхода, обозначенных Q и $\bar{Q}$ (НЕ-Q). Выход Q используется более часто и называется *нормальным*. Выход Q называется *дополнительным* или *инверсным* выходом триггера. Графическое обозначение «>» на входе CK логической схемы D-триггера указывает, что этот триггер передает данные со входа на выход при положительном фронте (↑) тактовых импульсов.

Операционные состояния D-триггера приведены в левой колонке таблицы истинности[1] (табл. 3.6). *Установить* (или *активизировать*) триггер означает, что на нормальном выходе Q устанавливается 1. Первая строка таблицы истинности показывает, что подачей 1 на вход D-триггера при положительном тактовом импульсе CK на выходе Q устанавливается 1. Вторая строка соответствует сбросу (установке в нуль) триггера. *Сбросить* триггер означает вернуть выход Q в состояние 0. *Установить состояние ожидания* это значит сохранить на выходе данные предыдущего состояния. Когда триггер находится в состоянии ожидания, изменения логических состояний данных на входе не влияют на состояние выходов.

Таблица 3.6. *Таблица истинности статических состояний D-триггера*

Операционные состояния	Входы		Выходы	
	D	CK	Q	$\bar{Q}$
Активизация	1	↑	1	0
Сброс	0	↑	0	1
Ожидание	*	Отсутствие	Предыдущее состояние	

Примечание. 0 — LOW; 1 — HIGH; * — не имеет значения; ↑ — переход от LOW к HIGH тактового импульса.

Состояние ожидания характеризует способность триггера запоминать сигналы. Отметим, что операции установки и сброса рассматриваются по отношению к выходу Q.

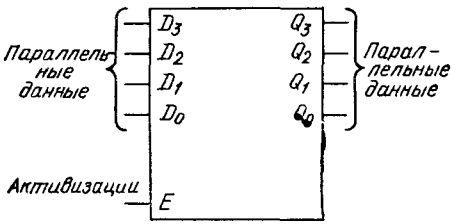


Рис. 3.11. Логическая схема прозрачной 4-разрядной защелки

Рассмотрим логическую схему, приведенную на рис. 3.11, на которой изображена 4-разрядная *прозрачная защелка*. Каждый триггер-защелка, входящий в это устройство, является устройством либо памяти, либо размещения данных. Для лучшего понимания состава регистра-защелки необходимо рассматривать его состоящим из четырех D-триггеров, тактовые входы которых объединены одним входом разрешения (или активизации). Согласно соответствующей таблице истинности (табл. 3.7) при поступлении 1 на вход E (Н-сигнал) данные со входов D_0—D_3 будут переданы на выходы Q_0—Q_3 соответственно. Четырехразрядное слово поступает на входы в параллельной (но не в последовательной) форме и передается на выходы в той же

Таблица 3.7. *Таблица истинности защелки*

Функциональное состояние	Входы		Выход
	D	E	Q
Признание данных	0	1	0
	1	1	1
Захват данных	*	0	Предыдущее состояние

Примечание. 0 — LOW; 1 — HIGH; * — не имеет значения.

[1] Такие таблицы называют еще таблицами переходов триггера. — *Прим. ред.*

форме. Такая передача называется параллельным вводом/выводом. Защелка размещения (или хранения) данных является одной из разновидностей среди разных типов триггеров.

JK-триггер является наиболее употребительным триггером в последовательных логических цепях. На рис. 3.12 приведена логическая схема типового *JK*-триггера. Она имеет два входа данных *J* и *K* и один тактовый вход *CK*. *JK*-триггер имеет два традиционных выхода — *Q* (нормальный) и $\bar{Q}$ (инверсный).

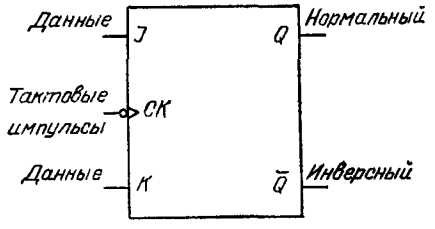


Рис. 3.12. *JK*-триггер

JK-триггер имеет четыре операционных режима (см. табл. 3.8). *Режим триггера* означает, что при каждом тактовом импульсе выходы перейдут в логическое состояние, обратное тому, которое он имел до этого импульса. В этом случае выход *Q* *JK*-триггера будет проходить состояния HIGH-LOW-HIGH-LOW и т. д. по мере следования тактовых импульсов.

Таблица 3.8. *Таблица истинности состояний JK-триггера*

Функциональные режимы	Входы			Выходы	
	J	*K*	*CK*	*Q*	$\bar{Q}$
Триггер	1	1	↓	Противоположно предыдущему	
Активизация	1	0	↓	1	0
Сброс	0	1	↓	0	1
Ожидание	0	0	↓	Нет изменений	

Примечание. 0 — LOW; 1 — HIGH; ↓ — переход от HIGH к LOW тактового импульса.

Таблица истинности (см. табл. 3.8) показывает, что *JK*-триггер будет переключаться, когда два входа *J* и *K* находятся в H-состоянии. В этот момент тактовый импульс поступает на вход *CK*. Эффективные переключения происходят, когда тактовый импульс переходит из H- в L-состояние, как показывает стрелка в таблице истинности.

Находится *JK*-триггер в *режиме инициализации* (активизации), когда два входа данных $J=1$ и $K=0$. Мы видим, что согласно второй строке таблицы истинности переход от H- к L-состоянию тактовых импульсов переводит выход *Q* в состояние 1. *Режим сброса* или дезактивации (установка *Q* в 0) представлен третьей строкой таблицы истинности. Далее в таблице истинности приведен *режим ожидания* (отсутствие каких-либо действий) *JK*-триггера. Когда оба входа данных (*J* и *K*) являются LOW, тактовый импульс на входе *CK* не оказывает никакого влияния на выход.

Запуск триггеров является важным этапом их функционирования. По способу запуска триггеры могут быть классифицированы на устройства, активизированные *фронтом импульсов* или *уровнем импульсов*. Логические схемы, приведенные на рис. 3.10 и 3.12, показывают, что соответствующие триггеры запускаются фронтом импульсов [наличие на схеме знака «>» (больше чем) на входе *CK* тактовых импульсов]. *D*-триггер запускается при переходе тактовых импульсов от L- к H-уровню. Это показано в таблице истинности и на логической схеме (нет кружка инверсии на входе *CK*, что указывает на необходимость 1 для активизации тактовых импульсов). *D*-триггер называется также триггером, запускаемым *положительным фронтом* тактовых импульсов, так как запуск осуществляется положительной частью тактовых импульсов.

Запускается *JK*-триггер *отрицательным фронтом* тактовых импульсов, что показано в таблице истинности (табл. 3.8) и на логической схеме (рис. 3.12). Кружок инверсии на входе *CK* *JK*-триггера указывает на необходимость L-сигнала для активизации входа тактовых импульсов. В случае, когда этот триггер запускается фронтом импульсов, для его запуска нужен переход тактовых импульсов от H- к L-уровню.

Прозрачная 4-разрядная защелка, приведенная на рис. 3.11, является устройством, запускаемым уровнем тактовых импульсов. Это означает, что когда на вход активизации *E* (подобный входам *CK* триггеров) поступает H-сигнал, все двоичные данные на входах (D_3—D_0) тотчас появляются на выходах (Q_3—Q_0). Такая защелка называется прозрачной.

Упражнения

3.15. Последовательные устройства содержат элементы, называемые _____.

3.16. Элементом памяти, который позволяет размещать данные, является обычно _____.

3.17. *D*-триггер называется также триггером _____.

3.18. Нормальным выходом триггера является выход _____ (Q, $\bar{Q}$).

3.19. Перечислить три функциональных состояния *D*-триггера на рис. 3.10.

3.20. Назвать функциональные состояния *D*-триггера, приведенного на рис. 3.13, соответствующие каждому тактовому импульсу.

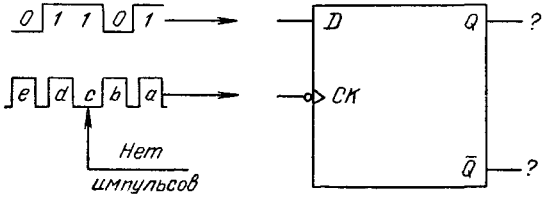


Рис. 3.13. К упражнениям 3.20 и 3.21

3.21. Перечислить двоичные значения на выходе Q *D*-триггера на рис. 3.13 после каждого тактового импульса.

3.22. Защелка — это устройство _____ (размещения, расчета) данных.

3.23. Обратиться к рис. 3.11. Входом активизации такой 4-разрядной защелки является логический сигнал _____ (0, 1).

3.24. Перечислить двоичные 4-разрядные числа на выходах защелки на рис. 3.14.

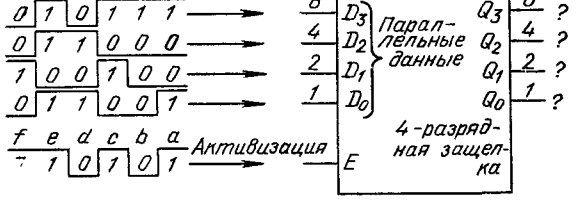


Рис. 3.14. К упражненню 3.24

3.25. Перечислить четыре функциональных режима *JK*-триггера на рис. 3.12.

3.26. Перечислить режимы *JK*-триггера (рис. 3.15), соответствующие каждому тактовому импульсу.

3.27. Перечислить двоичные состояния на нормальном выходе Q *JK*-триггера на рис. 3.15 после каждого тактового импульса.

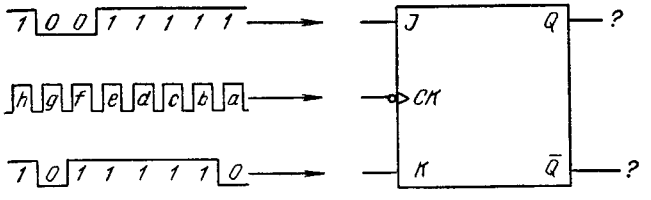


Рис. 3.15. К упражнениям 3.26 и 3.27

3.28. См. рис. 3.10. *D*-триггер является устройством, запускаемым _____ (уровнем, фронтом) импульсов.

3.29. См. рис. 3.12. Этот *JK*-триггер является устройством, запускаемым фронтом _____ (положительных, отрицательных) импульсов.

3.30. См. рис. 3.11. Такая 4-разрядная защелка является элементом, включаемым _____ импульсов.

Решения

3.15. Триггерами. **3.16.** Триггер. **3.17.** Данных. Иногда его называют триггером временной задержки. **3.18.** Q. Выход $\bar{Q}$ — инверсный. **3.19.** Установка, сброс, ожидание. **3.20.** См. таблицу истинности (табл. 3.6): импульс a — установка; импульс b — сброс; время c (нет импульсов) — ожидание; импульс d — установка; импульс e — сброс. **3.21.** См. табл. 3.6: импульс a — 1; импульс b — 0; время c (нет импульсов) — 0; импульс d — 1; импульс e — 0. **3.22.** Размещения. **3.23.** См. табл. 3.7. Логической 1 (Н-сигналом). **3.24.** См. табл. 3.7: импульс a — 1001; импульс b — 1001 (данные, соответствующие импульсу a, захвачены до того, как вход перешел от 1 к 0); импульс c — 1010; импульс d — 1010 (данные, соответствующие импульсу c захвачены до того, как изменилось состояние входов D_3, D_2, D_1, D_0); импульс e — 1101; импульс f — 0010. **3.25.** См. табл. 3.8. Режимы: триггера, установки, сброса, ожидания **3.26.** См. табл. 3.8: импульс a — установка; импульс b — триггер; импульс c — триггер; импульс d — триггер; импульс e — триггер; импульс f — сброс; импульс g — ожиданне; импульс h — ожидание. **3.27.** См. табл. 3.8: после импульса a — установка, 1; после импульса b — триггер, 0; после импульса c — триггер, 1; после импульса d — триггер, 0; после импульса e — триггер, 1; после импульса f — сброс, 0; после импульса g — ожи-

дание, 0; после импульса h — триггер, 1. **3.28.** Положительным фронтом импульса (при переходе от L к H). **3.29.** Отрицательных (при переходе от H к L). **3.30.** Уровнем.

3.4. ШИФРАТОРЫ, ДЕШИФРАТОРЫ И СЕМИСЕГМЕНТНЫЕ ИНДИКАТОРЫ

Рассмотрим функциональную схему калькулятора (рис. 3.16, *а*). В этой цифровой системе поступающая с *клавиатуры* десятичная информация должна быть переведена в двоично-десятичный код. Эта операция (*кодирование*) выполняется цифровым устройством, называемым *шифратором*. Выходящая из ЦП двоично-десятичная информация переводится *дешифратором* в специальный код *семисегментного индикатора*.

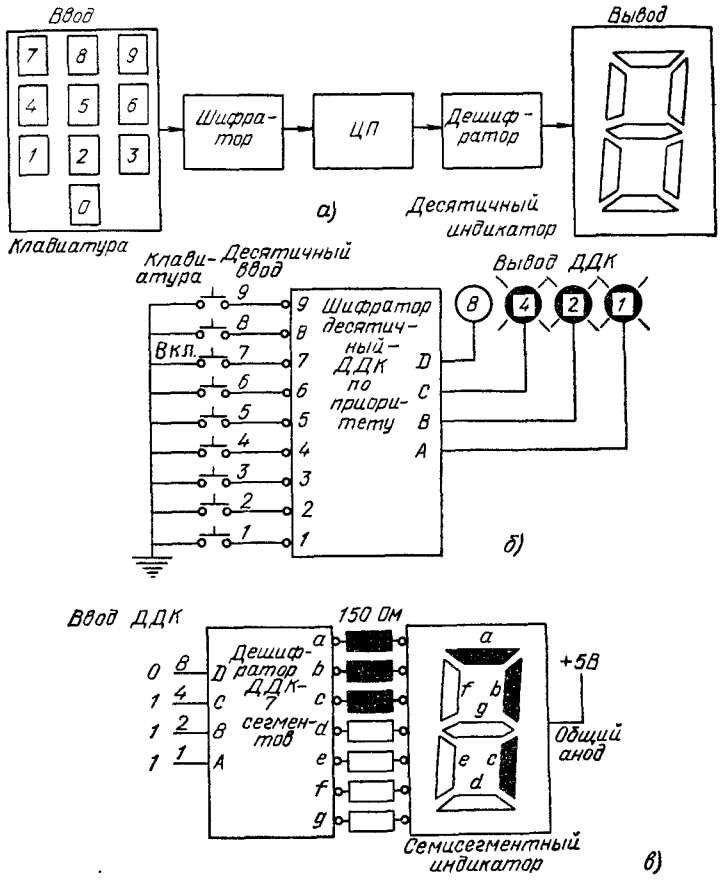


Рис. 3.16. Схемы:
а — функциональная калькулятора; *б* — логическая системы клавиатура — шифратор; *в* — логическая система дешифратор — индикатор

На рис. 3.16, *б* приведена логическая схема *шифратора десятичного ДДК с приоритетом*. У этого шифратора девять входов, активизируемых L-сигналом, и четыре выхода, соединенных со световым индикатором. Соединения с клавиатурой показаны слева, каждый занумерованный ключ подсоединен на соответствующий вход шифратора. В приведенном на рис. 3.16, *б* примере показано, что была нажата клавиша 7, в результате чего заземлился вход 7 шифратора. Это повлечет за собой вывод двоично-десятичного числа 0111, что мы можем видеть на индикаторе на рис. 3.16, *б*. Большинство шифраторов обладает свойством приоритета. Это означает, что если запрос поступает от двух клавиш одновременно, будут активизированы выходы той из них, которая соответствует более высокой десятичной величине. Совершенно очевидно, что соединения шифратора следовало бы дополнить полной схемой системы питания. В принципе, такой шифратор может выполняться в форме кристалла интегральной схемы, но он мог бы быть составлен и из отдельных элементов (для этого их понадобилось бы около 20).

Рассмотрим систему дешифратор-индикатор, приведенную на рис. 3.16, *в*. Этот дешифратор двоично-десятичного кода с выводом на семь сегментов переводит $0111_{ДДК}$ в его десятичный эквивалент 7, выводимый на семисегментный индикатор на фотодиодах. Используемый в этой системе индикатор имеет общий анод — все аноды семи диодов (формирующих семь сегментов) подсоединены к выводу +5 В общего источника питания. Индикатор на рис. 3.16, *в* имеет L-активные входы, что может быть установлено по наличию кружков инверсии на входах *a*—*g*. Таким образом, для активизации сегмента нужен один L-сигнал. Отметим также, что дешифратор имеет L-активные совместимые выходы.

Семь сопротивлений между дешифратором и индикатором являются элементами защиты, предназначенными для ограничения тока. В примере, приведенном на рис. 3.16, *в*, активизированы только выходы *a*, *b* и *c* дешифратора (логический 0). Индикатор на этом рисунке имеет только одно соединение с источником +5 В, тогда как шифратор имеет

их два, они не приведены на схеме, чтобы ее не усложнять. В действительности шифраторы двоично-десятичный код-индикатор содержат также входы для полного стирания (погасание всех сегментов) и проверки диодов (зажигание всех сегментов).

Упражнения

3.31. Шифратор на рис. 3.17 переводит десятичные сигналы с клавишного устройства в _____ (ASCII, двоично-десятичный код).

3.32. Шифратор на рис. 3.17 имеет _____ (H-, L-активные) входы.

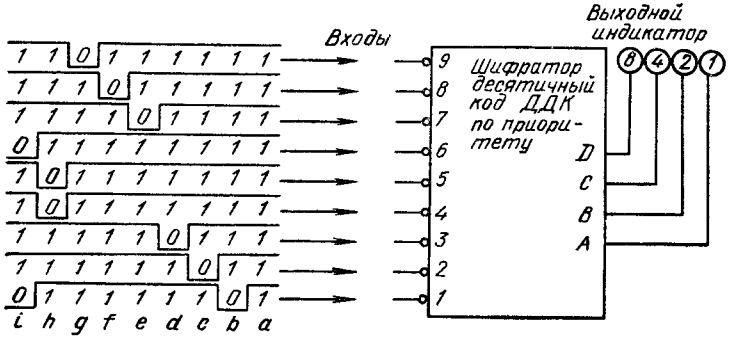


Рис. 3.17. К упражнениям 3.31—3.33

3.33. Перечислить 4-разрядные индикации в двоично-десятичном коде на выходе для каждого из входных импульсов, изображенных на рис. 3.17.

3.34. Дешифратор на рис. 3.18 позволяет перейти от двоично-десятичного кода к _____ (десятичному, шестнадцатеричному) коду.

3.35. Входные сигналы дешифратора на рис. 3.18 являются _____ (H-, L-активными), а их выходные сигналы _____ (H-, L-активными).

3.36. Перечислить десятичные выходные сигналы (показания индикатора) для каждого входного импульса на рис. 3.18.

3.37. Перечислить активные сегменты для каждого из импульсов на рис. 3.18.

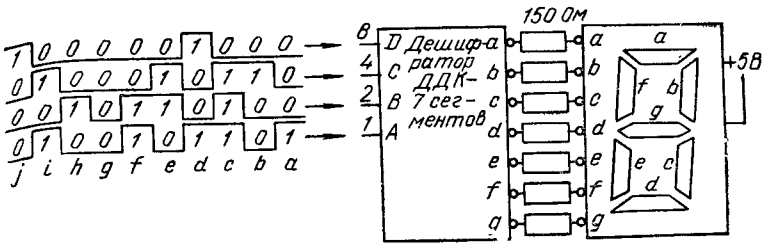


Рис. 3 18. К упражнениям 3.34—3.37

Решения

3.31. См. рис. 3.17. Этот шифратор переводит сигналы с клавишного устройства в двоично-десятичный код. **3.32.** Обратиться к рис. 3.17. Входы этого устройства активны (см. кружки инверсии). **3.33.** Импульсы: a — 0000 (нет активации входов); b — 0001 (вход *1* активизирован L-сигналом); c — 0010 (вход *2* активизирован L-сигналом); d — 0011; e — 0111; f — 1000; g — 1001; h — 0101 (активизированы входы *4* и *5*, но приоритет имеет большее число). **3.34.** От двоично-десятичного кода к десятичному с отражением на семисегментном индикаторе. **3.35.** Входы H-активные; выходы L-активные. **3.36.** Импульсы: a — 1; b — 4; c — 7; d — 9; e — 6; f — 3; g — 0; h — 2; i — 5; f — 8; **3.37.** Импульсы: a — b, c b — b, c, f, g; c — a, b, c; d — a, b, c, f, g; e — c, d, e, f, g (импульс a может быть активным на некоторых индикаторах для формирования цифры 6); g — a, b, c, d, e, f; h — a, b, d, e, g; i — a, c, d, f, g; j — a, b, c, d, e, f, g.

3.5. МУЛЬТИПЛЕКСОРЫ И ДЕМУЛЬТИПЛЕКСОРЫ

Мультиплексоры называют также *селекторами данных*. Действие механического коммутатора, показанного на рис. 3.19, *а*, идентично действию селектора данных электронного мультиплексора. Вращающийся коммутатор имеет восемь входов и один единственный выход. Вращением механической ручки данные с одного любого входа (*0*—*7*) могут быть переданы на выход и будут полностью идентичны входным.

На рис. 3.19, *б* приведена логическая схема мультиплексора-селектора данных на восемь входов. Обозначим их как I_0—I_7 и единственный выход как Y. Отметим также, что имеется один L-активный вход активизации $\overline{E}$. Эти входы и выход представляют собой устройство, полностью аналогичное механическому коммутатору, и мы можем

рассматривать вход активизации $\overline{E}$ как общий прерыватель. Внизу на логической схеме мы обозначим выводы управления селекцией данных как S_2, S_1, S_0. Двоичные данные, поданные на эти входы, определяют, какой из входов данных соединен с выходом.

Пример, приведенный на рис. 3.19, *б*, показывает, что цепь включена или активизирована L-сигналом на входе

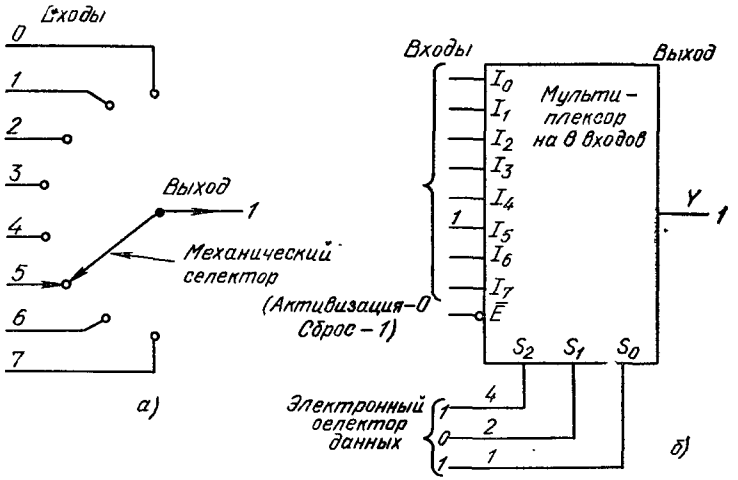


Рис. 3.19. Механический коммутатор — аналог мультиплексора/селектора данных (*а*) и мультиплексор на восемь выходов (*б*)

$\overline{E}$. На выводы селекции данных S_2, S_1, S_0 подано 101_2. Этот сигнал избирает вход *5*, что обеспечивает передачу логической 1 с I_5 на вход *Y*. Мультиплексор с восемью входами может быть использован для преобразования одного входного 8-разрядного слова в последовательный эшелон импульсов, переключая 3-разрядный счетчик на вводах селекции данных, что осуществляет последовательную селекцию входов (I_0, I_1, I_2 и т. д.). Мультиплексор-селектор данных используют также для решения сложных логических задач.

Демультиплексор, представленный на рис. 3.20, предназначен для выполнения действий, обратных действиям мультиплексора. Демультиплексор 1×8 обладает одним только входом данных *D* и восемью выходами (*0*—*7*). Схема имеет один L-вход активизации и три входа селекции данных.

В примере на рис. 3.20 имеется логическая 1 на входе *D*. Цепь активизируется одним L-сигналом и входы селекции данных избирают выход *5* (101_2). При этих условиях входные данные появляются на выходе *5*. Соединением входов селекции данных с 3-разрядным счетчиком последовательно входящие данные могут быть распределены на восемь выходов один за другим. Мультиплексоры и демультиплексоры могут быть использованы совместно для преобразования непрерывной информации в форму последовательностей. Мультиплексор будет представлять собой эмиттер, демультиплексор — приемник, который передает данные в их начальной форме.

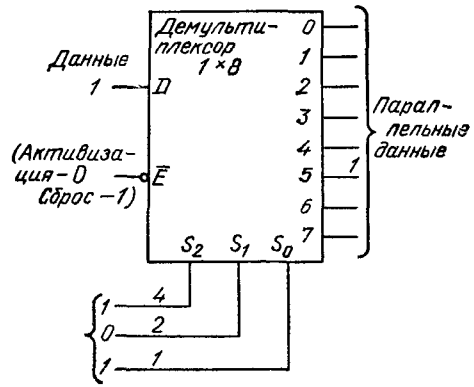


Рис. 3.20. Схема демультиплексора 1×8

Упражнения

3.38. См. рис. 3.21. Перечислить значения сигналов на выходе мультиплексора с восемью входами для каждой группы импульсов.

3.39. Обратиться к рис. 3.21. Мультиплексор осуществляет преобразование входных параллельных данных в _____ выходные.

3.40. См. рис. 3.22. Дать для каждой группы импульсов значения выходных сигналов (*0*—*7*).

Решения

3.38. Значения выходных сигналов: *а* — 0 ($S_2S_1S_0$=000; I_0=0); *b* — 1 ($S_2S_1S_0$=001; I_1=1); *c* — 1 ($S_2S_1S_0$=010; I_2=1); *d* — 0 ($S_2S_1S_0$=011; I_3=0); *e* — 1 ($S_2S_1S_0$=100; I_4=1); *f* — 0 ($S_2S_1S_0$=101; I_5=0); *g* — 0 ($S_2S_1S_0$=110; I_6=0); *h* — 1 ($S_2S_1S_0$=111; I_7=1); *i* — 1 (мультиплексор не активизирован, пока входы выбора данных содержат 000). **3.39.** Последовательные. **3.40.** Значения на выходах: *а* — активизирован выход *0* ($S_2S_1S_0$=000); *b* — активизирован выход *6* ($S_2S_1S_0$=110; *c* — активизирован выход *1* ($S_2S_1S_0$=001); *d* — активизированных выходов нет ($\overline{E}$=1 — сброс); *f* — актнвизирован выход *7* ($S_2S_1S_0$=111).

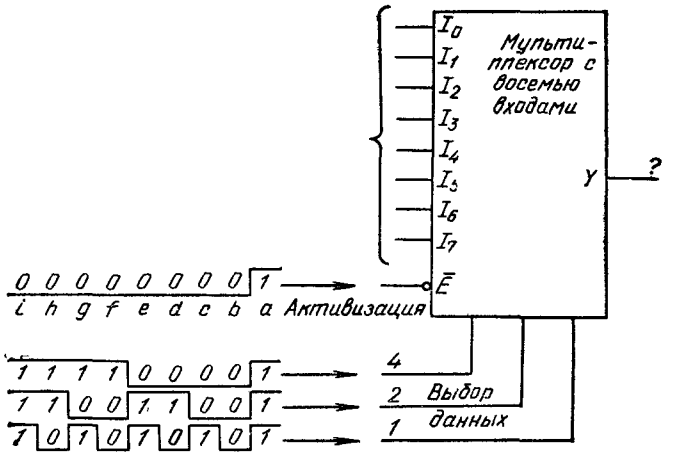


Рис. 3.21. К упражненням 3.38 и 3.39

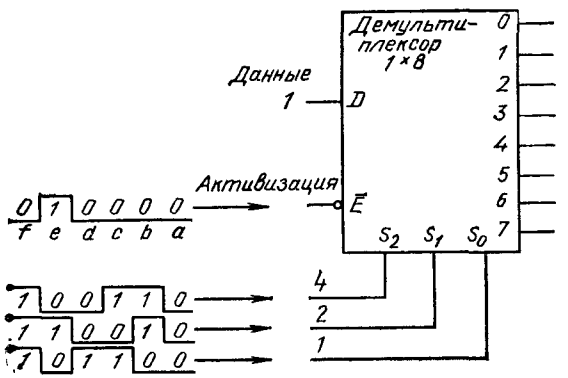


Рис. 3.22. К упражненню 3.40

3.6. ТРИСТАБИЛЬНЫЕ ЭЛЕМЕНТЫ

Интегральные элементы семейства ТТЛ (транзисторно-транзисторной логики) используются очень широко. Выход обычного устройства ТТЛ может быть либо логической 1 либо 0. Поэтому невозможно подсоединять выходы стандартных элементов ТТЛ на общую шину микро-ЭВМ (шину данных, например). Были введены специальные элементы ТТЛ, выходы которых могут быть объединены на общей шине. Эти элементы называются *тристабильными* (имеющими три состояния). Им присущи состояние на выходе логического 0, логической 1 и особое состояние высокого сопротивления Z. Когда тристабильный элемент находится в состоянии высокого сопротивления Z, его выход отсоединяется от шины.

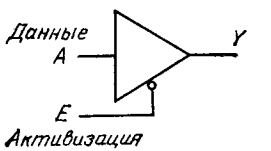


Рис. 3.23. Шинный тристабильный буфер

Логическая схема элемента шинного буфера представлена на рис. 3.23. Этот буфер обладает тристабильными выходами, и его действие описывается таблицей истинности (табл. 3.9). У шинного буфера имеется один L-активный вход активизации, а проходящие через элемент данные не инвертируются. Когда элемент шинного буфера сброшен, его выход находится в состоянии высокого сопротивления (плавает) и не оказывает на шины никакого влияния. В этом состоянии выход элемента отсоединяется от шины, т. е. не выдает на шину и не принимает от нее никакой информации.

Таблица 3.9. *Таблица истинности тристабильного шинного буфера*

Фуикциональное состояние	Входы		Выход
	$\bar{E}$	A	Y
Активизирован	0	0	0
	0	1	1
Сброшен	1	0	Высокое сопротивленне
	1	1	

Упражнения

3.41. Выход элемента ТТЛ может быть либо в состоянии логической 1, либо в состоянии _____ (высокого, низкого) сопротивления.

3.42. Элемент шинного буфера на рис. 3.24 имеет один вход активизации, активизируемый _____ (H-, L-сигналом).

3.43. См. рис. 3.24. Перечислите выходы элементов шинного буфера, соответствующие каждой из семи групп импульсов на входе.

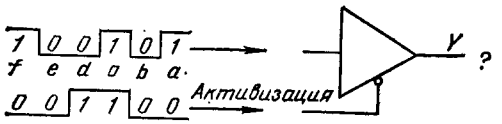


Рис. 3.24. К упражнениям 3.42—3.43

Решения

3.41. Высокого. **3.42.** L-сигналом. **3.43.** Группам импульсов соответствуют выходы: a — 1 (элемент активизирован); b — 0 (элемент активизирован); c и d — состояние высокого сопротивления (нет активизации); e — 0 (элемент активизирован); f — 1 (элемент активизирован).

3.7. ПОЛУПРОВОДНИКОВАЯ ПАМЯТЬ

Триггер (или защелка) является основным элементом памяти, используемым в многочисленных устройствах вычислительной техники. Память в основном делится на две группы: постоянную и оперативную. Сами названия показывают разницу между этими двумя типами памяти. В постоянной памяти нули и единицы располагаются в соответствии с определенной программой разработчика. Оперативная память может быть просто запрограммирована, уничтожена и перепрограммирована пользователем.

Программирование означает процесс *записи* в память. Копирование данных из памяти без разрушения ее содержимого представляет собой операцию *считывания* из памяти. Устройства постоянной памяти называются ПЗУ *(постоянные запоминающие устройства)*, а оперативной — ОЗУ *(оперативные запоминающие устройства)*. Как правило, программы ОЗУ сменяемы; это означает, что они теряются, если прерывается питание ИС даже на короткое время. Типичными являются системы, содержащие оба типа памяти — ПЗУ и ОЗУ. Наиболее часто оба типа памяти являются содержимым различных ИС.

Существуют четыре разновидности ПЗУ. Стандартные ПЗУ программируются разработчиком. Программируемые ПЗУ (ППЗУ) могут быть запрограммированы постоянно пользователем или заготовителем на специальном оборудовании, это программирование необратимо.

Стираемые ППЗУ (СППЗУ) могут быть программируемы и стираемы пользователем. Данные, расположенные в СППЗУ, могут быть стерты ультрафиолетовым облучением большой интенсивности. Стираемые электрически ПЗУ — это другой тип стираемых ППЗУ (СЭПЗУ), операция стирания осуществляется на специальном оборудовании в переменном электрическом поле без применения ультрафиолетового облучения. Рассмотренные ПЗУ, ППЗУ, СППЗУ и СЭПЗУ являются устройствами постоянной памяти и не теряют своего содержимого при прерывании питания ИС[1].

Группа ОЗУ делится на две подгруппы[2]. Если ОЗУ содержит в качестве элементарных ячеек памяти цепи типа триггеров, оно называется *статическим* ОЗУ (или статической оперативной памятью); *динамические* ОЗУ строятся более просто и основаны на свойствах электрической емкости, но они должны подтверждать содержимое ячеек примерно несколько сотен раз в секунду. Статические ОЗУ не нуждаются ни в каком подтверждении, и их двоичное содержимое сохраняется до тех пор, пока сохраняется питание ИС.

Оперативное запоминающее устройство микро-ЭВМ служит для размещения на определенное время программ и данных пользователя. Постоянное запоминающее устройство используется для размещения команд на машинном языке, которые представляют собой программу-монитор (сокращенно — монитор). Монитор содержит в себе неизменяющиеся подпрограммы инициализации, ввода/вывода и арифметических алгоритмов.

Упражнения

3.44. В вычислительной технике сокращение ПЗУ означает _____.

3.45. В практике оперативная память обозначается сокращением _____ (ОЗУ, ПЗУ).

3.46. В вычислительной технике сокращение ОЗУ означает _____.

3.47. Сокращение ППЗУ означает _____.

3.48. Постоянное запоминающее устройство и _____ (ОЗУ, ППЗУ) являются устройствами размещения данных, сохраняемых постоянно и необратимых.

[1] Такая память называется еще энергонезависимой. — *Прим. ред.*

[2] Оперативная память является энергозависимой, т. е. она теряется при прерываниях питания ИС. — *Прим. ред.*

3.49. Оперативное запоминающее устройство предназначено для хранения (стираемых, нестираемых) данных.

3.50. Сокращение СППЗУ означает _____ .

3.51. Сокращение СЭПЗУ означает _____ .

3.52. Электрически стираемые ПЗУ являются устройствами размещения _____ (переменных, постоянных) данных.

3.53. Устройства оперативной памяти делятся на статические и _____ ОЗУ.

3.54. Элементарные ячейки памяти _____ (статических, динамических) ОЗУ практически состоят из стандартных триггеров.

3.55. _____ (статические, динамические) ОЗУ нуждаются в регенерации памяти несколько сотен раз в секунду.

3.56. Программируемые постоянные запоминающие устройства в микро-ЭВМ могут предназначаться для размещения _____ (программы монитора, временной программы пользователя).

Решения

3.44. Постоянное запоминающее устройство. **3.45.** ОЗУ. **3.46.** Оперативное запомнинающее устройство. **3.47.** Программируемое постоянное запоминающее устройство. **3.48.** ППЗУ. **3.49.** Стираемых. **3.50.** ПЗУ, программируемое и стираемое ультрафиолетовым облучением. **3.51.** ПЗУ, стираемое электрически. **3.52.** Постоянных. **3.53.** Динамические. **3.54.** Статических. **3.55.** Динамические. **3.56.** Программы монитора.

3.8. ИСПОЛЬЗОВАНИЕ ОПЕРАТИВНОЙ И ПОСТОЯННОЙ ПАМЯТИ

Организация ПЗУ или ОЗУ может быть наглядно представлена своеобразной таблицей истинности. Таблица 3.10 представляет собой один из возможных вариантов организации ячеек памяти. Здесь речь идет об ОЗУ 16×4 бит, о чем мы можем сделать вывод, имея 16 4-разрядных групп (эти группы составляют слова памяти). В табл. 3.10 большинство ячеек памяти пусты, за исключением слова 12, которое содержит данные 0101. В действительности пустые ячейки памяти могут содержать неизвестные сочетания нулей и единиц.

На рис. 3.25 представлена логическая схема ОЗУ 16×4 бит. В этом случае ОЗУ с объемом памяти 64 бит выполняет операцию записи поступающих данных 0101 в

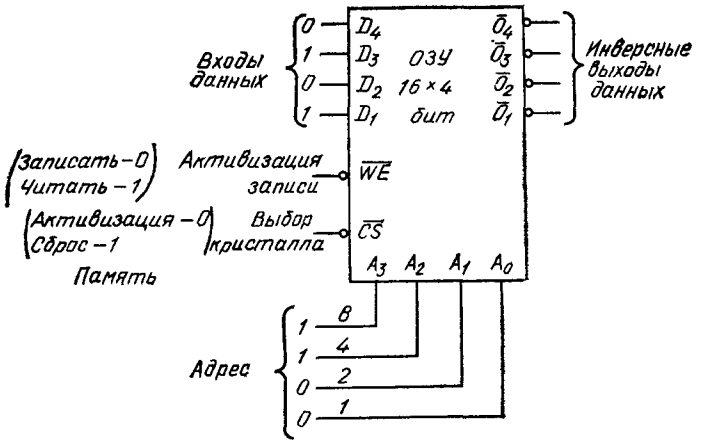


Рис. 3.25. Логическая схема ОЗУ 16×4 бит

Таблица 3.10. *Структура ОЗУ 16×4 бит*

Адрес	Бит *D*	Бит *C*	Бит *B*	Бит *A*
Слово 0				
Слово 1				
. . .	. . .	. . .	. . .	. . .
Слово 11				
Слово 12	0	1	0	1
Слово 13				
Слово 14				
Слово 15				

ячейку памяти 12_{10} (1100_2). На входы данных поступает для записи в память слово 0101_2, а положение слова в ячейке с адресом 12_{10} определено величиной 1100_2 (12_{10}), поступающей на адресные входы.

Затем две команды управления $\overline{CS}$ и $\overline{WE}$ переводят ОЗУ в состояние записи. Заметим, что входы *активизации записи* $\overline{WE}$ и *выбора кристалла* $\overline{CS}$ должны быть в L-состоянии для того, чтобы выполнялась операция записи. Данные 0101_2 помещаются тогда на место слова памяти по адресу 12_{10}, как показано на рис. 3.25. Некоторые конструкторы вызывают активизацию памяти по входу управления $\overline{CS}$. В табл. 3.11 приведена таблица истинности такого ОЗУ с объемом памяти 64 бит.

Таблица 3.11. *Таблица истинности ОЗУ 16×4 бит*

Функциональное состояние	Входы управления		Выходы
	$\overline{CS}$	$\overline{WE}$	$\overline{O}$
Запись	0	0	Состояние логической 1
Считывание	0	1	Инверсия размещенных данных
Ожидание	1	*	Состояние логической 1

Примечание: * — не имеет значения.

Оперативное запоминающее устройство находится в состоянии записи, когда две линии управления $\overline{CS}$ и $\overline{WE}$ находятся в L-состоянии. В ходе операции записи 4 бит данных (D_4, D_3, D_2, D_1) загружаются в ячейки памяти, на которую указывает адрес, и в течение этого времени выходы ($\overline{O}_4$, $\overline{O}_3$, $\overline{O}_2$, $\overline{O}_1$) держатся в H-состоянии. Когда входы команд $\overline{CS}=0$ и $\overline{WE}=1$, ОЗУ находится в состоянии считывания из него данных.

В ходе операции инверсное значение слова данных, на которое указывают адресные входы, появляется на выходах. Данные, расположенные в ОЗУ, не разрушаются операцией считывания. В состоянии ожидания все выходы переходят к H-уровню и никакие данные не проходят через входы D.

Упражнения

3.57. Оперативное запоминающее устройство с памятью объемом 64 бит, приведенное на рис. 3.26, находится в состоянии _____ (ожидания, записи) во время прохождения импульсов *a*. Следовательно, все выходы находятся в _____ (H-, L-состоянии).

3.58. См. рис. 3.26. Во время прохождения импульсов *b*, *c*, *d* и *e* ОЗУ находятся в состоянии _____ (записи, чтения).

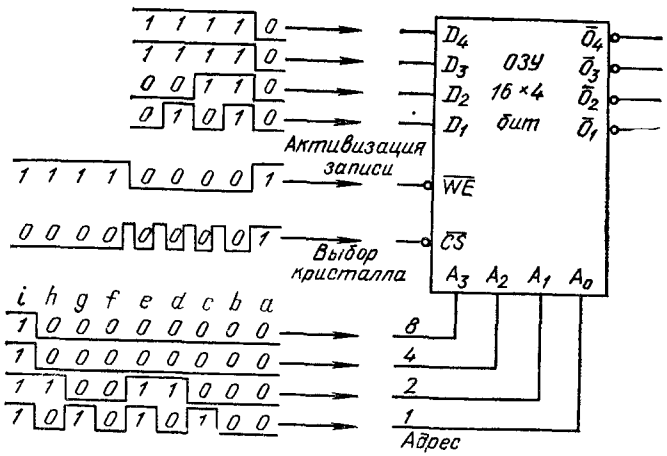


Рис. 3.26. К упражнениям 3.57—3.62

3.59. Обратиться к рис. 3.26. В течение импульсов *f*, *g*, *h* и *i* ОЗУ находятся в состоянии _____ (записи, чтения).

3.60. Обратиться к рис. 3.26. Перечислить адреса размещения и данные, помещаемые в ОЗУ, в ходе операции записи в течение импульсов *b c*, *d*, *e*.

3.61. См. рис. 3.26. Перечислить адреса и данные, которые появляются на выходах ОЗУ во время операции считывания, в течение прохождения импульсов *f*, *g*, *h*, *i*.

3.62. Если ОЗУ, показанное на рис. 3.26, стало бы ПЗУ, какие входы на рисунке следует устранить?

Решения

3.57. Во время импульса *a* обе линии управления $\overline{CS}$ и $\overline{WE}$ имеют H-уровень. Таблица истинности состояния (табл. 3.11) указывает, что ОЗУ находится в состоянии ожидания; выходы находятся, таким образом, в H-состоянии. **3.58.** Во время импульсов *b*, *c*, *d* и *e* обе линии управления $\overline{CS}$ и $\overline{WE}$ находятся на L-уровне. Таблица истинности состоя-

ния (табл. 3.11) показывает, что речь идет о состоянии записи в ОЗУ. Данные от источника на входах D копируются в ячейку памяти, адресованную входами A, в течение длительности этих четырех импульсов. **3.59.** В течение длительности этих импульсов $\overline{CS}=0$ и $\overline{WE}=1$. Таблица истинности состояния (см. табл. 3.11) показывает, что речь идет о состоянии чтения. Данные, помещенные в ячейку памяти, указанную адресными входами, появятся на выходе в инверсной форме. Операция считывания не разрушает содержимого ОЗУ. **3.60.** Импульс b — адрес 0000, данные 1111; импульс c — адрес 0001, данные 1110; импульс d — адрес 0011, данные 1100, 1101; импульс e — адрес 0011, данные 1100. **3.61.** Импульс f — адрес 0000, данные 1111, выходные инверсные данные 0000; импульс g — адрес 0001, данные 1110, выходные инверсные данные 0001; импульс h — адрес 0010, данные 1101, выходные инверсные данные 0010; импульс i — адрес 0011, нет данных в этой ячейке памяти, выходные данные неизвестны (случайны). **3.62.** Входы активизации (или разрешения) записи и четыре входа данных. Постоянное запоминающее устройство программируется постоянно конструктором и не имеет состояния записи.

Дополнительные упражнения к гл. 3

3.63. Для образования последовательной логической схемы используют триггеры, для построения комбинационной логической схемы используют _____.

3.64. Начертить логическую схему элемента И с тремя входами.

3.65. Начертить логическую схему элемента ИЛИ с тремя входами.

3.66. Логическая функция может быть описана четырьмя способами: своим названием, логической схемой, таблицей истинности, _____ функцией.

3.67. См. рис. 3.27. Привести последовательность импульсов на выходе Y, зная, что логическая схема — элемент И с двумя входами.

3.68. См. рис. 3.27. Описать последовательность импульсов на выходе Y, зная, что логическая схема — элемент НЕ-И с двумя входами.

3.69. См. рис. 3.27. Привести последовательность импульсов на выходе Y, зная, что логическая схема — элемент ИЛИ ИСКЛЮЧАЮЩЕЕ с двумя входами.

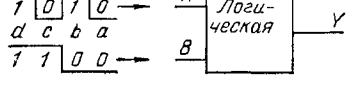


Рис. 3.27. К упражнениям 3.62—3.69

3.70. Обычно МП снабжаются четырьмя командами логических действий в своем составе команд. Какими?

3.71. Записать булеву функцию для таблицы истинности (табл. 3.12).

Таблица 3.12. *Таблица истинности*

Входы			Выход
C	B	A	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

3.72. Начертить логическую схему, удовлетворяющую таблице истинности (табл. 3.12).

3.73. Инициализировать триггер — значит добиться 1 на нормальном выходе, т. е. на выходе _____ (Q, $\overline{Q}$).

3.74. Сбросить или дезактивизировать триггер — значит получить 0 на выходе _____ (Q, $\overline{Q}$).

3.75. Когда триггер работает защелкой, он используется как устройство _____ (счета, памяти).

3.76. Перечислить функциональные режимы JK-триггера на рис. 3.28 для каждого тактового импульса.

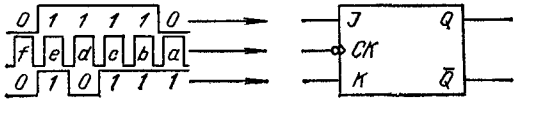


Рис. 3.28. К упражнениям 3.76 и 3.77

3.77. Перечислить двоичные сигналы на нормальном выходе Q JK-триггера на рис. 3.28 после каждого тактового импульса.

3.78. Трансляторы цифрового кода называются обычно _____ или _____.

3.79. Семисегментный индикатор управляется устройством, называемым _____ в код семи сегментов.

3.80. См. рис. 3.16, *в*. Чтобы подать напряжение на сегмент d, вход d индикатора должен находиться в _____ (H-, L-состоянии).

3.81. См. рис. 3.29. В течение импульсов *a* цепь мультиплексор-селектор данных _____ (активна, неактивна).

3.82. См. рис. 3.29. Перечислить выходы мультиплексора с восемью входами для каждой из групп импульсов (*b*—*i*).

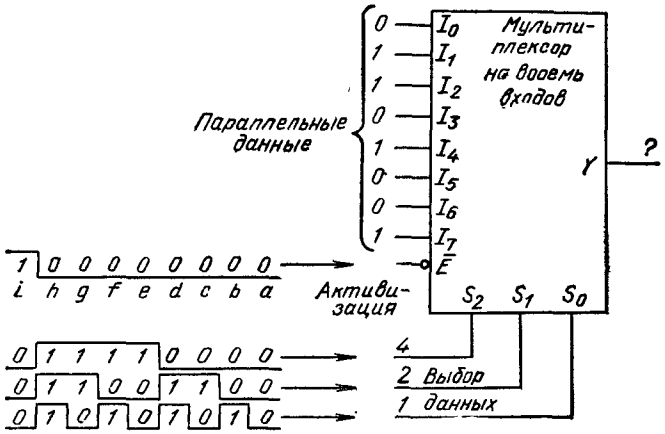


Рис. 3.29. К упражнениям 3.81—3.83

3.83. Включенный по схеме рис. 3.29 мультиплексор функционирует в соответствии с таблицей истинности элемента _____ (ИЛИ, НЕ-И).

3.84. См. рис. 3.29. Логический сигнал _____ (0, 1) передается на выход _____ мультиплексора при группе импульсов *f*.

3.85. В вычислительной технике сокращение ТТЛ означает _____.

3.86. Когда элементы ТТЛ подсоединены на шину микро-ЭВМ, выходы могут находиться в _____.

3.87. Когда выход тристабильного элемента ни 0, ни 1, но элемент подсоединен на напряжение шины, он находится в состоянии высокого _____.

3.88. Какие два типа распространенных устройств памяти используются в микро-ЭВМ?

3.89. Постоянное запоминающее устройство и _____ (ППЗУ, ОЗУ) очень сходны в том смысле, что оба являются устройствами постоянной памяти.

3.90. Поместить данные в память соответствует операции _____ (записи, чтения).

3.91. Копировать данные, существующие в памяти, соответствует операции _____ (записи, чтения).

3.92. На практике сокращением, относящимся к памяти с произвольным доступом (оперативная память), является _____ (ППЗУ, ОЗУ).

3.93. Каковы два типа существующих ОЗУ?

3.94. В нормальных режимах использования _____ (ППЗУ, ОЗУ) могут только считывать.

3.95. Программы пользователя в микро-ЭВМ помещаются только в _____ (ОЗУ, ПЗУ).

3.96. Какая длина слова ОЗУ 256×4 бит?

3.97. Сколько слов данных содержит ОЗУ 32×8 бит?

3.98. Каков общий объем памяти (в бит) ППЗУ 512×8 бит?

3.99. См. рис. 3.30. Оперативное запоминающее устройство объемом 64 бит находится в состоянии ожидания при импульсах _____ (*d*, *f*), и, следовательно, все выходы находятся в _____ (H-, L-) состоянии.

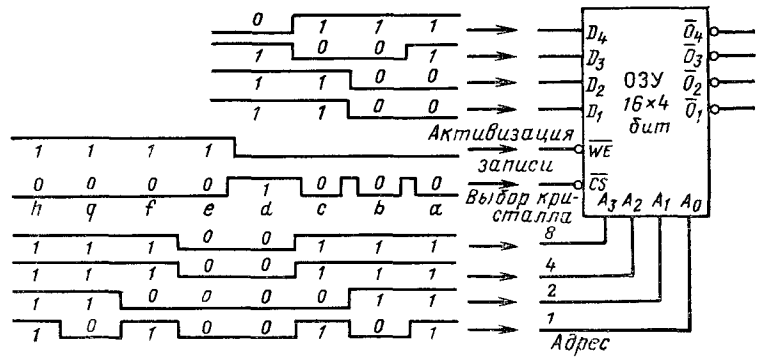


Рис. 3.30. К упражнениям 3.99—3.104

3.100. Оперативное запоминающее устройство на рис. 3.30 находится в состоянии _____ (записи, чтения) во время импульсов *a*, *b* и *c*.

3.101. См. рис. 3.30. Оперативное запоминающее устройство находится в состоянии _____ (записи, чтения) во время импульсов *e*, *f*, *g* и *h*.

3.102. См. рис. 3.30. Перечислить адреса и данные, помещенные в ОЗУ, при операции записи во время импульсов *a*, *b* и *c*.

3.103. См. рис. 3.30. Назвать адреса и данные на выходе ОЗУ при чтении во время импульсов *e*.

3.104. См. рис. 3.30. Назвать адреса и данные, выходящие из ОЗУ при чтении во время импульсов *f*, *g* и *h*.

Решения

3.63. Логических элементов. **3.64.** См. рис. 3.31. **3.65.** См. рис. 3.32. **3.66.** Булевой. **3.67.** Импульс *a* — 0; импульс *b* — 1; импульс *c* — 1; импульс *d* — 0. **3.68.** Импульс *a* — 1; импульс *b* — 1; импульс *c* — 1; импульс *d* — 0. **3.69.** Импульс *a* — 0; импульс *b* — 1; импульс *c* — 1; импульс *d* — 0. **3.70.** НЕ (инверсия или отрицание), И, ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ. **3.71.** $\overline{C}\cdot B\cdot\overline{A}+C\cdot B\cdot A=Y$. **3.72.** См. рис. 3.33. **3.73.** *Q*.

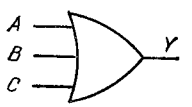


Рис. 3.31. Элемент И с тремя входами

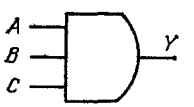


Рис. 3.32. Элемент ИЛИ с тремя входами

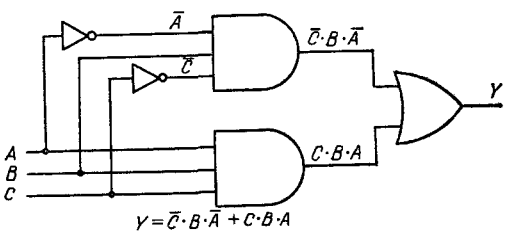


Рис. 3.33. К упражнению 3.72

3.74. *Q*. **3.75.** Памяти. **3.76.** Импульс *a* — сброс; импульс *b* — триггер; импульс *c* — триггер; импульс *d* — установка; импульс *e* — триггер; импульс *f* — ожидание. **3.77.** Импульс *a* — 0; импульс *b* — 1; импульс *c* — 0; импульс *d* — 1; импульс *e* — 0; импульс *f* — 0. **3.78.** Шифраторами, дешифраторами. **3.79.** Дешифратором двоично-десятичного кода. **3.80.** L-состоянии. **3.81.** Неактивна. **3.82.** Импульс *b* — 1; импульс *c* — 1; импульс *d* — 1; импульс *e* — 1; импульс *f* — 1; импульс *g* — 1; импульс *h* — 1; импульс *i* — 0. **3.83.** НЕ-И. **3.84.** 1,7. **3.85.** Транзисторно-транзисторная логика. **3.86.** Трех состояниях. **3.87.** Сопротивления. **3.88.** ОЗУ, ПЗУ. **3.89.** ППЗУ. **3.90.** Запнси. **3.91.** Чтения. **3.92.** ОЗУ. **3.93.** Статнческие, динамические. **3.94.** ППЗУ. **3.95.** ОЗУ. **3.96.** 4 бит. **3.97.** 32. **3.98.** 4096 бнт. **3.99.** *d*, Н-. **3.100.** Запнси. **3.101.** Чтения. **3.102.** Импульсы *a* — адрес 1111, данные 1100; импульсы *b* — адрес 1110, данные 1000; импульсы *c* — адрес 1101, данные 1011. **3.103.** Адрес 0000, данные нензвестны (нет данных, запнсанных в этой ячейке памятн). **3.104.** Импульсы *f* — адрес 1101, данные 1011, выходные данные (нверсные) 0100; импульсы *g* — адрес 1110, данные 1000, выходные данные (инверсные) 0111; импульсы *h* — адрес 1111, данные 1100, выходные данные (инверсные) 0011.

Глава 4

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

Когда программист воспринимает информацию о новом типе МП, он должен выяснить следующие вопросы: архитектура МП; система команд; простейшие системы, использующие данный МП; сигналы управления; назначение выводов.

Понятие архитектуры относится к организации регистров центрального устройства, числа бит шин адресов, данных и т. д.

Система команд — это список операций, которые МП может выполнить. Она включает в себя передачу данных, арифметические и логические операции, команды тестирования данных и ветвлений, операции ввода/вывода (ВВ). В то же время команды используют различные способы адресации.

Схема, представляющая простую информационную систему, показывает, как соединены различные устройства с МП. Простая система может содержать МП, генератор тактовых импульсов (ГТИ), ОЗУ, ПЗУ, порты ВВ, дешифратор адресов и источник питания. Иногда эти функции выполняются ИС или различными прочими составляющими; однако многие микропроцессорные устройства содержат большинство названных элементов.

Сигналы управления являются выходными и управляют другими ИС (ОЗУ, ПЗУ и портами ВВ). Некоторые сигналы могут управлять операциями чтения/записи в памяти или чтения/записи в устройствах ВВ (УВВ).

Наконец, изучение выводов каждой ИС обеспечивает дополнительную информацию о специальных входах и выходах МП. Среди прочих выводов мы найдем выводы питания, ГТИ, ввода-вывода последовательных данных, входов прерываний и управления шинами.

4.1. АРХИТЕКТУРА ПРОСТОЙ МИКРО-ЭВМ

На рис. 4.1 приведена архитектура простой микро-ЭВМ. Микропроцессор является центром всех операций. Ему необходимы питание и тактовые импульсы. Генератор тактовых импульсов может быть отдельным устройством или входить в состав кристалла МП. Типовой МП может содержать 16 адресных линий, которые составляют *однонаправленную шину адресов,* а также обычно восемь линий, которые составляют *двунаправленную шину данных.*

Архитектура, представленная на рис. 4.1, различает два типа полупроводниковой памяти, используемой системой. Постоянное запоминающее устройство ПЗУ представляет собой постоянную память, которая содержит программу-монитор системы. Оно содержит адресные входы, а также входы активизации только чтения и выбора кристалла, а

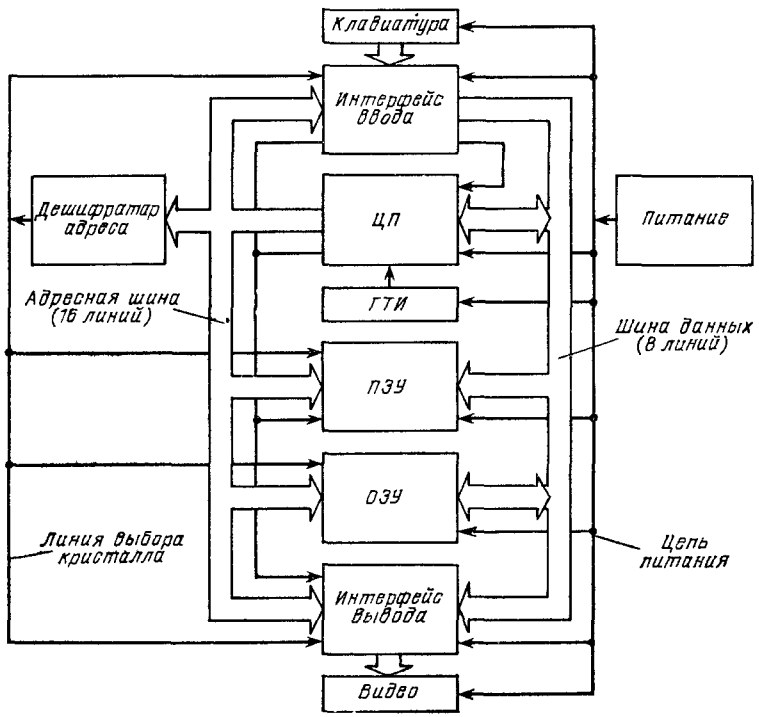


Рис. 4.1. Архитектура микро ЭВМ

также тристабильные выходы, подсоединяемые на шины. Очевидно, ПЗУ имеет также подсоединение питания, которое на схемах обычно не показывается.

Архитектура на рис. 4.1 содержит ОЗУ, т. е. устройство временного размещения данных. В него входят адресные входы, а также входы выбора кристалла и активизации чтения/записи. Это ОЗУ имеет восемь выходов с тремя состояниями, подсоединенных к шине данных. Здесь показан также *источник питания.*

Приведенная система микро-ЭВМ использует клавишное устройство ввода. На этой схеме показаны цепи питания, а также соответствующие линии данных, связанные со специальными ИС — *интерфейсом ввода с клавиатуры.* В задачу интерфейса входит размещение данных и управление их вводом с клавиатуры. В нужный момент интерфейс клавиатуры *прерывает МП* по специальной линии прерывания. Сигнал прерывания заставляет МП: 1) закончить выполнение текущей команды; 2) поддерживать свою нормальную работоспособность; 3) перейти к выполнению специальной группы команд в своем мониторе, по которым ведется управление вводом данных, исходящих с клавишного устройства. Система интерфейса с клавиатурой снабжена адресными входами, линиями выбора кристалла и команд активизации устройства. Активизированное один раз устройство интерфейса с клавиатурой передает данные, поступающие с клавишного устройства на шину данных, микропроцессор их принимает. Если тристабильные выходы интерфейса не активизированы, они возвращаются в свое состояние высокого сопротивления.

Приведенная на рис. 4.1 микро-ЭВМ имеет в качестве выхода группу *семисегментных индикаторов.* Индикаторы запитаны от источника, показанного на схеме справа. Система или специальная ИС *интерфейса вывода на индикаторы* служит для размещения данных и управления состоянием индикаторов. При активизации этого интерфейса по адресной шине, линиям выбора кристалла и активизации он принимает данные, поступающие с шины данных, и размещает их, а также управляет индикатором, на котором размещенные данные высвечиваются.

Адресная линия содержит 65 536 различных сочетаний 0 и 1 (2^{16}). Линии адресной шины могут быть подсоединены ко многим устройствам, таким, как ОЗУ, ПЗУ, другие интерфейсы. Для того чтобы активизировать (включить в работу) требуемое устройство, дешифратор адреса считы-

6*

вает данные с адресной шины. Комбинационной логикой дешифратора адреса активизируется линия выбора соответствующего кристалла, активизируя, таким образом, выбранное устройство. Заметим, что для упрощения схемы все 16 линий адресной шины не показываются.

Упражнения

Все упражнения этого раздела связаны с рис. 4.1. Настоятельно рекомендуем обращаться к нему постоянно

4.1. Центром всех операций управления микро-ЭВМ является _____ (МП, ОЗУ, ПЗУ).

4.2. Шина _____ (адреса, данных) является однонаправленной.

4.3. Посредством 16 линий адресной шины можно получить доступ к _____ (16 384, 65 536) ячейкам памяти и портам ВВ.

4.4. Назвать по меньшей мере три типа выходов МП.

4.5. Назвать по меньшей мере четыре типа входов МП.

4.6. Обычно ПЗУ содержит программу _____ (монитора, изменяемую).

4.7. Назвать по меньшей мере четыре входа ПЗУ.

4.8. Назвать выходы ПЗУ.

4.9. Назвать по меньшей мере четыре типа входов ОЗУ.

4.10. Назвать выходы ОЗУ.

4.11. Назвать по меньшей мере пять типов входов специального элемента интерфейса клавиатуры.

4.12. Назвать по меньшей мере два типа выходов элемента интерфейса клавиатуры.

4.13. Какова роль выхода прерываний элемента интерфейса клавиатуры?

4.14. Какие действия предпринимаются МП, когда линия прерываний активизируется клавишным устройством?

4.15. Назвать по меньшей мере пять типов входов специального элемента интерфейса индикатора.

4.16. Назвать выходы интерфейса индикатора.

4.17. Какова роль дешифратора адреса?

Решения

4.1. МП, однако работой микро-ЭВМ управляют команды, содержащиеся в ПЗУ и/нли в ОЗУ. **4.2.** Адреса. **4.3.** 65 536. **4.4.** Выводы адресной шины (16 линий), шины данных (восемь линий) и шины управления. **4.5.** Входы питания, ГТИ, прерываний и шины данных (восемь лиий).

4.6. Монитора. **4.7.** Входы питания, адресов (адресная шина), выбора кристалла (1) и активизации чтения (1). **4.8.** Выходы шины данных (восемь линий) с тремя состояниями. **4.9.** Входы питания, адресов (адресная шина), выбора кристалла (1), активизации чтения/записи (шина управления) и шина данных (8). **4.10.** Выходы данных на тристабильную шину (8). **4.11.** Входы питания, адреса (с адресной шины), выбора кристалла (1), управления (с управляющей шины) и данных (с клавишного устройства). **4.12.** Входы прерывания и тристабильной шины данных (8). **4.13.** Активизация линии прерываний, когда интерфейс готов к передаче данных, поступающих в МП с клавишного устройства. **4.14.** а) МП завершает выполненне текущей команды; б) МП прерывает свою последовательную работу; в) МП обращается к специальной группе команд, управляющих вводом данных с клавишного устройства **4.15.** Входы питания, адреса (адресная шина), данных (шина данных), управления (шина управления). **4.16.** Выводы данных для возбуждения индикатора. **4.17.** Он выбирает и активизирует единственное требуемое устройство (интерфейс с клавиатурой ПЗУ, ОЗУ или интерфейс индикатора).

4.2. СТРУКТУРА ПРОСТЕЙШЕЙ ПАМЯТИ

Запись в память или считывание из нее происходит при наличии доступа в память. Обычно память выполняется с *последовательным* или *произвольным* доступом. Последовательный доступ означает, что к требуемым данным нужно последовательно пройти через всю память, расположенную до размещения искомых данных.

В случае произвольного доступа данные могут быть записаны в любую ячейку памяти или считаны из нее за определенное фиксированное время, называемое *временем доступа в память*. Оперативные и постоянные запоминающие устройства микро-ЭВМ являются устройствами памяти с произвольным доступом, существенно более быстродействующими, чем устройства с последовательным доступом.

Изучаемый тип микро-ЭВМ обладает адресной шиной из 16 линий, которые могут обеспечить 65 536 (2^{16}) различных комбинаций 0 и 1. На рис. 4.2 приведено множество двоичных комбинаций. Обычно принято двоичный адрес представлять в шестнадцатеричной форме. Как видно из рис. 4.2, $0000\ 0000\ 0000\ 0000_2 = 0000H$ (0000_{16}).

Напомним, что здесь H указывает на то, что речь идет о шестнадцатеричной системе счисления (H-код). Наиболее значимым адресом на рис. 4.2 будет $1111\ 1111\ 1111\ 1111_2$ или FFFFH.

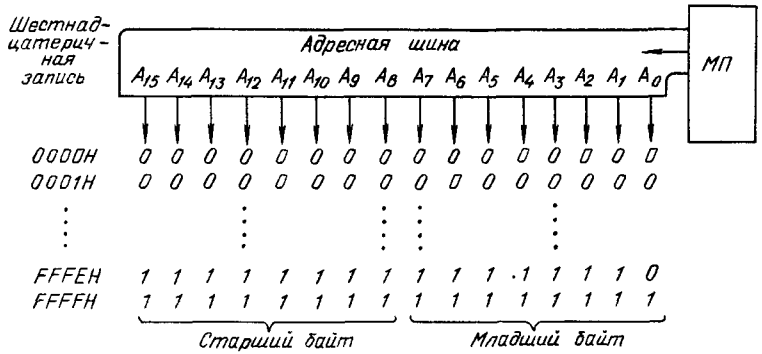


Рис. 4.2. Адресная система микро-ЭВМ

Схема на рис. 4.3 представляет собой воображаемую память микро-ЭВМ. Адресная шина микропроцессора 16-разрядная, она может сформировать 65 536 (0000—FFFFH) индивидуальных адресов (некоторые из них представлены в шестнадцатеричной записи слева на рис. 4.3). В случае этой специальной микро-ЭВМ первые 256 (00—FFH) ячеек памяти являются содержимым ПЗУ с объемом памяти 256×8 бит (т. е. 256 слов по 8 бит). Если адресная шина формирует адрес 0000H, ПЗУ передает программированную комбинацию из 0 и 1, содержащуюся в постоянной памяти (слово 1100 0011).

Для людей, начинающих работать в области микропроцессорной техники, удобно, когда 256 (00—FFH) первых бит или слов данных в ПЗУ помещены на странице 00H. Заметим, что номер страницы равен представлению в шестнадцатеричном коде старшего байта адреса (рис. 4.3). При этих условиях, манипулируя данными на странице 00H, рассматриваются только две шестнадцатеричные цифры младшего разряда адреса. Некоторые микропроцессоры используют упрощенные команды для доступа к ячейкам памяти первой страницы памяти.

Воображаемая элементарная память на рис. 4.3 показывает, что страницы от 01H до 1FH (адреса от 0100H до 1FFFH) в этом отдельном случае не содержат данных в памяти. Получение доступа в эту открытую зону повлечет непредвиденный результат, поскольку она не содержит ни определенных данных, ни программы. Оперативная память для чтения-записи расположена на странице 20H (рис. 4.3). Устройство размещения данных—это ОЗУ 256×8 бит.

Составители этой системы могли бы расположить это ОЗУ на любой другой странице. Доступ к страницам 21H—FFH в этом случае вызовет появление непредвиденного результата по мере вхождения в эту зону, здесь не существует никакой определенной информации.

Если понадобится записать данные в ячейку памяти 2000H (рис. 4.3), восемь элементарных ячеек памяти в начале ОЗУ будут заполнены сочетанием нулей и единиц, поступающих с шины данных. Если микропроцессор захочет затем считать из ячейки памяти по адресу 2000H, в ОЗУ будет прочитано то же сочетание нулей и единиц.

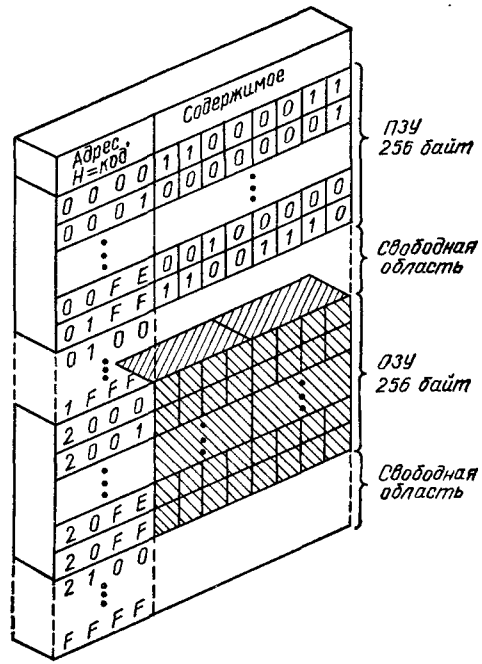


Рис. 4.3. Упрощенное представленне памяти

Физически ОЗУ должно быть, очевидно, составлено единственной ИС. Но получается, однако, что очень часто ОЗУ строятся иначе. На рис. 4.3 заштрихованная часть размещения данных в ОЗУ разделена посредине вертикально. Это указывает на то, что в этом случае зона размещения данных на странице 20H физически состоит из двух различных ИС. Здесь для реализации зоны памяти 256×8 на странице 20H микро-ЭВМ использованы два ОЗУ 256×4 бит.

На рис. 4.4 приведены три варианта реализуемых устройств ИС полупроводникового ОЗУ. На рис. 4.4, *а* единая ИС организована в ОЗУ 256×8 бит, на рис. 4.4, *б* две ИС составляют такую же зону размещения данных и, следовательно, два ОЗУ 256×4 бит позволяют составить слово из 8 бит. Наконец, восемь ОЗУ на рис. 4.4, *в* формируют

также устройство размещения для записи-чтения данных 256×8 бит.

Устройство, представленное на рис. 4.4, *в*, широко используется для составления обширной памяти. Обычно используют ИС ОЗУ 1024×1, 4096×1, 16384×1 бит. Используя способ, показанный на рис. 4.4, *в*, восемь ОЗУ 4096×1

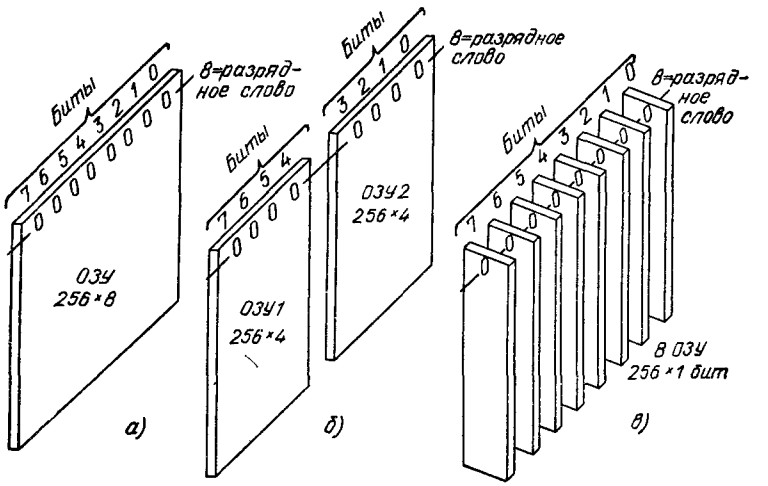


Рис. 4.4. Три способа формирования памяти микро-ЭВМ:
а — на одном кристалле; *б* — на двух кристаллах; *в* — на восьми кристаллах

бит будет достаточно для формирования оперативной памяти 4096×8 бит. Отметим, что в этом типе конфигурации такие ОЗУ будут активизированы одной и той же линией выбора кристалла, исходящей из дешифратора адресов.

В системе микро-ЭВМ объем памяти (или память 4096×8 бит) составляет 4 Кбайт (4 К=4096_{10} байт памяти)[1].

Упражнения

4.18. Запись в ячейки памяти или считывание из них представляет собой _____ (доступ в, поиски) память.

4.19. Оперативное и постоянное запоминающие устройства, используемые в микро-ЭВМ, являются примерами

[1] 1 К=2^{10}=1024_{10}. — *Прим. ред.*

памяти с _____ (произвольным, последовательным) доступом.

4.20. См. рис. 4.2. Зная адрес, выставленный на адресную шину, 0010 0000 0000 0000, определить какая ячейка памяти (в шестнадцатеричном коде) доступна микропроцессору?

4.21. См. рис. 4.3. Зная, что адресуется ячейка памяти микро-ЭВМ 0001Н, _____ (ПЗУ, ОЗУ) выдаст _____ (привести 8 бит).

4.22. См. рис. 4.3. Что появится на шине данных, если МП считывает содержимое ячейки памяти FFFFH?

4.23. См. рис. 4.3. Зная, что адресуется ячейка памяти 2001Н микро-ЭВМ и что элемент памяти находится в состоянии записи, какой _____ (байт, тетрада) будет записан в _____ (ОЗУ, ПЗУ)?

4.24. На рис. 4.3 _____ (ОЗУ, ПЗУ) состоит из единственной ИС.

4.25. Оперативное запоминающее устройство на рис. 4.3 расположено на странице _____.

4.26. Восемь ИС ОЗУ 16 384×1 бит могли бы быть составлены как ИС на рис. 4.4, *в* для того, чтобы образовать память 16 384×8 бит, что составит _____ К.

4.27. См. рис. 4.3. Какая наибольшая емкость памяти может быть адресована 16 адресными линиями?

Решения

4.18. Доступ в. **4.19.** Произвольным. **4.20.** 0010 0000 0000 0000 = =2000Н, т. е. микропроцессору доступна ячейка памяти 2000Н, если на адресную шину подано 0010 0000 0000 0000. **4.21.** Согласно рис. 4.3 ПЗУ, доступное по адресу 0001Н, постоянно будет выдавать помещенные в него данные 0000 0001. **4.22.** В ячейке памяти FFFFH данных нет, результат непредвидим. **4.23.** Согласно рис. 4.3 адрес 2001Н принадлежит ОЗУ и каждое слово содержит 8 бит, т. е. байт. **4.24.** ПЗУ. Что касается ОЗУ, оно, очевидно, составлено из двух ИС ОЗУ 256×4 бит. **4.25.** 20Н. **4.26.** 16 К, содержащих 16 384 байт или слов. **4.27.** 64К. Адресная часть содержит от 0000Н до FFFFH или 65536_{10} ячеек памяти. В данном случае слово памяти составляет 8 бит. В других системах длина слова может составлять от 4 до 16 бит.

4.3. СОСТАВ КОМАНД

Группа команд, которые может выполнять данный МП, называется его *составом команд.* Состав команд МП

может содержать как малое число (восемь), так и большое число (200) основных команд. Составы команд не являются нормализованными. Это неудобство связано как с индивидуальным подходом к разработке, так и с различиями архитектуры и назначений МП.

Имеется много способов классификации команд одного состава. В этой главе согласно нормативам, предложенным научным обществом инженеров-электронщиков, мы изучим следующие команды: арифметические, логические, передачи данных, вызова подпрограмм, возврата из подпрограмм, прочие.

Элементарный МП будет представлен следующим составом арифметических команд: сложение, вычитание, инкрементирование, сравнение, отрицание.

Некоторые конкретные МП могут обладать другими арифметическими командами, такими, как сложение с переносом, вычитание с заемом, умножение и деление.

Элементарный МП наделяется следующими логическими командами: И, ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ, НЕ (отрицание), сдвиг вправо, сдвиг влево.

Некоторые МП, кроме того, наделены такими логическими командами, как арифметический сдвиг вправо, циклические сдвиги вправо и влево, циклические сдвиги вправо и влево с переносом и тестирование.

Наш же элементарный процессор всегда наделяется *командами передачи данных:* загрузки, размещения, перемещения, ввода, вывода.

Более сложный состав будет содержать команды обмена, сброса и инициализации.

Что касается *команд ветвления,* они следующие: безусловный переход; переход, если нуль; переход, если не нуль; переход, если равенство; переход, если неравенство; переход, если положительно; переход, если отрицательно.

Другие команды условных переходов, имеющиеся в некоторых микропроцессорах, могут зависеть от таких условий, как: больше или меньше, сдвиг или нет, переполнение или нет. Команды ветвления являются командами принятия решений.

Элементарный микропроцессор будет наделен командой *вызова подпрограммы* (обычно CALL — вызов), чтобы программа могла перейти к специальной группе команд, которые решают поставленную задачу. Все МП обладают командой безусловного вызова, а некоторые наделены командой условного вызова, например, CALL, если нуль; CALL, если не нуль; CALL, если положительно или не положительно, и т. д.

В конце выполнения подпрограммы МП должен иметь возможность возврата в точку отправления из начальной программы. Эта операция выполняется *командой возврата.* Эта команда обычно безусловна, но некоторые МП снабжены и условным возвратом.

Наконец, *прочими командами* элементарного МП будут: нет операции, поместить в стек, выйти из стека, ожидание, останов.

Возможны и другие команды: прерывания активизации или сброса, останова, десятичной коррекции.

Пользователи микропроцессорных систем встречаются с многочисленными способами выражения одних и тех же команд. Из этого множества приведем команду сложения двух чисел для микропроцессора Motorola 6800. Имя команды является *активной формой глагола* и называется операцией. В табл. 4.1 выполняемой операцией является сложение.

Таблица 4.1. *Описание команды ADD*

Операция	Мнемоника	КОП	Символика (все обозначения регистров являются их содержимым)
Сложить	ADD *A*	8BH	$A+M \rightarrow A$

Пользователи работают часто с сокращенными формами выражения операции, которые являются обычно *мнемоническими.* В табл. 4.1 мнемоникой сложения является ADD *A* (отметим, что мнемоники всегда записываются большими буквами). Регистр команд и схема декодирования понимают только язык нулей и единиц. *Код операции* (КОП) является шестнадцатеричным представлением 8-разрядного двоичного кода, который заставляет МП выполнить эту команду. В табл. 4.1 КОП для МП Motorola 6800 для ADD *A* будет 8BH ($1000\ 1011_2$). В колонке символики в табл. 4.1 показывается, что содер-

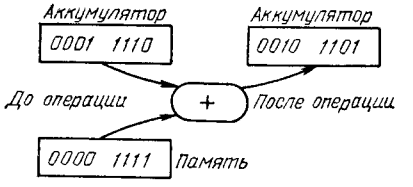


Рис. 4.5. Команда ADD

жимое памяти (М) складывается с содержимым аккумулятором (А) в МП, стрелка указывает, что результат помещается в аккумулятор А.

На рис. 45 приведен пример использования команды сложения ADD *A*. Содержимое (А) аккумулятора ($0001\ 1110_2$) складывается с содержимым (М) памяти ($0000\ 1111_2$), сумма ($0010\ 1101_2$) помещается в аккумулятор (справа). Заметим, что содержимое ячейки памяти не изменилось, тогда как содержимое аккумулятора стало другим.

Упражнения

4.28. Группа операций, которые может выполнить МП, представляет его _____.

4.29. Перечислить семь групп состава команд МП.

4.30. Операция ADD выполняется по команде арифметических действий, тогда как операция ИЛИ выполняется по команде _____ действий.

4.31. Операция сдвига выполняется командой _____ действий.

4.32. Перечислить пять основных операций передачи данных, которые можно найти в составе команд МП.

4.33. Операции принятия решений выполняются по командам _____.

4.34. Перечислить команды условного перехода состава команд МП.

4.35. Специальная группа команд, которая в ходе решения текущей задачи может часто повторяться, называется _____ (индексом, подпрограммой).

4.36. Какую текущую команду отрабатывает МП, ответвляясь в подпрограмму?

4.37. Какая команда в конце подпрограммы приводит МП в основную программу?

4.38. Перечислить пять различных команд, принадлежащих составу команд многих МП?

4.39. См. табл. 4.2. Что следует записать в верхней строке таблицы при операции вычитания?

Таблица 4.2. *Операция вычитания*

Операция	?	?	Символика (обозначение регистра—его содержимое)
Вычесть	SUB *A*	80H	*A*—*M*→*A*

4.40. См. рис. 4.6. Число 80H (Н-код) представляет собой _____.

4.41. См. рис 4.6. После выполнения операции SUB *A* в аккумуляторе будет _____.

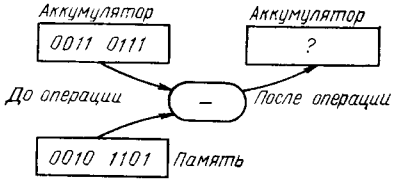


Рис. 4.6. Пример вычитания

Решения

4.28. Состав команд. **4.29.** Арифметические, логические, передачи данных, ветвления, вызова подпрограмм, возврата из подпрограмм, прочие. **4.30.** Логических. **4.31.** Логическнх. **4.32.** Загрузка, размещение, перемещение, ввод, вывод. **4.33.** Ветвления. **4.34.** Переход, если 0, если не 0, если равно, если не равно, если положительно, если отрицательно, если больше, если меньше, если сдвиг, если сдвига нет, если переполнение, если переполнения нет. **4.35.** Подпрограммой. **4.36.** Команду CALL. **4.37.** Команда возврата RETURN (RET). Эта команда может также восстанавливать некоторые регистры в их начальное состояние. **4.38.** Нет операции, поместнть в стек, извлечь из стека, ожидание, останов, прерыванне активизацни, сброс, захват, десятичная коррекция. **4.39.** Сокращение SUB *A* является мнемоннческой записью операции вычитания из содержимого аккумулятора. Следовательно, над SUB *A* должно быть записано «Мнемоника». **4.40.** Код операции (КОП). **4.41.** В двоичном коде 0011 0111—0010 1101=0000 1010 (десятичные 55—45=10). Таким образом, аккумулятор будет содержать $0000\ 1010_2$ после выполнения операции SUB *A*.

4.4. СТРУКТУРА ЭЛЕМЕНТАРНОГО МИКРОПРОЦЕССОРА

Основным устройством всех информационных систем является *центральный процессор* (ЦП). Из многочисленных ИС роль ЦП систем выполняют микропроцессоры. Обычно в технологии микроинформационной техники программную память, память данных, интерфейс ввода-вывода, дешифратор адресов выполняют на различных ИС, как это показано на рис. 4.1.

Центральным устройством системы является микропро-

цессор, который содержит обычно элементы размещения данных, называемые регистрами, и устройство счета, называемое арифметико-логическим устройством (АЛУ). Центральное устройство содержит также цепь декодирования команд и секцию управления и синхронизации. Оно снабжено также необходимыми соединениями с устройством ввода/вывода.

Основными функциями центрального устройства микроЭВМ являются следующие:

1) извлечение, декодирование и выполнение команд программы в указанном порядке;

2) передача данных из памяти и в память и из УВВ и в УВВ;

3) ответы на внешние прерывания;

4) установка общей синхронизации и сигналов управления для всей системы.

Большинство центральных устройств содержит по меньшей мере элементы, схематически представленные на рис. 4.7. Наиболее важные секции содержат различные регистры, АЛУ, дешифратор команд, устройства управления и синхронизации, а также УВВ. В настоящее время большинство микропроцессоров содержат множество дополнительных специальных регистров (на рис. 4.7 не показаны).

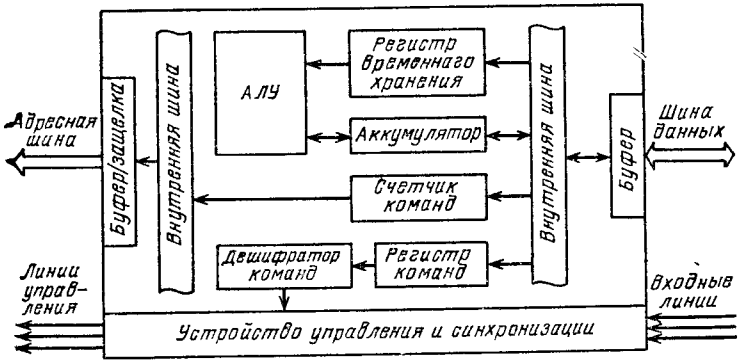


Рис. 4.7. Упрощенная архитектура центрального процессора

Арифметико-логическое устройство ЦП выполняет такие операции, как сложение, сдвиг/перестановка, сравнение, инкремент, декремент, отрицание, И, ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ, дополнение, сброс, инициализация.

Если АЛУ должно выполнить операцию сложения посредством команды ADD, процедура могла бы быть аналогичной представленной на рис. 4.8, *а*. Здесь содержимое аккумулятора 0АН складывается с содержимым регистра временного хранения данных 05Н. Сумма 0FH помещена в аккумулятор.

На рис. 4.8, *б* приведены основные функциональные элементы типового АЛУ. Оно содержит сумматор и устройство сдвига, а результаты пересылаются в аккумулятор

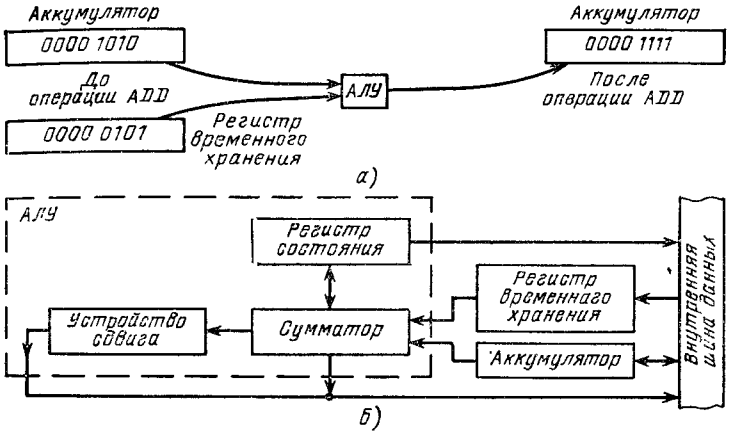


Рис. 4.8. Выполнение операции ADD (*а*) и структура АЛУ (*б*)

посредством внутренней шины данных. *Регистр состояния* слова в АЛУ является устройством чрезвычайно важным (его называют иногда *регистром кода условий* или *индикатором*)[1]. Этот регистр состоит из группы триггеров, которые могут быть установлены или сброшены исходя из результатов последней операции, выполненной АЛУ. Эти триггеры или индикаторы содержат указатели нуля, отрицательного результата, переноса и т. д. Индикаторы используются для принятия решений, когда вводятся команды ветвлений. Аккумулятор обычно используется в ходе большинства операций, выполняемых центральным устройством, например, передачи данных.

Устройство управления и синхронизации (см. рис. 4.7)

[1] Широко распространен термин *флажковый регистр*. Тогда соответствующие триггеры или индикаторы этого регистра называются *флагами* — флаг нуля, флаг знака и т. д. — *Прим. ред.*

является наиболее сложным в центральном процессоре. Оно влияет на все события и управляет их протеканием внутри центрального устройства и во всей микро-ЭВМ. Мы упоминали в предыдущей главе, что каждая команда программы может быть разделена на этапы извлечения и выполнения. Каждый из них в свою очередь может быть разделен на элементарные микропрограммы. Микропрограммы каждой команды находятся в секции декодирования и выполняются блоком управления и синхронизации центрального устройства.

Шестнадцатиразрядный регистр, называемый *счетчиком команд,* представлен на рис. 4.7 как элемент, составляющий часть центрального устройства. Этот регистр служит для хранения адреса следующей команды, чтобы извлечь ее из памяти. Так как команды выполняются последовательно, счетчик команд считает прямым счетом, если только нет контрпорядка. Большая часть выпускаемых микропроцессоров имеет 16-разрядный счетчик команд, который может адресовать 64 К слов памяти посредством адресной шины. Нормальная последовательность выполнения команд программы может быть изменена специальными командами ветвления, вызова подпрограмм, возврата из подпрограмм или прерывания. Эти команды повлекут переход содержимого счетчика команд на другую величину, отличную от следующего старшего адреса. Чтобы вернуть программу в исходное состояние после последовательности ее запуска, оператор должен восстановить в счетчике команд номер первой команды программы.

Последовательность извлечение-декодирование-выполнение команд является основой функционирования вычислительной машины. Первая команда, извлеченная из памяти программы, определяет код операции первой команды и помещается в регистр команд устройством управления центральным процессором. Код операции истолковывается дешифратором команд, который указывает затем процессору процедуру управления и синхронизации, которой должна следовать программа для выполнения заданной команды.

Центральное устройство, показанное на рис. 4.7, является элементарным. Большая часть центральных устройств МП содержит, по меньшей мере, несколько дополнительных регистров (8 и 16 бит). Существуют очень большие различия в количестве и типе регистров в зависимости от типов МП.

Упражнения

4.42. Какую часть микро-ЭВМ обозначают сокращением ЦП?

4.43. См. рис. 4.1. Где располагается ЦП в блоке всей системы микро-ЭВМ?

4.44. Центральный процессор обычно содержит: а) устройство размещения данных, называемое _____; б) устройство счета, называемое _____; в) устройство _____; г) устройство _____ и синхронизации.

4.45. Какие четыре основных назначения ЦП в микро-ЭВМ?

4.46. Какое устройство микро-ЭВМ сокращенно называется АЛУ?

4.47. См. рис. 4.9. Каково содержимое аккумулятора после операции И?

4.48. См. рис. 4.9. После операции И индикатор нуля будет _____ (установлен, сброшен).

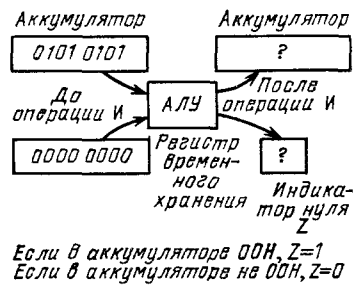


Рис. 4 9. К упражнениям 4.47 и 4.48

4.49. Регистр состояний АЛУ называется также регистром кода _____, а триггеры называются _____.

4.50. Какая важная часть ЦП предназначена для управления всеми событиями внутри системы?

4.51. Регистр ЦП, удерживающий адрес последующей команды, извлекаемой из программной памяти, называется _____.

4.52. Обычно счетчик команд инкрементируется в прямом направлении для адресации команд в программной памяти в порядке возрастания адреса, за исключением случаев, когда его содержимое изменяется командами _____.

4.53. В начале процедуры выполнения команды КОП первой команды помещается в регистр _____ (аккумулятора, команд) МП.

4.54. Часть МП, интерпретирующая КОП, помещенный в регистр команд, и определяющая последующую процедуру управления и синхронизации для выполнения команды, является _____.

4.42. Центральный процессор. **4.43.** Внутри микропроцессора. **4.44.** а) регистром; б) АЛУ; в) декодирования команд; г) управления. **4.45.** а) Извлечь, декодировать и выполнить команды программ в заданной последовательности. б) Передать данные из (в) памяти, из (в) УВВ. в) Ответить на внешние прерывания. г) Обеспечить требуемые сигналы управления и синхронизации. **4.46.** Арифметико-логическое устройство. **4.47.** 0000 0000. Таблица истинности функции И приведена в табл. 3.1. **4.48.** Условия установки индикатора нуля определены содержимым аккумулятора после операции И. Так как в этом случае его содержимым будет 0, индикатор нуля в регистре состояния станет равным 1 после операции И. **4.49.** Условий; индикаторами. **4.50.** Секция управления и синхронизации. **4.51.** Счетчиком команд. **4.52.** Ветвления, возврата, вызова прерываний. **4.53.** Команд. **4.54.** Дешифратором команд.

4.5. ФУНКЦИОНИРОВАНИЕ МИКРО-ЭВМ

Пусть требуется выполнить простую операцию сложения трех чисел, например $10+5+18=33_{10}$. Короткая и простая микропрограмма выполнения этой операции могла бы быть записана в следующей последовательности.

Команда *1*: загрузить (LOAD) первое число (10_{10}) в ЦП.
Команда *2:* сложить (ADD) второе число (5_{10}) с первым.
Команда *3* сложить (ADD) третье число (18_{10}) с двумя предыдущими.
Команда *4:* поместить (STORE) сумму (33_{10}) в ячейку памяти 2000H.

После загрузки в память программы эти команды могли бы извлекаться из нее как команды памяти, показанной на рис. 4.10. Заметим, что первая команда программы начинается с адреса 0000H. Эта команда (LOAD число 0AH) использует 2 байт памяти. Первый байт памяти содержит оперативную часть команды, другой — операнд. Код операции LOAD для микропроцессора, используемого в этом примере, будет 86H (1000 0110_2). Операнд 0AH (0000 1010_2) является первым числом, подлежащим загрузке в аккумулятор микропроцессора. Заметим, что рис. 4.10 является широко распространенным представлением содержимого памяти и адресов в шестнадцатеричной записи. В реальной действующей машине такая информация представляется в форме напряжения H- и L-уровней.

Предположим, что программа размещена в блоке ОЗУ микро-ЭВМ (рис. 4.1), в которую входят устройства, представленные на рис. 4.7. В таком случае рис. 4.11 иллюстрирует каждую операцию программы (LOAD, ADD, ADD, STORE).

Операция загрузки (LOAD) первой команды подробно приведена на рис. 4.11, *а* и показывает, что содержимое ячейки памяти 0001H загружено в аккумулятор, который содержит после этого 0000 1010_2 — первое слагаемое число. В результате операции загрузки стирается предыдущее и записывается новое содержимое аккумулятора. Вторая команда, операция ADD, детализирована на рис. 4.11, *б*. Содержимое ячейки памяти 0003H (0000 0101_2) складывается с содержимым аккумулятора 0000 1010_2, что дает сумму 0000 1111_2, помещаемую в аккумулятор, и мы можем заметить, что содержимое аккумулятора изменяется при операции ADD.

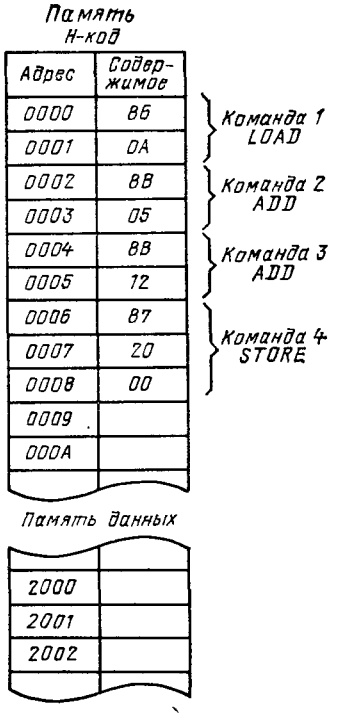


Рис. 4.10. Часть программы сложения 10+5+18

На рис. 4.11, *в* показана вторая операция ADD (команда *3*); содержимое аккумулятора — сумма 0000 1111_2 сложена с содержимым ячейки памяти 0005H, т. е. выполняется операция 0000 1111+0001 0010=0010 0001. Окончательно 0010 0001 появляется в аккумуляторе после второй операции ADD.

Операция STORE (РАЗМЕСТИТЬ) по команде *4* представлена на рис. 4.11, *г*. Содержимое аккумулятора (0010 0001_2) передано и размещено в ячейке памяти по адресу 2000H. Заметим, так как это важно, что ячейки памяти данных были идентифицированы в памяти программы двумя раздельными байтами (0007H и 0008H). Ячейка памяти программы 0006H содержит КОП B7H прямой команды STORE (см. рис. 4.10).

Рассмотрим извлечение, декодирование и выполнение команды LOAD по адресам 0000H и 0001H в программе.

Этот тип команды будет выполнен, вероятно, за время около 2—6 мкс большинством микро-ЭВМ. Рисунок 4.12 иллюстрирует процедуру выполнения центральным процессором этой специальной операции.

Рассмотрим сверху слева направо последовательность действий на рис. 4.12: счетчик команд прежде всего уста-

Рис. 4.11. Команды:
а — LOAD; *б* — ADD; *в* — ADD; *г* — STORE

навливает адрес первого этапа программы. После этого 16-разрядный адрес передается в адресный регистр, затем на адресную шину и в память программы. Для активизации памяти программы ЦП выдает сигнал считывания программы (1 на линии $R/\overline{W}$), в то время как дешифратор адресов (который не входит в состав ЦП) активизирует выбор кристалла $\overline{CS}$ нулем. Затем счетчик команд инкрементируется до 0001H, ячейка памяти программы 0000H становится доступной и ее содержимое считывается на шину данных. Код операции (86H) команды LOAD передается в регистр ЦП. Этап извлечения КОП команды LOAD завершен.

Затем КОП (86H), содержащийся в регистре команды ЦП, интерпретируется дешифратором команд. В данном случае ЦП определяет, идет ли речь о команде LOAD непосредственно, что означает загрузку им содержимого памяти, адрес которой следует непосредственно за КОП, в аккумулятор. Содержимое счетчика команд (0001H) передается в адресный регистр, на адресную шину и в память.

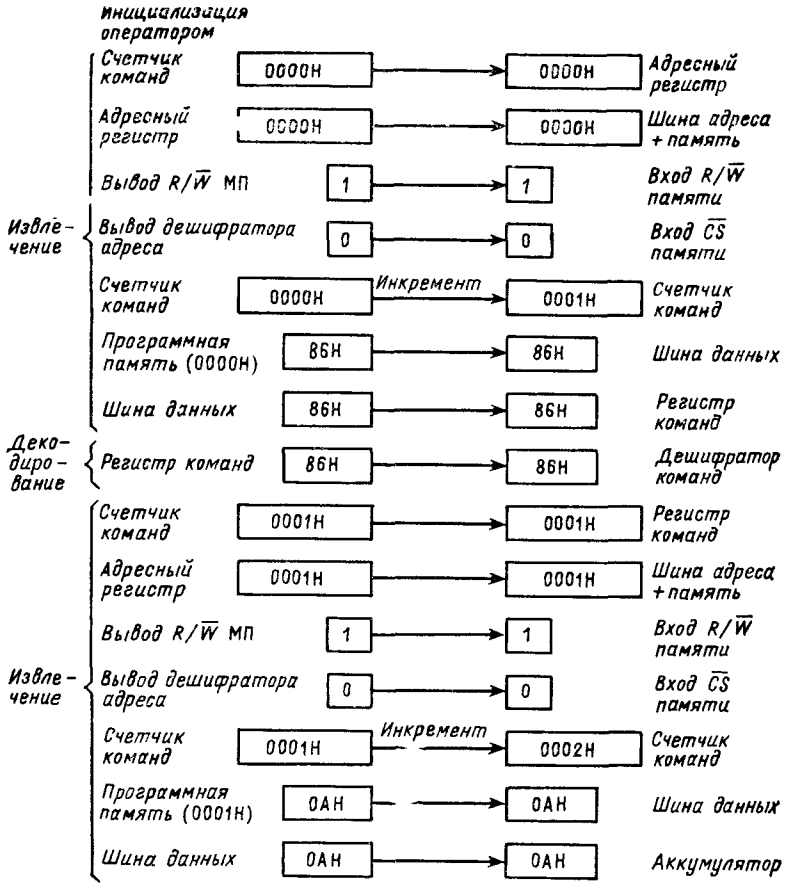


Рис. 4.12. Подробный анализ команды LOAD непосредственных данных

Центральный процессор выдает импульс HIGH считывания на вход $R/\overline{W}$ памяти. Импульс LOW поступает на вход $\overline{CS}$ памяти, что активизирует память. В третьей строке (снизу) на рис. 4.12 счетчик команд инкрементируется до 0002H, что подготавливает его к извлечению последующей команды. Ячейка памяти 0001H становится доступной, и ее содержимое (0AH) поступает на шину данных, затем передается в аккумулятор. В ходе этапа извлечения КОП всегда помещается в регистр команд.

Упражнения

4.55. Две части команды микропроцессоров называются операцией и_____.

4.56. См. рис. 4.10. Когда ячейка памяти 0002H доступна, ее содержимое передается в регистр _____ ЦП через шину _____ (адреса, данных).

4.57. См. рис. 4.10. Ячейка памяти 0003H содержит 05H, что рассматривается ЦП как _____ (операция, операнд) команды.

4.58. Двоичное 16-разрядное число может быть представлено шестнадцатеричным числом _____ цифрами.

4.59. Привести три последовательных этапа цикла команды микропроцессора.

4.60. См. рис. 4.12. Этап извлечения является основным при операции _____ (считывания, записи), содержимое памяти помещается в _____ (аккумулятор, регистр команд) микропроцессора.

4.61. В ходе извлечения _____ (дешифратор команд, счетчик команд) указывает адрес КОП извлечения из памяти.

4.62. См. рис. 4.12. Какие другие входы памяти необходимы для активизации считывания байта кроме адресного входа?

4.63. См. рис. 4.10. Какое шестнадцатеричное число должно быть помещено в ячейку памяти 2000H после выполнения ЦП команды *4*?

4.64. См. рис. 4.12. После последовательного выполнения последовательности действий счетчик команд восстанавливается _____ (дешифратором адреса, оператором).

Решения

4.55. Операндом. **4.56.** Команд; данных. **4.57.** Операнд. **4.58.** Четырьмя. Например, $0010\ 0000\ 0000\ 1111_2$ представляется в шестнадцатеричном коде как 200FH. **4.59.** Извлечение — декодирование — выполнение. **4.60.** Считывания; регистр команд. **4.61.** Счетчик команд. **4.62.** Входы $R/\overline{W}$ (чтение/запись) и $\overline{CS}$ (выбор кристалла). **4.63.** Программа выполняет сложение 0AH+05H+12H, сумма будет 21H. Эта сумма помещена в ячейку памяти 2000H (см. рис. 4.11, *г*). **4.64.** Оператором.

Дополнительные упражнения к гл. 4

4.65. Перечислить темы, которые следует изучить при знакомстве с новым микропроцессором.

4.66. Однонаправленная 16-разрядная шина на рис. 4.1 является шиной _____.

4.67. Двунаправленная 8-разрядная шина на рис. 4.1 является шиной _____.

4.68. См. рис. 4.1. Устройство постоянной памяти сокращенно называют_____, тогда как устройство переменной памяти сокращенно называют _____.

4.69. См. рис. 4.1. Системы, называемые _____, связывают порты УВВ с периферийными устройствами системы.

4.70. См. рис. 4.1. Подключение МП на шину данных выполняется по принципу _____ (ввода, вывода, двунаправленного обмена).

4.71. См. рис. 4.1. Зная, что на шине установлен адрес 0010 0000 1000 1111, определить, какая ячейка памяти будет адресована микропроцессором (шестнадцатеричное значение)?

4.72. Когда микро-ЭВМ получает доступ к ячейке памяти 00FEH, ПЗУ _____ (выдает, получает) 8 бит _____.

4.73. См. рис. 4.3. Постоянное запоминающее устройство представляет собой память емкостью 256×_____ бит.

4.74. См. рис. 4.3. На какой странице располагается ПЗУ?

4.75. Для формирования оперативной памяти 4096× × _____ бит или памяти _____ К могут использоваться 8 ИС ОЗУ 4096×1, как показано на рис. 4.4, *в*.

4.76. Список выполняемых процессором операций представляет собой состав _____ микропроцессора.

4.77. Команды микропроцессора классифицируются на различные команды вызова подпрограмм, возврата из подпрограмм, ветвления, передачи данных, _____

4.78. Команда ADD является арифметической, команда STORE — _____ .

4.79. Команды ветвления являются командами _____.

4.80. Команды вызова подпрограмм и _____ (арифметические, возврата из подпрограмм) в подпрограмме всегда объединяются попарно.

4.81. Операция ИЛИ ИСКЛЮЧАЮЩЕЕ является _____ командой.

4.82. Системой счета в МП является_____.

4.83. Центральный микропроцессор микро-ЭВМ обычно содержит модификацию оперативной памяти, составляемой _____ (ОЗУ, регистрами).

4.84. Центральный процессор микро-ЭВМ содержит обычно устройство интерпретации команд, называемое _____.

4.85. Центральный процессор микро-ЭВМ содержит цепи упорядочивания событий. Это цепи _____.

4.86. Центральный процессор микро-ЭВМ представляет собой устройство, называемое _____.

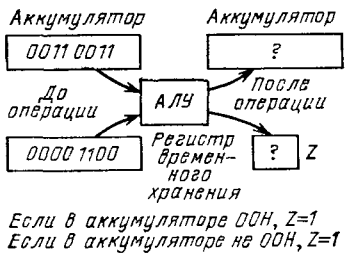


Рис. 4.13. Команда ИЛИ

4.87. См. рис. 4.13. Каково двоичное содержимое аккумулятора после операции ИЛИ?

4.88. См. рис. 4.13. После операции ИЛИ индикатор нуля будет _____ (установлен, сброшен).

4.89. Индикаторы принадлежат специальному регистру — регистру кода условия или регистру _____, расположенному в АЛУ.

4.90. Счетчик команд МП хранит _____ (адрес, время) следующей выполняемой команды.

4.91. В ходе извлечения _____ (КОП, операнд) читается в памяти и передается в регистр команд МП.

4.92. В ходе декодирования дешифратор команд, расположенный в _____ (памяти, ЦП) интерпретирует КОП, расположенный в регистре _____ (команд, синхронизации).

4.93. См. рис. 4.14, *а*, *б*. Каково содержимое аккумулятора после операции И?

4.94. См. рис. 4.14, *а* и *в*. Каково содержимое аккумулятора после операции ИНКРЕМЕНТ?

4.95. См. рис. 4.14, *а* и *г*. Какой адрес (шестнадцатеричный код) ячейки памяти, приведенной на рис. 4.14, *г*?

4.96. См. рис. 4.14, *а* и *г*. Каково двоичное содержимое ячейки памяти данных на рис. 4.14, *г* после операции STORE?

4.97. Каково двоичное содержимое аккумулятора (см. рис. 4.14, *г*) после операции STORE?

4.98. См. рис. 4.14, *а*. Какие две ячейки памяти содержат адрес операции STORE?

4.99. См. рис. 4.14, *а* и *б*. Результатом операции И с 00Н (команда 1) будет: _____ (сбросить, установить) аккумулятор, обращая все его биты в _____ (0,1).

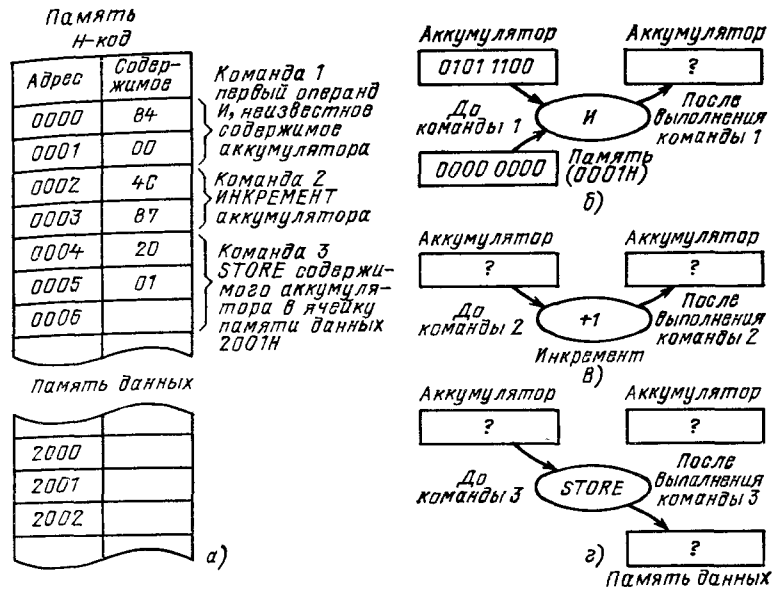


Рис. 4.14. Часть программы (*а*) и выполненне команд:
б — И; *в* — инкремента; *г* — STORE прямой адресации

4.100. См. рис. 4.14, *а*. Код операции *1* _____, а ее операнд _____.

4.101. См. рис. 4.14, *а*. Код операции *2*, а ее операнд _____.

4.102. См. рис. 4.14, *а*. Код операции *3*, а ее операнд _____.

Решения

4.65. Архитектура, состав команд, простейшне системы пользователя, сигналы управления и назначення выводов. **4.66.** Адресный. **4.67.** Данных. **4.68.** ПЗУ; ОЗУ. **4.69.** Интерфейсом. **4.70.** Двунаправленного обмена. **4.71.** 208FH. **4.72.** Выдает; 0010 0000. **4.73.** 8. **4.74.** 00H (нулевая страница). **4.75.** 8; 4. **4.76.** Команд. **4.77.** Арифметические и логические. **4.78.** Передачн данных. **4.79.** Принятия решений. **4.80.** Возврата из подпрограмм. **4.81.** Логической. **4.82.** АЛУ. **4.83.** Регистрамн **4.84.** Дешифратором команд. **4.85.** Управления и синхронизации. **4.86.** Микропроцессором. **4.87.** 0011 1111. **4.88.** Сброшен. **4.89.** Состояния. **4.90.** Адрес. **4.91.** КОП. **4.92.** ЦП; команд. **4.93.** $0000\ 0000_2$.

4.94. 0000 0001_2. **4.95.** 2001H. **4.96.** 0000 0001_2. **4.97.** 0000 0001_2. **4.98.** 0004H и 0005H. **4.99.** Сбросить; 0. **4.100.** 84H; 00H. **4.101.** 4CH; операнда иет для операции инкремента. **4.102.** B7H; сочетание двух байт операнда для формирования адреса памяти 2001H.

Глава 5
МИКРОПРОЦЕССОР

5.1. ПОСТАВЛЯЕМАЯ РАЗРАБОТЧИКОМ ИНФОРМАЦИЯ

Какими бы ни были рассматриваемые микропроцессоры, касающаяся их информация содержит много общего. Типовая документация содержит информацию о структуре ИС, схеме выводов ИС и назначении каждого из них. Схематизируется архитектура МП, описываются его основные свойства. Одновременно даются временные диаграммы и состав команд МП. Документация содержит также схемы различных систем, использующих рассматриваемый микропроцессор.

Обычно микропроцессор помещается в корпус интегральной схемы с 40 двусторонними выводами (корпус с двухрядной упаковкой выводов DIP — *dual-in-line package*). На рис. 5.1 приведены два типа микропроцессоров — в пластмассовом корпусе (рис. 5.1, *а*) и в керамическом (рис. 5.1, *б*) с 40 выводами. Микропроцессор в керамическом DIP-корпусе используется при высоких температурах. Микропроцессоры могут поставляться также с 28, 42, 50 и 64 выводами.

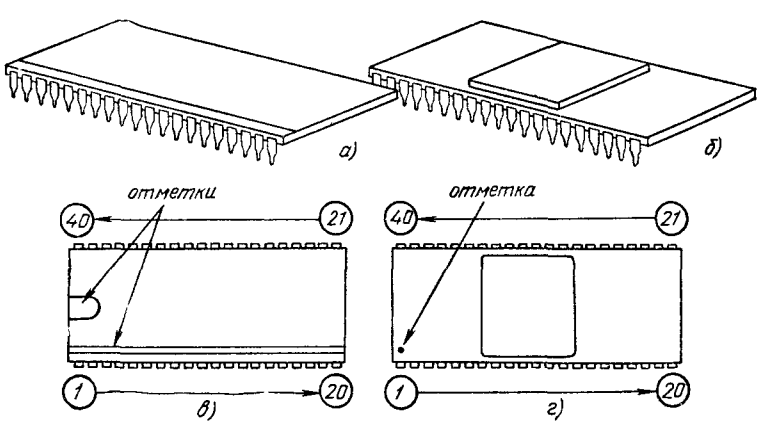


Рис. 5.1. Микропроцессоры в корпусах:
а — керамическом; *б* — пластмассовом с двухрядной упаковкой выводов (DIP); *в* и *г* — отметки для определения порядка нумерации выводов

На рис. 5.1, *в* и *г* приведены два способа определения положения вывода *1*. Заметим вырез и желобок по всей длине, являющиеся отметками на пластмассовом корпусе (рис. 5.1, *в*). Непосредственно после этой отметки в направлении, обратном ходу часовой стрелки, находится вывод *1* ИС. На рис. 5.1, *г* отметкой, позволяющей определить вывод *1* ИС, является маленькая точка слева. Затем выводы нумеруются в направлении, обратном ходу часовой стрелки при виде на ИС сверху.

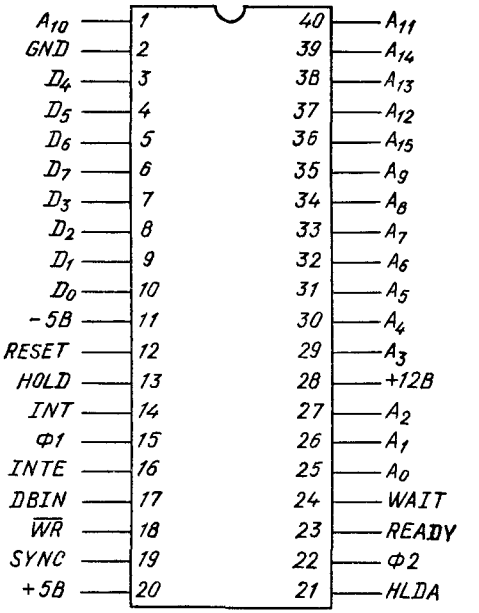


Рис. 5.2. Схема выводов МП Intel 8080

Выводы	Назначение	Вход или выход
GND, +5 В, −5 В, +12 В	Питание	Входы
Ф1, Ф2	Тактовые импульсы	Входы
D_0—D_7	Линии данных	Двунаправленные
A_0—A_{15}	Линии адресов	Выходы
SYNC	Синхронизация	Выход
DBIN	Строб входных данных	Выход
WAIT	Ожидание	Выход
$\overline{WR}$	Строб записи	Выход
HLDA	Подтверждение захвата	Выход
INTE	Разрешение прерывания	Выход
READY	Готовность ввода данных	Вход
HOLD	Захват	Вход
INT	Требование прерывания	Вход
RESET	Сброс	Вход

Схема выводов (рис. 5.2) приводится в документации. Разработчики представляют все сведения о названиях и назначениях каждого из выводов микропроцессора. Схема на рис. 5.2 соответствует микропроцессору Intel 8080. Отметим, что выводы *2*, *11*, *20*, *28* являются выводами питания.

Выводы *15*, *22* (*Ф1*, *Ф2*) являются входами внешнего двухфазного генератора тактовых импульсов — часов. Выводы *3—10* (Intel 8080) двунаправленные (это значит, что они являются то входами, то выходами). Эти выводы данных (D_0—D_7) являются восемью подсоединениями на шину данных системы. Адресная 16-разрядная шина системы будет связана выходами A_0—A_{15}. Шесть других выходов (*SYNC*, *DBIN*, *WAIT*, $\overline{WR}$, *HLDA*, *INTE*) несут сигналы управления и синхронизации всем прочим элементам системы. Наконец, четыре входа (*READY*, *HOLD*, *INT*, *RESET*) являются входами управления, которые воспринимают информацию, поступающую из системы. На рис. 5.2 приведена вся информация по каждому выводу микропроцессора Intel 8080.

Типовая документация содержит также структурную схему микропроцессора. На рис. 5.3, *а* представлена функциональная схема МП Intel 8080, которая содержит внутренние регистры — аккумулятор, пары регистров *B* и *C*, *D* и *E*, *H* и *L*,*указатель стека SP*[1], *регистр состояния (индикатор)*, несколько *регистров временного хранения данных*. Эта схема содержит также *регистр команд, дешифратор команд*, а также *устройство управления и синхронизации*. Наконец, она содержит также АЛУ, его *объединенный индикатор* и блок *десятичного корректора*. Все восемь линий данных, так же как и 16-разрядные адресные выходы, снабжены буферами. Микропроцессор Intel 8080 содержит также несколько внутренних линий управления, цепей данных и шины.

На рис. 5.3, *б* представлены используемые программистом регистры МП Intel 8080. Отметим, что основным является регистр *A* или аккумулятор. Регистры *B* и *C*, *D* и *E*, *H* и *L* являются универсальными. Указатель стека, счетчик команд и индикатор состояния являются специальными регистрами. Пара регистров *HL* может быть использована также в качестве адресного регистра.

Документация содержит разработанные временны́е диаграммы, которые показывают соотношения между входами тактовых импульсов и другими внешними сигналами (синхронизации, записи, адресных выходов, ВВ данных и т. д.) и внутренними операциями. Разработчик дает также указания о способе, по которому микропроцессор используется в случае минимальной системы

[1] От *Stack Pointer* (англ.) — указатель стека. — *Прим. пер.*

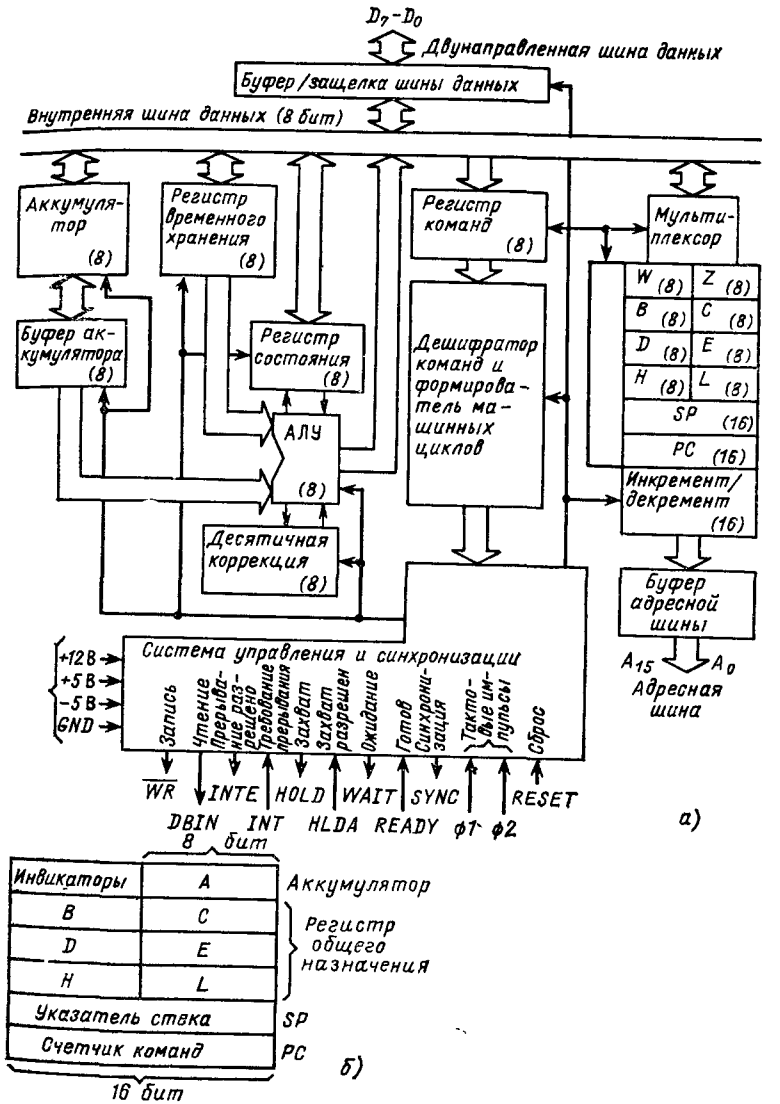


Рис. 5.3. Функциональная схема (архитектура) микропроцессора Intel 8080 (*а*) и регистры, доступные программисту (*б*)

Такая система, основанная на МП Intel 8080, могла бы содержать микропроцессор, генератор тактовых импульсов, устройство управления системой, ПЗУ, ОЗУ и интерфейс портов ВВ.

Документация содержит подробную информацию о системе команд. В табл. 5.1 приведено краткое изложение системы команд микропроцессоров Intel 8080/8085 (Intel 8085 — это улучшенная версия Intel 8080 и имеет приблизительно тот же состав команд; мы будем рассматривать МП Intel 8085 более подробно в гл. 8 и 9).

Рассмотрим первые команды МП Intel 8080/8085 в табл. 5.1. Команда СЛОЖИТЬ с непосредственным переносом символически обозначается мнемоникой ACI фирмы Intel. Код операции команды CE мы можем найти во второй колонке табл. 5.1. В третьей колонке «Число байт» показан объем памяти, необходимый для этой команды. Для нашего случая из таблицы видно, что необходимы 2 байт. Первый будет содержать КОП (CE), а второй — число для сложения (данные или операнд). Колонки «Число тактов» приводят необходимую длительность выполнения команды. Далее расположены основные внешние операции, выполняемые МП. В рассматриваемом случае F означает извлечение команды, R — считывание. Такие краткие выдержки помогают всем желающим программировать в системах, построенных на основе микропроцессора Intel 8080. Кроме того, указания пользователю (также поставляемые конструктором) содержат дополнительные детали использования каждой команды.

Упражнения

5.1. Перечислить по меньшей мере пять типов информации, которая должна содержаться в документации, поставляемой с микропроцессором.

5.2. При виде сверху вывод *1* находится _____ (по ходу против хода) часовой стрелки сразу после метки на корпусе ИС.

5.3. См. рис. 5.2. Выводы МП Intel 8080 D_0—D_7 являются _____ (входами, выходами, двунаправленными), соединенными с шиной данных системы.

5.4. См. рис. 5.2. Какой единственный вывод МП Intel 8080 будет в L-состоянии при операции считывания?

5.5. См. рис. 5.2. Какие уровни напряжения питания МП Intel 8080?

Таблица 5.1. *Сокращенный состав команд микропроцессоров Intel 8080/8085*

Команда	Код	Число байт	Число тактов		Машинный цикл
			8080	8085	
ACI DATA	CE data	2	7	7	FR
ADC REG	1000 1SSS	1	4	4	F
ADC M	8 E	1	7	7	FR
ADD REG	10000 SSS	1	4	4	F
ADD M	86	1	7	7	FR
ADI DATA	C6 data	2	7	7	FR
ANA REG	10100 SSS	1	4	4	F
ANA M	A6	1	7	7	FR
ANI DATA	E6 data	2	7	7	FR
CALL LABEL	CD addr	3	17	17	SRRWW*
CC LABEL	DC addr	3	9/18	11/17	SR·/SRRWW*
CM LABEL	FC addr	3	9/18	11/17	SR·/SRRWW*
CMA	2 F	1	4	4	F
CMC	3 F	1	4	4	F
CMP REG	10111 SSS	1	4	4	F
CMP M	BE	1	7	7	FR
CNC LABEL	D4 addr	3	9/18	11/17	SR·/SRRWW*
CNZ LABEL	C4 addr	3	9/18	11/17	SR·/SRRWW*
CP LABEL	F4 addr	3	9/18	11/17	SR·/SRRWW*
CPI DATA	FE data	2	7	7	FR
CPE LABEL	EC addr	3	9/18	11/17	CR·/SRRWW*
CPO LABEL	E4 addr	3	9/18	11/17	CR·/SRRWW*
CZ LABEL	CC addr	3	9/18	11/17	CR·/SRRWW*
DAA	27	1	4	4	F
DAD RP	00RP 1001	1	10	10	FBB
DCR REG	00SSS 101	1	4	5	F*
DCR M	35	1	10	10	FRW
DCX RP	00RP 1011	1	6	5	S*
DI	F3	1	4	4	F
EI	FB	1	4	4	F
HLT	76	1	5	7	FB
IN PORT	DB data	2	10	10	FRI
INR REG	00SSS 100	1	4	5	F*
INR M	34	1	10	10	FRW
INX RP	00RP 0011	1	6	5	S*
JC LABEL	DA addr	3	7/10	10	FR/FRR+
JM LABEL	FA addr	3	7/10	10	FR/FRR+
JMP LABEL	C3 addr	3	10	10	FRR
JNC LABEL	D2 addr	3	7/10	10	FR/FRR+
JNZ LABEL	C2 addr	3	7/10	10	FR/FRR+
JP LABEL	F2 addr	3	7/10	10	FR/FRR+
JPE LABEL	EA addr	3	7/10	10	FR/FRR+
JPO LABEL	E2 addr	3	7/10	10	FR/FRR+
JZ LABEL	CA addr	3	7/10	10	FR/FRR+

Продолжение табл. 5.1

Команда	Код	Число байт	Число тактов 8080	Число тактов 8085	Машинный цикл
LDA ADDR	3A addr	3	13	13	FRRR
LDAX RP	000 1010	1	7	7	FR
LHLD ADDR	2A addr	3	16	16	FRRR
LXI RP, DATA16	00RP0001 data16	3	10	10	FRR
MOV REG, REG	01DDDSSS	1	4	5	F*
MOV M, REG	01110SSS	1	7	7	FW
MOV REG, M	01DDD110	1	7	7	FR
MVI REG, DATA	00DDD110 data	2	7	7	FR
MVI M, DATA	36 data	2	10	10	FRW
NOP	00	1	4	4	F
ORA REG	10110SSS	1	4	4	F
ORA M	B6	1	7	7	FR
ORI DATA	F6 data	2	7	7	FR
OUT PORT	D3 data	2	10	10	FRO
PCHL	E9	1	6	5	S*
POP RP	11RP0001	1	10	10	FRR
PUCH RP	11RP0101	1	12	11	SWW*
RAL	17	1	4	4	F
RAR	1F	1	4	4	F
RC	D8	1	6/12	5/11	S/SRR*
RET	C9	1	10	10	FRR
RIM (8085)	20	1	4	—	F
RLC	07	1	4	4	F
RM	F8	1	6/12	5/11	S/SRR*
RNC	D0	1	6/12	5/11	S/SRR*
RNZ	C0	1	6/12	5/11	S/SRR*
RP	F0	1	6/12	5/11	S/SRR*
RPE	E8	1	6/12	5/11	S/SRR*
RPO	E0	1	6/12	5/11	S/SRR*
RRC	0F	1	4	4	F
RST N	11 111	1	12	11	SWW*
RZ	C8	1	6/12	5/11	S/SRR*
SBB REG	1001 1SSS	1	4	4	F
SBB M	9E	1	7	7	FR
SBI DATA	DE data	2	7	7	FR
SHLD ADDR	22 addr	3	16	16	FRRWW
SIM (8085)	30	1	4	—	F
SPHL	F9	1	6	5	S*
STA ADDR	32 addr	3	13	13	FRRW
STAX RP	00 000	1	7	7	FW
STC	37	1	4	4	F
SUB REG	10010SSS	1	4	4	F
SUB M	96	1	7	7	FR

Продолжение табл 5.1

Команд	Код	Число байт	Число тактов 8080	Число тактов 8085	Машинный цикл
SUI DATA	D6 data	2	7	7	FR
XCHG	EB	1	4	4	F
XRA REG	10101SSS	1	4	4	F
XRA M	AE	1	7	7	FR
XRI DATA	EE data	2	7	7	FR
XTHL	E3	1	16	18	FRRWW

Типы машинных циклов:

F — извлечение команды длительностью четыре периода тактовых импульсов;
S — извлечение команды длительностью шесть периодов тактовых нмпульсов;
R — считывание из памяти;
I — считывание с УВВ;
W — запись в память;
O — запись в УВВ;
B — неиспользуемая шина;
X — переменная или избираемая двоичная цифра;
DDD — двоичные цифры, идентифицирующие регистр назначения;
SSS — двоичные цифры, идентифицнрующие регистр источника: B-000; C=001; D=010; M=110; E=011; H=100; A=111;
RP — пара регистров: *BC*=00; *HL*=10; *DE*=01; *SP*=11;
* — извлечение команды длительностью пять периодов тактовых импульсов с МП Intel 8080A;
· — дополнительный цикл READ (R) с МП Intel 8080A;
\+ — наиболее длительная последовательность машннного цикла с МП Intel 8080.

5.6. См. рис. 5.3, *б*. Перечислить по крайней мере шесть универсальных регистров МП Intel 8080.

5.7. См. рис. 5.3, *а*. Какова разрядность регистра счетчика команд (*PC*)?

5.8. Индикатор (регистр состояния) тесно связан с _____ (АЛУ, дешифратором команд).

5.9. См. табл. 5.1. Мнемоника команды непосредственного сложения для МП Intel 8080 — ADI, ее КОП _____.

5.10. См. табл. 5.1. Команда ADI требует _____ байт памяти.

5.11. См. табл. 5.1. В ходе выполнения команды ADI МП Intel 8080 выполняет один цикл извлечения и один цикл _____ (записи, считывания).

Решения

5.1. Архитектура микропроцессора, схемы выводов ИС и назначение каждого из них, временны́е диаграммы, состав команд и типовые системы, использующие микропроцессор. 5.2. Против хода. 5.3. Двунаправленными. 5.4. См. рис. 5.2, *б*. Активизируется вывод $\overline{WR}$ для сигнализации в устройство памяти о том, что данные записаны в память. 5.5. —5 В, +5 В, +12 В. 5.6. *B*, *C*, *D*, *E*, *H* и *L*. Обычно говорят, хотя не всегда категорично, что аккумулятор *A* также является универсальным регистром. 5.7. 16 бит. 5.8. АЛУ. 5.9. СБН. 5.10. 2. 5.11. Считывания.

5.2. СХЕМА И НАЗНАЧЕНИЕ ВЫВОДОВ

Рассмотрим теперь более сложный микропроцессор, обладающий большинством уже рассмотренных свойств. Однако с целью некоторого упрощения мы пренебрегаем некоторыми сигналами управления выводом информации из МП.

Схема выводов такого МП представлена на рис. 5.4. Микропроцессор заключен в DIP-корпус с 40 двухрядными выводами. Эта ИС питается напряжением +5 В по выводам *1* и *2*, что соответствует новой концепции использования для питания МП единственного источника питания +5 В.

Выводы X_1 и X_2 вверху справа предназначены для подсоединения кристалла управления частотой ГТИ МП. Для наиболее распространенных устройств характерно наличие ГТИ на кристалле МП, тогда как для более старых устройств был необходим внешний ГТИ. Выход *CLK* (вывод *38*) предназначен для выдачи сигналов ГТИ в систему. Частота сигнала на выводе *38* (*CLK*), очевидно, подчинена частоте внутреннего ГТИ.

Адресная шина системы будет подсоединена к выводам ИС A_0—A_{15} (рис. 5.4). Эти 16 адресных линий (может быть и другое количество) могут обеспечить доступ к 65 536 (2^{16}) ячейкам памяти или/и портам ВВ.

Поток данных и команд от микропроцессора и в него обеспечивается выводами D_0—D_7 на ИС рис. 5.4. Эти выводы (*21*—*28*) двунаправленные, т. е. являются то выходами, то входами. Кроме того, обычно они могут переводиться в третье состояние (высокого сопротивления).

Вывод *30* является выходом управления записью. Сигнал L-уровня на выходе $\overline{WR}$ указывает, что данные, имеющиеся на шине данных, должны быть записаны в область

памяти или выбранное УВВ. Выход управления считыванием $\overline{RD}$ (вывод *31*) активизируется L-сигналом, который указывает, что избранные места в памяти или УВВ должны быть считаны.

Результатом активизации входа сброса (рис. 5.4) является остановка работы МП по текущей программе и переход к подпрограмме сброса. Сигнал L-уровня на входе $\overline{RESET}$

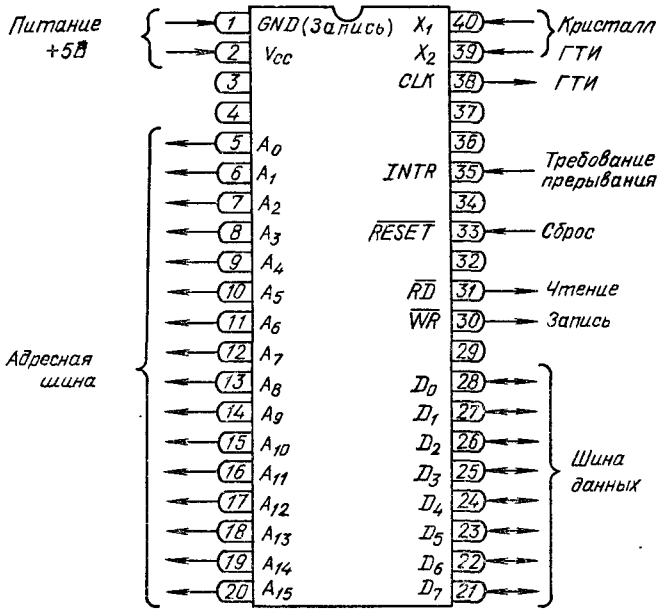


Рис. 5.4. Схема выводов типового микропроцессора

МП сбрасывает счетчик команд до заранее предопределенного адреса, например 0000H. Другие внутренние регистры МП могут быть также сброшены или их содержимое изменяется в течение операции сброса. Когда вход $\overline{RESET}$ переходит в состояние HIGH, МП начинает выполнение команд с нового адреса памяти, т. е. с адреса 0000H в данном случае (или с другого заранее предопределенного адреса памяти).

Этот адрес соответствует началу подпрограммы новой инициализации системы, содержащейся обычно в ПЗУ.

Большинство микропроцессоров находятся в фазе с ГТИ, следовательно, они являются синхронными. Вход $\overline{RESET}$ МП асинхронный и может вмешаться и приостановить наполовину выполненную команду.

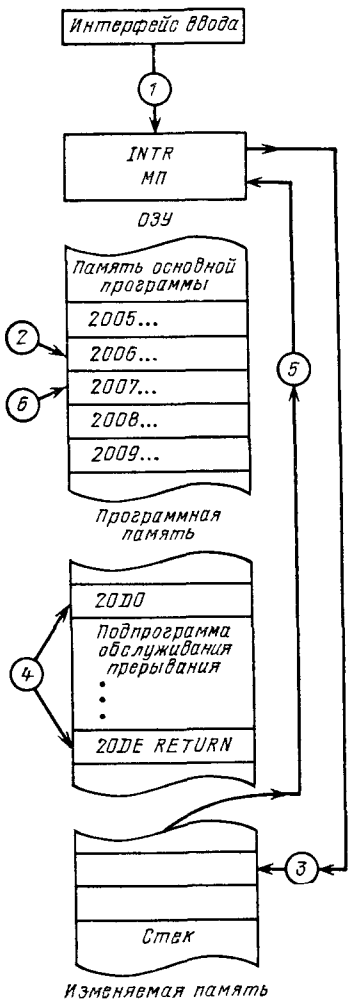


Рис. 5.5. Этапы отработки требования прерывания в типовом МП

Вход требования прерывания помещен на вывод *35*. Вход *INTR* отвечает на Н-сигнал внешнего устройства. Рисунок 5.5 нам поможет понять, что происходит, когда МП реагирует на такой сигнал. Предположим, что устройство интерфейса ввода на рис. 5.5 загружено 8 бит параллельных данных, готовых для передачи в МП: процесс может быть продолжен в порядке, показанном на рис. 5.5.

1. Интерфейс ввода выдает сигнал требования прерывания в направлении МП (см. *1* в кружке на рис. 5.5).

2. Микропроцессор завершает выполнение текущей команды, находящейся в памяти по адресу 2006Н.

3. Поскольку управление должно обеспечить последующее обращение к команде по адресу 2007Н, содержимое счетчика команд (именно 2007Н) и содержимое большинства регистров МП помещается в специальную зону ОЗУ, называемую *стеком*. Это содержимое будет позже извлечено в определенном порядке в регистры МП и в счетчик команд.

4. А сейчас МП разветвляется в предопределенный адрес памяти и начинает выполнение подпрограммы обслуживания прерывания (в нашем примере 20DOH). Микропроцессор выполняет тогда команды подпрограммы, которые всегда в нашем примере обеспечивают выполнение операций ввода. По адресу 20DEH МП находит конец этой подпрограммы обслуживания и получает приказ вернуться в основную программу.

5. Перед возвращением в основную программу данные регистров и счетчик команд, помещенные в стек, возвращаются в МП.

6. Теперь счетчик команд отсылает МП в память по адресу 2007Н, т. е. в основную программу, и нормальное выполнение ее продолжается.

Прерывание является очень нужным способом, позволяющим периферии вмешаться и заставить МП выполнять требуемую операцию почти сразу. Многие микропроцессоры обладают одним или несколькими прерываниями. Входы прерывания могут быть названы также сбросами, новым запуском, маскируемыми прерываниями или сетками.

Упражнения

5.12. Обратиться к рис. 5.4. Этот МП питается напряжением _____, обладает _____, (внутренним, внешним) ГТИ, а его выводы *39*, *40* подсоединены на _____ (переменное напряжение, кристалл ГТИ).

5.13. См. рис. 5.4. Стрелка указывает от вывода *5*, потому что он является _____ (выходом, входом, двунаправленным).

5.14. См. рис. 5.4. Уходящая с вывода *30* линия рассматривается как часть шины _____ (данных, адреса, управления).

5.15. См. рис. 5.4 L-сигнал на выводе *31* _____ (активизирует, сбрасывает) вход сброса, что заставляет МП сбросить _____ (счетчик команд, память) в 0000Н.

5.16. См. рис. 5.4. Вывод сброса является _____ (синхронным, асинхронным) входом в том смысле, что сигналы на нем не являются синхронными с сигналами ГТИ.

5.17. См. рис. 5.4. Здесь передача данных осуществляется параллельными словами длиной _____ бит.

5.18. См. рис. 5.4. Требование прерывания является _____ (входным, выходным) сигналом МП.

5.19. Требование прерывания приводит МП к ветвлению и выполнению _____ прерывания, находящейся в памяти, а затем к возврату в основную программу.

5.20. См. рис. 5.4. Микропроцессор заключается в _____, т. е. имеет двустороннюю упаковку 40 выводов.

5.12. +5 В; внутренним; кристалл ГТИ. **5.13.** Выходом. **5.14.** Управления. **5.15.** Активизирует; счетчик команд. Линия над $\overline{RESET}$ показывает, что речь идет об активном L-сигнале. **5.16.** Асинхронным. **5.17.** & **5.18.** Входным. **5.19.** Подпрограммы обслуживания **5.20.** Корпус с DIP-выводами.

5.3. АРХИТЕКТУРА МИКРОПРОЦЕССОРА

Практически все микропроцессоры содержат по меньшей мере следующие элементы: АЛУ; несколько регистров; счетчик команд; систему декодирования команд; секцию управления и синхронизации; буферы и защелки; внутренние шины цепей управления; несколько входов и выходов управления.

Кроме того, кристалл микропроцессора может также содержать функциональные устройства: ПЗУ; ОЗУ; ряд портов ВВ; внутренние цепи ГТИ — часов, программируемый таймер; систему выбора приоритета прерываний; логику интерфейса последовательно-параллельных взаимодействий при ВВ; логическое управление прямым доступом к памяти (ПДП).

В предыдущем параграфе мы изучили схему выводов и их назначение у типового микропроцессора. Здесь рассмотрим архитектуру того же МП, приведенную на рис. 5.6.

Начнем с внутренних соединений. Микропроцессор обладает восемью двунаправленными связями с шиной данных, по которым они выводятся на внутреннюю шину. Слева от МП показаны 16 выходов на адресную шину с буферами/защелками на внутренней адресной шине. Выходы управления показаны внизу слева; это линии записи, чтения и ГТИ. Внизу справа от МП принимаются два сигнала по линиям сброса и требования прерывания. У нашего МП есть внутренняя цепь ГТИ, и ему нужен только один внешний кристалл (или в некоторых случаях — одна емкость) для запуска МП. Наконец, этот микропроцессор запитан от единственного источника напряжением +5 В.

Функциональные назначения большинства устройств, составляющих этот микропроцессор, были уже изучены, мы их напомним только коротко.

Регистр команд: это устройство является 8-разрядным регистром и содержит первый байт команды (ее КОП).

Дешифратор команд: это устройство интерпретирует

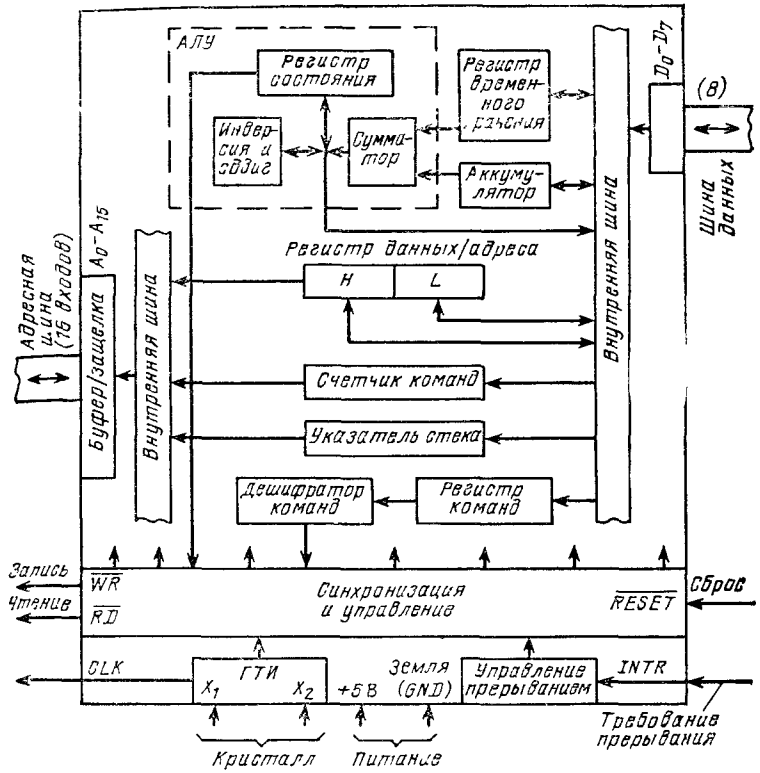


Рис 5.6 Функциональная схема (архитектура) типового МП

(декодирует) содержимое регистра команд, определяет микропрограмму для выполнения нужной из всего множества команд и последовательно вводит в действие секцию управления.

Арифметико-логическое устройство (АЛУ): это устройство выполняет операции арифметические, логические и сдвига, в результате которых устанавливает регистр состояния (индикаторы). Результаты помещаются в аккумулятор, связанный с внутренней шиной. Часто внутренние регистры и аккумулятор рассматриваются как часть АЛУ. Условия индикатора передаются в устройство управления и синхронизации.

Аккумулятор: это устройство является универсальным 8-разрядным регистром, где концентрируется большинство

результатов выполнения команд — арифметических, логических, загрузки, запоминания результата, ввода/вывода.

Счетчик команд: это устройство является разновидностью 16-разрядной памяти, которое постоянно указывает на следующую выполняемую команду. Оно всегда содержит 16-разрядный адрес. Счетчик может быть инкрементирован или сброшен устройством управления или изменен командой передачи данных.

Устройство управления и синхронизации: это устройство получает сигналы дешифратора команд для определения природы выполняемой команды. Оно получает также информацию от регистра состояния в случае условного перехода. Сигналы управления и синхронизации передаются во все устройства системы для координации выполнения команд, и, наконец, оно вырабатывает сигналы управления внешними устройствами (ОЗУ, ПЗУ, УВВ и т. д.).

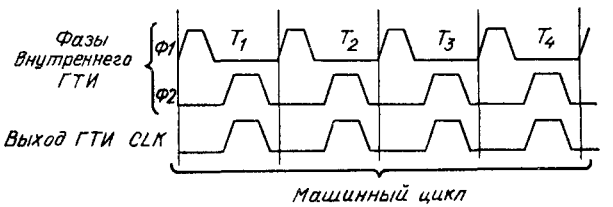


Рис. 5.7. Диаграмма тактовых импульсов

Регистр состояния: элементарный микропроцессор на рис. 5.6 содержит в своем регистре состояния индикаторы только нуля и переноса.

Новые дополнительные устройства этого микропроцессора содержат внутренний ГТИ, систему управления прерываниями, указатель стека и универсальный регистр данных/адреса.

Цепь внутреннего ГТИ с внешним кристаллом генерирует сигналы, аналогичные показанным на рис. 5.7. Генератор тактовых импульсов выдает сигналы с двумя различными фазами для использования их внутри микропроцессора. Выход *CLK* МП имеет сигнал, идентичный сигналу *Ф2*, и служит для синхронизации событий во всей системе. На рис. 5.7 сигналы ГТИ разделены на периоды T (T_1, T_2 и т. д.), определенное число которых формирует машинный цикл. Периоды T имеют всегда постоянную длительность, тогда как длительность машинного цикла может меняться

(на рис. 5.7 приведен машинный цикл, включающий в себя четыре периода T).

Соотношения между периодами T (T_1, T_2 и т. д.) и машинными циклами (M_1, M_2 и т. д.) приведены на рис. 5.8, *а*. Машинные циклы определяют функционирование МП: ечи-

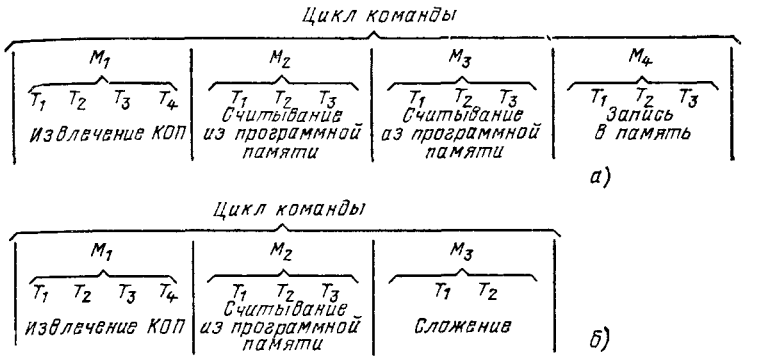


Рис. 5.8. Синхронизация в машинном цикле команды STORE (*а*) и ADD (*б*) с непосредственной адресацией. (Сложить содержимое аккумулятора с содержимым следующего непосредственно за КОП байта программной памяти и сумму сохранить в аккумуляторе)

тывание, запись, извлечение или выполнение. В нашем МП типовыми циклами являются: считывание (извлечение КОП); считывание из памяти или УВВ; запись в память или УВВ; выполнение внутренней операции.

Команда STORE (разместить данные) разделена на четыре цикла (M_1—M_4), как показано на рис. 5.8, *а*: во-первых, извлечение КОП (считывание); во-вторых, считывание из программной памяти; в-третьих, еще одно считывание из программной памяти; в-четвертых, операция записи в память. Сочетание этих четырех действий (считывание, считывание, считывание, запись) составляет цикл команды. Отметим, что все машинные циклы выполняются не одинаковое время. Первый M_1 занимает 4 периода T, тогда как все остальные занимают только по 3. Полный цикл команды занимает 13 периодов.

На рис. 5.8, *б* приведен пример выполнения во времени команды непосредственного сложения. Первым машинным циклом M_1 является извлечение КОП (считывание). В течение периода T_4 машинного цикла МП декодирует команду сложения ADD и решает, что для ее завершения ему на-

до два дополнительных цикла. Во втором цикле M_2 считывается следующий байт из памяти, которым является операнд, и в течение цикла M_3 выполняет операцию ADD в АЛУ. Заметим на рис. 5.8, что машинные циклы занимают не одинаковое время, но не больше, чем цикл команды.

Устройство управления прерываниями (см. рис. 5.6) принимает сигнал прерывания с внешнего устройства через вход *INTR*. Оно управляет по этому сигналу МП в соответствии с ранее рассмотренными этапами (см. § 5.2). Таким образом, МП ветвится в подпрограмму обслуживания прерываний, которая отвечает на требование прерывания, и по окончании ее МП возвращается в основную программу.

Указатель стека подобен счетчику команд в том смысле, что в нем содержится адрес, который он инкрементирует или декрементирует, он может быть также загружен новым адресом. Емкость указателя стека составляет 16 бит, т. е. он может посылать адрес по 16 линиям. Мы увидим использование указателя стека более подробно в § 5.5.

Регистр данных/адреса (см. рис. 5.6) составляется из двух 8-разрядных регистров, которые могут быть использованы вместе или раздельно; они обозначены *H* и *L* соответственно старшему (HIGH) и младшему (LOW) байтам.

Когда эти два регистра используются вместе, мы обращаемся к *паре HL*. Регистры *H* и *L* являются универсальными подобно аккумулятору в том смысле, что они могут быть инкрементированы, декрементированы, загружены данными и служить источником данных. Пара *HL* может служить также адресным регистром и хранить адрес назначения в ходе размещения данных в памяти или адресом источника в ходе загрузки аккумулятора. Таким образом, регистры *H* и *L* могут быть использованы для размещения данных и манипуляций с ними или как *указатель адреса.* Подробнее использование регистра данных/адреса показано в § 5.4. Некоторые микропроцессоры обладают специальным регистром — *счетчиком данных,* который указывает на ячейку памяти (он используется подобно паре регистров *HL* нашего МП).

Упражнения

5.21. Перечислить по меньшей мере шесть функциональных устройств, содержащихся на кристалле МП.

5.22. Обратиться к рис. 5.6. Какое функциональное устройство расположено между шиной данных МП и внешней шиной данных?

5.23. См. рис. 5.6. По каким трем выходным линиям обеспечивается синхронизация системы?

5.24. Счетчик команд является 16-разрядной областью памяти, предназначенной для хранения _____ (адресов, данных).

5.25. См. рис. 5.6. Сигналы управления чтением и записью посылаются устройством _____ микропроцессора.

5.26. Назвать четыре типа машинных циклов, рассмотренных в этом параграфе.

5.27. См. рис. 5.8. У какой команды — STORE или ADD более короткое время выполнения?

5.28. Если период *T* длится 500 нс, какой будет длительность команды ADD на рис. 5.8?

5.29. См. рис. 5.6. Активизация входа *INTR* заставляет устройство _____ привести микропроцессор к ветвлению в подпрограмму обслуживания прерываний.

5.30. Указатель стека является аналогом счетчика команд в том смысле, что он содержит _____ (адрес, команду) длиной _____ бит.

5.31. См. рис. 5.6. Когда объединяются два регистра адреса/данных и используются как область памяти длиной 16 бит, они составляют пару регистров_____ .

5.32. См. рис. 5.6. Два регистра памяти, названные *H* и *L*, являются регистрами _____.

5.33. См. рис. 5.6. Регистры *H* и *L* могут быть использованы для размещения и обработки данных, а в форме пары *HL* — как _____ (указатель адреса, избиратель данных).

Решения

5.21. Арифметико-логическое устройство, несколько регистров, счетчик команд, дешифратор команд, устройство управления и синхронизации, шинные буферы, внутренние шины и несколько входов и выходов. **5.22.** Шинные буферы/защелки. **5.23.** Выводы считывания, записи и синхронизации. **5.24.** Адресов. **5.25.** Управления и синхронизации. **5.26.** Считывание и извлечение КОП, считывание из памяти или УВВ, запись в память, выполнение внутренних операций. **5.27.** ADD. **5.28.** Для выполнения команды ADD нужно девять периодов *T*, следовательно, $9\times500=4500$ нс$=4{,}5$ мкс. **5.29.** Управления прерываниями. **5.30.** Адрес; 16. **5.31.** *HL*. **5.32.** Адреса/данных. **5.33.** Указатель адреса.

5.4. ИСПОЛЬЗОВАНИЕ РЕГИСТРА АДРЕСА/ДАННЫХ

Использование пары регистров *HL* в качестве указателя адреса является интересным свойством нашего типового МП. Обычно рассматривают ее использование в качестве указателя адреса, когда она временно берет на себя роль

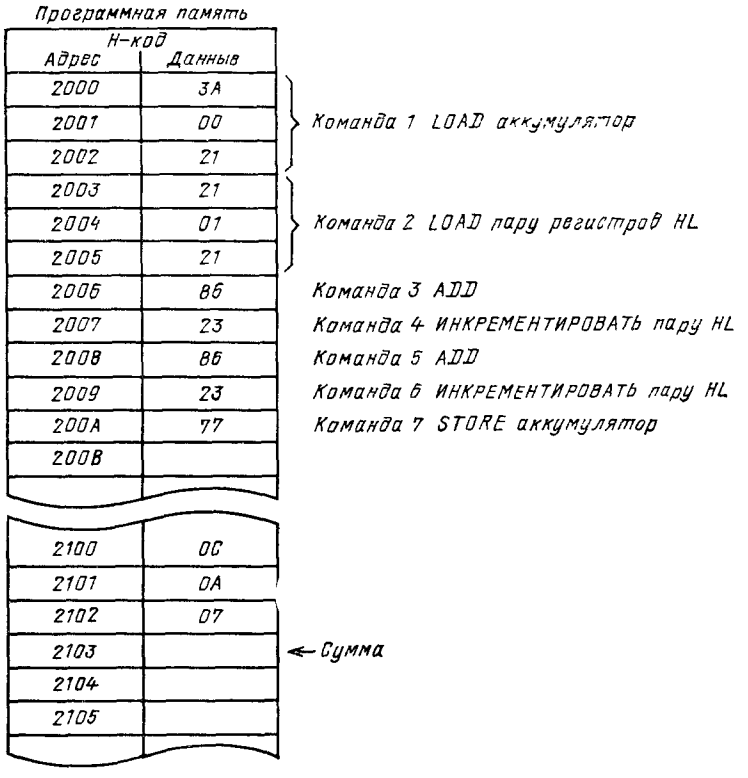


Рис. 5 9. Воображаемая память и команды в примере сложения

основного счетчика команд, указывая адрес ячейки памяти или УВВ. Многие широко распространенные МП (например, Intel 8080/8085, Z 80) содержат регистры такого типа. Регистры адреса/данных в рассматриваемом типовом МП называются также парой *HL*-регистров, регистром адреса, счетчиком данных или указателем адреса.

Рассмотрим простую задачу сложения содержимого трех последовательных ячеек памяти с размещением суммы в следующей ячейке памяти (выполнение ее показано на рис. 5.9). Программа загружена в ячейки памяти 2000H—200AH, а три слагаемых (OCH+OAH+07H) — в ячейки памяти 2100H—2102H. Программа содержит 6 команд, записанных справа на рис. 5.9. Не следует забывать, что текущая сумма будет всегда помещаться в аккумулятор, содержащий вначале первое слагаемое (OCH).

Команда *1* имеет КОП ЗАН (рис. 5.9) и приказывает МП ЗАГРУЗИТЬ (LOAD) в аккумулятор содержимое ячейки памяти 2100H. Выполнение этой команды прямой загрузки аккумулятора показано на рис. 5.10, *а*. После выполнения команды аккумулятор будет содержать первое слагаемое (OCH).

Команда *2* приказывает МП загрузить (LOAD) 2101H в пару регистров *HL*, емкость которых 16 бит. Это число представляет собой адрес памяти данных. Более точно команду *2* можно сформулировать так: загрузить пару регистров *HL* непосредственно следующими за КОП данными, ее выполнение приведено на рис. 5.10, *б*. Заметим, что содержимое первой ячейки памяти (2004H) затружается в младший байт *L*, следующей за ней — в старший байт *H* пары регистров *HL*.

Команда *3* приказывает МП выполнить операцию сложить (ADD) содержимое аккумулятора с содержимым ячейки памяти, адрес которой содержится в паре регистров *HL*. Ее выполнение приведено на рис. 5.10, *в* (команда ADD). Пара регистров *HL* указывает на ячейку памяти 2101H, и АЛУ складывает свое содержимое ($0000\ 1010_2$) с содержимым аккумулятора ($0000\ 1100_2$), что дает сумму ($0001\ 0110_2$), помещаемую в аккумулятор.

Команда *4* приказывает МП инкрементировать (увеличить на 1) содержимое пары регистров *HL* (см. рис. 5.10, *г*). Заметим, что изменился только младший байт пары регистров *HL*.

Команда *5* снова приказывает МП сложить (ADD) содержимое аккумулятора и ячейки памяти с адресом 2102H, на которую указывает пара регистров *HL* (см. рис. 5.10, *д*).

Оба содержимых складываются, что дает сумму ($0001\ 1101_2$), помещаемую в аккумулятор.

По команде *6* содержимое пары регистров *HL* снова инкрементируется (см. рис. 5.10, *е*).

Команда *7* приказывает МП поместить (STORE) содер-

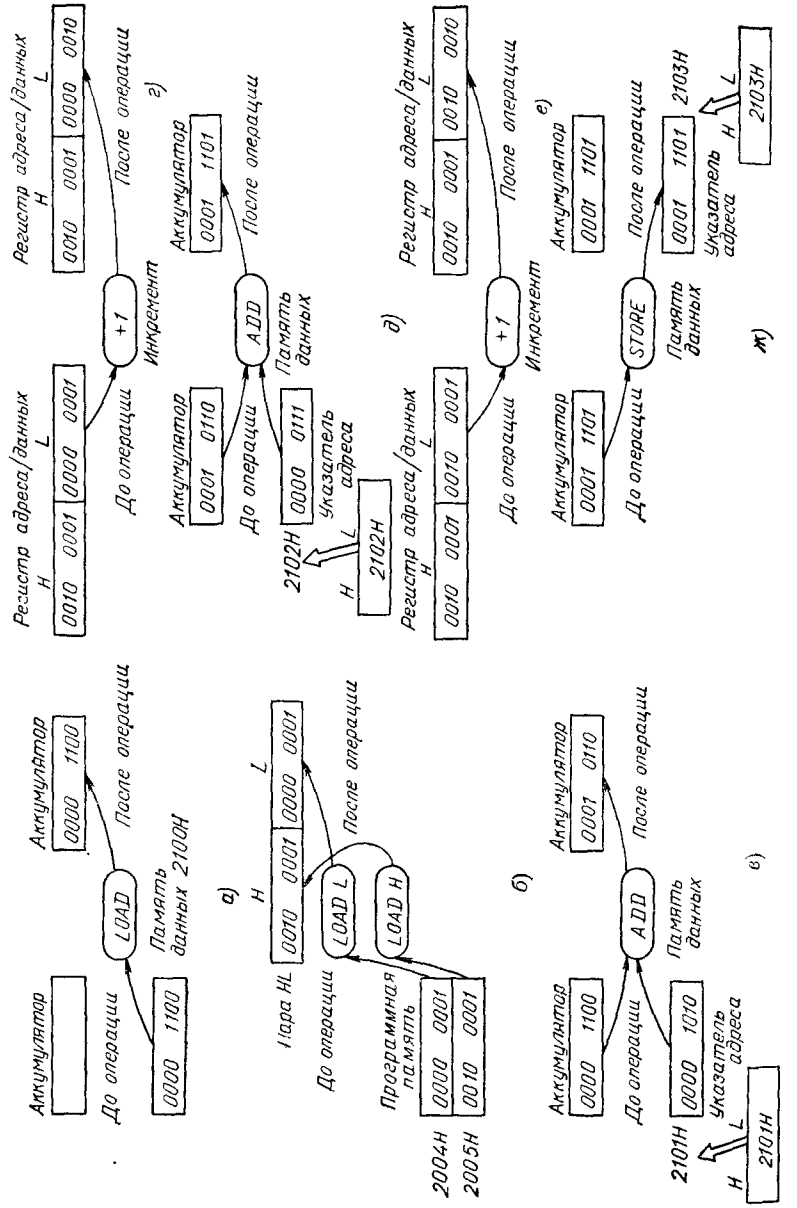


Рис. 5.10. Выполнение операций в соответствии с программой рис. 5.9: *а* — команда *1* загрузки аккумулятора; *б* — команда *2* загрузки пары регистров *HL*; *в* — команда *3* сложения, *г* — команда *4* инкремента пары регистров *HL*; *д* — команда *5* сложения; *е* — команда *6* инкремента пары регистров *HL*; *ж* — команда *7* размещения в памяти содержимого аккумулятора

жимое аккумулятора, т. е. окончательную сумму ($0001\ 1101_2$) в ячейку памяти, на которую указывает пара регистров *HL* (см. рис. 5.10, *ж*), т. е. по адресу 2103Н.

Команды *3*, *5*, *7*, взаимодействующие с парой регистров *HL* как с указателем адреса, используют косвенно-регистровый способ адресации. Его мы изучим в следующей главе.

Упражнения

5.34. Счетчик _____ (данных, команд) — одно из названий адресного регистра.

5.35. См. рис. 5.9. Какой будет шестнадцатеричная сумма, переданная в ячейку памяти 2103Н после выполнения программы?

5.36. См. рис. 5.9. и 5.10. Аккумулятор всегда содержит _____ (сумму, команду), тогда как пара регистров *HL* содержит _____ (адрес, номер команды).

5.37. См. рис. 5.9. Какой способ адресации используется, когда пара регистров *HL* играет роль указателя адреса?

5.38. См. рис. 5.11. Каково содержимое (в Н-коде) аккумулятора и пары регистров *HL* в начале программы?

5.39. См. рис. 5.11. Каково содержимое аккумулятора после выполнения команды *1*?

5.40. Каково содержимое пары регистров *HL* после выполнения команды *2*?

5.41. См. рис. 5.11. Каково содержимое пары регистров *HL* после выполнения команды *3*?

5.42. Рассмотреть команду *5* на рис. 5.11. Где поместится содержимое аккумулятора после выполнения команды поместить?

Решения

5.34. Данных. **5.35.** ОСН+ОАН+07Н=1DH ($12+10+7=29_{10}$, т. е. IDH). **5.36.** Текущую сумму; адрес. **5.37.** Косвенная регистровая адресация. **5.38.** Аккумулятор: $0000\ 1111_2$ или OFH; пара *HL*: $0010\ 0001\ 0000\ 0010_2$ или 2102Н. **5.39.** См. рис. 5.12 **5.40.** 2102+1=2103Н. **5.41.** См. рис. 5.13. **5.42.** 2103+1=2104Н. **5.43.** В ячейке памяти 2104Н.

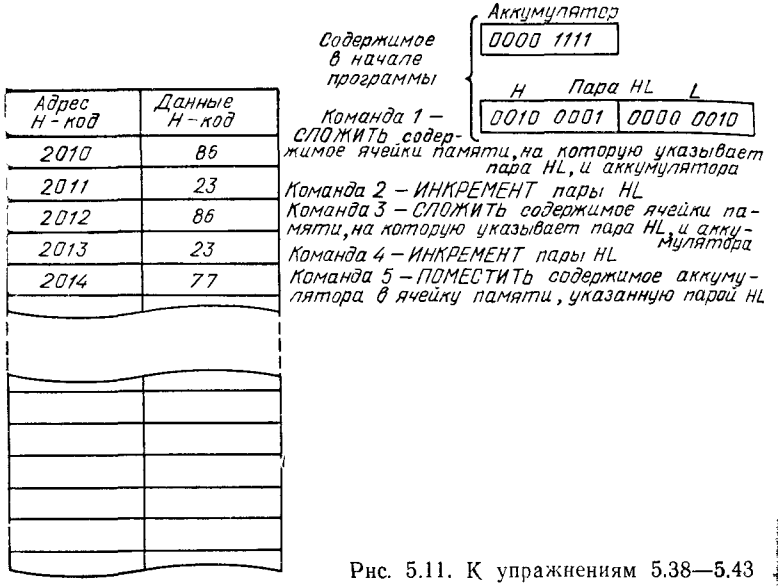


Рис. 5.11. К упражнениям 5.38—5.43

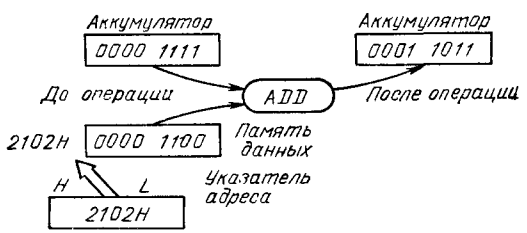


Рис. 5 12. К упражнению 5 39

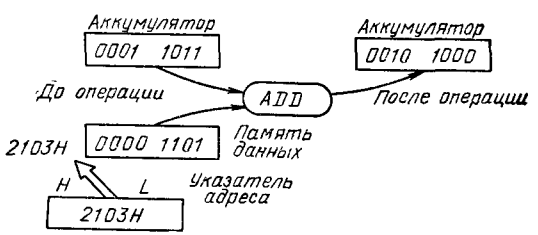


Рис. 5.13. К упражнению 5 41

5.5. ИСПОЛЬЗОВАНИЕ УКАЗАТЕЛЯ СТЕКА

Наш типовой микропроцессор содержит указатель стека — специализированный 16-разрядный регистр-счетчик, содержимым которого всегда является адрес. Этот адрес принадлежит особой группе ячеек памяти данных, которая называется *стеком*. В некоторых МП стек может быть составлен из группы физически локализованных на кристалле МП ячеек памяти. Мы видели уже (см. § 5.2), что когда микропроцессором выполнялась подпрограмма обслуживания прерывания, текущие данные во всех регистрах МП должны были временно сохраняться. Эта сохранность обеспечена стеком. А когда подпрограмма полностью выполнена, содержимое счетчика команд должно быть сохранено таким образом, чтобы МП мог возвратиться в соответствующее место в программной памяти. Зона временной памяти является стеком. Напомним, что подпрограмма является короткой, часто используемой, специализированной программой (например, умножить).

Стек типового микропроцессора будет содержаться в ОЗУ, и его положение определяется программистом. Указатель стека загружается старшим адресом, представляющим собой вершину стека (рис. 5.14). В этом случае указатель

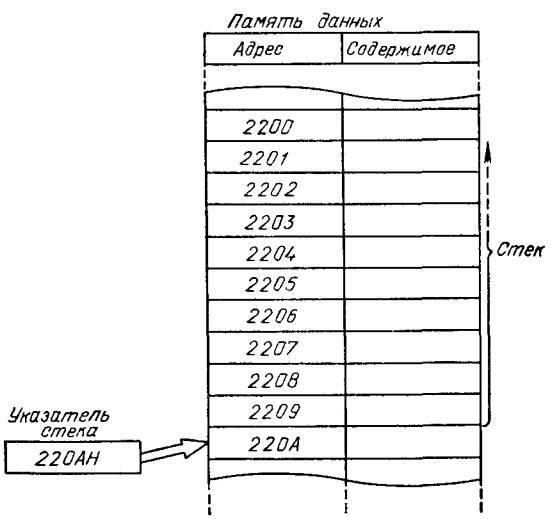


Рис. 5.14. Расположение стека в ОЗУ

стека содержит 220АН, что на единицу старше первой ячейки памяти стека 2209Н.

Данные можно записать в стек, используя команды PUSH (поместить) или CALL (вызвать). Они могут быть считаны из стека по командам POP (извлечь) или RETURN (возврат). Стек функционирует как память с последовательным доступом по типу: данные, поступившие последними, извлекаются первыми (тип LIFO от *Last In — First Out* — последний входит — первый выходит, или FILO от *First In — Last Out* — первый входит — последний выходит).

Команда загрузки в стек (PUSH) приводит к результату, показанному на рис. 5.15, *а*. Содержимое пары регистров *HL* помещается в стек. Отметим, что двухбайтовая пара регистров *HL* должна быть размещена в двух ячейках памяти стека. Последовательность событий может быть описана в соответствии с номерами, показанными в кружках на рис. 5.15.

1. Указатель стека МП декрементируется от 220АН до 2209Н.

2. Указатель стека показывает на ячейку памяти 2209Н по адресной шине и старший байт ($0000\ 0000_2$) помещается в стек.

3. Указатель стека снова декрементируется от 2209Н до 2208Н.

4. Указатель стека указывает на ячейку памяти 2208Н (по адресной шине системы) и младший байт из регистра данных ($0000\ 1111_2$) загружается в стек.

На рис. 5.15, *б* показано выполнение другой операции загрузки. На этот раз в стек загружается содержимое аккумулятора и регистра состояния. Проследим снова за событиями, отмеченными цифрами в кружках.

5. До операции указатель стека указывает на последнюю ячейку стека. Ее называют вершиной стека. Затем указатель стека декрементируется до 2207Н.

6. Указатель стека указывает на ячейку памяти 2207Н, и содержимое аккумулятора ($0101\ 0101_2$) загружается в стек по этому адресу.

7. Указатель стека декрементируется от 2207Н до 2206Н.

8. Указатель стека указывает на ячейку памяти 2206Н. Содержимое регистра состояния ($1111\ 1111_2$) загружается по этому адресу.

Стек может продолжать расти, пока длится процесс загрузки в него (на рис. 5.14 показано, что стек растет вверх).

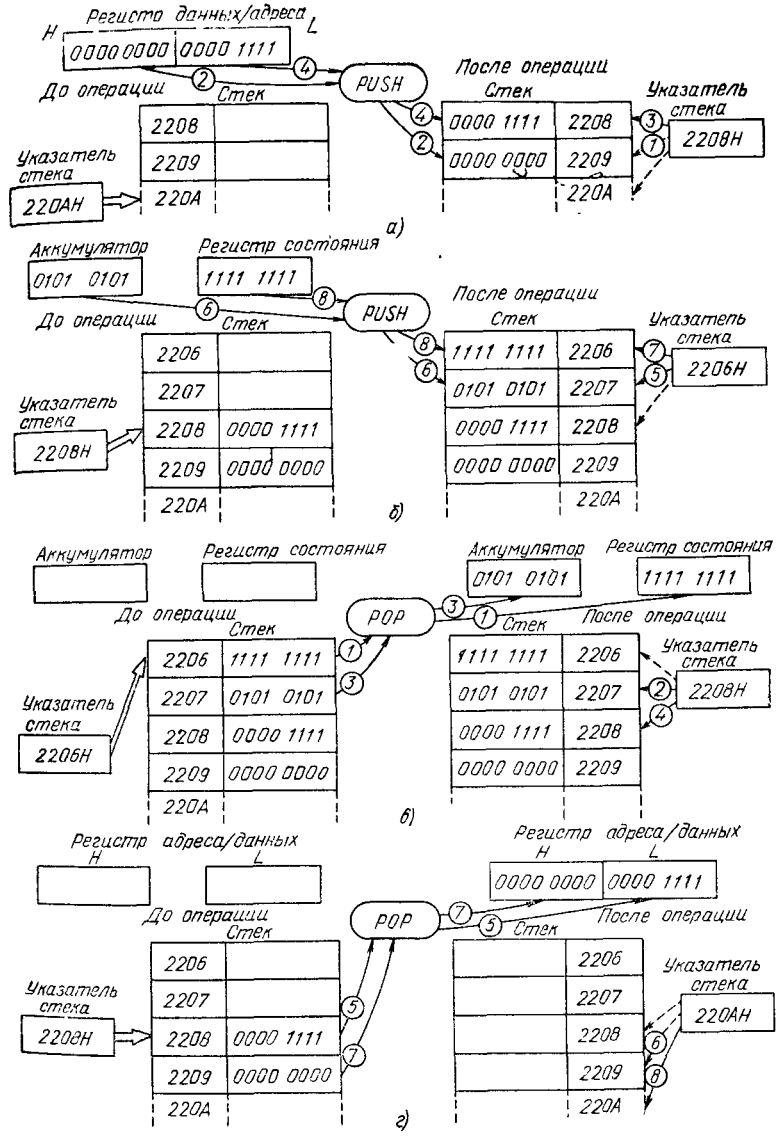


Рис. 5.15. Операции со стеком:

а — помещение в стек содержимого регистра адреса/данных; *б* — помещение в стек содержимого аккумулятора и регистра состояния; *в* — извлечение из стека содержимого аккумулятора и регистра состояния; *г* — извлечение из стека содержимого регистра адреса/данных

Стек не имеет ограничений, за исключением тех, которы обусловлены наличием других программ в ОЗУ.

Обычно каждой команде загрузки в стек (PUSH) позж будет соответствовать команда извлечения из стека (POP) по которой данные берутся из вершины стека. Поскольку стек является памятью типа LIFO (последний вошел — первый вышел), данные должны извлекаться из стека в порядке, обратном загрузке. На рис. 5.15, *в* и *г* подробно показаны операции извлечения из стека.

Рассмотрим команду POP на рис. 5.15, *в*. Аккумулятор и регистр состояния свободны до операции извлечения из стека. Следуем последовательности, указанной цифрами в кружках.

1. Указатель стека указывает на вершину стека, т. е. на адрес 2206H. Содержимое регистра состояния ($1111\ 1111_2$) извлечено из стека и переслано в АЛУ.
2. Указатель стека инкрементирован с 2206H до 2207H.
3. Указатель стека указывает на адрес 2007H стека. Вершина стека извлекается, и ее содержимое пересылается в аккумулятор АЛУ.
4. Указатель стека инкрементирован до 2208H и указывает теперь на следующий адрес извлечения из стека.

Содержимое аккумулятора и регистра состояния было восстановлено до тех значений, которые были до операции PUSH, показанной на рис. 5.15, *б*.

Затем (на рис. 5.16, *г*) содержимое регистра адреса/данных в свою очередь извлекается из стека. Снова последуем согласно заключенным в кружки цифрам:

5. Указатель стека указывает на вершину стека (адрес 2208H). Содержимое этой ячейки памяти стека извлекается и пересылается в младший байт пары регистров *HL*.
6. Указатель стека инкрементируется до 2209H.
7. Указатель стека показывает на вершину стека, т. е. теперь адрес 2209H, содержимое которого ($0000\ 0000_2$), передается в старший байт пары регистров *HL*.
8. Указатель стека инкрементируется от 2209H до 220AH для последующей операции загрузки в стек (PUSH) или извлечения из стека (POP).

Извлечение данных из стека и их восстановление в регистре адреса/данных является действием, обратным операции загрузки в стек (PUSH), выполненной на рис. 5.15, *а*. Команды PUSH и POP используются всегда совместно, однако между ними располагаются другие команды, которые меняют данные, содержащиеся в регистрах МП.

Наш типовой МП загружает в стек и извлекает содержимое пары регистров. Некоторые МП осуществляют эти операции только с однобайтовыми регистрами, единственной командой. В других МП указатель стека может указывать на пустую ячейку памяти по ближайшему большему по отношению к вершине стека адресу вместо указания на вершину стека, как в предыдущем случае. При этих условиях имеется возможность сохранить в памяти то, что программист загрузил в начале (адрес 220AH в нашем случае) в указатель стека для определения его адреса.

Упражнения

5.44. Указатель стека на рис. 5.6 является специализированным регистром-счетчиком с объемом памяти 16 бит, который всегда содержит _____ (адрес, КОП команды).

5.45. Стек является областью памяти, обрабатываемой последовательно по типу_____.

5.46. В информационной технике LIFO означает _____.

5.47. В типовом микропроцессоре адрес стека в ОЗУ определяет _____ (приемник, программист).

5.48. См. рис. 5.14. Адресная ячейка _____ (шестнадцатеричное значение) является первой обрабатываемой ячейкой при команде PUSH или CALL.

5.49. Загрузка в стек является операцией _____ (записи, чтения) по принципу LIFO.

5.50. См. рис. 5.16, *а*. Этап *1* соответствует декременту _____ от 0009H до 0008H.

5.51. См. рис. 5.16, *а*. Этап *2* соответствует _____ (загрузке в стек, извлечению из стека) старшего байта регистра адреса/данных.

5.52. См. рис. 5.16, *б*. Этап *4* соответствует загрузке в стек младшего байта регистра адреса/данных по адресу _____ (шестнадцатеричное значение).

5.53. См. рис. 5.16, *б*. До операции указатель стека указывает на _____ (вершину, глубину) стека, т. е. адрес 0007H.

5.54. См. рис. 5.16, *б*. Этап *5* соответствует восстановлению _____ (старшего, младшего) байта регистра адреса/данных с содержимым _____ (указателя стека, вершины стека).

5.55. См. рис. 5.16, *б*. Этап *6* соответствует _____ (инкременту, декременту) указателя стека от 0007H до _____.

5.56. См. рис. 5.16, *б*. Этап *7* соответствует _____ (загрузке в стек, извлечению из стека) данных стека.

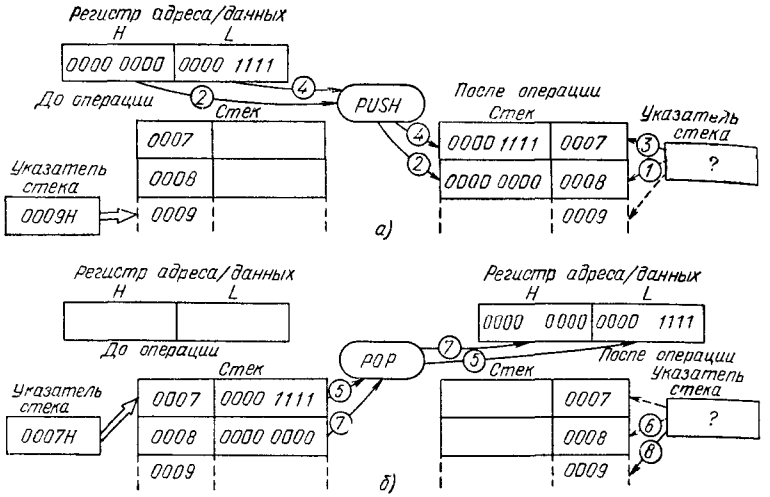


Рис 5.16 К упражнениям 5.50—5.58, 5 82, 5.83

5.57. См. рис. 5.15 и 5.16. Указатель стека _____ (инкрементируется, декрементируется) до операции PUSH и _____ (инкрементируется, декрементируется) после операции POP.

5.58. См. рис. 5.16,*б*. После этапа *8* в указателе стека содержится адрес _____ (шестнадцатеричное значение).

Решения

5.44. Адрес. **5.45.** Памяти LIFO (последний вошел — первый вышел). **5.46.** Последний вошел — первый вышел, что эквивалентно FILO, т. е. первый вошел — последний вышел. **5.47.** Программист. **5.48.** 2209H. **5.49.** Записи. **5.50.** Указатель стека. **5.51.** Загрузка в стек. **5.52.** 0007H. **5.53.** Вершину. **5.54.** Младшего; вершины стека. **5.55.** Инкременту; 0008H **5.56.** Извлечению из стека. **5.57.** Декрементируется; инкрементируется. 5.58. 0009H.

Дополнительные упражнения к гл. 5

5.59. Обычно МП выполняются в форме _____ корпуса ИС.

5.60. Документация, поставляемая конструктором, содержит, вероятно, _____ (технический список программирования, схему выводов).

5.61. При виде сверху вывод *2* следует непосредственно после вывода *1* _____ (по ходу часовой стрелки, против часовой стрелки).

5.62. См. рис. 5.2. Требование прерывания в МП Intel 8080 является _____ (входным, выходным) выводом.

5.63. См. табл. 5.1. Команда CALL МП Intel 8080 имеет КОП (шестнадцатеричный) _____ и требует _____ байт памяти.

5.64. См. рис. 5.4. Когда вывод $\overline{WR}$ имеет L-уровень, он _____ (управляет, управляем) и МП находится в состоянии _____ (чтения, записи) в память или УВВ.

5.65. См. рис. 5.4, *а*. Имеем вывод *28*. Стрелка указывает в обе стороны, потому что этот вывод подсоединяется на _____ (однонаправленную, двунаправленную) шину данных.

5.66. См. рис. 5.4. Какой вход устройства заставляет перейти счетчик команд к адресу 0000H, а МП — приступить к началу выполнения подпрограммы инициализации?

5.67. Типовой МП содержит _____ (АЛУ, ПЗУ).

5.68. Типовой МП содержит _____ (внутренний, внешний) генератор тактовых импульсов.

5.69. Во время первого считывания из памяти программы за один цикл команды данные с шины данных переданы в_____ (аккумулятор, регистр команд).

5.70. Микропроцессор обладает четырьмя типами машинных циклов: цикл извлечения или считывания КОП, цикл _____ из памяти, цикл записи в память и цикл выполнения внутренней операции.

5.71. Машинные циклы и цикл команды имеют переменную длительность, но интервалы времени _____ имеют заданную фиксированную длительность.

5.72. В типовом МП какая длительность меньше: цикла команды или машинного цикла?

5.73. На рис. 5.6 _____ интерпретирует содержимое регистра команды, определяет внутреннюю микропрограмму для выполнения и сигнал в устройство синхронизации и управления, который устанавливает последующую процедуру.

5.74. _____МП (см. рис. 5.6) осуществляет операции сложения, вычитания, И, ИЛИ и сдвига.

5.75. Типовой микропроцессор содержит индикатор нуля и переноса. Эти индикаторы являются содержимым _____ (управления прерыванием, регистра состояния).

5.76. Счетчик команд и регистр адреса/данных являют-

ся аналогами в том смысле, что оба могут _____ (указать ячейку памяти, поместить команды).

5.77. См. рис. 5.6. Какие три устройства могут указать ячейку памяти?

5.78. См. на рис. 5.9 команду *2*. Пара регистров *HL* загружена _____ (адресом, данными) в этой программе.

5.79. См. рис. 5.9 и 5.10. Команда *5* является командой ADD _____ (непосредственной, косвенно регистровой) адресации.

5.80. См. рис. 5.9 и 5.10, *г*. Регистр _____ в этой программе инкрементирован от _____ до _____ (шестнадцатеричные).

5.81. См. рис. 5.9 и 5.10, *ж*. Команда *7* является командой STORE _____ (прямой, косвенной регистровой) адресации.

5.82. См. рис. 5.16, *а*. На этой схеме микропроцессор _____ (загружает в стек, извлекает из стека).

5.83. См. рис. 5.15, *а*. Содержимое указателя стека после операции должно быть _____ (шестнадцатеричное).

5.84. Стек является специализированной областью памяти с _____ (произвольным, последовательным) доступом, используемой при выполнении команд _____ (PUSH, STORE) и _____ (LOAD, POP).

Решения

5.59. DIP (с двойной упаковкой выводов). **5.60.** Наряду с другими сведениями информацию о диаграмме выводов. **5.61.** Против часовой стрелки. **5.62.** Входным. **5.63.** CDH; 3. **5.64.** Управляет; записи. **5.65.** Двунаправленную. **5.66.** Сброс *RESET*. **5.67.** АЛУ. **5.68.** Внутренний. **5.69.** Регистр команд. Эти данные составляют КОП команды. **5.70.** Считывания. **5.71.** *Т*. **5.72.** Машинного цикла. **5.73.** Дешифратор команд. **5.74.** АЛУ. **5.75.** Регистра состояния. **5.76.** Указать ячейку памяти. **5.77.** Счетчик команд, указатель стека, регистр адреса/данных. **5.78.** Адресом. **5.79.** Косвенно-регистровой. **5.80.** Адреса/данных (или HL); 2101H; 2102H. **5.81.** Косвенной регистровой. **5.82.** Загружает в стек. **5.83.** 0007H. **5.84.** Последовательным; PUSH; POP.

Глава 6

ПРОГРАММИРОВАНИЕ МИКРОПРОЦЕССОРА

6.1. МАШИННЫЙ ЯЗЫК И АССЕМБЛЕР

На своем рабочем уровне микропроцессор реагирует на список операций, называемый машинной программой. На рис. 6.1, *а* приведено содержимое памяти, являющееся программой на машинном языке. Эта программа начинается с адреса 2000H с содержимым КОП 0011 1110_2 и оканчивается адресом 2006H с содержимым 0111 0110_2. Человеку практически невозможно понять программу, представленную в такой форме.

Адрес, H-код	Двоичное содержимое	
2000	0011 1110	← Начало программы
2001	1011 0100	
2002	0010 1111	
2003	0011 0010	
2004	0000 0000	
2005	0010 0001	
2006	0111 0110	← Конец программы
2007		

а)

Адрес, H-код	Содержимое, H-код	
2000	3E	← Начало программы
2001	B4	
2002	2F	
2003	32	
2004	00	
2005	21	
2006	76	← Конец программы
2007		

б)

Рис. 6.1. Программы:
а — в двоичном машинном коде; *б* — в шестнадцатеричном машинном коде

Программа на машинном языке на рис. 6.1, *а* становится несколько проще для восприятия, когда она представлена в шестнадцатеричном коде (H-коде), как показано на рис. 6.1, *б*. Однако, хотя двоичные данные приведены в шестнадцатеричном коде, эта часть программы всегда рассматривается как заданная на машинном языке и оказывается трудной для понимания.

В более приемлемой форме записанная на машинном языке она могла бы выглядеть так:

1. Загрузить двоичное число (1011 0100) в аккумулятор.

2. Инвертировать каждый двоичный бит содержимого аккумулятора.

3. Поместить результаты инверсии в ячейку памяти данных 2100H.

В этой части осуществляется перевод двоичного 8-разрядного числа в его эквивалент в инверсной форме.

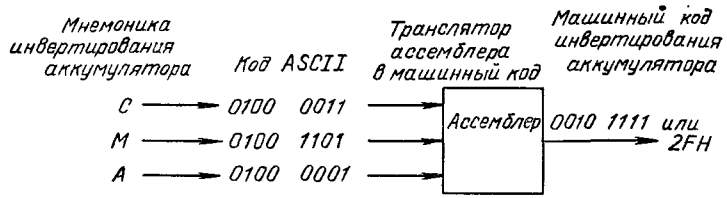


Рис. 6.2 Трансляция мнемоники ассемблера в машинный код программы

Возникает вопрос: как перейти от этой формы человеческого языка, иногда длинной и сложной, к машинному языку? Ответ состоит в использовании языка простого программирования — от самого высокого уровня до машинного, представленного на рис. 6.1. *Ассемблер* использует слова и фразы, преобразуя их в машинный код микропроцессора.

Обычно фраза или заданная величина на ассемблере будет соответствовать выражению длиной от одного до трех байт машинного языка. Суть и процедура ассемблирования показаны на рис. 6.2, где, например, вторая команда программы представлена единственной *мнемоникой* из трех букв CMA (инвертировать содержимое аккумулятора)[1]. Сначала три буквы переведены в их эквивалент в коде AS CII, затем три кода AS CII преобразованы в определенный порядок специальной программой ассемблера, которая выдает код инверсии содержимого аккумулятора на машинном языке, т. е. $0010\ 1111_2$ в данном случае или 2FH. Мнемоника преобразована в один единственный байт машинного языка.

Программа на языке ассемблер, записанная человеком, могла бы быть представлена в виде табл. 6.1.

Обычным является деление объявлений на машинном

[1] CMA от *Complement Accumulator* (англ.) — дополнить аккумулятор. Имеется в виду дополнение до 1, т. е. инвертирование или формирование обратного кода числа, содержащегося в аккумуляторе. — *Прим. ред.*

Таблица 6.1. *Программа на языке ассемблер*

Метка	Мнемоника	Операнд	Комментарии
	MVI	*A*, B4H	Загрузить в аккумулятор данные, следующие непосредственно: B4H
	CMA		Инвертировать содержимое аккумулятора
	STA	2100H	Разместить содержимое аккумулятора
	HLT		Остановить МП

языке на четыре поля: метка; мнемоника; операнд и комментарий. *Поле метки* используется не всегда и в этой программе остается пустым. *Поле мнемоники* содержит точную мнемонику, установленную разработчиком, которая обычно указывает программе ассемблера операцию для выполнения. *Поле операнда* содержит информацию о регистрах, данных и адресах, объединенных соответствующей операцией. Используя информацию только полей мнемоники и операнда, ассемблер может выдать соответствующий код на машинном языке. Можно также назначить ячейки памяти списку команд. Поле комментариев не учитывается ассемблером и ограничивается его перепечаткой. Это поле очень важно, поскольку позволяет понять события в программе.

Как только программа составлена (см. табл. 6.1), она представляется затем в виде табл. 6.2. Таким образом, задача ассемблирования (или составление программы на ассемблере) состоит из этапов: 1) перевод мнемоники и

Таблица 6.2. *Программа в машинных кодах и на языке ассемблер*

Адрес, Н-код	Содержимое, Н-код	Метка	Мнемоника	Операнд	Комментарий
2000	3E		MVI	*A*, B4H	Загрузить аккумулятор данными, следующими непосредственно за КОП, B4H
2001	B4				
2002	2F		CMA		Инвертировать содержимое аккумулятора
2003	32		STA	2100H	Поместить содержимое аккумулятора в ячейку памяти 2100H
2004	00				
2005	21				
2006	76		HLT		Остановнть МП

операндов на машинный язык; 2) назначение последовательно ячейки памяти каждому КОП и операнду. Переход от версии табл. 6.1 к ассемблированной версии табл. 6.2 может быть выполнен либо вручную, либо на машине при помощи специальной программы ассемблера.

Программа, состоящая из символических команд (фрагмент показан в табл. 6.1), иногда называется *исходной программой,* а переведенная однажды на машинный язык — уже *объектной программой.*

Программирование на языке ассемблер является способом «очеловечивания» действий микропроцессора. Языки высокого уровня (БЕЙСИК, ФОРТРАН и т. д.) при их использовании делают программирование более удобным. Например, одна команда на БЕЙСИКе или ФОРТРАНе может соответствовать 20 или 30 машинным командам. Название этой главы относится к программированию микропроцессора (в противоположность программированию микро-ЭВМ), потому что везде мы будем использовать его состав команд. Будет использовано программирование на языке ассемблер, помогающее пониманию состава команд микропроцессора и его действий.

Упражнения

6.1. Два сегмента программы, приведенные на рис. 6.1, написаны на _____ (ассемблере, машинном языке).

6.2. Для формулировки команды микропроцессору язык _____ (машинный, ассемблер) использует слова и фразы.

6.3. _____ (Ассемблер, Монитор) является специальной программой ЭВМ, позволяющей перевести программу пользователя на машинный язык.

6.4. Перечислить четыре поля программ на языке ассемблер.

6.5. Составленная на языке ассемблер программа будет содержать адреса ячеек памяти и _____ (машинные, БЕЙСИКа) коды для каждой команды на языке ассемблер.

6.6. Программа, составленная из символических команд, называется _____ (объектной, исходной).

Решения

6.1. Машинном языке. **6.2.** Ассемблер. **6.3.** Ассемблер. **6.4.** См. обозначения в табл. 6.1, а именно четыре поля: метка, мнемоника, операнд и комментарий. **6.5.** Машинные. **6.6.** Исходной.

6.2. ПРОСТОЙ СОСТАВ КОМАНД

Введем состав команд, относящихся к типовому микропроцессору, приведенному на рис. 5.6. Мнемоника и КОП, которые будут использованы, представляют собой подсостав команд микропроцессоров Intel 8080/8085 (мы благодарны компании Intel, предоставившей нам возможность это сделать). Ранее мы уже рассматривали произвольно (см. § 4.4) отдельные мнемоники, но предпочли использовать часть состава команд МП Intel 8080/8085, так как это позволит читателю ближе познакомиться с ним перед изучением этих МП в гл. 8 и 9. На рис. 6.3 в сокращенном ва-

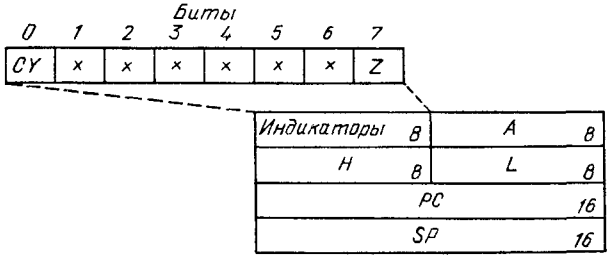


Рис. 6.3. Регистры, доступные для программирования типового МП

рианте приведены регистры типового МП, предоставляемые программисту. Вверху справа мы видим универсальный 8-разрядный регистр-аккумулятор *A*, а слева показан 8-разрядный регистр состояния. В составе этого регистра индикатор переноса *CY* находится в позиции 7, а индикатор нуля *Z* — в позиции 0. Позиции бит от первого до шестого регистра состояния в типовом МП не используются, но в выпускаемых МП индикаторов больше, чем здесь.

Во второй строке на рис. 6.3 расположены регистры *H* и *L*. Это универсальные регистры адреса/данных. Они могут использоваться раздельно или в форме пары регистров (мы говорим тогда о паре регистров *HL*). В последнем случае они используются как указатель адреса.

Снизу на рис. 6.3 находятся два специальных 16-разрядных регистра. Счетчик команд *PC** указывает МП на

* От *Program Counter* (англ.) — программный счетчик. — *Прим. пер.*

следующую для выполнения команду. Указатель стека *SP** содержит адрес вершины стека. Сам стек находится в ОЗУ.

Состав команд такого типового микропроцессора разделен на семь категорий, которые мы уже встречали в гл. 4, а именно: арифметические; логические; передачи данных; ветвления; вызова подпрограмм; возврата из подпрограмм; прочие.

Типовой микропроцессор в состоянии выполнить только 67 различных команд из 239, которые входят в состав МП Intel 8085.

Упражнения

6.7. В типовом микропроцессоре 67 команд представляют собой подсостав состава команд МП_____.

6.8. См. рис. 6.3. Перечислить три универсальных 8-разрядных регистра типового МП.

6.9. См. рис. 6.3. Какие индикаторы находятся в регистре состояния типового МП?

6.10. См. рис. 6.3. Назвать два специальных 16-разрядных регистра типового МП.

Решения

6.7. Intel 8080/8085. **6.8.** Аккумулятор *A*, регистры адреса/данных *H* и *L*. **6.9.** Индикаторы нуля *Z* и переноса *CY*. **6.10.** Счетчик команд *PC* и указатель стека *SP*.

6.3. СОСТАВ КОМАНД АРИФМЕТИЧЕСКИХ ДЕЙСТВИЙ

Первыми рассмотрим команды арифметических действий. Они сведены в табл. 6.3 и содержат команды сложить, вычесть, инкрементировать, декрементировать, сравнить. Заметим по данным табл. 6.3, что существуют четыре команды сложения. Аккумулятор (регистр *A*) содержит одно из слагаемых. Каждая команда точно оговаривает различные источники другого слагаемого.

Рассмотрим первую из команд сложения в табл. 6.3. Команда сложить непосредственно является двухбайтовой. Ее формат КОП (в этом случае С6Н) содержится в первом байте команды, непосредственно за ним — во втором

* См. § 5.1. — *Прим. ред.*

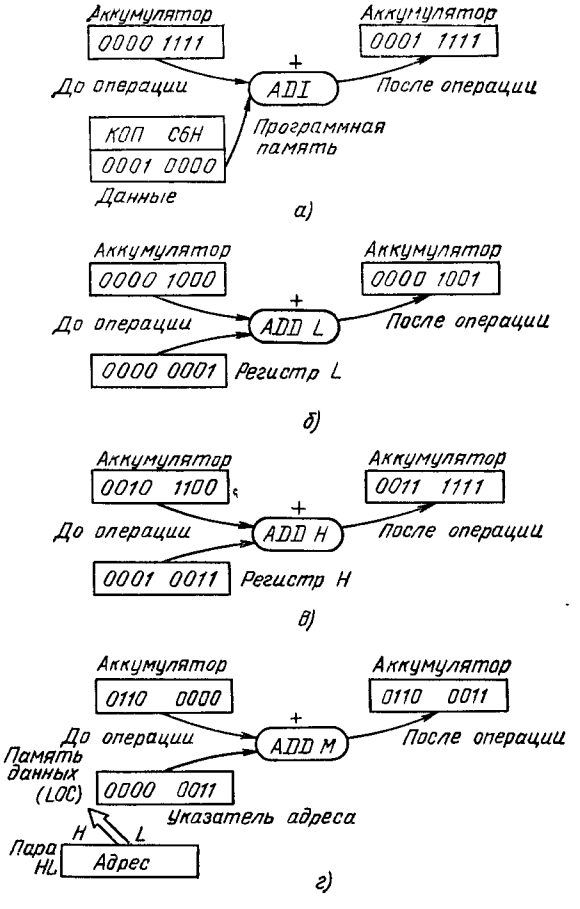


Рис. 6.4. Интерпретация операции ADD *M* в кодах табл. 6.3:
а — сложение (А) с даннымн, следующимн непосредственно за КОП; *б* — сложение (L) и (А); *в* — сложение (Н) и (А); *г* — сложение с косвенной адресацией

байте — находятся данные для сложения с содержимым аккумулятора. Команда ADI выполняется по схеме, приведенной на рис. 6.4, *а*. Данные, находящиеся в памяти непосредственно за КОП, складываются с содержимым аккумулятора ($0000\ 1111_2$). Сумма ($0001\ 1111_2$) помещается в аккумулятор.

Таблица 6.3. *Команды арифметических операций типового микропроцессора*

Операция	Адресация	Мнемоника	КОП	Байты	Формат команды	Символика	Индикаторы
Сложить *A* с данными	Непосредственная	ADI	C6	2	КОП данные	(A) ← (A) + (байт 2)	*Z*, *CY*
Сложить *L* с *A*	Регистровая	ADD *L*	85	1	КОП	(A) ← (A) + (L)	*Z*, *CY*
Сложить *H* с *A*	»	ADD *H*	84	1	КОП	(A) ← (A) + (H)	*Z*, *CY*
Сложить LOC (HL) с *A*	Косвенная регистровая	ADD *M*	86	1	КОП	(A) ← (A) + ((H) (L))	*Z*, *CY*
Вычесть данные из *A*	Непосредственная	SUI	6	2	КОП данные	(A) ← (A) — (байт 2)	*Z*, *CY*
Вычесть *L* из *A*	Регистровая	SUB *L*	95	1	КОП	(A) ← (A) — (L)	*Z*, *CY*
Вычесть *H* из *A*	Регистровая	SUB *H*	94	1	КОП	(A) ← (A) — (H)	*Z*, *CY*
Вычесть LOC (HL) из *A*	Косвенная регистровая	SUB *M*	96	1	КОП	(A) ← (A) — ((H) (L))	*Z*, *CY*
Инкремент *A*	Регистровая	INR *A*	3C	1	КОП	(A) ← (A) + 1	*Z*
Инкремент *HL*	»	INX *H*	23	1	КОП	(HL) ← (HL) + 1	
Декремент *A*	»	DCR *A*	3D	1	КОП	(A) ← (A) — 1	*Z*
Декремент *HL*	»	DCX *H*	2B	1	КОП	(HL) ← (HL) — 1	
Сравнить *A* с данными	Непосредственная	CPI	FE	2	КОП данные	(A) ← (байт 2)	*Z* =1, если (A)=(байт 2); *CY*=1, если (A)<(байт 2)
Сравнить *A* с *L*	Регистровая	CMP *L*	BD	1	КОП	(A) — (L))	*Z*=1, если (A)=(*L*); *CY*= =1, если (A)<(*L*)
Сравнить *A* с *H*	»	CMP *H*		1	КОП	(A) — (H)	*Z* = 1, если (A) = (H); *CY* = 1, если (A) < (H)
Сравнить *A* с LOC (HL)	Косвенная регистровая	CMP *M*		1	КОП	(A) — ((H) (L))	*Z*=1, если (A)=((H)(L)); *CY*=1, если (A)<((H)(L))

() — содержимое; (()) — косвенная регистровая адресация.

Второй является команда сложить содержимое регистров L и A (мнемоника ADD L), на рис. 6.4, *б* показано ее выполнение. Содержимое аккумулятора ($0000\ 1000_2$) складывается с содержимым регистра L ($0000\ 0001_2$), получающаяся в результате выполнения команды ADD L сумма ($0000\ 1001_2$) помещается в аккумулятор.

Третья команда в табл. 6.3 представляет собой однобайтовую команду сложить содержимое регистров H и (мнемоника ADD H). Это другая команда сложения с держимого одного регистра с содержимым другого (ка и предшествующая), и она выполняется в последовател ности, приведенной на рис. 6.4, *в*. Содержимое регистра ($0010\ 1100_2$) сложено с содержимым регистра H ($0001\ 0011_2$), сумма ($0011\ 1111_2$) помещена в аккумулятор.

Четвертая строка таблицы представляет собой однобайтовую команду сложить с косвенным регистром (мнемоника ADD M). Адрес данного слагаемого числа задан в более сложной форме с использованием способа косвенной регистровой адресации. Рисунок 6.4, *г* является примером выполнения команды ADD M. Пара HL указывает 16-разрядный адрес памяти, т. е. LOC*. Содержимое LOC ($0000\ 0011_2$) сложено с содержимым аккумулятора ($0110\ 0000_2$), сумма ($0110\ 0011_2$) помещена в аккумулятор. Команды косвенного сложения используют в качестве указателя адреса 16-разрядный регистр (обычно пару HL).

Рассмотрим снова команду сложить с косвенным регистром (ADD M) в табл. 6.3. Предписание символики указывает: сложить LOC (H+L) с A, которое мы прочтем как сложить содержимое памяти, на которую указывает пара регистров HL, с содержимым регистра A. Следуя соответствующей строке в табл. 6.3, мы найдем, что способом адресации является косвенная регистровая и соответствующей мнемоникой будет ADD M с КОП 86H. Эта команда является однобайтовой. Представленный формат ее показывает, что единственным байтом является КОП. Следуя предписанию, данному фирмой Intel (и, очевидно, обратному обычной записи), символическую запись читают справа налево. Справа стрелки первый операнд идентифицирован как содержимое A (акумулятора). Скобки в рамках этой записи означают содержимое. Знак «+» означает выполняемую операцию, а двойная скобка (()) указывает команду косвенной адресации. Выражение

* От *Location* (англ.) — место расположения. — *Прим. пер.*

((H)(L)) означает: содержимое пары HL указывает адрес расположения в памяти второго операнда. Иначе говоря, второй операнд расположен в ячейке памяти, на которую указывает пара регистров HL. После того как операция выполнена, ее результат помещается в аккумулятор. Все действия выполняются по такой схеме:

Результат	Первый операнд	Операция	Второй операнд
(A) ←	(A)	+	((H)(L))
Содержимое регистра A	Содержимое регистра A	Сложить	Содержимое пары указывает на положение операнда в памяти

В последней правой колонке табл. 6.3 приведен список устанавливаемых по результатам операции индикаторов. Пример сложения двоичных чисел 1111 1111 и 0000 0001 приведен на рис. 6.5. Выполнение такой операции обычным способом дает следующий результат:

```
  1111 1111
+
  0000 0001
-------------
1 0000 0000
```

Перенос — Содержимое аккумулятора

Решение этой задачи микропроцессор выполняет в соответствии с рис. 6.5 (операция ADD M.

Аккумулятор содержит $1111\ 1111_2$, тогда как ячейка памяти, на которую указывает пара регистров HL, содержит другое слагаемое ($0000\ 0001_2$). Выполнив операцию сложения, аккумулятор содержит 8 младших бит суммы,

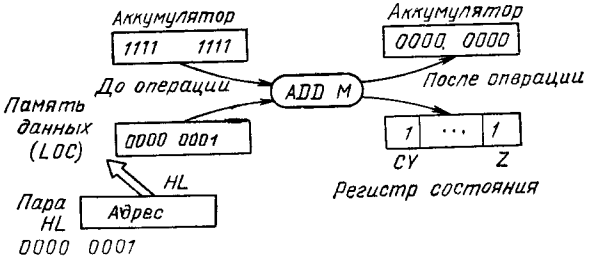


Рис. 6.5. Влияние результата операции ADD M на содержимое аккумулятора и индикаторы

т. е. 0000 0000_2. Индикатор переноса регистра состояния установлен в 1, что показывает наличие переноса старшего бита результата. Индикатор нуля проверяет содержимое аккумулятора после операции и находит 0000 0000_2. Так как это нуль, индикатор нуля становится 1, показывая, что содержимое аккумулятора 0000 0000_2 после операции сложения.

Рассмотрим теперь четыре операции вычитания, приведенные в табл. 6.3, относящиеся к составу команд типового микропроцессора. Каждая команда вычитает содержимое некоторого регистра или ячейки памяти из содержимого аккумулятора. В силу внутренних особенностей АЛУ не обладает возможностями вычитания, оно осуществляет сложение, представляя вычитаемое в форме дополнительного кода и затем складывая его.

В качестве примера рассмотрим процесс вычитания, представленный на рис. 6.6. Двоичное число 0000 0001 вы-

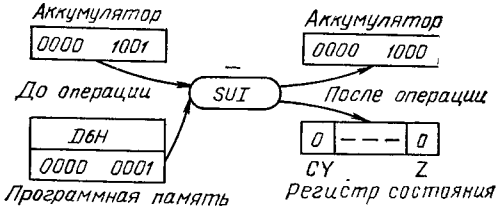


Рис. 6.6. Команда вычитания данных с непосредственной адресацией

читается из 0000 1001 (09H—01H=08H). Выполнение этой операции обычным способом дает следующий результат

CY				
	Инверсия	+	0000 1001 1111 1111	Дополнительный код
0	←———	1	0000 1000	
Нет переноса		Переполнение	Содержимое аккумулятора (8 бит)	

Напомним, что вычитание двоичных чисел осуществляют сложением первого числа со вторым, представленным форме дополнительного кода, пренебрегая переполнением. В этой задаче вторым числом будет 0000 0001, которое будет преобразовано в дополнительный код и даст

	Инверсия		
0000 0001	———→	1111 1110	Добавить 1
1111 1110	+1=	1111 1111	Дополнительный код

Дополнительный код 1111 1111 другого числа складывается тогда с первым числом, что дает сумму 1 0000 1000. В старшем бите суммы 1 является переполнением и не принадлежит разности 0000 1000_2. Микропроцессор использует это переполнение для установления индикатора переноса. Вычитая, МП инвертирует переполнение, и результат становится содержимым индикатора переноса *CY*. Переполнение 1 инвертируется и сбрасывает индикатор переноса в 0. Когда в ходе вычитания индикатор переноса сбрасывается, это значит, что переноса не было и что первое число больше второго.

Команда вычесть непосредственные данные (мнемоника SUI) используется в примере на рис. 6.6. Следующие непосредственно за КОП данные второго байта памяти 0000 0001_2 вычитаются из содержимого аккумулятора 0000 1001_2, разность передается в аккумулятор, после чего операция завершается. Индикатор переноса сбрасывается в 0. Это означает, что переноса не было или содержащееся в аккумуляторе число до операции было больше числа в памяти. Индикатор нуля проверяет содержимое аккумулятора. Операция вычитания завершается, содержимое 0000 1000— не нуль, индикатор нуля также сброшен в 0.

Рассмотрим другой пример вычитания, когда уменьшаемое меньше вычитаемого. Пусть надо вычесть двоичное число 0000 0110 из двоичного 0000 0101 (05H—06H= =FFH=-1_{10}). В этом случае:

			+	0000 0101 1111 1010	Дополнительный код
1	Инверсия ←———	0		1111 1111	
Перенос		Переполнение	Содержимое аккумулятора (8 бит)		

Вычитаемое здесь представлено в дополнительном коде

	Инверсия		
0000 0110	———→	1111 1001	Добавить 1
1111 1001	+ 1 =	1111 1010	Дополнительный код

Уменьшаемое 0000 0101_2 сложено с результатом преобразования вычитаемого в дополнительный код 1111 1010, что дает 1111 1111. Как можно увидеть из табл. 2.10, результат 1111 1111 является представлением в дополнительном коде числа -1_{10}. Сложение не вызывает перепол-

нения тогда, когда имеем 0 в регистре переполнения. Этот результат инвертируется (так как речь идет о вычитании), что дает 1 в регистре переноса *CY*. Когда индикатор *CY* устанавливается в 1 после вычитания, это означает, что число, содержащееся в аккумуляторе, младше числа в па

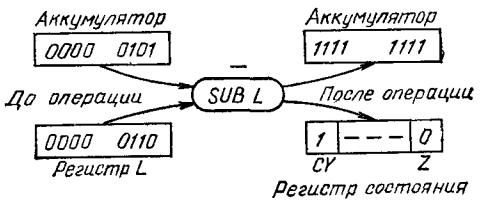


Рис. 6.7. Команда вычитания (L) из (A)

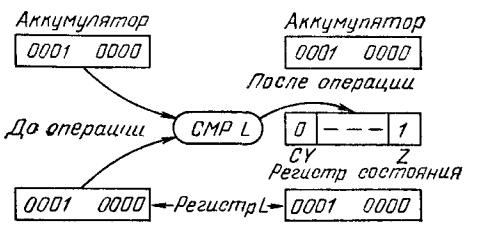


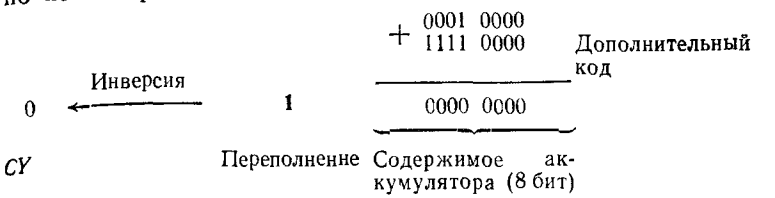
Рис. 6.8. Команда сравнения (A) с (L)

мяти или в регистре. Индикатор переноса в состоянии 1 показывает, что число, содержащееся в аккумуляторе после того, как вычитание было выполнено, является дополнительным кодом, представляя отрицательное число. 1111 1111 в аккумуляторе на рис. 6.7 представляет собой -1_{10}.

Наш типовой микропроцессор обладает четырьмя командами сравнения (см. четыре последние строки в табл. 6.3). Команды сравнения вычитают содержимое регистра или ячейки памяти из содержимого аккумулятора, но не изменяют содержимое ни того, ни другого. Индикаторы же подвержены воздействию команд сравнения.

На рис. 6.8 приведен пример использования команды СРАВНИТЬ регистр *L*. Равные числа $0001\ 0000_2$ являются содержимым аккумулятора и регистра *L* и сравниваются микропроцессором. Заметим на рис. 6.8, *а*, что ни одно, ни другое из содержимых не изменяется после операции сравнения. Индикаторы подвержены влиянию результата сравнения.

Согласно приведенному в табл. 6.3 символическому представлению информации содержимое регистра *L* вычтено из содержимого аккумулятора *A* по команде CMP *L*:



Второе число (содержимое регистра *L*) переведено в дополнительный код:

	Инверсия		
0001 0000	⟶	1110 1111	Добавить 1
1110 1111	+ 1 =	1111 0000	Дополнительный код

Первое число $0001\ 0000_2$ и дополнительный код второго числа 1111 0000 складывается, что дает 1 0000 0000. Затем проверяется равенство 0 восьми младших разрядов. Индикатор нуля принимает значение 1. Переполнение 1 инвертируется АЛУ, и в этом примере индикатор переноса принимает значение 0. Сброшенный индикатор переноса *CY* означает, что содержимое аккумулятора больше или равно содержимому регистра *L*.

Таким образом, арифметические операции типового микропроцессора могут выполняться по большому числу команд из всего состава. Многие МП снабжены несколькими дополнительными арифметическими командами и индикаторами в их регистре состояния. Использование индикатора при ветвлениях и переходах по результатам операций будет впоследствии рассмотрено.

Упражнения

6.11. См. табл. 6.3. Мнемоника команды ВЫЧЕСТЬ прямо _____, а ее КОП—D6H. Формат команды _____ байт.

6.12. См. табл. 6.3. Инкрементировать — значит _____ (прибавить 1, вычесть 1) в/из содержимое регистра.

6.13. См. табл. 6.3. Какая команда инкрементирования не влияет на индикатор нуля?

6.14. См. рис. 6.9. После операции декрементирования содержимым аккумулятора будет _____ (двоичное).

6.15. См. табл. 6.3. После операции декрементирования индикатор нуля на рис. 6.9 _____ (сброшен, установлен), т. е. равен _____ (0, 1).

6.16. См. табл. 6.3. Каково состояние индикатора переноса на рис. 6.9 после операции декрементирования?

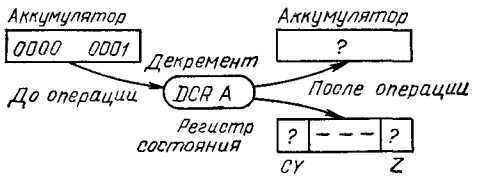


Рис. 6.9. К упражнению 6.14

6.17. См. табл. 6.3. Какой КОП команды СРАВНИТЬ прямо (шестнадцатеричный)?

6.18. См. табл. 6.3. Дать содержимое аккумулятора и двух ячеек памяти программы после операции CPI на рис. 6.10.

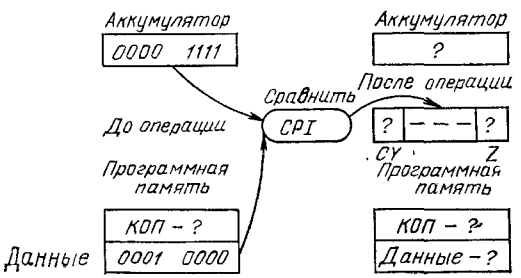


Рис. 6.10. К упражнениям 6.18 и 6.19

6.19. См. табл. 6.3. Каким будет состояние индикатора нуля после выполнения команды CPI на рис. 6.10?

6.20. См. табл. 6.3. Индикатор переноса _____ (установлен, сброшен) после выполнения команды CPI.

6.21. См. табл. 6.3. Каким будет содержимое пары регистров *HL* (шестнадцатеричное) после операции инкрементирования?

6.22. См. табл. 6.3. Каково состояние пары регистров *HL* после инкрементирования (рис. 6.11)?

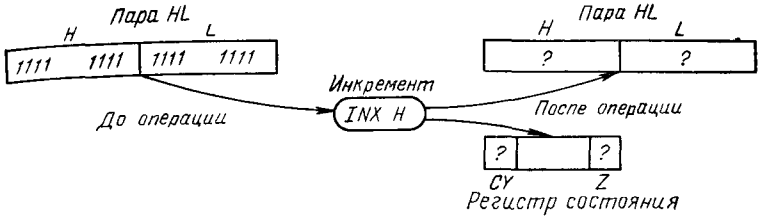


Рис 6.11. К упражнению 6.22

6.23. Индикаторы подвержены влиянию многих арифметических операций, но используются микропроцессором в ходе операций _____ (ветвления, логических).

Решения

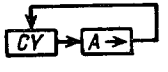
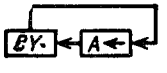
6.11. SUI; 2. **6.12.** Прибавить 1. **6.13.** Согласно табл. 6.3 это команда INX *H* (Инкрементировать пару регистров *HL*). **6.14.** Команда DCR *A* означает, что нужно вычесть 1 из содержимого аккумулятора, т. е. 0000 0001—1=0000 0000, результат 0000 0000 является содержимым аккумулятора после операции. **6.15.** Проверка после декрементирования показывает, что содержимое аккумулятора — 0, следовательно, индикатор нуля регистра состояния установлен в 1. **6.16.** Индикатор переноса *CY* не подвержен влиянию со стороны операции декрементирования, следовательно, его состояние непредсказуемо. **6.17.** Команда CPI, ее КОП — FEH. **6.18.** Команда сравнения не изменяет содержимое регистров или ячеек памяти, поэтому содержимым аккумулятора всегда будет 0000 1111, памяти — 0001 0000, КОП команды CPI—FEH (см. табл. 6.3). **6.19.** На рис. 6.10 индикатор нуля после операции CPI сбрасывается (0). Колонка символики в табл. 6.3 показывает, что команда сравнения выполняется вычитанием с проверкой результата. Результат — не 0, индикатор не устанавливается. **6.20.** Установлен в 1. Результат операции вычитания показывает, что первое число (содержимое аккумулятора) было меньше содержимого памяти (второго числа). **6.21.** Инкремент *HL* означает 1111 1111 1111 1111+1=1 0000 0000 0000 0000. Старший разряд суммы (1) исключается, следовательно, содержимым пары регистров *HL* будет 0000 0000 0000 0000 или 0000H.

6.22. Индикаторы не подвержены влиянию операции инкрементирования пары регистров *HL* (см. табл. 6.3), следовательно, результат непредсказуем. **6.23.** Ветвления (условного).

6.4. СОСТАВ КОМАНД ЛОГИЧЕСКИХ ОПЕРАЦИЙ

Логические команды составляют еще одну группу команд типового микропроцессора. Они сведены в табл. 6.4 и содержат команды И, ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ,

Таблица 6.4. *Команды логических операций типового микропроцессора*

Операция	Адресация	Мнемоника	КОП	Байты	Формат команды	Символика	Установка индикаторов
A И данные	Непосредственная	ANI	E6	2	КОП данные	(A) ← (A) · (байт 2)	Z *CY* — Сброс
A И *L*	Регистровая	ANA *L*	A5	1	КОП	(A) ← (A) · (L)	Z *CY* — Сброс
A И *H*	»	ANA *H*	A4	1	КОП	(A) ← (A) · (H)	Z *CY* — Сброс
A И LOC (HL)	Косвенная регистровая	ANA *M*	A6	1	КОП	(A) ← (A) · (H)	Z *CY* — Сброс
A ИЛИ данные	Непосредственная	ORI	F6	2	КОП данные	(A) ← (A) + (байт 2)	Z *CY* — Сброс
A ИЛИ *L*	Регистровая	ORA	B5	1	КОП	(A) ← (A) + (L)	Z *CY* — Сброс
A ИЛИ *H*	»	ORA *H*	B4	1	КОП	(A) ← (A) + (H)	Z *CY* — Сброс
A ИЛИ LOC (HL)	Косвенная регистрация	ORA *M*	B6	1	КОП	(A) ← (A) + ((H)(L))	Z *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ данные	Непосредственная	XRI	EE	2	КОП данные	(A) ← (A) ⊕ (байт 2)	Z *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ *A*	Регистровая	XRA *A*	AF	1	КОП	(A) ← (A) ⊕ (A) Сброс A	Z = 1 *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ *L*	»	XRA *L*	AD	1	КОП	(A) ← (A) ⊕ (L)	Z *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ *H*	»	XRA *H*	AC	1	КОП	(A) ← (A) ⊕ (H)	Z *CY* — Сброс
A ИЛИ ИСКЛЮЧАЮЩЕЕ LOC (HL)	Косвенная регистровая	XRA *M*	AE	1	КОП	(A) ← (A) + ((H)(L))	Z *CY* — Сброс
Инвертировать *A*	Неявная	CMA	2F	1	КОП	(A) ← ($\overline{A}$)	
Сдвиг *A* вправо	»	RAR	1F	1	КОП		*CY*
Сдвиг *A* влево	»	RAL	17	1	КОП		*CY*

() — содержимое; (()) — косвенная регистровая адресация; · — И; + — ИЛИ; ⊕ — ИЛИ ИСКЛЮЧАЮЩЕЕ.

НЕ (инверсия) и сдвига. Здесь именно аккумулятор составляет ядро большинства операций. Как и при арифметических командах, способ адресации и здесь влияет на способ и место нахождения других данных в системе.

Рассмотрим выполнение типовым микропроцессором команды И непосредственно (рис. 6.12, *а*). Содержимое

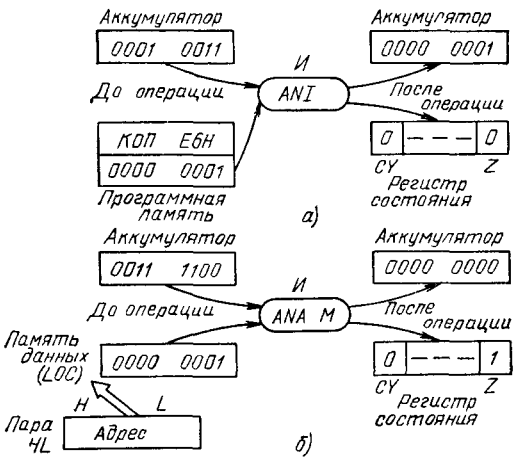


Рис. 6.12. Команда И с непосредственной адресацией (*а*) и команда ANA M (*б*)

аккумулятора (0001 0011) подвержено операции И побитно. Согласно таблице истинности для операции И (см. табл. 3.1) только самые младшие биты каждого числа равны 1, следовательно, результатом будет 0000 0001, он помещается в аккумулятор. Согласно последней колонке табл. 6.4 результатом всех операций И будет сброс индикатора переноса, мы это видим также на рис. 6.12, *а*. Результат операции И проверяется с целью определения — не нуль ли он, и если нет, то индикатор нуля сбрасывается в 0. Отметим использование точки (·) в табл. 6.4 в колонке символов для обозначения операции И.

На рис. 6.12, *б* приведен другой пример использования команды И, в этом случае — И косвенной адресации (мнемоника ANA *M*). Содержимое аккумулятора подвержено операции И (бит с битом) с содержимым ячейки памяти, указанной парой *HL*. После операции 0011 1100 И

0000 0001 полученный результат 0000 0000 помещен в аккумулятор. Индикатор переноса сбрасывается (СБРОС) согласно табл. 6.4. Помещенный в аккумулятор результат проверяется, и, поскольку он равен 0, индикатор нуля устанавливается в 1.

Внимательно рассмотрим рис. 6.12, *б*. В этих двух примерах второй операнд — 0000 0001, он используется как *маска*. Маска 0000 0001 на рис. 6.12, *а* и *б* может быть использована для сброса в 0 семи старших бит или, с учетом наличия индикатора нуля, для тестирования значений 0 или 1 в позиции младшего бита аккумулятора (в этом случае на единственную 1 надевается маска 0000 0001). Но нужно быть осторожным. Заметим, что содержимое аккумулятора изменилось после операции И. Некоторые микропроцессоры снабжены специальными командами тестирования битов, которые выполняют операции И с содержимым аккумулятора и с маской байта без изменения содержимого всего аккумулятора, изменяя состояние индикаторов.

Четыре команды ИЛИ (см. табл. 6.4) выполняются с содержимым аккумулятора и содержимым какой-либо другой ячейки памяти и регистра. На рис. 6.13 приведен при-

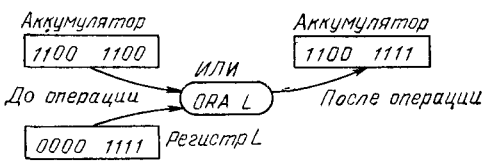


Рис. 6.13. Команда (A) ИЛИ (L)

мер операции ИЛИ, когда содержимым аккумулятора будет 1100 1100, тогда как регистр *L* содержит 0000 1111. Результат — 1100 1111. Числа подвержены операции ИЛИ побитно согласно таблице истинности ИЛИ (табл. 3.1). Содержимое 0000 1111 регистра *L* может рассматриваться как маска, которая всегда будет обращать 4 младших бита в 1111. Отметим использование логического знака (+) для обозначения операции ИЛИ в колонке символов в табл. 6.4.

В табл. 6.4 приведены пять команд ИЛИ ИСКЛЮЧАЮЩЕЕ для типового микропроцессора. На рис. 6.14 показан процесс выполнения команды ИЛИ ИСКЛЮЧАЮЩЕЕ *A* с *A*. Результатом ИЛИ ИСКЛЮЧАЮЩЕЕ

1010 1010 с самим собой (рис. 6.14) будет 0000 0000. Выполнение этой операции любого числа с самим собой всегда дает результат 0000 0000, индикатор нуля всегда устанавливается в 1, что означает нулевое содержимое индикатора, а в соответствии с табл. 6.4 индикатор переноса *CY* всегда будет сброшен в 0.

Две последние логические операции табл. 6.4 осуществляются командами циклического сдвига с переносом.

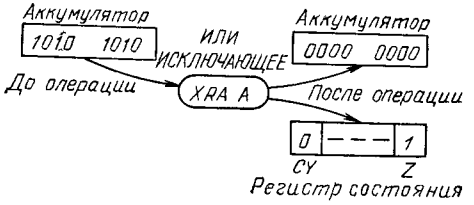


Рис. 6.14. Команда XRA *A*

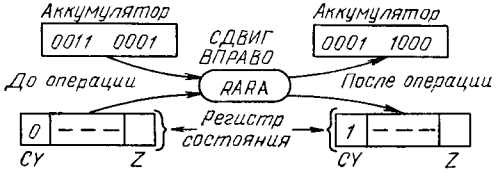


Рис. 6.15. Команда RAR *A*

Схемы в колонке символики показывают результат сдвига, выполняемого этими командами.

Рассмотрим приведенный на рис. 6.15 пример использования команды циклического сдвига вправо с переносом: содержимое аккумулятора (0011 0001) сдвинуто на одну позицию вправо и его младший бит (1 в этом примере) передается в позицию бита индикатора переноса, тогда как имевшийся там бит занимает позицию старшего бита аккумулятора, в котором содержится 0001 1000 после завершения операции. Индикатор переноса установится в 1, индикатор нуля не изменится.

Используя одну или несколько команд циклического сдвига, можно тестировать весь заданный состав бит, а индикатор переноса может быть сброшен или установлен. Индикатор переноса может быть тестирован затем командой условного ветвления. Тест паритета является другим приложением использования команд сдвига. Паритет двоичного числа определяется числом содержащихся в нем единиц: четный паритет — общее число единиц четное; нечетный паритет — общее число единиц нечетное.

Заметим, что команды сдвига оперируют только данными аккумулятора и не требуют других операндов, расположенных в памяти или регистрах. Поэтому здесь способ адресации называется неявным, а иногда вообще не указывается. Многие МП обладают несколькими иными, чем типовой микропроцессор, типами команд сдвига.

Таким образом, команды логических операций используются для минипуляции с переменными по законам алгебры логики. Они могут быть использованы для тестирования и сравнения бит.

Упражнения

Все упражнения связаны с табл. 6.4, обращаться к которой следует постоянно.

6.24. Какой КОП (шестнадцатеричный) имеет команда ANA *H* на рис. 6.16? Ее формат _____ байт.

6.25. Каково содержимое аккумулятора на рис. 6.16 после операции И?

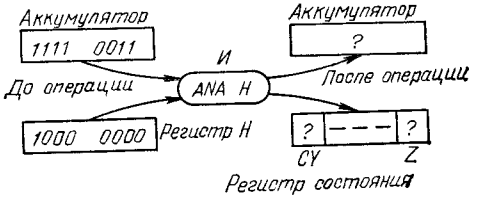


Рис. 6.16. К упражнениям 6.24—6.27

6.26. Назвать состояние индикаторов на рис. 6.16 после операции И.

6.27. Если роль байта в регистре *H* на рис. 6.16 состоит в тестировании старшего бита аккумулятора (0 или 1), слово 1000 0000 в регистре *H* называется _____ (маской, словом теста).

6.28. Огкуда поступает операнд, подверженный операции ИЛИ непосредственное (ORI) с содержимым аккумулятора?

6.29. Знак (+) в табл. 6.4 означает операцию _____ (сложение, ИЛИ).

6.30. Каков источник адреса (LOC) операнда на рис. 6.17 в памяти данных?

6.31. Каково содержимое аккумулятора на рис. 6.17 после операции ИЛИ?

6.32. Назвать состояния индикаторов в регистре состояния на рис. 6.17 после операции ИЛИ.

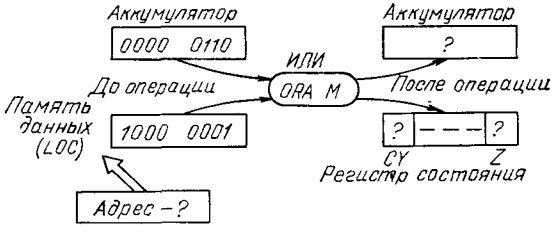


Рис. 6.17. К упражнениям 6.30—6.32

6.33. Мнемоника команды ИЛИ ИСКЛЮЧАЮЩЕЕ на ассемблере на рис. 6.18 _____.

6.34. Что является источником операнда, подверженного операции ИЛИ ИСКЛЮЧАЮЩЕЕ с содержимым аккумулятора на рис. 6.18?

6.35. Каково содержимое аккумулятора на рис 6.18 после операции ИЛИ ИСКЛЮЧАЮЩЕЕ?

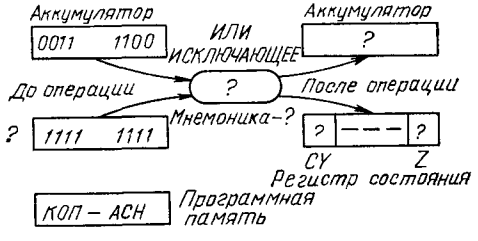


Рис. 6.18. К упражнениям 6.33—6.36

6.36. Каково состояние индикаторов на рис. 6.18?

6.37. Каково содержимое аккумулятора на рис. 6.19 после выполнения команды сдвига?

6.38. Какие состояния индикаторов на рис. 6.19 после операции циклического сдвига влево?

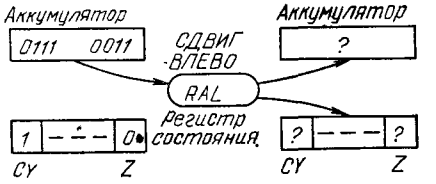


Рис. 6.19. К упражнениям 6.37 и 6.38

Решения

6.24. A4H; 1. **6.25.** 1111 0011 и 1000 0000 выполняется побитно и дает 1000 0000, что и будет содержимым аккумулятора. **6.26.** В индикаторах переноса и нуля будут нули. **6.27.** Слово 1000 0000 является маской, если его роль состоит в тестировании наиболее значимого бита аккумулятора. Используя эту маску с командой ANA *H*, получаем: если в индикаторе нуля 1, наиболее значимый бит байта в аккумуляторе 0; если в индикаторе нуля 0, старший бит байта в аккумуляторе 1. **6.28.** Команда ORI извлекает операнд из 2-го байт памяти. **6.29.** ИЛИ. **6.30.** Пара регистров *HL*. **6.31.** 1000 0111. **6.32.** Индикатор переноса сброшен; индикатор нуля сброшен — 0. **6.33.** КОП ИЛИ ИСКЛЮЧАЮЩЕЕ АСН (см. табл. 6.4), его мнемоника XRA *H*. **6.34.** Операцией является ИЛИ ИСКЛЮЧАЮЩЕЕ *A* с *H*. Значит, операнд поступит из регистра *H*. **6.35.** 1100 0011. Заметим, что ИЛИ ИСКЛЮЧАЮЩЕЕ с 1111 1111 дает инвертирование байта аккумулятора. **6.36.** Индикатор переноса сброшен; индикатор нуля сброшен — 0 **6.37.** 1110 0111. **6.38.** Только индикатор переноса подвержен влиянию операции циклического сдвига влево; следовательно, индикатор переноса принимает 0 (значение бывшего старшего бита в аккумуляторе), а индикатор нуля остается 0.

6.5. СОСТАВ КОМАНД ОПЕРАЦИЙ ПЕРЕДАЧИ ДАННЫХ

Команды передачи данных составляют третью категорию команд нашего типового микропроцессора. Они выполняют передачу данных из регистра в регистр, размещение данных в памяти, размещение извлеченных из памяти данных в УВВ и установление индикатора переноса (табл. 6.5). Почти все команды передачи не влияют на индикаторы МП. Эта группа содержит множество команд, данные могут быть переданы из любой ячейки памяти в любой регистр или наоборот. Каждая команда передачи содержит адреса источника и назначения данных. Способы адресации ориентированы на то, где и как осуществляется поиск данных.

Таблица 6.5. *Команды передачи данных типового микропроцессора*

Операция	Адресация	Мнемоника	КОП	Байты	Формат команд	Символика
Передать *L* в *A*	Регистровая	MOV *A*, *L*	7D	1	КОП	(A) ← (L)
Передать *H* в *A*	»	MOV *A*, *H*	7C	1	КОП	(A) ← (H)
Передать *A* в *L*	»	MOV *L*, *A*	6F	1	КОП	(L) ← (A)
Передать *A* в *H*	»	MOV *H*, *A*	67	1	КОП	(H) ← (A)
Передать *HL* в *PC*	»	PCHL	E9	1	КОП	(PC) ← (HL)
Передать *HL* в *SP*	»	SPHL	F9	1	КОП	(SP) ← (HL)
Загрузить *A* данными	Непосредственная	MVI *A*	3E	2	КОП данные	(A) ← (байт 2)
Загрузить *L* данными	»	MVI *L*	2E	2	КОП данные	(L) ← (байт 2)
Загрузить *H* данными	»	MVI *H*	26	2	КОП данные	(H) ← (байт 2)
Загрузить LOC (HL) в *A*	Косвенная регистровая	MOV *A*, *M*	7E	1	КОП	(A) ← ((H)(L))
Загрузить *HL* данными	Непосредственная	LXI *H*	21	3	КОП мл. байт ст. байт	(L) ← (байт 2) (H) ← (байт 3)
Загрузить *SP* данными	»	LXI *P*	31	3	КОП мл. байт ст. байт	(SP) ← (байт 2 + 3)
Загрузить HL из LOC	*Прямая*	LHL *D*	2A	3	КОП мл. адрес ст. адрес	(L) ← (байт 2 + 3) (H) ← ((байт 2 + 3)+1)
Загрузить *A* из LOC	»	LDA	3A	3	КОП мл. адрес ст. адрес	(A) ← ((байт 2 + 3))
Поместить *A* в LOC*aa*	»	STA	32	3	КОП мл. адрес ст. адрес	(адрес) ← (A)
Поместить *HL* в LOC	»	SHLD	22	3	КОП мл. адрес ст. адрес	(адрес) ← (L) (адрес + 1) ← (H)
Поместить *A* в LOC*aa* (HL)	Косвенная регистровая	MOV *M*,*A*	77	1	КОП	((H)(L)) ← (A)
Поместить *L* в LOC (HL)	То же	MOV *M*,*L*	75	1	КОП	((H)(L)) ← (L)
Поместить *H* в LOC (HL)	»	MOV *M*,*L*	74	1	КОП	((H)(L)) ← (H)
Ввести в *A* из порта в LOC*a*	Прямая	IN	DB	2	КОП адр. порта	(A) ← (адрес порта)
Вывести *A* в порт в LOC*a*	»	OUT	D3	2	КОП адр. порта	(адрес порта) ← (A)
Установить индикатор переноса	Неявная	STC	37	1	КОП	(CY) ← 1

() — содержимое; (()) — косвенная регистровая адресация; *PC*—счетчик команд; *SP* — указатель стека; устанавливается индикатор *CV* только для команды *STC*.

Рассмотрим первую команду передачи, записанную в табл. 6.5: необходимо их передать из регистра *L* в регистр *A* (аккумулятор). Команда ассемблера (MOV *A, L*) означает перемещение[1]. Следующая за мнемоникой буква (*A* — в нашем примере) указывает назначение, тогда как последняя (*L*) идентифицирует источник данных. Это покажется немного несовременно, но эта условность принята фирмой Intel для команд микропроцессоров Intel 8080/8085.

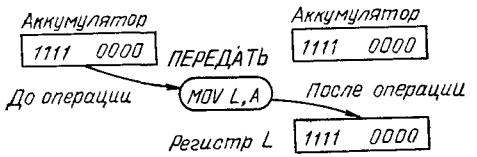


Рис. 6.20. Команда передачи данных из *A* в *L*

Обратимся к примеру на рис. 6.20. Источником данных является регистр *A* (аккумулятор), приемником — регистр *L*. Выполненная один раз команда не меняет содержимое аккумулятора.

На рис. 6.21 приведен другой пример. Здесь источником данных является пара регистров *HL*, приемником — 16-разрядный указатель стека. Если рассмотреть внимательно

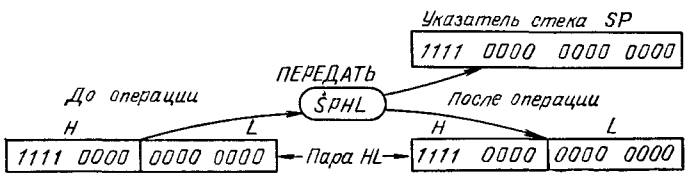


Рис. 6.21. Команда передачи данных из пары регистров *HL* в указатель стека *SP*

мнемонику команды передачи содержимого пары регистров *HL* в указатель стека *SP*, подтвердится, что назначение указывается первым (*SP*), а источник — последним (*HL*), что дает нам мнемонику SPHL.

Существует пять команд непосредственной загрузки данных. Эти команды используются очень часто для помещения начального значения в регистр МП в заданный момент, предшествующий программе. На рис. 6.22 приведен пример такой команды. Пара регистров *HL* емкостью 16 бит должна быть загружена данными, следующими непосредственно за КОП в памяти. Заметим, что согласно табл. 6.5 команда ЗАГРУЗИТЬ *HL* данными является трехбайтовой командой. Первый байт — это КОП (21H в нашем примере), тогда как два последующих байта являются байтами данных. Второй байт содержит МБ данных

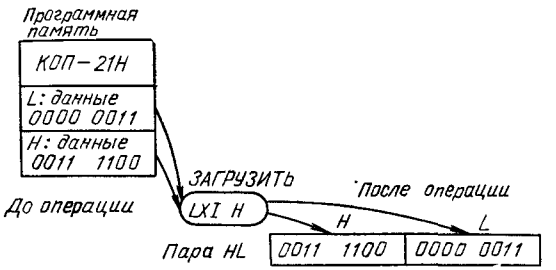


Рис. 6.22 Команда загрузки пары регистров *HL* с непосредственной адресацией

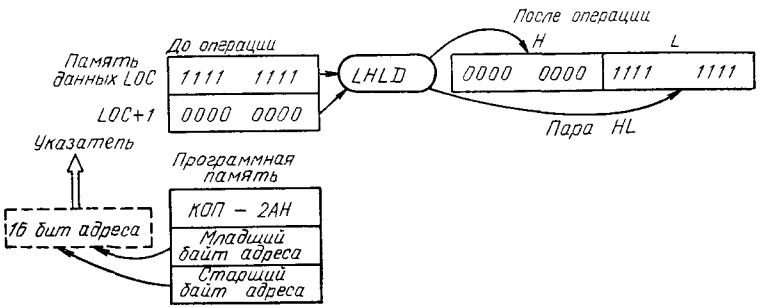


Рис 6.23. Команда загрузки пары регистров *HL* с прямой адресацией

(здесь 0000 0011), он помещается в регистр *L* (см. рис. 6.22), третий — СБ данных (здесь 0011 1100), который помещается в регистр *H* пары *HL*.

Следующий пример — использование команды ЗАГРУЗИТЬ прямо *HL* (LHLD) — приведен на рис. 6.23. Команда LHLD использует прямой способ адресации (согласно табл. 6.5), второй и третий байт являются 16-разрядными адресами памяти данных для загрузки. Следуем за событиями на рис. 6.22 исходя из трехбайтовой команды (внизу слева памяти), КОП второй — 2AH. Микропроцессор преобразует два следующих байта в 16-разрядный адрес

[1] MOV от *move* (англ.) — перемещать. — *Прим. пер.*

(в штриховой рамке), который служит указателем адреса памяти (LOC) и загружается в регистр *L*. Содержимое следующего байта памяти [адрес (LOC+1) в нашем примере] загружается в регистр *H*.

Типовой МП снабжен пятью командами размещения, которые пояснены в табл. 6.5. Рассмотрим пример использования команды прямого размещения содержимого *A*, приведенный на рис. 6.24. Содержимое аккумулятора (ре-

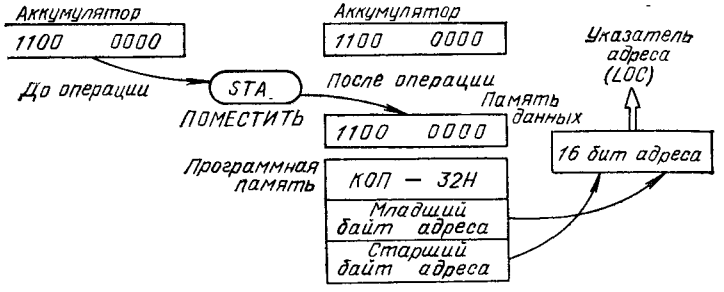


Рис. 6.24. Команда загрузки данных аккумулятора в память с прямой адресацией

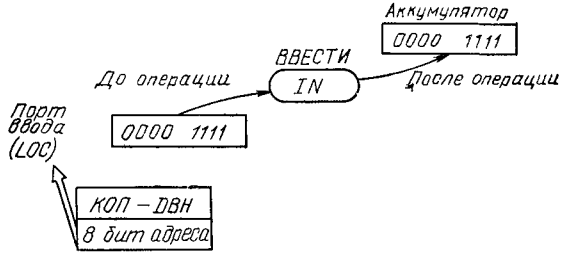


Рис. 6.25. Команда ввода с прямой адресацией

гистр *A*) помещается в память (LOC), на которую указывает 16-разрядный адрес, составленный вторым и третьим байтом команды. После выполнения ячейка памяти (LOC) и аккумулятор содержат те же данные, т. е. 1100 0000.

Команда ввода (третья строка снизу в табл. 6.5) идентична команде загрузки. Источник данных передачи является портом ввода, идентифицированным двоичным 8-разрядным числом ($0—255_{10}$), назначение этих данных — аккумулятор МП. Пример выполнения команды представлен

на рис. 6.25. Данные порта ввода 0000 1111, на который указывает второй байт команды, передаются в аккумулятор, исходя из порта ввода, идентифицированного LOC.

Упражнения

Все последующие упражнения рекомендуем выполнять, обращаясь к табл. 6.5.

6.39. Что является источником и что назначением данных в команде MOV *H*, *A*?

6.40. Что является источником и что назначением данных в команде PCHL?

6.41. Индикаторы _____ (устанавливаются, не устанавливаются) большинством операций передачи данных.

6.42. На языке ассемблер мнемоникой команды загрузки на рис. 6.26 является _____.

6.43. Каково содержимое аккумулятора после операции загрузки на рис. 6.26?

6.44. Какой способ адресации используется командой MVI *A* на рис. 6.26?

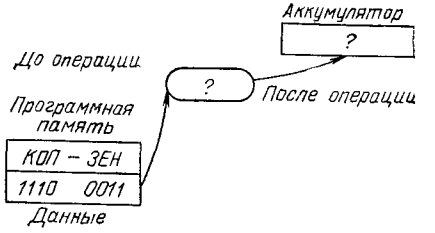


Рис. 6.26. К упражнениям 6.42—6.44

6.45. Команда MOV *A*, *M* имеет КОП 7EH. Это команда _____ (прямой, косвенной регистровой) адресации, и источником данных является область памяти _____ (данных, программы), на которую указывает пара регистров *HL*.

6.46. Код операции команды SHLD — 22H, это команда _____ (прямой, непосредственной) адресации.

6.47. Формат команды поместить *HL* прямо в LOC с КОП 22H _____ байта. Два последних байта содержат информацию о _____ (данных, адресе).

6.48. В команде OUT (ВЫВЕСТИ) на рис. 6.27 источником данных является регистр _____ (*A*, *H*, *L*).

6.49. В примере на рис. 6.27 первый байт памяти соответствующей команды содержит КОП _____ (шестнадцатеричный).

6.50. После операции вывода, приведенной на рис. 6.27, порт вывода номер _____ (шестнадцатеричный) содержит данные _____ (назвать 8 бит).

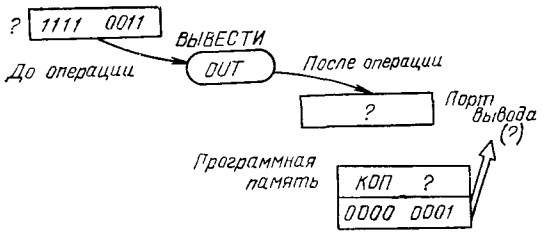


Рис. 6.27. К упражнениям 6.48—6.50

Решения

6.39. Источник — регистр *A*, назначение — регистр *H*. **6.40.** Источник — пара регистров *HL*, назначение — 16-разрядный счетчик команд. **6.41.** Не устанавливаются. **6.42.** MVI *A*. **6.43.** 1110 0011. **6.44.** Непосредственная адресация. **6.45.** Косвенной регистровой; данных. **6.46.** Прямой. **6.47.** 3; адресе. **6.48.** *A* или аккумулятор. **6.49.** D 3H. **6.50.** 01H; 1111 0011.

6.6. СОСТАВ КОМАНД ОПЕРАЦИЙ ВЕТВЛЕНИЯ

Команды ветвления составляют четвертую группу команд, которой снабжен типовой МП. Они приведены в табл. 6.6. Заметим, что описание относится к командам перехода. Действительно, термины *ветвление* и *переход* приводятся в этой главе как синонимы. Однако некоторые разработчики усматривают разницу между ними. Эти команды называют иногда *командами передачи управления*. Там, где они будут встречаться, мы все-таки их будем квалифицировать как команды перехода, следуя договору с фирмой Intel.

Обычно микро-ЭВМ выполняет команды последовательно. Шестнадцатиразрядный счетчик команд типового микропроцессора хранит всегда адрес следующей извлекаемой из памяти команды до ее выполнения. Содержимое его обычно повышается в каждом счете. Команды ветвления или перехода являются средством изменения значения содержимого счетчика команд и, следовательно, изменения нормальной последовательности выполнения программы.

Эти команды разделены на две группы: *безусловного перехода* и *условного перехода*. Первая команда в табл. 6.6 является командой безусловного перехода. Команда ПЕРЕЙТИ непосредственной адресации является трехбайтовой, используемой для изменения специфического адреса

Таблица 6.6. Команды ветвления или перехода

Операция	Адресация	Мнемоника	КОП	Байты	Формат команды	Символика
Перейти в LOC	Непосредственная	JMP	C3	3	КОП мл. адрес ст. адрес	(PC) ← (адрес)
Перейти в LOC, если 0	»	JZ	CA	3	КОП мл. адрес ст. адрес	Если $Z=1$, то (PC) ← (адрес)
Перейти в LOC, если не 0	»	JNZ	C2	3	КОП мл. адрес ст. адрес	Если $Z=0$, то (PC) ← (адрес)
Перейти в LOC, если перенос	»	JC	DA	3	КОП мл. адрес ст. адрес	Если $CY=1$, то (PC) ← (адрес)
Перейти в LOC, если нет переноса	»	JNC	D2	3	КОП мл. адрес ст. адрес	Если $CY=0$, то (PC) ← (адрес)

в счетчике команд МП. На рис. 6.28 приведен пример использования такой команды безусловного перехода. Здесь адрес 2000Н загружен в счетчик команд, информация о нем следует непосредственно за КОП, поэтому адресация называется непосредственной. Заметим на рис. 6.28, что младшая часть адреса находится во 2-м байт памяти,

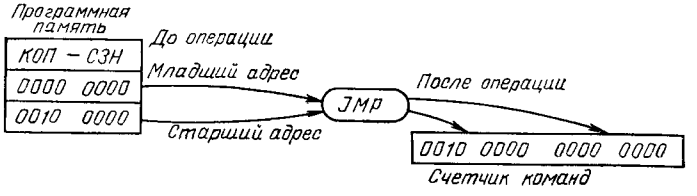


Рис. 6.28. Команда безусловного перехода

а старшая является содержимым 3-го байт памяти. Одна команда этого типа будет использована для запуска счетчика команд в момент начала выполнения новой программы. Мы можем рассматривать команду ветвления или безусловного перехода как способ загрузки новой информации об адресе в счетчик команд.

Четыре последние команды ветвления в табл. 6.6 являются командами условного перехода. Эти команды повлекут за собой непосредственно загрузку адреса, только если будут выполнены специальные условия. В противном случае счетчик команд будет инкрементирован нормально. На рис. 6.29 показан пример команды перейти, если 0. В этом случае счетчик команд 2013Н до операции будет нормально инкрементирован, если только индикатор нуля не установлен в 1. Микропроцессор проверяет это и находит 1, значит, результатом последней логической или арифметической операции был 0. Условия перехода выполнены, и МП загружает новый адрес 2008Н в счетчик команд. Этот новый адрес поступает из памяти программы; команда опроса использует непосредственную адресацию. Следующей выполняемой командой будет команда размещения данных в памяти по адресу 2008Н (но не по адресу 2013Н), как мы могли бы предположить при последовательном выполнении программы. Наш МП перешел к новой ячейке памяти программы. Отметим, что пример на рис. 6.29 показывает переход назад, что случается наиболее часто.

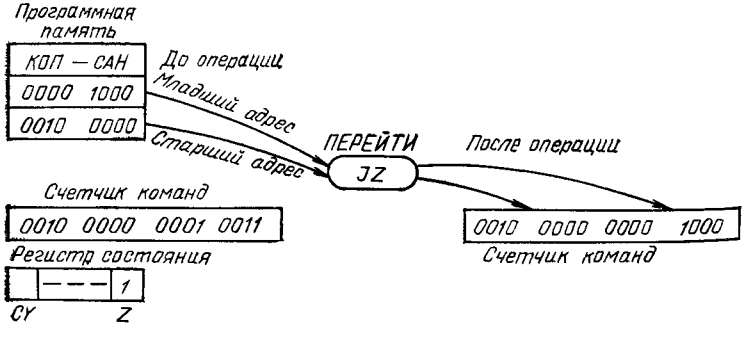


Рис. 6.29. Команда перехода, если нулевой результат

Команды перехода или ветвления существуют практически во всех программах МП. Они очень эффективны как средство принятия решений и удобны для формирования циклов программы.

Упражнения

6.51. Какие два других названия используются для команд ветвления?

6.52. Команды _____ (логические, условного перехода) являются командами принятия решений.

6.53. См. табл. 6.6. Мнемоникой команды перехода ассемблера на рис. 6.30 будет_____.

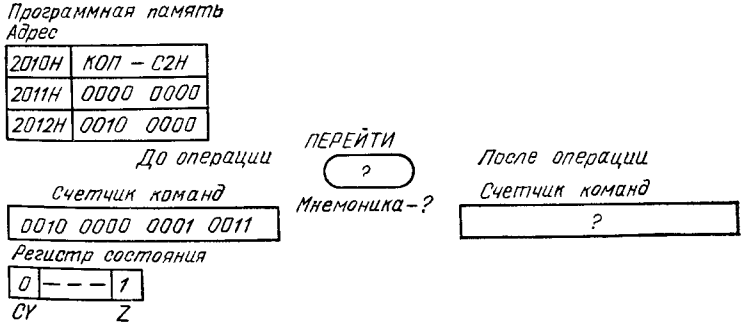


Рис. 6.30. К упражнениям 6.53 и 6.54

6.54. См. табл. 6.6. Содержимым счетчика команд на рис. 6.30 станет _____ (шестнадцатеричное).

6.55. См. табл. 6.6. Мнемоника команды перейти, если перенос на рис. 6.31 — JC, ее КОП _____.

6.56. Каким будет содержимое (шестнадцатеричное) счетчика команд после команды перехода, приведенной на рис. 6.31?

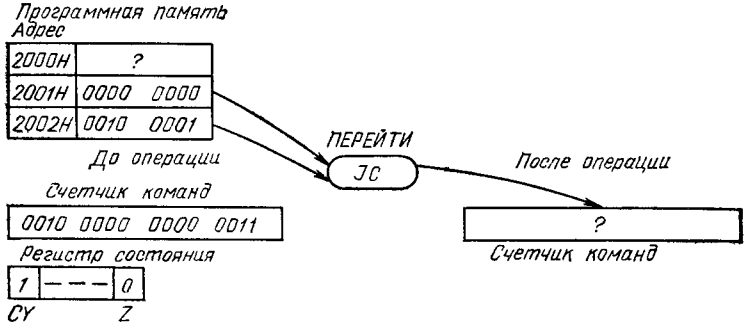


Рис. 6.31. К упражнениям 6.55 и 6.56

Решения

6.51. Перехода, принятия решения. **6.52.** Условного перехода. **6.53.** Операция перейти, если не 0, ее КОП C2H, мнемоника JNZ. **6.54.** Индикатор нуля установлен в 1, что означает равенство нулю результата выполненной последней логической или арифметической операции. Условие перехода не выполнено, и содержащийся во 2-м и 3-м байт адрес команды не загружается в счетчик команд. Его содержимое 2013H (что было до команды JNZ) будет адресом следующей выполняемой команды. **6.55.** DAH. **6.56.** Индикатор переноса содержит 1, условие перехода выполнено. Микропроцессор загружает новый адрес 2100H (выполняемой затем команды) в счетчик команд.

6.7. СОСТАВ КОМАНД ОПЕРАЦИЙ ВЫЗОВА ПОДПРОГРАММ И ВОЗВРАТА В ОСНОВНУЮ ПРОГРАММУ

Эти команды составляют пятую категорию состава команд типового МП. Их только две, и они приведены в табл. 6.7. Команды вызова (CALL) и возврата (RET) всегда используются парами. При их выполнении индикаторы не изменяются.

Трехбайтовая команда CALL используется основной программой для перехода МП (или ветвления) к подпро-

Таблица 6.7. *Команды вызова подпрограмм и возврата типового микропроцессора*

Операция	Адресация	Мнемоника	КОП	Байты	Формат команды	Символика
Вызов подпрограммы	Непосредственная, косвенная регистровая	CALL	CД	3	КОП мл. адрес ст. адрес	((SP)—1)←(PCH) ((SP)—2)←(PCL) (SP)←(SP—2) (PC)←(адрес)
Возврат из подпрограммы	Косвенная регистровая	RET	C9	1	КОП	(PCL)←((SP)) (PCH)←((SP)+1) (SP)←(SP)+2

() — содержимое; (()) — косвенная регистровая адресация; *PC* — счетчик команд; *SP* — указатель стека.

граммe. В примере на рис. 6.32 подпрограмма является короткой последовательностью команд, целью которой является создание интервалов времени в течение 1 с. Когда МП передает первую команду CALL по адресу 1000H, он находит адрес перехода в двух следующих байтах программы. Адрес следующей команды за CALL отправляется в стек (не показанный на рис. 6.32), и МП переходит тогда в начало подпрограммы по адресу 1000H. Команды, кото-

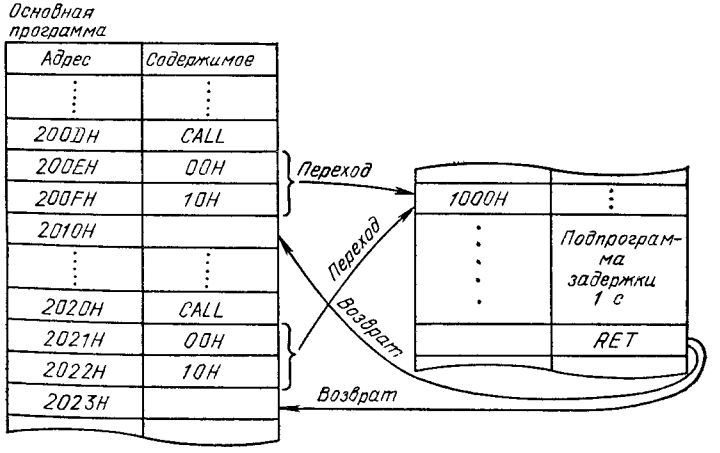


Рис. 6.32. Взаимодействие основной программы и подпрограммы при командах CALL (ВЫЗОВ) и RET (ВОЗВРАТ)

рые составляют эту программу счета времени, выполняются, пока МП не передаст команду возврата (RET).

Сохраняющийся в стеке адрес (2010H) отыскивается счетчиком команд, и МП продолжает выполнение основной программы, принимая ее там, где он ее покинул. Это нормальное выполнение продолжается до тех пор, пока МП не встретит другую команду вызова по адресу 2020H. Микропроцессор сохраняет адрес следующей команды (2023H) в стеке и переходит на подпрограмму, начинающуюся адресом 1000H. После завершения выполнения этой подпрограммы команда возврата извлекает из стека адрес следующей команды основной программы и загружает его в счетчик команд. Данная подпрограмма может быть использована много раз в ходе выполнения одной и той же основной программы. Подпрограмма может быть расположена в ОЗУ или ПЗУ.

Команда вызова сочетает функции операций загрузки в стек и перехода. Использование ее показано на рис. 6.33. Сначала она загружает в стек содержимое счетчика команд. Затем счетчик команд должен быть загружен новым адресом для выполнения перехода в подпрограмму. Последовательность операций на рис. 6.33 отмечена номерами в кружках.

1. Указатель стека декрементирован от 210AH до 2109H.

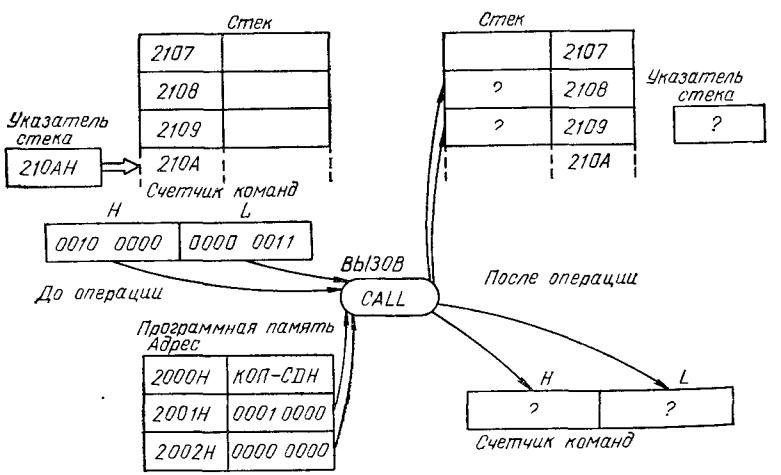


Рис. 6.33. Команда вызова подпрограммы

2. Старший байт счетчика команд загружается в стек по адресу 2109H.

3. Указатель стека декрементирован от 2109H до 2108H.

4. Младший байт счетчика команд загружается в ячейку памяти по адресу 2108H.

5. Младший байт адреса (второй байт памяти) загружается в младший байт счетчика команд.

6. Старший байт адреса (третий байт памяти) загружается в старший байт счетчика команд.

Здесь МП ответвляется по адресу, на который указывает счетчик команд (в приведенном примере 1000H), являющемуся адресом начала подпрограммы.

Рассмотрим теперь пример применения команды возврата в основную программу RET (рис. 6.34). По команде

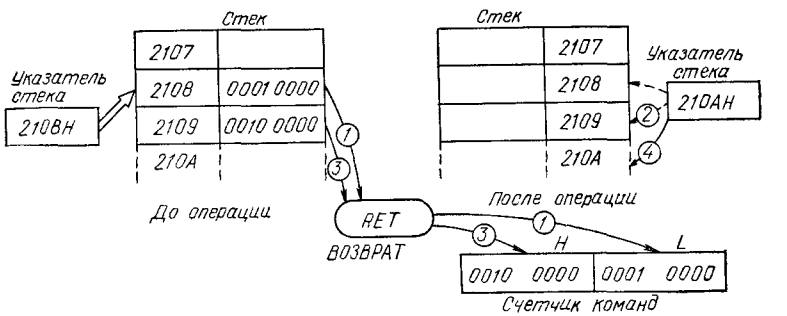


Рис. 6.34. Команда возврата из подпрограммы

возврата содержимое стека должно быть передано в счетчик команд. Проследим последовательность выполнения команды по номерам, взятым на рис. 6.34 в кружки.

1. Вершина стека (адрес 2108H) извлекается, ее содержимое передается в младший байт счетчика команд.

2. Указатель стека инкрементируется от 2108H до 2109H.

3. Вершина стека (теперь ее адрес 2109H) извлекается, ее содержимое передается в старший байт счетчика команд.

4. Указатель стека инкрементируется от 2109H до 210AH.

Теперь счетчик команд содержит 16-разрядный адрес (2010H) следующей извлекаемой из памяти команды.

Упражнения

6.57. Команда _____ (вызова, возврата) заставляет МП уйти из основной программы, тогда как команда _____ (вызова, возврата) заставляет уйти из подпрограммы.

6.58. Команда возврата является _____ байтовой командой.

Последних 2 байта содержат адрес _____ (основной программы, подпрограммы).

6.59. Операция вызова выполняет одновременно функции _____ (загрузки в стек, извлечения из стека) и _____ (сложения, перехода).

6.60. См. рис. 6.35. Каково содержимое стека по адресу 2108H после операции вызова?

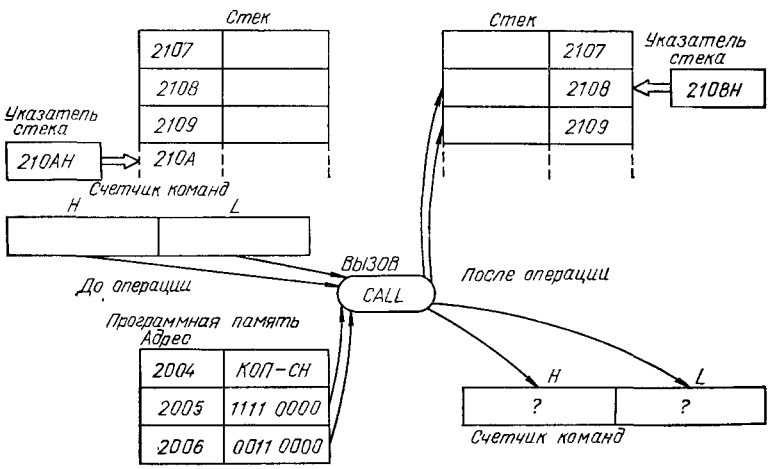


Рис. 6.35. К упражнениям 6.60—6.63

6.61. См. рис. 6.35. Каково содержимое стека по адресу 2108H после операции вызова?

6.62. См. рис. 6.35. Каким будет содержимое указателя стека после операции вызова (в H-коде)?

6.63. См. рис. 6.35. Каково содержимое счетчика команд после команды вызова (в H-коде)?

6.64. Команды вызова и возврата всегда используются _____ (в одиночку, парно).

Решения

6.57. Вызова; возврата. **6.58.** Трех; подпрограммы. **6.59.** Загрузки в стек; перехода. **6.60.** 0010 0000, старший байт счетчика команд. **6.61.** 0000 0011. **6.62.** 2108H. **6.63.** 0010H. **6.64.** Парно.

6.8. СОСТАВ КОМАНД ПРОЧИХ ОПЕРАЦИЙ

Эти команды составляют последнюю категорию, которыми наделен типовой микропроцессор. Они сведены в табл. 6.8 и содержат команды помещения в стек, извлечения из стека, отсутствия операции и команду остановки. При их выполнении индикаторы не изменяются.

Команды помещения в стек и извлечения из него уже упоминались в § 5.5. Они используются всегда парно, так как то, что в стек помещается, должно быть из него извлечено. Они широко распространены при использовании под-

Таблица 6.8. *Прочие команды типового микропроцессора*

Операция	Адресация	Мнемоника	КОП	Байт	Формат команды	Символика
Поместить в стек *A* и индикаторы	Косвенная регистровая	PUSH *PSW*	F5	1	КОП	((SP) — 1) ← (A) ((SP)—2) ← (индикаторы) (SP) ← (SP) — 2
Поместить в стек *HL*	То же	PUSH *H*	E5	1	КОП	((SP) — 1) ← (H) ((SP) — 2) ← (L) (SP) ← (SP) — 2
Извлечь из стека *A* и индикатора	»	POP *PSW*	F1	1	КОП	(индикаторы) ← ((SP)) (A) ← ((SP) + 1) (SP) ← (SP) + 2
Извлечь из стекла *HL*	»	POP *H*	E1	1	КОП	(L) ← ((SP)) (H) ← ((SP) + 1) (SP) ← (SP) + 2
Нет операций	Неявная	NOP	00	1	КОП	(L) ← ((SP)) (H) ← ((SP) + 1)
Останов	Неотделим	HLT	76	1	КОП	(SP) ← (SP) + 2 (PC) ← (PC) + 1

() — содержимое; (()) — косвенная регистровая адресация; *A* — аккумулятор; *SP* — указатель стека.

программ. Команда поместить в стек содержимое аккумулятора *A* и индикаторов могла бы быть, например, первой командой подпрограммы, приведенной на рис. 6.32. Она сохранила бы содержимое аккумулятора и индикаторов независимо от подпрограммы. Точно перед операцией возврата на рис. 6.32 команда извлечь из стека *A* и индикаторы восстановила бы начальное содержимое аккумулятора и индикаторов.

Рассмотрим первую команду поместить в стек *A* и индикаторы (PUSH *PSW*). Часть *PSW* соответствует слову состояния программы [1], которое в данном случае является содержимым аккумулятора и регистра состояния (индикаторов). Команда PUSH *PSW* является однобайтовой, содержимое аккумулятора помещается первым, а регистра состояния — вторым. Для более подробного ознакомления с командами помещения и извлечения из стека (PUSH) (POP) соответственно следует обратиться к § 5.5.

Команда НЕТ ОПЕРАЦИЙ соответствует отсутствию всякого выполнения операций в течение 1 или 2 мкс. Это однобайтовая команда, единственным эффектом которой является инкремент счетчика команд. Никакой другой регистр не затрагивается. Эта команда используется как дополнение (когда одна или две команды отменены в ходе наладки) и связывает две части программы так, чтобы МП мог обратиться от одной к другой. Она может также служить для ввода интервала времени в цикл временно́й задержки.

Команда ОСТАНОВ используется в конце программы для остановки микропроцессора. В этом случае только СБРОС или команда вызова прерывания может позволить новый запуск типового микропроцессора.

Упражнения

Все следующие упражнения имеют ссылку на табл. 6.8, к которой следует обращаться постоянно.

6.65. Какой КОП команды PUSH *PSW* использован на рис. 6.36?

6.66. В ходе выполнения команды PUSH *PSW* (рис. 6.36) указатель стека_____(инкрементирован, декрементирован) первым, затем содержимое_____(аккумулятора, регистра состояния) передается в стек по адресу _____.

[1] *Program Status Word* (англ.) — слово состояния программы. — *Прим. пер.*

6.67. Каково содержимое указателя стека и его ячеек памяти 2208H и 2209H после операции извлечения из стека на рис. 6.36?

6.68. Если команда PUSH *PSW* использоваиа для размещения содержимого регистров МП в стек, команда с мнемоникой_____будет использована для восстановления содержимого регистров.

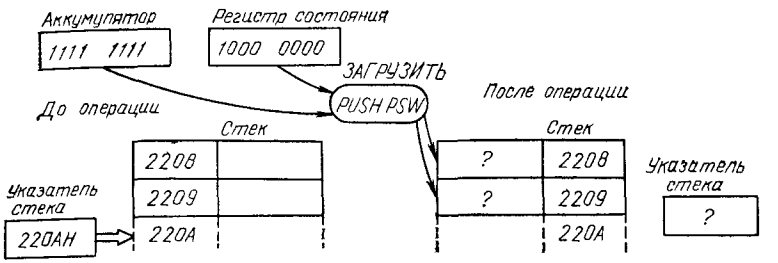


Рис. 6.36. К упражнениям 6.65—6.67

6.69. Команды загрузки в стек и извлечения из стека используются всегда парами в_____(сбросе, подпрограмме).

6.70. Команда_____(останов, нет операций) приводит к прекращению обработки (ни извлечения, ни выполнения команд) до получения микропроцессором внешнего сигнала сброса или вызова прерывания.

6.71. Обычно команда_____(останов, нет операций) помещается в конце программы.

Решения

6.65. F5H. **6.66.** Декремеитирован; аккумулятора; 2209H. **6.67.** 2208H, затем для стека 2208H=1000 0000, регистра состояния 2209H= =1111 1111. **6.68.** POP *PSW*. **6.69.** Подпрограмме. **6.70.** Останов. **6.71.** Останов.

6.9. ЗАПИСЬ ПРОГРАММЫ

Программист должен отлично знать соответствующий состав команд, быть хорошо знаком с расположением регистров, т. е. знать общую архитектуру микро-ЭВМ.

Этапы развития программы могут быть представлены следующим образом:

1. Определить и проанализировать задачу.
2. Начертить структурную схему решения.
3. Записать программу на ассемблере.
4. Записать или генерировать программу в кодах машины.
5. Запустить программу на решение.
6. Документировать программу.

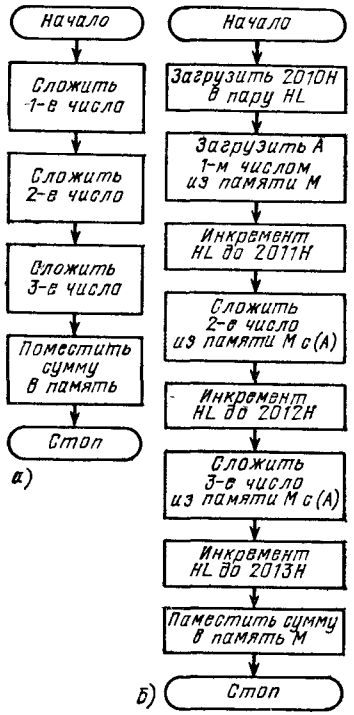


Рис. 6.37. Функциональная структурная схема программы сложения трех чисел (*а*) и ее развитие (*б*)

Рассмотрим задачу записи программы, целью которой является сложение содержимого трех последовательных ячеек памяти и размещение суммы в памяти. На рис. 6.37, *а* представлена функциональная структурная схема решения этой задачи. Она не содержит адекватных деталей, которые позволяют перевести ее прямо в сегмент программы на ассемблере или машинном языке.

На рис. 6.37, *б* показана подробная функциональная схема организации решения, которая в ходе программирования зависит от системы, в то время как функциональная схема может быть использована для любого МП.

Затем следует записать версию подробной структурной схемы организации решения на ассемблере (такой сегмент программы приведен в табл. 6.9) в четыре типовых поля: метка; мнемоника; операнд; комментарий. Метки нас пока не интересуют. Комментарии оказывают большую помощь в понимании действия каждой команды программы. Сравнивая подробную функциональную схему организации решения на рис. 6.37, *б* и список команд на ассемблере в табл. 6.9, мы установим, что каждая рамка схемы соответствует одной эквивалентной команде. Возьмем, например, первую прямоугольную рамку на рис. 6.37, *б*: команда загрузить 2010H в пару *HL* эквивалентна команде на ассемблере *L XI H* 2010H, т.е. загрузить адрес 2010H в пару *HL* в табл. 6.9.

Таблица 6.9. *Программа на ассемблере сложения трех чисел и размещения их суммы в памяти*

Метка	Мнемоника	Операнд	Комментарий
	LXI	*H*, 2010H	Загрузнть адрес 2010H в пару *HL*. Пара *HL* используется как указатель адреса
	MOV	*A*, *M*	Поместить первое число в ячейку памяти 2010H в аккумулятор
	INX	*H*	Инкрементировать пару *HL* до 2010H
	ADD	*M*	Сложить второе число в ячейке памяти 2011H с содержимым аккумулятора
	INX	*H*	Инкрементировать пару *HL* до 2012H
	ADD	*M*	Сложить третье число в ячейке памятн 20112H с содержимым аккумулятора
	INX	*H*	Инкрементировать пару *HL* до 2013H
	MOV	*M*, *A*	Поместить сумму, содержащуюся в аккумуляторе, в ячейку памяти 2013H
	HLH		Остановить МП

Записанная версия должна быть переведена в состав бит из единиц и нулей, понимаемых микропроцессором, называемых машинным кодом. Это можно сделать вручную или с помощью специальных программ (на ассемблере). Учебное кодирование выполняется следующим образом:

1. Найти КОП каждой мнемоники в таблице состава команд (в нашем случае табл. 6.3—6.8).

2. Определить операнды (данные и адреса), когда это необходимо, передать командами из нескольких байт.

3. Установить адреса памяти в последовательности каждой команды и операнда.

Выполним наш пример сложения содержимых трех последовательных ячеек памяти и размещения их суммы в четвертой. Версия на ассемблере представлена в табл. 6.10. Две левые колонки содержат установленные адреса памяти и шестнадцатеричный КОП (или операнд). Содержимое соответствует 8-разрядным словам, помещенным в память.

Мнемоникой и операндом первой строки команды на машинном языке являются LXI *H*, 2010H, для которых должен быть установлен КОП и в этом случае два операнда. Необходимая информация для этого типа команд (передать данные) находится в табл. 6.5. Мнемоника LXI *H*

Таблица 6.10. *Программа сложения и размещения суммы в машинном коде и на ассемблере*

Шестнадцатеричные		Мнемоника	Операнд	Комментарий
адрес	содержимое			
2020	21	LXI	*H*, 2010*H*	Загрузнть адрес 2010Н в пару *HL* указатель адреса
2021	10			
2020	20			
2023	7E	MOV	*A*, *M*	Загрузнть первое числ в ячейке памяти 2010Н в аккумулятор
2024	23	INX	*H*	Инкрементировать пару *HL* до 2011Н
2025	86	ADD	*M*	Сложнть второе число в ячейке памятн 2011Н с содержнмым аккумулятора
2026	23	INX	*H*	Инкрементнровать пару *HL* до 2012Н
2027	86	ADD	*M*	Сложить третье чнсло в ячейке памятн 2012Н с содержнмым аккумулятора
2028	23	INX	*H*	Инкрементировать пару *HL* до 2013Н
2029	77	MOV	*M*, *A*	Поместить содержащуюся в аккумуляторе сумму в ячейку памятн 2013Н
202A	76	HLT		Остановить МП

в середине таблицы и ее КОП 21Н — то, что упоминается в колонке «содержимое» в табл. 6.10. Отметим, что эта команда согласно табл. 6.5 требует два дополнительных байта данных. Младший (10Н в этом примере) занесен первым по адресу 2021Н, а старший (здесь 20Н) занесен по адресу 2022Н. Другие мнемоники в табл. 6.10 соответствуют однобайтовым командам. Шестнадцатеричные КОП находятся в соответствующей таблице и переносятся в колонку «содержимое». Программа в левых колонках записана в машинном коде и может быть введена в систему.

Затем нужно проверить программу — удовлетворительно ли она проходит, эта операция представляет собой *процесс отладки программы* (поиск и устранение ошибок). В конечном счете программа вводится в систему обычно с известными данными и результаты проверяются. Выполняют несколько тестов.

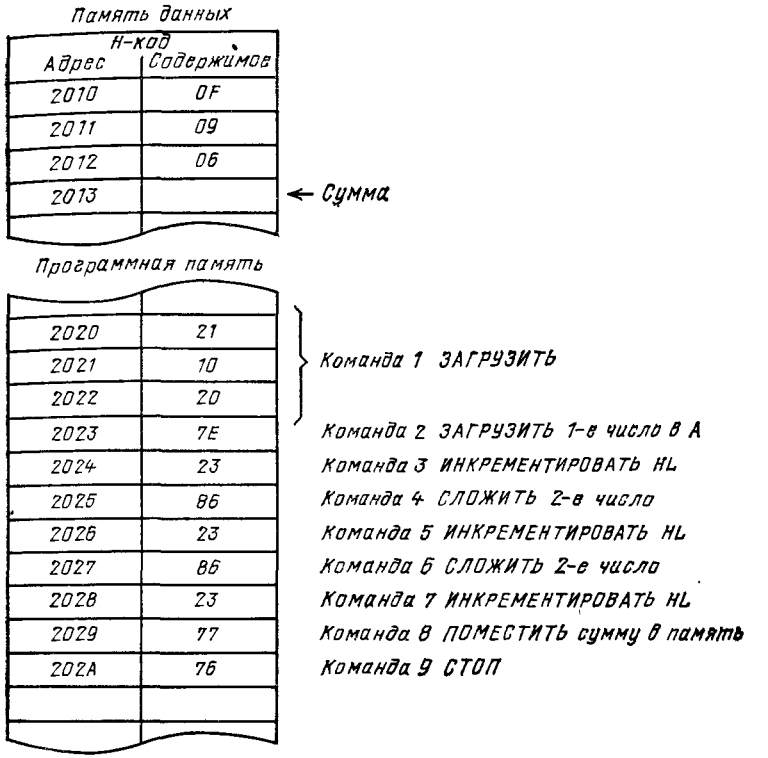


Рис. 6.38. Воображаемая память программы сложения трех чисел и помещения суммы в память

Последним этапом программирования является *документирование*. Документировать программу значит дать ее описание, указывающее, в какой последовательности должны выполняться операции. Сюда входят схемы вычислений, списки, данные, адреса, используемые подпрограммы и комментарии. Сильно развитые программы требуют тщательно разработанной документации.

Рассмотрим пример, представленный на рис. 6.38. В этом случае машинный код был введен в память программы по адресам 2020Н—202АН. Слагаемые (0FH, 09H, 06H) в рассматриваемом примере введены в память по адресам 2010Н—2012Н. Мы можем проследить этап за этапом выполнение программы сложения.

Хотя программы для реальных микро-ЭВМ довольно большие, они состоят из групп сегментов, идентичных тем, что мы только что видели. Обычно каждый сегмент записан и тестирован прежде, чем собрана для окончательного тестирования полная программа. Описанный состав команд является только одним из многих способов решения поставленной задачи. Программа этого типа называется последовательной в том смысле, что она не содержит ни ветвлений, ни переходов.

Упражнения

6.72. Программист должен знать архитектуру системы, состав команд МП, воображаемую память системы и регистры МП. Таким образом, регистры, предоставляемые программисту, даются обычно на схеме, называемой _____(моделью, составом) программирования.

6.73. Перечислить последовательность шести приведенных этапов программы.

6.74. Графическое представление задачи (такое, как на рис. 6.37) называется_____.

6.75. После разработки подробной структурной схемы решения (см. рис. 6.37) следующим этапом является запись программы на_____(ассемблере, машинном языке).

6.76. При записи версии программы в машинных кодах _____(адреса, прерывания) памяти устанавливаются командами.

6.77. Версия программы в машинных кодах использует коды _____ (восьмеричные, операции) и операнды в _____ (десятеричной, шестнадцатеричной) форме.

6.78. См. рис. 6.38. Каково содержимое пары регистров *HL* после выполнения команды *1*?

6.79. См. рис. 6.38. Каково содержимое аккумулятора после выполнения команды *4*?

6.80. См. рис. 6.38. Каково содержимое аккумулятора после выполнения команды *6*?

6.81. См. рис. 6.38. Каково содержимое ячейки памяти 2013H после выполнения команды *7*?

6.82. См. рис. 6.38. Каково содержимое ячейки памяти 2013H после выполнения команды *8*?

6.83. Как называется этап тестирования программы?

6.84. См. рис. 6.38. Программа без ветвлений и переходов называется _____.

Решения

6.72. Моделью. **6.73.** 1. Постановка и анализ задачи. 2. Построение структурной схемы решения. 3. Запись программы на языке ассемблер. 4. Запись или выдача программы в машинных кодах. 5. Запуск программы. 6. Документированне программы. **6.74.** Структурной схемой решения. **6.75.** Ассемблере. **6.76.** Адреса. **6.77.** Операции; шестнадцатеричной. **6.78.** 2010H. **6.79.** OFH+09H=18H, что является содержимым аккумулятора. **6.80.** OFH+09H+06H=1EH. **6.81.** Сумма еще не помещена, содержнмое этой ячейкн памятн непредсказуемо. **6.82.** OFH++09H+06H=1EH. **6.83.** Запуск программы (отладка). **6.84.** Последовательной.

6.10. СПОСОБЫ АДРЕСАЦИИ

Рассмотрим команду сложения. В случае типового МП мы предположили, что одно из слагаемых было в аккумуляторе. В таком случае откуда поступает второе слагаемое и как оно находится? Многочисленные способы решения этой задачи называются способами адресации. В настоящей главе мы упоминали уже способы адресации во многие регистры, однако в этом вопросе имеется ряд тонкостей. Рассмотрим их подробнее.

Способы адресации нашего типового МП следующие: 1) неявный; 2) регистровый; 3) непосредственный; 4) прямой; 5) косвенный регистровый.

Два первых (регистровый и неявный) касаются операндов, расположенных в самом МП. Три последних (непосредственная, прямая и косвенная регистровая) — операндов, расположенных вне МП, т. е. в ячейках памяти или портах УВВ. Эти способы адресации присущи МП Intel 8080/8085.

Рассмотрим пример команды с *неявной адресацией.* Команда ВОССТАНОВИТЬ индикатор переноса принадлежит к группе команд передачи данных (см. табл. 6.5) и является однобайтовой, потому что дополнительные данные бесполезны для ее выполнения. Все события происходят в МП. Это действие представлено на рис. 6.39, *а*, не нужно искать данные или адреса в других регистрах МП, в памяти или портах УВВ. Команда STC восстанавливает индикатор переноса без воздействия на другие регистры или индикаторы.

В случае *регистровой адресации* операнд отыскивается во внутреннем регистре МП. Рассмотрим пример команды

ВЫЧЕСТЬ *L* из *A*, принадлежащей к группе арифметических команд (см. табл. 6.3). Регистровые операции (как и неявные) всегда являются однобайтовыми, потому что они не требуют данных и адресов вне МП. Все события происходят в МП. Операция показана на рис. 6.39, *б*. Первое число (здесь 0000 1111) находится в аккумуляторе, другое (0000 0011) является содержимым регистра *L*. После выполнения команды результат (0000 1100) помещается в аккумулятор.

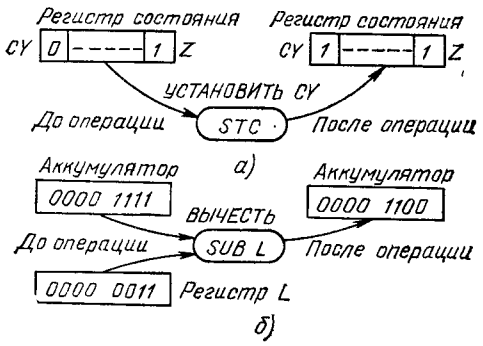


Рис. 6.39. Команды установки индикатора переноса (*а*) и вычитания (А) из (L) (*б*)

В случае *непосредственных команд* операнд поступает в программную память из следующего байта (иногда из двух следующих байтов). Операнд следует в команде непосредственно за КОП. Рассмотрим пример этого типа команд. Команда загрузить данные в *SP*, принадлежащую группе передачи данных (табл. 6.5) — это трехбайтовая команда. Обычно непосредственные команды точно определяются 2 или 3 байт. В примере, показанном на рис. 6.40,

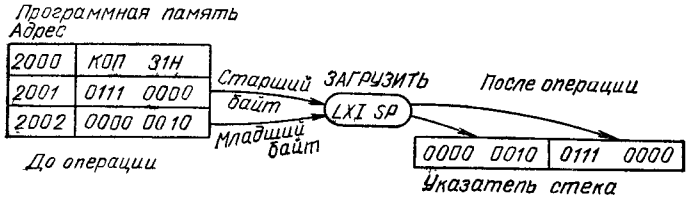


Рис. 6.40. Команда загрузки указателя стека

2 байт данных берутся в памяти программы и помещаются в указатель стека. Младший байт 0111 0000 загружен первым, затем старший байт 0000 0010. Непосредственные команды удобны для осуществления загрузок инициализированных регистрами МП или указателем стека.

В случае *прямых команд* 2-й и 3-й байт памяти прямо указывают на адрес операнда. Они являются адресами в прямом способе адресации, тогда как при непосредственной адресации эти же байты были операндами. Операция передать данные из порта УВВ в *A* (группа команд передачи данных, см. табл. 6.5) является примером прямой команды из 2 байт*. Обычно эти команды состоят из 2 или 3 байт. В примере, приведенном на рис. 6.41, мы можем

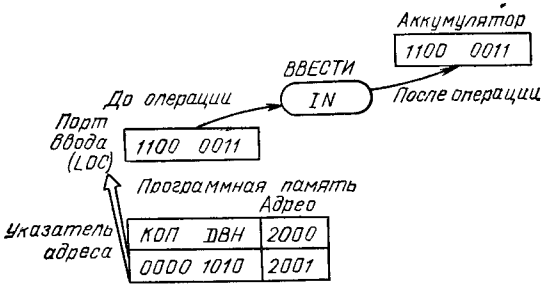


Рис. 6.41. Команда ввода

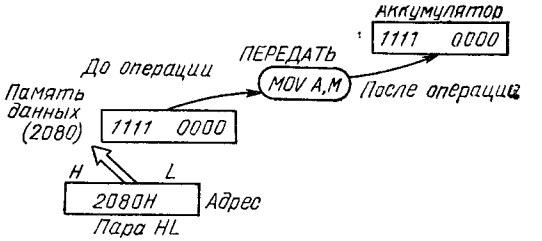


Рис. 6.42. Команда загрузки аккумулятора из памяти

проследить передачу содержимого 1100 0011 из порта ввода в аккумулятор. Адрес порта указан адресом, являющимся содержимым 2-го байт команды (здесь 0000 1010, или 10_{10}).

В случае *косвенных регистровых команд* пара регистров *HL* указывает на адрес операнда в памяти. Рассмотрим пример: загрузить LOC (*H* и *L*) в *A* с мнемоникой MOV *A*, *M* (группа команд передачи данных, см. табл. 6.5).

* Здесь двухбайтовая команда, поскольку для указания 8-разрядного адреса порта УВВ достаточно 1 байт (второго), 1-м байт команды является КОП. — *Прим. ред.*

Этот тип команд всегда однобайтовый. Обратимся к рис. 6.42, содержимое памяти по адресу .2080H (1111 0000 в этом случае) загружено в аккумулятор. Соответствующий адрес (2080H в нашем случае) памяти данных указан содержимым пары *HL*, которая здесь играет роль адресного регистра.

Другие МП снабжены иногда иными способами прямой адресации, а именно: нулевой или основной страницы; абсолютной; адресации действующей страницы или ожидаемой.

Упражнения

6.85. Назвать пять способов адресации в типовом микропроцессоре.

6.86. При использовании _____(прямой, неявной) адресации нет необходимости отыскивать операнд.

6.87. Когда 2-й и 3-й байт команды указывают на адрес операнда, речь идет о _____ (прямой, непосредственной) адресации.

6.88. Когда используют команду непосредственного сложения, операнд поступает из памяти_____(программы, данных).

6.89. В_____(регистровой, косвенной регистровой) команде пара регистров *HL* МП указывает на адрес операнда.

6.90. Когда два операнда (например, два числа при вычитании) являются содержимым внутренних регистров МП, может быть использован_____(непосредственный, регистровый) способ адресации.

6.91. Внутренние и регистровые команды являются _____ байтовыми.

6.92. Команды прямой адресации являются_____ байтовыми.

6.93. Команды косвенной регистровой адресации являются _____байтовыми.

Решения

6.85. Неявная, регистровая, непосредственная, прямая, косвенная регистровая. **6.86.** Неявной. **6.87.** Прямой. **6.88.** Программы. **6.89.** Косвенной регистровой. **6.90.** Регистровый. **6.91.** Одно. **6.92.** Двух или трех. **6.93.** Одно.

6.11. ВЕТВЛЕНИЕ ПРОГРАММЫ

Рассмотрим простую задачу сравнения двух чисел. Нужно найти наибольшее и поместить его в определенную ячейку памяти (см. структурную схему решения этой задачи на рис. 6.43). В двух первых прямоугольных кадрах представлена операция загрузки двух чисел в регистры *A* (аккумулятор) и *L* (данные). Следующий прямоугольный кадр соответствует операции сравнения, где содержимое регистра *L* вычитается из содержимого регистра *A*. Команды сравнения не разрушают содержимого регистров, но влияют на индикатор. Следующий кадр называется *знаком принятия решения*. Он завершает в программе этап решения и содержит поставленный вопрос [$(A) \geqslant (L)$?]. Если ответ на этот вопрос *Да*, программа продолжается последовательно и содержимое регистра *A* помещается в память, затем процессор останавливается. Если *Нет*, программа ответвляется вправо и содержимое регистра *L* помещается в память, затем МП останавливается. Использование знаков принятия решения приводит в программах к тому, что называется внутренним ветвлением программы.

В табл. 6.11 представлена на ассемблере задача, которую мы только что описали и в которой наибольшее число помещено в память 2040H. Заметим, что здесь аккумулятор *A* загружен числом 15H, а регистр *L* — числом 6H. Сравнивая (третья команда в этом примере), находим, что (A) > (L), и, следовательно, индикатор переноса (*CY*) сброшен в 0. Команда перехода (мнемоника JC в примере) проверяет индикатор переноса и находит (*CY*) = 0. Микропроцессор обращается тогда к следующей последовательной команде ПОМЕСТИТЬ (A) в память по адресу 2040H и наибольшее число помещено командой STA в память по адресу 2040H. Затем МП останавливает-

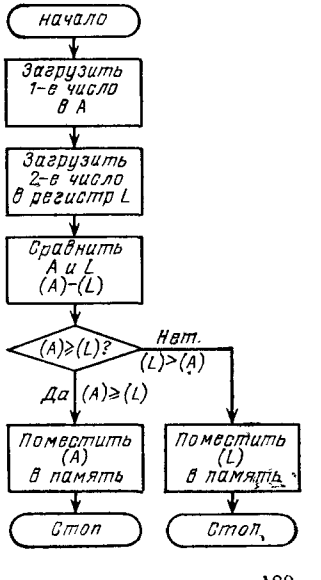


Рис. 6.43. Структурная схема программы сравнения двух чисел и помещения в память большего

Таблица 6.11. *Ветвящаяся программа на ассемблере*

Метка	Мнемоника	Операнд	Комментарий
	MVI	*A*, OFH	Загрузить первое число (15_{10}) в аккумулятор
	MVI	*L*, 06H	Загрузить второе число (6_{10}) в регистр *L*
	CMP	*L*	Сравнить (A) и (L). Индикатор $CY=1$, если (A)<(L)
	JC	STORE *L*	Перейти к STORE *L*, если $CY=1$ (если A<*L*); если нет, продолжать последовательно
	STA	2040H	Поместить (A) в ячейку памяти 2040H
	HLT		Остановить МП после помещения (A)
STORE *L*	MOV	*A*, *L*	Передать (L) в аккумулятор
	STA	2040H	Поместить (A) в ячейку памяти 2040H
	HLT		Остановить МП после помещения

ся (команда HLT). Три последние команды не были использованы в этом примере. Если обратиться к рис. 6.43, мы установим, что программа прошла последовательно.

Рассмотрим теперь выполнение такой же программы, но в случае, когда содержимое аккумулятора меньше содержимого регистра *L*. Программа остается та же (табл. 6.11). Содержимое аккумулятора — 0AH (10_{10}), содержимое регистра *L* — 0EH (14_{10}) здесь больше. Команда сравнения устанавливает индикатор *CY* в 1, так как (A)<(L). Команда ПЕРЕЙТИ, если индикатор переноса установлен в 1 (JC здесь), проверяет индикатор *CY* и определяет, установлен ли он. При этом изменяется содержимое счетчика команд, т. е. загружается адрес следующей выполняемой команды (STORE *L* в нашем примере). Заметим, что МП перешел через две команды для того, чтобы обратиться туда, где находится *символический адрес* STORE *L*. Микропроцессор выполняет тогда три последние команды, которые помещают содержимое регистра *L*, т. е. наибольшее число 0EH в ячейку памяти по адресу 2040H. Обратившись к схеме на рис. 6.43, мы установим, что программа выполнена последовательно до знака принятия решения, затем ответвилась вправо, заканчиваясь оператором *Стоп*.

Рассмотрим снова три последние команды в табл. 6.11. Целью этих команд является размещение содержимого регистра *L* в память по адресу 2040H. Наш микропроцессор не позволяет поместить содержимое регистра *L* прямо в память. Нужно последовательно передать сначала содержимое *L* в *A*, затем поместить содержимое *A* в память. Таким образом, часто оказывается, что МП с ограниченным составом команд (как в случае типового МП) использовать сложнее, чем устройства, более мощные по своим возможностям.

Таким образом, рассмотренный на структурной схеме способ ветвления осуществляется командами перехода (или ветвления), согласно которым принимаются решения, основанные на состоянии специальных индикаторов. Мы встретились здесь с использованием символического адреса (метки) при команде перехода. Ветвление очень широко используется при программировании.

Упражнения

6.94. Ромбовидный кадр на рис. 6.43 называется_____ (ромбом решения, знаком принятия решения).

6.95. Команда сравнить *A* и *L* в табл. 6.11 _____(изменяет, не изменяет) содержимое аккумулятора.

6.96. Команда сравнить *A* и *L* служит в приведенном примере для установки индикатора _____.

6.97. См. рис. 6.43. Если содержимые *A* и *L* равны, *CY* будет_____(установлен в 1, сброшен в 0) командой сравнения и использована ветвь _____ (*Да*, *Нет*).

6.98. См. рис. 6.43. Если в регистре *A* содержится 1AH, а в регистре *L* — A5H, в результате команды перехода МП _____(продолжит выполнение команд последовательно, перейдет к STORE *L*).

6.99. См. табл. 6.11. Операндом команды JC является STORE *L*; здесь используется _____ (абсолютная, символьная) адресация.

6.100. Команды перехода и ветвления в программах используются_____ (редко, часто).

Решения

6.94. Знаком принятия решения. **6.95.** Не изменяет. **6.96.** *CY*. **6.97.** Если (A)=(L), CY=0, то ветвь *Да*. **6.98.** Если (A)=1AH, а (L)=A5H, переход к STORE *L*, так как *CY*=1, потому что (A)<(L). **6.99.** Символьная. **6.100.** Часто.

6.12. ЦИКЛЫ

Микропроцессоры особенно эффективны в случае выполнения повторяющихся задач. Структурная схема на рис. 6.44, например, представляет программу, которая будет производить счет от 0 до 254 (0—FEH) и выводить результат счета на печать. Исходя из вершины, она является обычной для установки регистров. В этом примере содержимым аккумулятора A является 00H. Затем содержимое аккумулятора выводится на избранную периферию: это действие является повторяющимся процессом. Затем содержимое аккумулятора инкрементируется или изменяется. Знак сравнения и принятия решения выполняет процедуру проверки, чтобы определить, содержит ли аккумулятор значение FFH. Если ответ на поставленный в знаке принятия решения вопрос (будет ли A=FFH?) отрицателен (*Нет*), программа ветвится влево от знака принятия решения и возвращается в блок введения. Это составляет цикл. Программа будет продолжаться, повторяясь 254_{10} раз.

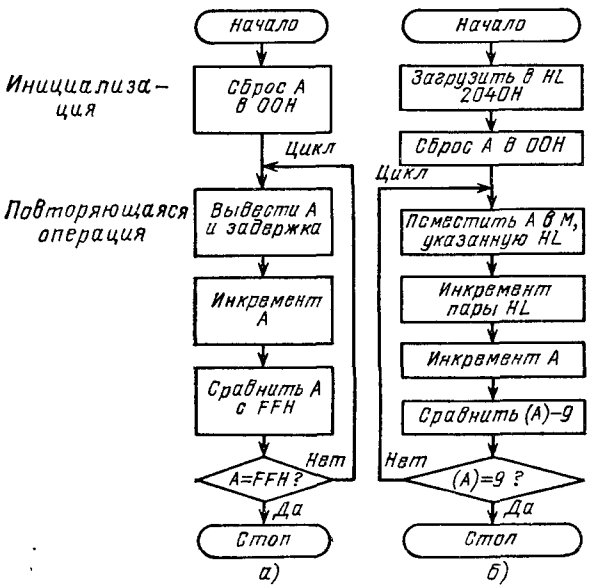


Рис. 6.44. Структурная схема последовательного счета с использованием циклов (*а*) и ее развитие (*б*)

Структурная схема на рис. 6.44, *а* представляет собой программу, которая может содержать только 20 или 30 команд. Если бы 255_{10} пропусков через программу были запрограммированы последовательно, их список составил бы тысячи команд. Циклы в программе являются очень эффективным методом ее сокращения.

На рис. 6.44, *б* приведена другая циклическая программа размещения ряда чисел (от 0 до 8) последовательно в память с адресами от 2040H до 2048H. Две первые прямоугольные рамки соответствуют начальной загрузке пары регистров *HL* и аккумулятора *A* значениями 2040H и 00H соответственно. Третья рамка соответствует процессу размещения данных в памяти, который будет повторен 9 раз в ходе выполнения этой программы. Микропроцессор, повторяя свои действия, разместит содержимое аккумулятора в памяти по адресу, указанному парой *HL*. При первом прохождении цикла содержимое (00H) аккумулятора будет помещено в память по адресу 2040H.

Четвертый и пятый кадры представляют операции, которые изменяют адрес в паре *HL* и счет в регистре *A*. Например, в ходе первого прохождения пара *HL* инкрементируется до 2041H, а аккумулятор — до 01H.

Прямоугольник сравнения и знак принятия решения составляют операцию тестирования. Команда сравнения вычитает 09H из содержимого *A* для восстановления или сброса индикатора нуля. Если (A)<9, индикатор нуля будет сброшен, если (A)=9, индикатор нуля будет установлен в 1. Знак принятия решения опрашивает: Равно ли (A) девяти? Если ответ *Да*, программа выходит из цикла и заканчивается, если, однако, ответ *Нет*, ветвится снова к команде размещения по соседству с вершиной структурной схемы. Команды сравнения и условного перехода используются для проверки изменяющегося счета для определения момента выхода из цикла. Программа, показанная на рис. 6.44, *б*, повторит цикл 9 раз прежде, чем выйти на знак *Стоп*.

В табл. 6.12 приведена версия программы на ассемблере, соответствующая структурной схеме на рис. 6.44, *б*.

Эта программа последовательно поместит числа от 0 до 9 в память по адресам от 2040H до 2048H. Две первые команды эквивалентны двум первым кадрам схемы и предназначены для установки в паре *HL* и в аккумуляторе начальных значений 2040H и 00H соответственно. Команда MOV *M*, *A* помещает содержимое аккумулятора в ячейку

Таблица 6.12. *Версия на ассемблере программы, соответствующей рис. 6.44, б*

Метка	Мнемоника	Операнд	Комментарий
	LXI	*H*, 2040H	Загрузить 2040H в пару *HL* (указатель адреса)
	XRA	*A*	Сброс аккумулятора в 00H, т. е. (A)⊕(A)=00H в аккумулляторе
LOOP	MOV	*M*, *A*	Поместить содержимое аккумулятора ячейку памяти, указанную парой *HL*
	INX	*H*	Инкрементировать пару *HL*
	INR	*A*	Инкрементировать аккумулятор
	CPI	09H	Сравнить (A)=09H? Если (A)=09H, индикатор нуля установлен в 1
	JNZ	LOOP	Перейтн к LOOP, еслн *Z*==0, т. е. еслн (A)=09H; если иет, продолжать последовательно
	HLT		Остановить МП

памяти, указанную парой *HL*. Две следующие команды (INX *H* и INR *A*) инкрементируют пару *HL* и *A*. Команда CPI сравнивает содержимое *A* с константой 09H. Если *A* содержит значение между 0 и 8, индикатор нуля сброшен в 0, но если *A* содержит 9, индикатор нуля устанавливается в 1.

Команда JNZ проверяет индикатор нуля. Если индикатор нуля сброшен в 0, значит результат вычитания (A) — 09H не равен 0, следовательно, осуществляется переход по символическому адресу LOOP. Если индикатор нуля установлен в 1, результат вычитания (A) — 09H нулевой; при этих условиях программа выходит из цикла и выполняет следующую последовательную команду (HLT), по которой завершается выполнение программы.

Упражнения

6.101. Способом сокращения программ, выполняющих повторяющиеся задачи, является_____(формирование циклов, модуляция программы).

6.102. См. рис. 6.44, *б*. Кадр сброс *A* в 00H соответствует команде_____(тестирования, восстановления) аккумулятора.

6.103. См. рис. 6.44, *б*. Если содержимым аккумулятора, соответствующим блоку сравнения, является 8, программа _____(циклится, останавливается) после знака принятия решения.

6.104. См. табл. 6.12. Команда CP1 _____(сравнивает, инвертирует) содержимое регистра_____(*A*, *HL*) с константой 09H операцией вычитания.

6.105. См. табл. 6.11. Если индикатор нуля установлен в 1, результатом команды JNZ будет_____(переход к последующей команде, ветвление на команду MOV *M*, *A*) МП.

6.106. Образование циклов в программе зависит от самой программы, по которой МП выполняет группу команд повторяющимся образом, а также от способности_____(изменить, прекратить) их повторяемость в заданный момент.

6.107. См. рис. 6.44, *б*. Эта программа повторяет операцию поместить *A* в память 9 раз, и программа после ромба принятия решения циклится_____раз.

Решения

6.101. Формирование циклов. **6.102.** Восстановления. **6.103.** Циклится. **6.104.** Сравнивает; *A*. **6.105.** Переход к последующей команде. **6.106.** Прекратить. **6.107.** 8 (помещение начального значения имело место до первого цикла).

6.13. ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ

Существуют подпрограммы, очень часто используемые одной и той же основной программой или одним и тем же программистом в различных программах. Бывает, что одна подпрограмма так часто используется, что она помещается в ПЗУ постоянно. Передача управления программой в подпрограмму выполняется одной командой вызова. Возврат или восстановление управления основной программой осуществляется командой возврата. Кроме того, команды помещения в стек, извлечения из стека и загрузки указателя стека используются также как текущие вместе с подпрограммой.

На рис. 6.45, *а* приведен пример функциональной структурной схемы. В задачу МП входит ввести два отобранных

числа, образовать их сумму и сохранить ее. Этой процедуре соответствуют три кадра в начале схемы. Затем сумма должна быть умножена на масштабный коэффициент и результат размещается в памяти. Это связано с тем, что сумма (в третьем кадре) должна быть восстановлена в аккумуляторе. Затем она проверяется знаком принятия решения. В том случае, если сумма равна или больше 10Н (16_{10}), программа выводится на сигнализацию тревоги, если меньше 10Н, циклится и снова вводит выборку чисел. Такой тип программ мог бы быть использован в промышленных устройствах, когда сумма двух входов должна постоянно контролироваться; быть подвержена масштабированию, а затем размещена в памяти; контролироваться по крайней мере один раз в секунду для определения, не приняла ли она опасного значения, и если да, то программа должна выдать аварийный сигнал АС.

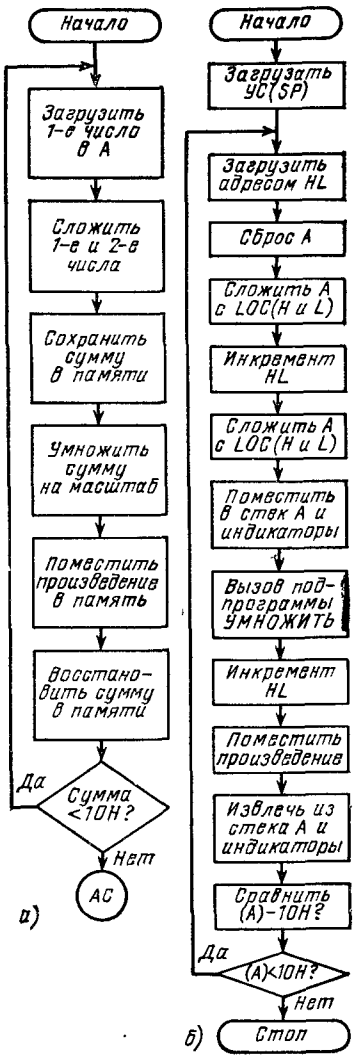


Рис. 6.45. Структурная схема программы контроля оборудования (*а*) и ее развитие (*б*)

Программа задумана для образования циклов неопределенно долго и принимает новые наборы данных в каждом цикле. Она покидает цикл для процедуры аварийной сигнализации только в случае возникновения соответствующей ситуации. В этом случае оператор или другая программа выдает процедуру отработки аварийного состояния. Это показано на структурной схеме кружком.

На рис. 6.45, *б* представлена схема программы. Каждый кадр ее эквивалентен одной операции МП. Выделенная рамкой ячейка, представляет собой *группу команд*, называемую подпрограммой умножить.

В табл. 6.13 представлена программа на ассемблере к рис. 6.45, *б*. Этот список привлекает только команды, которые составляют часть состава команд нашего типового МП (список, относящийся к подпрограммам умножения, на рисунке не приведен).

Таблица 6.13. *Программа на ассемблере к рис. 6.45, б*

Метка	Мнемоника	Операнд	Комментарий
	LXI	*P*, 20C0H	Поместить 20C0H в указатель стека
START	LXI	*H*, 2040H	Поместить 2040H в пару *HL* (указатель адреса основной программы)
	XPA	*A*	Сброс аккумулятора в 00H
	ADD	*M*	Сложить (А) с содержнмым ячейки памяти 2040H (сложнть первое слагаемое с содержимым ячейкн памяти 2040H)
	INX	*H*	Инкрементировать пару *HL* до 2041H
	ADD	*M*	Сложить *А* с содержнмым ячейки памяти 2041H (сложить второе число с содержимым ячейкн памяти 2041H)
	PUSN	*PSW*	Поместнть в стек (А) и индикаторы
	CALL	MULTIPLY	Вызов подпрограммы MULTIPLY в ячейку памятн 2050H
	INX	*H*	Инкрементнровать пару *HL* до 2042H
	MOV	*M*, *A*	Поместить произведение в ячейку памяти 2042H
	POP	*PSW*	Извлечь нз стека и восстановить *А* и ндикаторы
	CPI	10H	Сравнить (А) и 10H, т. е. (А) — 10H. Если (А)<10H, *CY*=1; если нет, *CY*=0
	JC	START	Перейти к START (ячейка памятн 2003H), еслн *CY*=1; еслн нет, продолжать последовательно
	HLT		Остановить МП

Три первые команды (табл. 6.13) восстанавливают указатель стека, пару *HL* и аккумулятор *A* соответственно до 20C0H, 2040H и 00H. Четвертая складывает содержимое (A) (т. е. 00H) с содержимым ячейки памяти, на которую указывает пара *HL* (адрес 2040H). Обратимся к рис. 6.46, который представляет собой основную программу, данные, подпрограмму и память стека, использованные в этом примере. Заметим, что содержимым памяти по адресу 2040H, сложенным с (A) является 05H. После выполнения четвертой команды содержимым (A) является 05H (00H+ +05H=05H).

Пятая команда основной программы инкрементирует пару *HL*. Шестая складывает содержимое памяти по адресу 2041H с (A). Эта ячейка памяти содержит 09H (см. рис. 6.46), *A* содержит 05H, сумма в *A* после второго ADD M—0EH является суммой двух выбранных чисел.

Следующей командой основной программы является PUSH *PSW*, которая сохраняет содержимое аккумулятора и индикатора. Это должно быть выполнено, потому что следующая команда вызова разрушит содержимое этих регистров, выполняя подпрограмму умножения. Операция вызова помещает адрес команды в последовательности основной программы (в нашем примере 200EH) в стек, затем переходит к адресу первой команды подпрограммы (адрес 2050H).

Этот вызов показан на рис. 6.45, *б*. Теперь рассмотрим команду вызова как команду, которая просто умножает содержимое *A* на два. Подпрограмма в нашем примере осуществляет умножение 0EH×2=1CH, и произведение, помещенное в аккумулятор к моменту возврата, будет 1CH.

После возврата в основную программу адресом следующей последовательной команды становится 200EH. Команда INX *H* инкрементирует пару *HL*, и произведение, выполненное подпрограммой, помещается теперь в памяти по адресу 2042H командой MOV *M*, *A*.

Вспомним, что команды помещения в стек и извлечения из него парные. Только что мы поместили в стек содержимое аккумулятора и индикаторов командой PUSH *PSW*, мы их восстановим теперь командой POP *PSW*. Содержимое аккумулятора является не произведением, а суммой 0EH в нашем примере). Эта сумма будет тестирована последующей командой.

Важно отметить, что (0EH) отправлена в подпрограмму для обработки. Напротив, именно произведение было

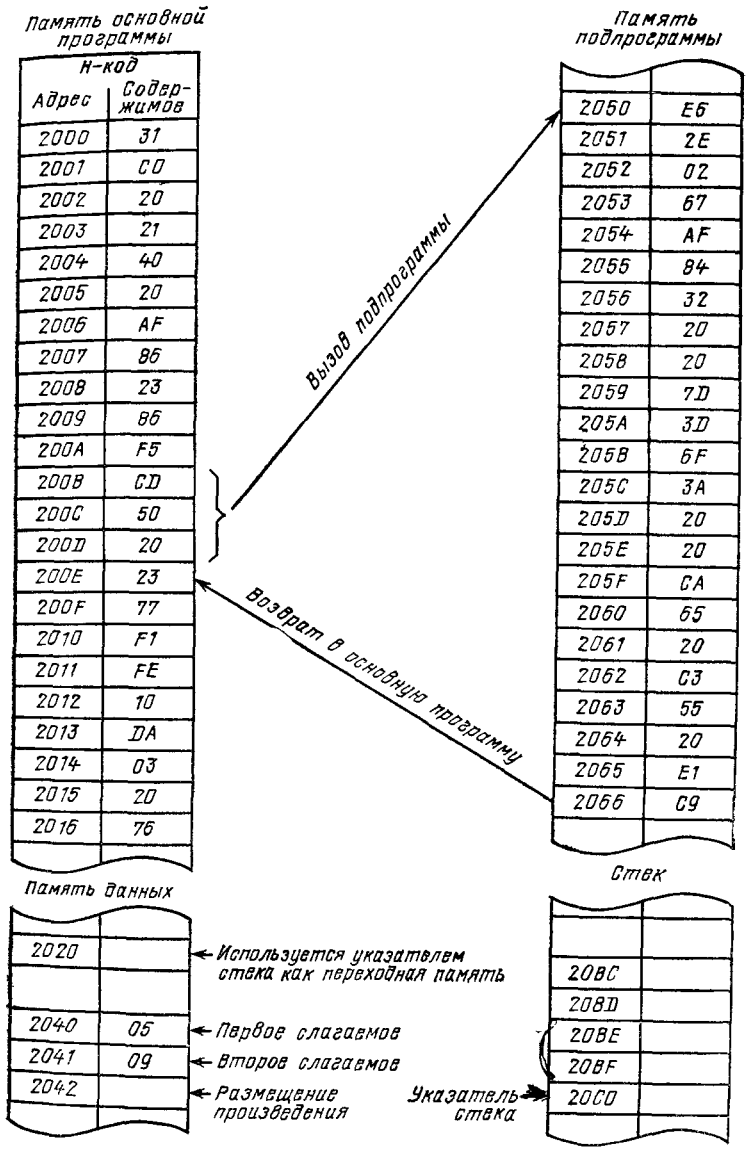


Рис. 6.46. Воображаемая память программы, приведенной в табл. 6.13

передано подпрограммой и, следовательно, размещено в памяти. На жаргоне информатики эта операция составляет прохождение параметра. В данном случае использован именно регистр *A*, но в других случаях это могло бы быть выполнено посредством памяти.

Две следующие команды основной программы предназначены для тестирования суммы в аккумуляторе. Команда сравнить непосредственно выполняет операцию вычитания 0EH—10H=FEH с использованием дополнительного кода числа -2_{10}. Содержимое *A* меньше чем 10H, индикатор переноса *CY* установлен в 1. Команда ПЕРЕЙТИ, если перенос, проверяет индикатор, он установлен в 1, и МП переходит к символическому адресу START, который является не чем иным, как началом цикла команды LXI *H* (адресация 2003H, см. рис. 6.46).

Для операции двоичного умножения могут быть использованы различные способы на базе команд типового МП. Вытекающие непосредственно из правил умножения методы приводят к повторяющемуся сложению. Рассмотрим типичное решение на примере выполнения операции $5\times3=15_{10}$.

Множимое Множитель Произведение

$$5 \times 3 = 15_{10}.$$

При повторяющемся сложении эта операция выполнится иначе:

Множитель = 3 Произведение

$$5+5+5 = 15_{10}$$

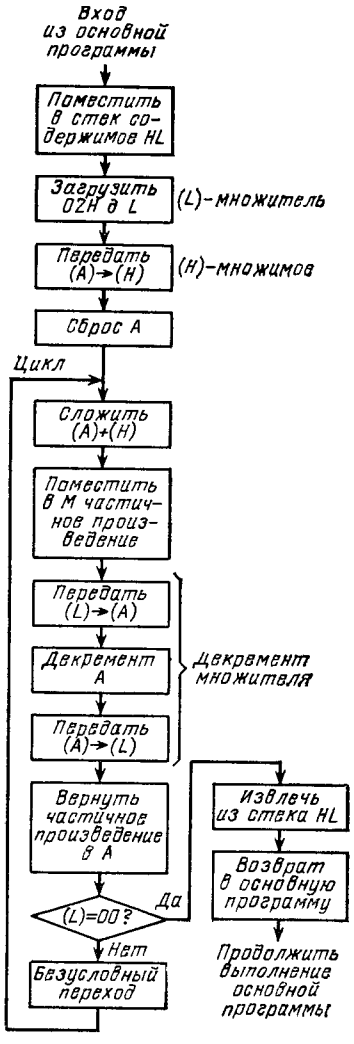
Множимое = 5

В этом случае множимое повторяется слагаемым число раз, равное множителю, а результатом этой операции будет произведение. В нашем примере подпрограмма эффективно выполняет операцию сложения (вместо $0E\times2=1CH$). Множимое складывается с самим собой ($0E+0E=1CH$), что дает искомое произведение.

На рис. 6.47 приведена подробная структурная схема подпрограммы умножения, и каждый ее шаг соответствует одной команде программы на ассемблере, приведенной в табл. 6.14.

Первый шаг на рис. 6.47 обеспечивает хранение текущего содержимого пары *HL* операцией помещения в стек. Это характерно для многих подпрограмм — помещать в

Рис. 6.47. Структурная схема подпрограммы умножения в задаче, приведенной на рис. 6.45



стек содержимое регистров МП, потому что они могут быть использованы во время выполнения подпрограммы, что изменит их содержимое. Совершенно очевидно, что содержимое пары *HL* будет восстановлено в конце программы командой извлечения из стека.

Три следующих шага восстанавливают регистры *L*, *H* и *A*. Регистр *L* будет содержать множитель (здесь 02H), и он будет декрементирован до 00H в ходе программы умножения. Регистр *H* будет содержать множимое (здесь 0EH). т. е. сумму, посланную в подпрограмму основной программой. Аккумулятор здесь сбрасывается в 00H.

Пятый шаг соответствует сложению множимого (здесь 0EH) с (A). Частичное произведение 0EH затем сохраняется во временной памяти по адресу 2020H, тогда как аккумулятор используется для декрементирования содержимого *L*. Один раз множитель декрементируется от 02H до 01H и помещается в *L*. Затем частичное произведение восстанавливается в *A*.

На шаге принятия решения возникает вопрос: (L)=00?. Если ответ отрицательный, программа переходит снова в цикл, если положительный — ветвится вправо. Содержимое пары *HL* восстанавливается командой извлечения из стека, и, наконец, команда возврата возвращает нас в основную программу. Согласно рис. 6.46 возврат осуществляет-

Таблица 6.14. *Программа на ассемблере, соответствующая рис. 6.47*

Метка	Мнемоника	Операнд	Комментарий
	PUSH	*H*	Поместить в стек содержимое регистров *H* и *L* для сохранения содержимого пары *HL*
	MVI	*L*, 02H	Поместить 02H (масштабный коэффициент) в *L*; 02H — множитель
	MOV	*H*, *A*	Передать (A)→(H); содержимое *H* — множимое
	XRA	*A*	Сброс *A*
LOOP	ADD	*H*	Сложить (H)+(A); сумму поместить в *A*
	STA	2020H	Поместить (A) в ячейку памяти 2020H (временная память)
	MOV	*A*, *L*	Передать (*L*) в *A*
	DCR	*A*	Декрементировать содержимое *A*
	MOV	*L*, *A*	Передать (A) в *L*
	LDA	2020H	Поместить содержимое ячейки памяти 2020H (временной памяти) в *A* (восстановление *A* из ячейки памяти 2020H)
	JZ	DONE	Перейти к DONE (ячейка памяти 2065H), если *Z*=1, т. е. декрементировано до 00H; если нет — продолжать последовательно
	JMP	LOOP	Перейти (всегда) к LOOP (адрес 2055H)
DONE	POP	*H*	Извлечь из стека содержимое *HL*
	RET		Возврат в основную программу

ся по адресу 200EH основной программы. В случае, когда множителем является 02H, программа дважды пройдет в последовательности сложить—поместить—передать—декрементировать—передать—загрузить перед возвратом в основную программу. Отметим, что окончательное произведение 1CH поступит в основную программу, оставаясь в аккумуляторе.

В табл. 6.14 приведена программа на ассемблере. Каждая команда соответствует одному шагу структурной схемы на рис. 6.47. Рекомендуем читателю попытаться проследить поток данных сначала по ней, а затем по программе, приведенной в табл. 6.14.

Используя команды вызова и возврата, следует быть очень внимательным и использовать их парами. Нужно следить за тем, чтобы требуемые регистры были правильно инициализированы, многократно проверять состояние указателя стека. Следует также убедиться, что парно используются команды загрузки и извлечения из стека, а также глубоко осознать способы обмена параметрами между подпрограммами и основными программами.

Упражнения

6.108. Подпрограмма умножения на рис. 6.47 для вычисления произведения использует метод_____(сложения-сдвига, повторяющегося сложения).

6.109. См. рис. 6.46. Какие типы команд используются обычно памятью, называемой стеком?

6.110. Назначением команды LXI *SP* в табл. 6.13 является _____(установка, тестирование) указателя стека.

6.111. Если индикатор переноса в табл. 6.13 установлен в 1, по команде JC МП перейдет к выполнению команды _____(HLT, LXI *H*).

6.112. Команда вызова в табл. 6.13 осуществляет помещение в стек содержимого_____(регистра *A*, счетчика команд) и загрузку счетчика команд адресом_____(шестнадцатеричный).

6.113. См. табл. 6.13. Команда POP *PSW* извлекает из стека содержимое_____(аккумулятора, счетчика команд) и индикаторов и передает их в МП.

6.114. Знак принятия решения (см. рис. 6.47) заменяется обычно командой_____(сравнить, JUMP).

6.115. При выполнении подпрограммы УМНОЖИТЬ, приведенной в табл. 6.14, множимое находится в регистре _____(*H*, *L*), а множитель — в регистре_____(*H*, *L*).

6.116. В ходе выполнения подпрограммы умножить содержимое регистра_____(*H*, *L*) декрементируется до 00H.

6.117. См. табл. 6.13 и 6.14. Регистры *H* и *L* используются как_____ (регистры данных, указатели адреса) в основной программе и как_____(указатели адреса, регистры данных) в подпрограмме.

Решения

6.108. Повторяющегося сложения. **6.109.** Помещения в стек, извлечения из стека (PUSH и POP) соответственно, а также вызова подпро-

грамм и возврата из них (CALL и RET). Это память типа LIFO. **6.110.** Установка. **6.111.** LXI *H* по символьному адресу START. **6.112.** Счетчика команд; 2050H. **6.113.** Аккумулятора. **6.114.** JUMP — переход или ветвление. **6.115.** *H*; *L*. **6.116.** *L*. **6.117.** Указатели адреса; регистры данных.

Дополнительные упражнения к гл. 6

6.118. Список команд на языке_____(машинном, ассемблер) характеризуется использованием адресов, КОП и операндов в шестнадцатеричном коде.

6.119. Программа, приведенная в табл. 6.12, является примером программы на языке_____(машинном, ассемблер).

6.120. Обычно команда на ассемблере делится на четыре поля:_____.

6.121. Версия программы на ассемблере иногда называется _____.

6.122. БЕЙСИК является развитым языком, так как каждая команда на нем содержит_____(больше, меньше) команд микропроцессора, чем ассемблер или машинный код.

6.123. Индикаторы на рис. 6.3 находятся в ячейке памяти МП, называемой регистром _____.

6.124. Когда регистры *H* и *L* составляют пару *HL*, они обычно используются как указатели _____.

6.125. Стек типового микропроцессора (или МП Intel 8080/8085) находится в _____.

6.126. Аккумулятор и регистры *H* и _____ типового микропроцессора являются универсальными регистрами.

6.127. 16-разрядный регистр, называемый *SP* на рис. 6.3, является _____.

6.128. 16-разрядный регистр, называемый *PC* на рис. 6.3, является _____.

6.129. Декрементировать — значит_____(добавить 1, вычесть 1) в/из содержимое(го) регистра.

6.130. Обычно арифметические операции_____(влияют, не влияют) на индикаторы.

6.131. После операции сравнения установлен(ы) только _____(аккумулятор, индикаторы).

6.132. Каково содержимое аккумулятора после операции ADD *L* (рис. 6.48).

6.133. См. рис. 6.48. Каковы состояния индикаторов переноса и нуля после операции ADD *L*?

6.134. Микропроцессор не имеет внутренней аппаратуры вычитания. Эту операцию он выполняет, преобразуя _____(уменьшаемое, вычитаемое) в дополнительный код, затем — операцию сложения.

6.135. См. табл. 6.3. При выполнении команды вычитания, если индикатор переноса *CY* сброшен в 0, это означает _____(наличие, отсутствие переноса) или что уменьшаемое_____(больше, меньше) вычитаемого.

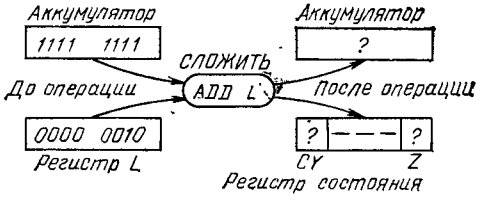


Рис. 6.48. К упражнениям 6.132 и 6.133

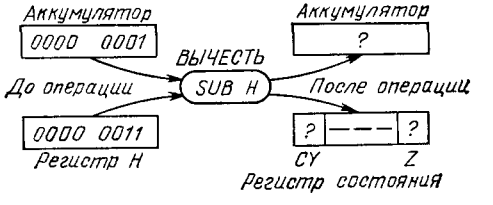


Рис. 6.49. К упражнениям 6.136 и 6.137

6.136. См. рис. 6.49. После операции SUB *H* содержимое аккумулятора_____(дать 8 бит), что является дополнительным кодом десятичного числа _____.

6.137. См. рис. 6.49. Каковы состояния индикаторов переноса и нуля после операции SUB *H*?

6.138. Как правило, индикаторы регистра состояния _____ (устанавливаются, не устанавливаются) логическими командами.

6.139. Знак (·) в табл. 6.4 означает операцию _____ (И, умножить).

6.140. См. рис. 6.50. Какой КОП команды XRI (шестнадцатеричный)?

6.141. См. рис. 6.50. Каково содержимое аккумулятора после операции XRI?

6.142. См. рис. 6.50. Каковы состояния индикаторов после операции XRI?

6.143. См. рис. 6.51. Каково содержимое аккумулятора после операции CMA?

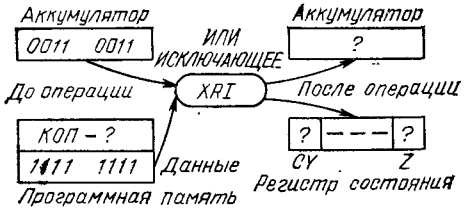


Рис. 6.50. К упражнениям 6.140—6.142

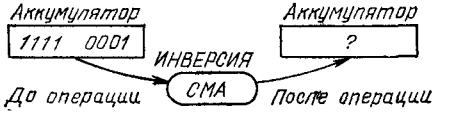


Рис. 6.51. К упражнению 6.143

6.144. См. табл. 6.4. Если аккумулятор содержит 1110 1110 и $CY=0$, каково содержимое A и CY после выполнения операции RAR?

6.145. Почти все команды передачи_____(влияют, не влияют) на индикаторы.

6.146. См. табл. 6.5. Команды IN и OUT являются командами_____адресации.

6.147. См. табл. 6.5. В ходе выполнения команды MOV L, A регистр_____(A, L) является источником данных, а регистр_____(A, L) — назначением.

6.148. См. табл. 6.5. Код операции команды LDA_____, ее формат — _____байт.

6.149. См. рис. 6.52. Каково содержимое аккумулятора после операции загрузки?

6.150. См. рис. 6.52. Какой адрес источника данных?

6.151. См. табл. 6.5. Какой способ адресации команды загрузки на рис. 6.52?

6.152. Команды перехода называют также командами_____.

6.153. Команды_____(условного, безусловного) перехода называются командами принятия решения.

6.154. Команды перехода устанавливают содержимое _____(счетчика команд, указателя стека).

6.155. См. табл. 6.6. Какой КОП команды JZ на рис. 6.53?

6.156. См. рис. 6.53. Каково содержимое счетчика команд после операции JZ?

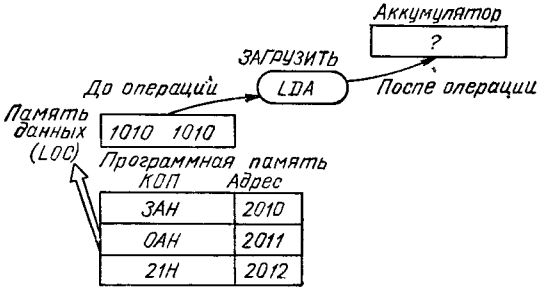


Рис. 6.52. К упражнениям 6.149 и 6.151

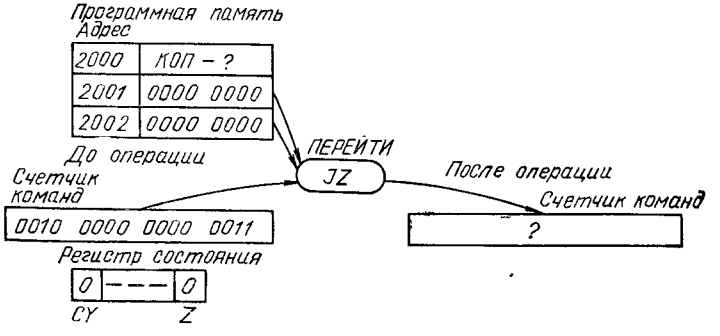


Рис. 6.53. К упражнениям 6.155 и 6.156

6.157. Команда_____(вызова, возврата) типового микропроцессора выполняет одновременно функции помещения в стек или извлечения из него.

6.158. Команда_____(вызова, возврата) следует в конце подпрограммы.

6.159. См. рис. 6.54. Каково содержимое ячейки памяти 2109Н после выполнения операции вызова?

6.160. Каково содержимое ячейки памяти 2108H в стеке (рис. 6.54) после вызова?

6.161. См. рис. 6.54. Каково содержимое счетчика команд после вызова?

6.162. См. рис. 6.54. Содержимое счетчика команд после вызова указывает на первую команду_____(стека, подпрограммы).

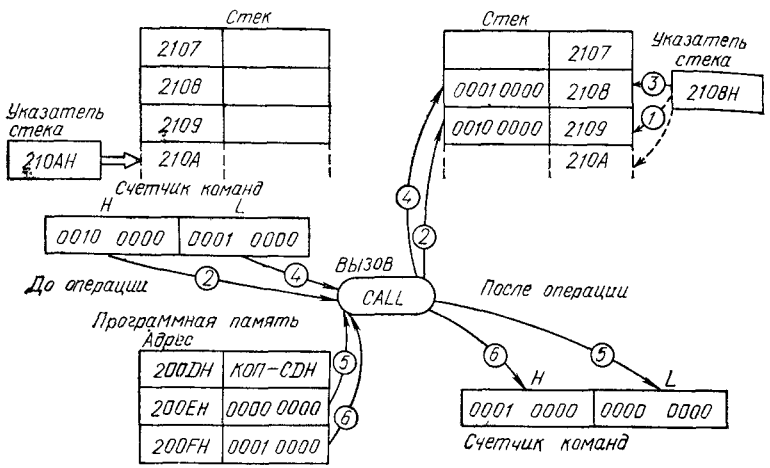


Рис. 6.54. К упражнениям 6.159—6.162

6.163. См. табл. 6.8. Какой КОП команды NOP на рис. 6.55?

6.164. Каково содержимое регистров после операции NOP на рис. 6.55?

6.165. См. табл. 6.8. Если команда PUSH *PSW* использована как первая команда подпрограммы, какая команда (дать ее мнемонику) используется непосредственно перед возвратом?

6.166. Когда типовой микропроцессор находится в состоянии останова после команды HLT, его может запустить только вход_____(ГТИ, сброса) или вызов прерывания.

6.167. Первым этапом записи программы является _____ (определение задачи, запись на ассемблере).

6.168. Программист должен хорошо знать архитектуру системы, ее воображаемую память, регистры МП и_____ МП.

6.169. После анализа и определения задачи следующим этапом программирования является составление_____ (документации, структурной схемы).

6.170. Формат команд непосредственной адресации в типовом микропроцессоре составляет _____ байт.

6.171. См. табл. 6.8. Команда PUSH *H* является_____ байтовой командой_____ адресации.

6.172. См. табл. 6.3. Команда сравнить *A* и *L* является _____ байтовой.

6.173. См. табл. 6.5. Команда LDA является _____ байтовой командой _____ адресации.

6.174. В знаке принятия решения _____(ставится вопрос, утверждается положение).

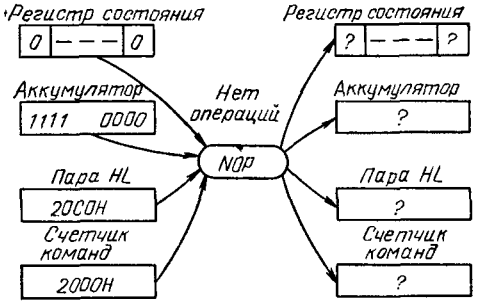


Рис. 6.55. К упражнениям 6.163—6.164

6.175. Для определения содержащейся внутри знака принятия решения команды используют команду _____ (логическую, условного перехода).

6.176. До операции условного перехода часто используются команды _____ (загрузки, сравнения).

6.177. Структурные схемы со знаками принятия решения_____(являются, не являются) последовательными.

6.178. См. табл. 6.12. Каково назначение команды XRA *A*?

6.179. Формирование_____используется для решения повторяющихся задач.

6.180. См. рис. 6.44, *а*. Самое большое число, выводимое этой программой, _____.

6.181. Наиболее часто используемые программы помещаются в _____ (ПЗУ, ОЗУ).

6.182. При команде JC проверяется индикатор_____ (*CY*, Z).

6.183. Команды PUSH и_____(ADD, POP) должны использоваться совместно.

6.184. Команда CALL и_____(RET, JUMP) должны использоваться парно.

6.185. Структурная схема на рис. 6.45, *а* является_____ (подробной, функциональной).

Решения

6.118. Машинном. **6.119.** Ассемблер. **6.120.** Метка, мнемоника, операнд, комментарий. **6.121.** Исходной программой. **6.122.** Больше. **6.123.** Состояния. **6.124.** Адреса. **6.125.** ОЗУ. **6.126.** *L*. **6.127.** Указателем стека. **6.128.** Счетчиком команд. **6.129.** Вычесть 1. **6.130.** Влияют. **6.131.** Индикаторы. **6.132.** 0000 0001. **6.133.** Переноса — устанавливается в 1, нуля — сбрасывается в 0. **6.134.** Вычитаемое. **6.135.** Отсутствие переноса; больше. **6.136.** 1111 1110; -2_{10}. **6.137.** $CY=1$; $Z=0$. **6.138.** Устанавливаются. **6.139.** И. **6.140.** EEH. **6.141.** 1100 1100. **6.142.** $CY=0$; $Z=0$. **6.143.** 0000 1110. **6.144.** (A) = 0111 0111 и $CY=0$. **6.145.** Не влияют. **6.146.** Прямой. **6.147.** *A*; *L*. **6.148.** 3AH; 3. **6.149.** 1010 1010. **6.150.** 210AH. **6.151.** Прямая адресация. **6.152.** Ветвления. **6.153.** Условного. **6.154.** Счетчика команд. **6.155.** CAH. **6.156.** 2003H (то же, что и до перехода). **6.157.** Вызова. **6.158.** Возврата. **6.159.** 0010 0000 (20H). **6.160.** 0000 0111 (07H). **6.161.** 30F0H (0011 10000 1111 0000). **6.162.** Подпрограммы. **6.163.** 00H. **6.164.** Регистр состояния: $CY=0$; $Z=0$ (без изменения).

Аккумулятор: (A)=1111 0000 (без изменения).

Пара регистров *H*: (HL)=20C0H (без изменения).

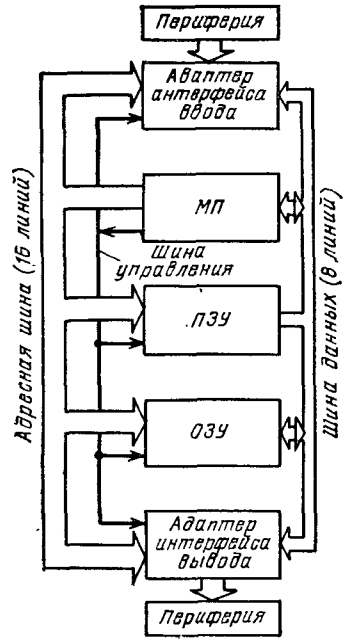
Счетчик команд инкрементирован до значения 2001H. **6.165.** POP *PSW*. **6.166.** Сброса. **6.167.** Определение задачи. **6.168.** Состав команд. **6.169.** Структурной схемы. **6.170.** 2 или 3. **6.171.** Одно; косвенной регистровой. **6.172.** Одно; регистровой. **6.173.** Трех; прямой. **6.174.** Ставится вопрос. **6.175.** Условного перехода. **6.176.** Сравнения. **6.177.** Не являются. **6.178.** Сбросить аккумулятор до 00H. **6.179.** Образования циклов. **6.180.** FEH или 254_{10}. **6.181.** ПЗУ. **6.182.** *CY*. **6.183.** POP. **6.184.** RET. **6.185.** Функциональной.

Глава 7
ИНТЕРФЕЙС МИКРОПРОЦЕССОРА

Большинство микропроцессоров сами по себе функционально ограничены. Большая часть из них содержит память и немногие порты ввода/вывода, которые напрямую соединяют их с периферией. Микропроцессоры функционируют как элементы системы. Соединения между элементами внутри системы составляют *интерфейс*. Обычно интерфейс является общей границей между двумя или несколькими устройствами, т. е. тем, что влечет за собой раздел информации[1]. Среди прочих свойств интерфейса отметим

[1] Согласно [1] под интерфейсом понимается: совокупность унифицированных технических и программиых средств, необходнмых для подключения данных устройств к системе илн одной системы к другой. — *Прим. ред.*

Рис. 7.1. Функциональная схема микропроцессорной информационной системы



решение им задач синхронизации, выбора направления передачи данных и иногда приведения в соответствие уровней или форм сигналов.

Рассмотрим схему простой системы на рис. 7.1. Мы будем изучать интерфейс (или взаимные присоединения) между каждым из показанных устройств и МП; подробно рассмотрим также интерфейс МП с ПЗУ, ОЗУ и УВВ; приведем подробную информацию о некоторых периферийных устройствах[1]. Согласно рис. 7.1 очевидно, что шины адресов, данных и управления играют важную роль во взаимных связях элементов системы, и, следовательно, нас будут интересовать как аппаратные, так и программные средства.

Обычно передача данных от и в МП через шины осуществляется в следующих формах:

1. Считывание из памяти.
2. Запись в память.
3. Считывание из УВВ.
4. Запись в УВВ.
5. Управление прерыванием или сбросом.

Когда говорят, что данные введены с другого устройства, это означает *вступить в отношения с МП*. Аналогично вывод данных — это уход из МП. Обычно МП является ядром всех операций. Однако некоторые МП оставляют на время управление шинами данных и адресов, чтобы периферийное устройство могло получить доступ к центральной

[1] Периферийное устройство [1] — устройство, входящее в состав внешнего оборудования микро-ЭВМ, обеспечивающего ввод и вывод данных, организацню промежуточного и длительного хранения данных. — *Прим. ред.*

памяти, минуя МП: эта операция называется *прямым доступом к памяти* (ПДП).

Согласно рис. 7.1 МП, ПЗУ, ОЗУ, *адаптер интерфейса ввода, адаптер интерфейса вывода* являются различными устройствами. В рассматриваемой системе это может быть и иначе. Иногда разработчики создают интерфейсы, совместимые с соответствующими устройствами МП. Они являются универсальными в том смысле, что могут быть программируемы как устройства интерфейса ввода или вывода. Некоторые разработчики встраивают ОЗУ и порты ВВ или ПЗУ в одну и ту же ИС с целью снижения числа элементов, составляющих систему. Они производят также специализированные интерфейсы в форме ИС: интерфейсы программируемых связей, программируемого управления ПДП, программируемого управления прерываниями, управления дискетами, контролерами связи синхронных данных, управления видеотерминалом и клавишным устройством.

7.1. ИНТЕРФЕЙС С ПЗУ

Рассмотрим задачу разработки интерфейса с ПЗУ или ППЗУ. На рис. 7.2 приведена часть системы, включающая МП и ПЗУ. С выходами O_0—O_7 ПЗУ соединены 8 линий шины данных. Единственный выход управления считыванием $\overline{RD}$ идет из МП на вход активизации $\overline{OE}$ ПЗУ.

С постоянным запоминающим устройством емкостью 4 Кбайт соединены 12 линий адресной шины младших разрядов (A_0—A_{11}). Дешифратор, встроенный в ИС ПЗУ, может получить доступ к любому из 4096 (2^{12}=4096) 8-разрядных слов ПЗУ. Адресные линии четырех старших разрядов (A_{11}—A_{15}) идут в устройство комбинированной логики — дешифратор адреса. Для доступа в ПЗУ и считывания из него данных МП должен: активизировать линии адреса A_0—A_{11}; установить L-сигнал на линии управления считыванием $\overline{OE}$; установить L-сигнал на линии дешифратора адреса и выбора кристалла.

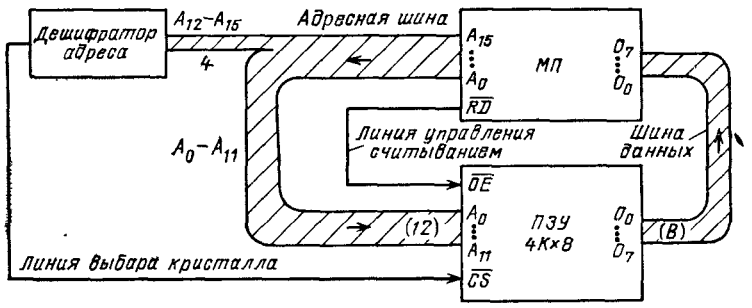


Рис. 7.2. Интерфейс ПЗУ и МП

Предположим, что МП нужно обратиться в память по адресу 0000H (0000 0000 0000 0000_2). Младших 12 бит подключаются по адресным линиям A_0—A_{11} к контуру дешифратора ПЗУ. К адресным принадлежат также старших 4 бит A_{12}—A_{15}. Они декодируются *дешифратором адреса*. Если (A_{12}—A_{15})=0000_2, дешифратор адреса выдает сигнал, который активизирует вход $\overline{CS}$ выбора кристалла ПЗУ (рис. 7.2).

Воображаемая память, приведенная на рис. 7.3, может помочь понять роль дешифратора адреса. Она представляет собой устройство емкостью 64К (т. е. 65 536 ячеек памяти), разделенное на 16 сегментов по 4 К каждый. Роль дешифратора адреса состоит в том, чтобы обеспечить МП доступ только к одному из этих сегментов одновременно. Если имеется четыре входа в дешифратор 0000, то доступным будет нулевой сегмент (пространство памяти 0000—0FFFH). Если на этих входах 0001, доступен первый сегмент (пространство памяти 1000—1FFFH) и т. д. Таким образом, старших 4 бит (передающих линии) выбирают сегмент памяти, а младших 12 бит определяют нужную ячейку памяти в этом сегменте.

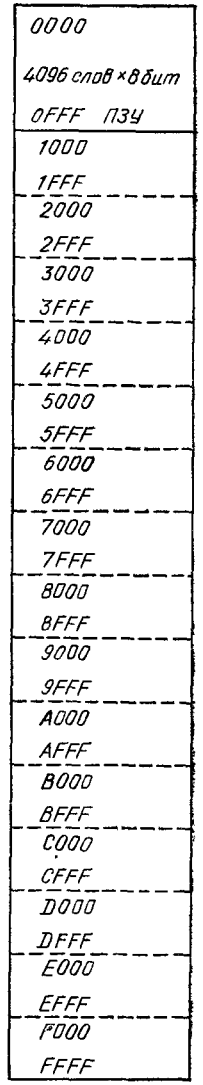


Рис. 7.3. Воображаемая память

В интерфейсе с ПЗУ важное значение имеют способы *адресации* и *синхронизации*. Адресацию мы сейчас рассмотрели, обратимся теперь к синхронизации. На рис. 7.4 приведена

временна́я диаграмма сигналов МП, управляющих считыванием 8-разрядного слова из ПЗУ.

Верхняя линия диаграммы представляет переход адресных линий A_0—A_{15} на их соответствующий логический уровень. Согласно рис. 7.2 адресные линии A_0—A_{11} активизируют адресные входы ПЗУ, тогда как адресные линии A_{12}—A_{15} декодируются дешифратором адреса и активизируют вход *CS* выбора кристалла ПЗУ. Спустя некоторое время выход управления считыванием *RD* МП активизирует процесс вывода данных из ПЗУ. Расположенные здесь данные помещаются на шину данных и принимаются МП.

На рис. 7.4 показаны критические ограничения синхронизации. После того как на адресных линиях установился

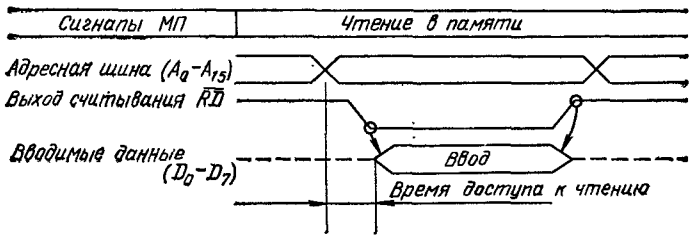


Рис. 7.4. Временна́я диаграмма. Сигналы МП в ходе операции считывания из ПЗУ

соответствующий логический уровень и активизировался вход $\overline{CS}$ ПЗУ, нужен определенный отрезок времени для извлечения слова данных. Это время необходимо внутренним дешифраторам ПЗУ для нахождения требуемого байта в памяти.

Обратим внимание на использование на рис. 7.4 кружков и стрелок. Эти индикаторы используются для обозначения *соотношений причины и следствия* на временно́й диаграмме. В качестве примера на рис. 7.4 переход от Н-к L-уровню (от HIGH к LOW) на выходе считывания осуществит переключение тристабильных выводов шины данных МП из состояния высокого сопротивления для приема данных на входе. Штриховая часть временно́й диаграммы по линии ввода соответствует состоянию высокого сопротивления. При переходе от L- к Н-уровню выхода $\overline{RD}$ выводы шины данных МП снова переключатся в третье состояние

и не примут данные с шины. Временна́я диаграмма на рис. 7.4 представляет собой только сигналы МП и не содержит сигналов входа и выхода ПЗУ.

Упражнения

7.1. Постоянное запоминающее устройство на рис. 7.2 может содержать _____ слов емкостью _____ бит каждое.

7.2. Постоянное запоминающее устройство емкостью 4 К требует _____ адресных входов для декодирования адресов 4096 содержащихся в нем ячеек памяти.

7.3. См. рис. 7.2. Дешифратором _____ (адреса, ПЗУ) декодируются четыре старшие линии.

7.4. См. рис. 7.3. Какой сегмент памяти будет доступен, если МП выдает 0010 0000 0000 1111_2 на адресную шину?

7.5. См. рис. 7.4. В ходе считывания из памяти последней будет активизироваться линия МП _____ (A_0—A_{15}, $\overline{RD}$).

7.6. Когда сигнал на выходе $\overline{RD}$ МП переходит к L-уровню, он активизирует вход _____ ($\overline{CS}$, $\overline{OE}$) ПЗУ и приводит выводы шины данных МП в состояние _____ (приема данных с шины, отказа от данных с шины, исходя из условия трех состояний).

7.7. Оценка ПЗУ, относящаяся к необходимому времени декодирования адреса и доступа в память по заданному адресу, составляет время _____.

Решения

7.1. 4096 (4К); 8. **7.2.** 12. **7.3.** Адреса. Это линии (A_{12}—A_{15}). **7.4.** $0010_2=2_{10}$, сегмент 2, содержащий 4096 ячеек памяти с адресами от 2000H до 2FFFH. **7.5.** $\overline{RD}$. **7.6.** $\overline{OE}$; приема данных с шин. **7.7.** Доступа к чтению в ПЗУ.

7.2. ИНТЕРФЕЙС С ОЗУ

Устройства размещения данных, допускающие их запись и считывание, обычно называются ОЗУ. Оперативные запоминающие устройства делятся разработчиками на два типа: статические и динамические. Этот параграф посвящен статическим ОЗУ, которые более просты для выполнения интерфейса, чем динамические (см. § 3.7).

Рассмотрим рис. 7.5, где представлены ОЗУ и МП некоторой системы. Отметим, что ОЗУ составлено устройствами 4 К×8 бит, т. е. может разместить 4096 слов емко-

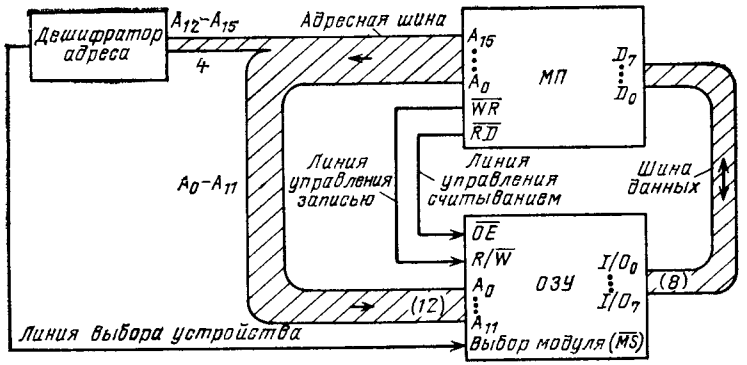


Рис. 7.5. Интерфейс ОЗУ и МП

стью 8 бит каждое. Еще совсем недавно ОЗУ такого типа составлялись из нескольких ИС (например, 32 ИС ОЗУ 1024×1 бит). Модуль памяти (или карта памяти) содержит также около десятка дополнительных ИС (порты и буферы), и ОЗУ, представленное как один блок, в действительности является сложной системой.

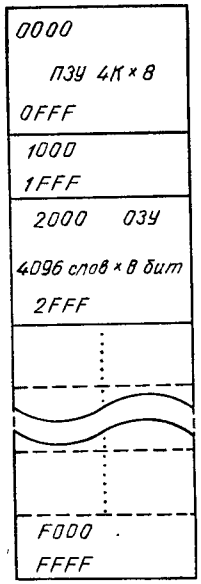


Рис. 7.6. Воображаемая память

На рис. 7.6 приведена воображаемая память системы. Оперативное запоминающее устройство емкостью 4 К произвольно помещено в третьем сегменте сверху. Как и в случае ПЗУ, дешифратор на рис. 7.5 будет иметь целью активизировать линии выбора устройства. Он выдает L-импульс в устройство выбора модуля $\overline{MS}$ для активизации ОЗУ только в случае, когда четыре адресные линии $(A_{12}$—$A_{15}) = 0010_2$. Как и в случае ПЗУ, декодирование младших 12 бит (A_0—A_{11}) осуществляется системой декодирования ОЗУ.

Шина данных становится двунаправленной 8-разрядной для считываемых и записываемых в ОЗУ данных. Двенадцать линий младших разрядов МП идут прямо на адресные входы модуля ОЗУ через адресную шину. Четыре линии старших разрядов соединены с дешифратором адреса. Выход $\overline{WR}$ записи в МП соединен по линии управления со входом $R/\overline{W}$ ОЗУ. Заметим, что вход ОЗУ является входом записи/чтения. Это означает, что когда МП не активизирует выход записи L-сигналом, $\overline{WR}$ выдает H-сигнал в ОЗУ, который точно определяет операцию считывания. Выход $\overline{RD}$ считывания МП соединен по линии управления считыванием с выходом $\overline{OE}$.

Временна́я диаграмма, иллюстрирующая изменения сигналов МП и ОЗУ в ходе операции считывания, приведена на рис. 7.7. Адресные линии МП активизированы и содержат требуемый адрес. Выход считывания $\overline{RD}$ переходит

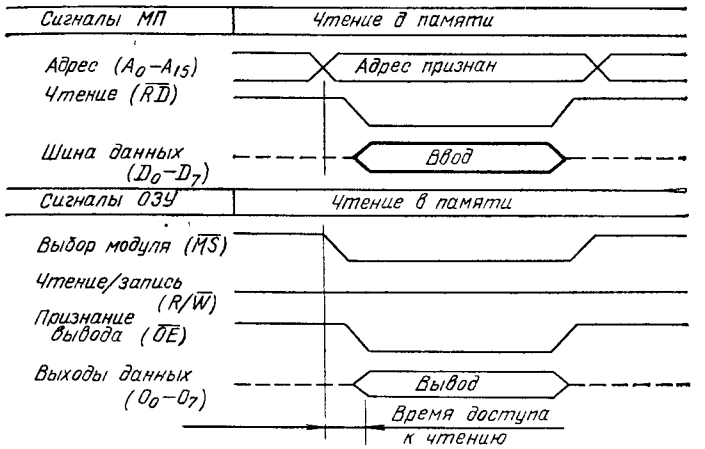


Рис. 7.7. Временна́я диаграмма. Сигналы МП и ОЗУ в ходе операции считывания из памяти

к L-уровню. Тристабильная шина данных переходит в состояние считывания, и МП готов принять с нее данные. Линия выбора модуля $\overline{MS}$ ОЗУ и вход активизации выхода $\overline{OE}$ — оба переходят к L-уровню или активизируются дешифратором адреса и линией управления считыванием микропроцессора. Сигнал входа $R/\overline{W}$ сохраняется на H-уровне или в состоянии считывания. За короткое время после того, как активизация выхода $\overline{OE}$ приняла L-уровень, активизируются выходы данных. Данные из памяти помещаются на шину данных по выходам ОЗУ. Как и в ПЗУ, время доступа в память при считывании является важным

показателем ОЗУ, это время может изменяться в различных статических ОЗУ от 250 до 1000 нс.

На рис. 7.8 приведена временна́я диаграмма процесса записи в ОЗУ. Последовательность событий в ходе этой операции начинается посылкой адреса в ОЗУ и дешифратор адреса, который в свою очередь активизирует линию выбора модуля $\overline{MS}$, составленную входами выбора кристалла на отдельных ИС. После интервала времени, называемого временем адресации, импульсом записи $\overline{WR}$ МП активизирует вход $R/\overline{W}$ ОЗУ и устанавливает его в состоя-

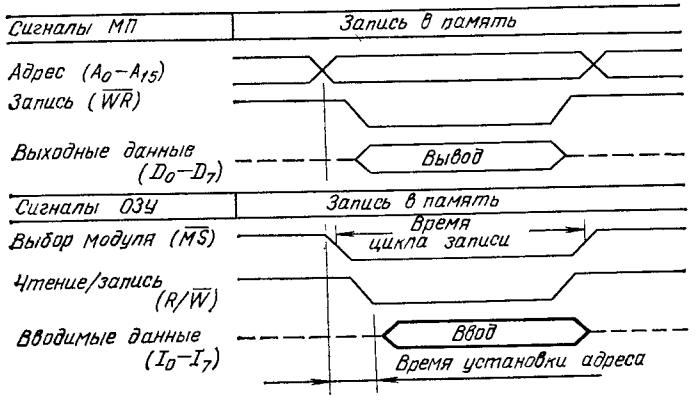


Рис. 7.8 Временна́я диаграмма. Сигналы МП и ОЗУ в ходе записи в память

ние записи. Импульс записи должен быть установлен в кратчайшее время, называемое временем импульса записи (или шириной импульса записи). Когда импульс выдан, записываемые в память данные помещаются МП на шину данных. Короткое время спустя ОЗУ принимает данные с шины и записывает их в ячейку памяти. Минимальное время цикла записи для определенного ОЗУ может изменяться в пределах от 250 до 1000 нс; время адресации составляет 20—200, а длительность импульса записи 180—750 нс.

Маркировка и число выходов выпускаемых ОЗУ изменяются в соответствии с выходными сигналами, вырабатываемыми различными микропроцессорами. В документации сигналы ОЗУ и МП не приводятся подобно тому, как это показано на диаграммах рис. 7.7 и 7.8. Их нужно рассматривать раздельно для оценки совместимости элементов, учи-

тывая, что ячейки памяти динамических ОЗУ требуют обновления (регенерации) их содержимого через каждые несколько микросекунд, устройства интерфейса динамических ОЗУ более сложны, чем статических.

Упражнения

7.8. Сокращением ОЗУ обозначают _____ память.

7.9. Какой тип ОЗУ наиболее прост по интерфейсу?

7.10. См. рис. 7.6. Если МП адресует ячейку памяти 2030H, дешифратор выбирает сегмент памяти _____. Расположенная по этому адресу память является _____.

7.11. См. рис. 7.5. Блок ОЗУ содержит _____ слов длиной по _____ бит каждое. Обычно ОЗУ этой части состоит из _____ (одной ИС, нескольких ИС).

7.12. См. рис. 7.5. Для записи в это ОЗУ адресные входы A_0—A_{11} должны быть активизированы, затем на вход $\overline{MS}$ должен поступить сигнал _____ (H-, L-) уровня, тогда как сигнал на входе R/W поддерживается на _____ (H-, L-) уровне. Наконец, на вход $\overline{OE}$ подается _____ (H, L-) сигнал и выводы I/O_0—I/O_7 становятся _____ (входами, выходами) при операции записи.

7.13. См. рис. 7.5. Какие две линии управления в этой системе составляют шину управления?

7.14. См. рис. 7.7. Какой смысл перехода от H- к L-уровню сигнала на линии $\overline{MS}$ в начале операции считывания из памяти?

7.15. См. рис. 7.7. Напомним, что время доступа в память при считывании является временем, необходимым для того, чтобы имеющиеся в памяти данные поступили на выход ОЗУ начиная с момента, когда установленный адрес поступил на адресный вход. Каково время доступа в память при считывании из типового ОЗУ?

7.16. См. рис. 7.5. В ходе операции записи выводы D_0—D_7 являются _____ (входами, выходами) и _____ (выдают, получают) данные на или с шины данных.

7.17. См. рис. 7.8. Время цикла записи _____ (длиннее, короче), чем время импульса считывания.

Решения

7.8. Оперативную. **7.9.** Статическое ОЗУ. **7.10.** 2 (0010_2); оперативной. **7.11.** 4К; 8 бит; нескольких ИС (около 40). **7.12.** L-; L-; L-сигнал; входами. **7.13.** Линии управления записью и считыванием. **7.14.** Четыре

старших бита адресной шины декодированы дешифратором адреса, который активизирует $\overline{MS}$-линию выбора модуля, так как получает 0010_2 на входе. **7.15.** Около 500 нс. **7.16.** Выходами; выдают. **7.17.** Длиннее.

7.3. ОСНОВНЫЕ ЭЛЕМЕНТЫ ИНТЕРФЕЙСА ПОРТОВ ВВОДА/ВЫВОДА

Операция ввода или вывода включает в себя передачу данных из (или) требуемой периферии. Микропроцессор является ядром всех операций. Ввод соответствует потоку данных в МП, вывод — из МП. Ячейки, куда данные входят или выходят, называются обычно портами ввода или вывода.

Согласно табл. 6.5 оказывается, что наш МП использует команды IN или OUT для передачи данных посредством портов ВВ. Реализация этих команд показана на рис. 7.9, *а*. Мнемониками являются IN и OUT для вводов и выводов соответственно; рис. 7.9, *а* дает форматы и КОП этих команд, которые соответствуют номеру периферии или адресу порта. Адрес порта (1 байт) позволяет выбрать один из 256 (2^8) портов, и адрес передается восемью адресными линиями младших разрядов (A_0—A_7). На рис. 7.9, *а* представлены также два дополнительных выхода сигналов управления. При операции OUT используется

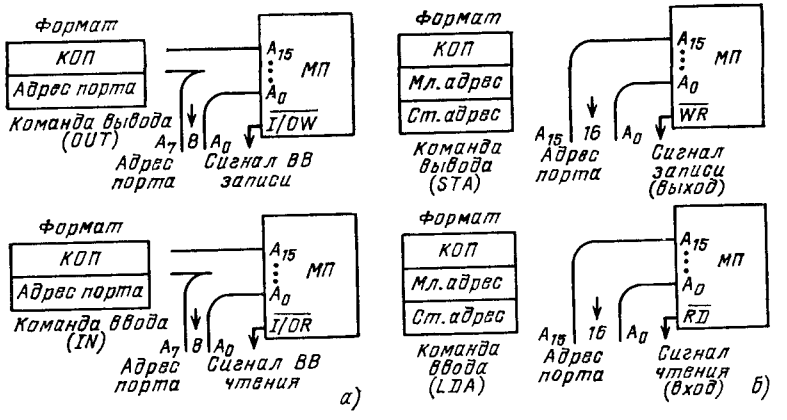


Рис. 7.9. Ввод/вывод данных и сигналы МП:
а — при изолированном ВВ; *б* — при ВВ по принципу доступа в память

особый сигнал *ВВ запись*, т. е. $\overline{I/OW}$; в случае операции IN — *ВВ считывание*, т. е. $\overline{I/OR}$. Это два L-активных сигнала. Использование специального управления, такого, как $\overline{I/OW}$ и $\overline{I/OR}$, соответствует *изолированному ВВ* или вводу/выводу через аккумулятор.

Передача данных, выполняемая на основе команд *IN* и *OUT*, классифицируется как *программно-контролируемый ВВ*, т. е. ею управляют команды программы. Передача данных может быть вызвана периферией, выдающей сообщение типа «Я готова выдать (или принять) данные». Чтобы вовлечь МП, периферия использует *прерывания* (вспомним, что когда МП получает запрос на прерывание, он завершает выполнение текущей команды, затем ветвится на подпрограмму обслуживания прерывания, эта подпрограмма может содержать или не содержать операции ВВ).

Программно-управляемый ВВ может быть выполнен двумя способами: первый — изолированный ВВ (см. рис. 7.9, *а*) выполняется посредством команд IN и OUT; по второму — положение входов и выходов определяется как адреса обычной памяти. Второй способ ВВ реализуется по *принципу доступа в память*, и тогда используются обычные команды. Обратимся к рис. 7.9, *б*, где мы использовали команду прямого размещения содержимого аккумулятора STA при выводе данных в порт вывода; на том же рисунке тот же тип команды использован затем и для ввода данных (команда LDA). Мы видим здесь, что адресные линии должны быть декодированы и служат для выбора адреса порта ввода или порта вывода.

Используются также обычные сигналы управления $\overline{WR}$ и $\overline{RD}$. Таким образом, для ввода и вывода данных по принципу доступа в память могут быть использованы все команды обращения.

Последний способ, очевидно, наиболее распространен и может быть применим в любом МП. Метод изолированного ВВ применим только в МП, снабженных как командами IN и OUT, так и специальными выводами управления ВВ при записи и считывании.

Обычно под выводом подразумевают вывод на периферию. Однако на практике выводы попадают не сразу на периферию, а в устройство памяти, где помещаются данные для периферии (см. рис. 7.1, где мы ввели адаптеры интерфейса ввода и вывода, которые являются ни чем

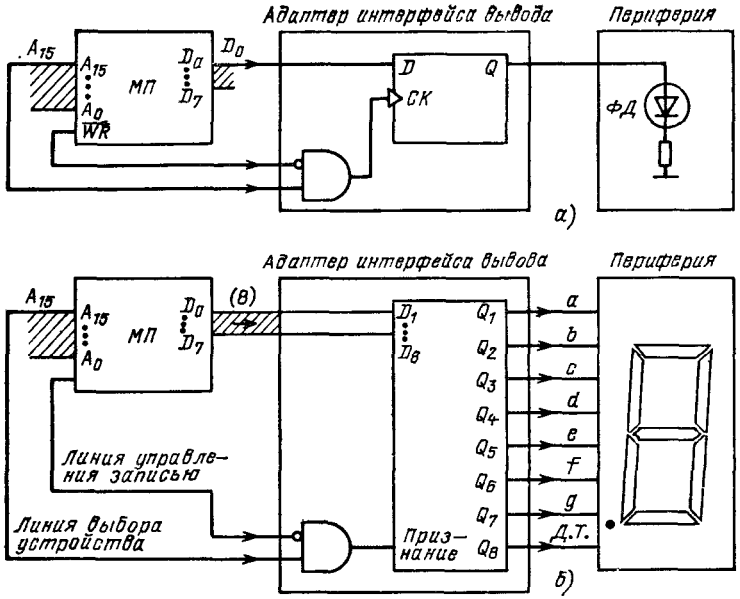


Рис. 7.10 Интерфейс:
а — периферии с отдельным битом; *б* — с семисегментным индикатором

иным, как устройствами памяти, и которые обладают всеми их свойствами).

На рис. 7.10, *а* изображен интерфейс МП с периферией. Отметим, что в качестве индикатора выхода здесь стоит фотодиод, а адаптер интерфейса вывода содержит *D*-триггер. Предположим, что МП выполняет команду прямого размещения содержимого аккумулятора $(A) = 0000\ 0001$. Если предположить также адрес 8000H, то линия A_{15} будет находиться в H-активном состоянии и активизирует нижний вход элемента И. Некоторое время спустя H-активный сигнал появляется на линии D_0 шины данных. Сигнал управления записью $\overline{WR}$ переходит к L-уровню и активизирует элемент И, который запирает 1 в *D*-триггере. На выходе *D*-триггера H-сигналом зажигается фотодиод (или индикатор бита). В качестве примера на рис. 7.10, *а* показан процесс индикации единственного бита данных.

Интерфейс вывода на рис. 7.10, *б* несколько сложнее. Он передает 8 бит данных МП в адаптер интерфейса вывода через шину данных. Адаптер помещает данные в

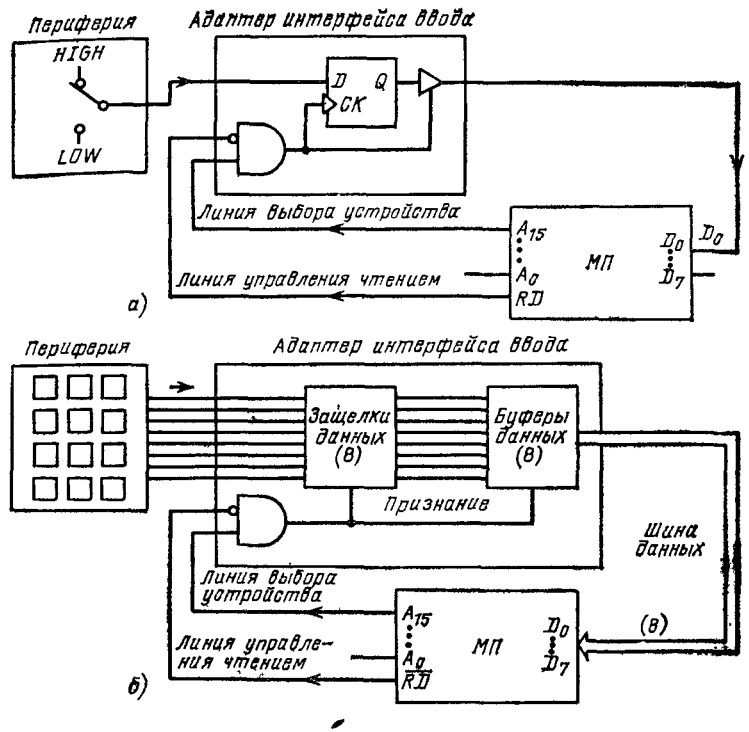


Рис. 7.11. Интерфейс:
а — при вводе единичной коммутации, *б* — при вводе с клавиатуры

8-разрядную защелку обычным способом. Заметим, что адресная линия A_{15} должна быть в H-состоянии, тогда как вывод управления записью $\overline{WR}$ должен быть в L-состоянии в течение времени, необходимого для того, чтобы запереть данные в семисегментном индикаторе. Каждый сегмент индикатора (*a*—*g* и десятичная точка) работает так же, как отдельный диод на рис. 7.10, *а*. Запертый в любом выводе защелки H-сигнал вызывает свечение соответствующего сегмента.

На рис. 7.11, *а* представлено другое периферийное устройство и его интерфейс с МП. Здесь речь идет о простом коммутаторе, позволяющем выбрать логический L- или H-уровень. Отдельная линия данных соединена с вводом *D*-триггера внутри адаптера. Когда МП выполняет команду типа ЗАГРУЗИТЬ А прямо из ячейки памяти 8000H,

элемент И активизируется, захватывая входные данные. Буфер вывода становится разрешенным, что позволяет данным расположиться вдоль отдельной линии D_0 шины данных. Микропроцессор принимает бит HIGH с шины данных и передает его в аккумулятор. Интервал спустя элемент И сбрасывается. Выход адаптера интерфейса ввода затем запрещается и переводится в состояние высокого сопротивления, он не оказывает больше влияния на другие передачи на шину данных.

На рис. 7.11 мы используем способ ВВ по принципу доступа в память. Обычно несколько адресных линий декодируются одним адресным дешифратором для того, чтобы активизировать линию выбора устройства. На рис. 7.11, *б* приведена расширенная система ввода одного отдельного бита. Здесь параллельные данные объемом 8 бит вводятся с клавишного устройства. Управление считыванием $\overline{RD}$ и простая адресация те же, что и в предыдущем примере.

Организация ВВ по принципу доступа в память используется очень распространенными микропроцессорами, такими, как Motorola 6800 и МОП-технологии[1] 6502. Изолированный ВВ (иногда называемый канальным ВВ данных) используется таким семейством микропроцессоров, как Intel 8080/8085 и Zilog 80.

Упражнения

7.18. Использование команды ВВ IN и OUT соответствует способу _____ (изолированного ВВ, ВВ по принципу доступна в память).

7.19. Способ, по которому УВВ обрабатывается, как обычные адреса памяти, называется _____.

7.20. Когда выводы управления считыванием $\overline{RD}$ и $\overline{WR}$ соединяются с адаптером интерфейса ВВ, используется способ _____.

7.21. См. рис. 7.11, *б*. Данные передаются в аккумулятор, после чего на адресную линию A_{15} поступает _____ (H-, L-) сигнал, на линию управления считыванием поступает _____ (H-, L-) сигнал, что в свою очередь разрешает защелки и триггеры.

7.22. См. рис. 7.11, *б*. В ходе операции ввода МП выполняет, вероятно, команду _____ (IN, OUT).

[1] МОП — металл — окисел — полупроводник. — *Прим. ред.*

7.23. Когда буферы шины разрешены, они _____ (запрещают, позволяют) передачу данных с защелок на шину.

7.24. См. рис. 7.11, *б*. На какой адрес ответит интерфейс?

Решения

7.18. Изолированного ВВ. **7.19.** По принципу доступа в память. **7.20.** По принципу доступа в память. **7.21.** H-сигнал; на линии $\overline{RD}$ устанавливается L-. **7.22.** По принципу доступа в память, команда передачи LDA. **7.23.** Позволяют. **7.24.** Каждый раз, когда адресная линия старшего разряда A_{15} находится в H-состоянии, линия выбора устройства активизируется. Этот метод адресации непрактичен в случае, когда 32К памяти (8 страниц по 4К каждая снизу на рис. 7.3) будут использоваться только этим устройством вывода.

7.4. ИНТЕРФЕЙС С РЕАЛЬНЫМИ ПОРТАМИ ВВ

Разработчики выпускают порты ВВ в виде ИС. Тому пример — 8-разрядный элемент ВВ Intel 8212. Интегральная схема Intel 8212 может быть использована как адаптер порта ввода или вывода.

На рис. 7.12 приведен МП, подобный типовому, имеющий интерфейсом с семисегментным индикатором элемент

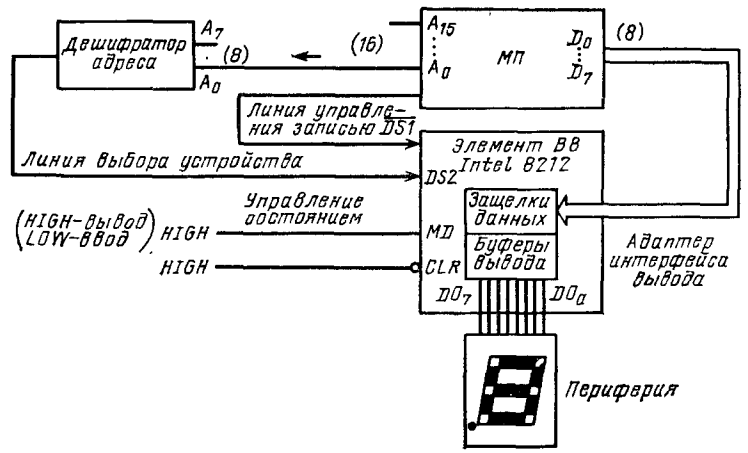


Рис. 7.12. Интерфейс семисегментного индикатора, построенный на элементе ВВ Intel 8212

Intel 8212. Этот индикатор является периферией. На схеме МП имеет изолированный ВВ. Отметим, что в этом случае выходная линия выбора устройства полностью декодируется по восьми адресным линиям младших разрядов (A_0—A_7). Имеется также выход управления записью $\overline{I/OW}$ (при изолированном ВВ можно только использовать команды IN и OUT).

На рис. 7.12 ИС 8212 используется в порте вывода, что можно установить, заметив, что на входе управления состоянием *MD* установлен Н-уровень. Линии шины данных подсоединены к восьми входным линиям DI_0—DI_7 элемента ВВ 8212, чьи выходные линии DO_0—DO_7 подсоединены к входам индикатора. Элемент Intel 8212 имеет восемь защелок данных и выходные буферы. Два входа выбора $\overline{DS1}$, *DS2* являются управляющими для элемента 8212, используемого в рассматриваемом способе ВВ. Когда $\overline{DS1}$ активизируется L-сигналом, а *DS2*—Н-сигналом, данные, поступающие с шины захватятся защелками данных и появятся на выходных выводах DO_0—DO_7, активизируя сегменты индикатора.

На рис. 7.13 приведена временна́я диаграмма работы порта Intel 8212 в состоянии вывода. Отметим, что как только активизируются входы сигналов управления $\overline{DS1}$ и

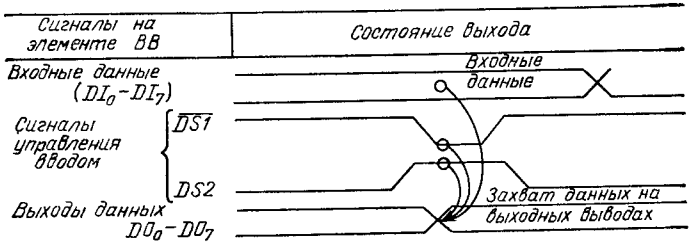


Рис. 7.13. Временна́я диаграмма. Сигналы элемента ВВ в состоянии вывода данных

DS2, выходные данные захватываются в порте ВВ и располагаются на выходных выводах. В состоянии вывода выходные буферы постоянно разрешены, следовательно, захваченные данные появляются даже после того, как линии управления $\overline{DS1}$ и *DS2* возвращаются в состояние сброса.

Упражнения

7.25. См. рис. 7.12. Данные, поступающие на выводы DI_0—DI_7 порта ВВ 8212, поступают с шины _____ (адреса, данных) системы.

7.26. См. рис. 7.12. Порт ВВ 8212 используется как порт _____ (ввода, вывода) параллельных восьмибитовых данных.

7.27. См. рис. 7.13. Входной сигнал $\overline{DS1}$ вызван _____ (дешифратором адреса, сигналом I/OW) МП.

7.28. См. рис. 7.13. Входной сигнал *DS2* вызван _____ _____ (дешифратором адреса, шиной данных).

7.29. Буферы вывода ИС Intel 8212 _____ (активизированы, сброшены) постоянно, когда устройство находится в состоянии вывода. Это означает, что они _____ (отправляют данные в порты, блокируют данные портов) DO_0—DO_7.

7.30. См. рис. 7.13. Данные захвачены портом ВВ 8212, когда одновременно активизированы вводы управления _____.

7.31. См. рис. 7.12. Система снабжена интерфейсом _____ (изолированного, действующего по способу обращения к памяти) ВВ.

Решения

7.25. Данных. **7.26.** Вывода. **7.27.** Сигналом I/OW. **7.28.** Дешифратором адреса. **7.29.** Активизированы; отправляют данные в порты. **7.30.** $\overline{DS1}$ и *DS2*. **7.31.** Изолированного.

7.5. СИНХРОНИЗАЦИЯ ПРЕРЫВАНИЕМ ПЕРЕДАЧИ ДАННЫХ В УВВ

До сих пор мы предполагали, что когда программа указывала МП ввести данные в порт, они уже имелись в наличии и были расположены в строго определенном месте. Однако не всегда это так, потому что периферия (например, клавишное устройство) имеет различное с МП быстродействие. В этом случае имеется несколько способов решения проблемы. К ним относятся методы *опросов* и *прерываний*.

Опрос называется еще программируемым ВВ. Это наиболее простой метод синхронизации, и используется

он в небольших специализированных устройствах. Основной идеей опроса является ввести и(или) вывести данные последовательно, используя в программе *цикл опроса.*

Предположим очень простую систему с одним только коммутатором на входе и единственным индикатором на выходе. Приведенные на рис. 7.10, *а* и 7.11, *а* системы могли бы быть составлены отдельным микропроцессором, одним только коммутатором с адаптером интерфейса ввода и одним фотодиодом с адаптером интерфейса вывода. Можно было бы предусмотреть предельно простой цикл опроса, в котором сначала считывалось бы состояние коммутатора, и затем оно записывалось бы в выходной индикатор. В таком случае МП опрашивал бы вход и одновременно воздействовал на выход через каждые несколько микросекунд.

В более общем случае в цикле можно опрашивать одно или несколько УВВ. Микропроцессор опрашивает первое устройство — нужно ли ему обслуживание, и если да, устанавливает индикатор состояния, т. е. устройство получает требуемое обслуживание. В противном случае (индикатор состояния не устанавливается) МП продолжает опрос.

Представленная на рис. 7.14 система снабжена линией требования прерывания, чтобы дать знать МП о готовности данных для передачи в ЦП. Вспомним, что типовой МП имеет только один ввод требования прерывания (INTR). Активизированный Н-сигналом МП выполняет текущую команду, передает на хранение в стек содержимое аккумулятора, регистра и счетчика команд и ветвится в подпрограмму обслуживания прерывания. Затем МП возвращается в основную программу. В рассматриваемом примере подпрограмма обслуживания прерывания могла бы быть простой, вводящей и помещающей данные, поступающие с клавишного устройства.

Рассмотрим систему, приведенную на рис. 7.14. Микропроцессор имеет интерфейс порта ВВ с клавишным устройством, построенный на ИС 8212. Управление состоянием (линия *MD*) устройства имеет *L*-уровень и воздействует на адаптер ввода. Вход сброса *CLR* в устройстве 8212 дезактивируется Н-сигналом. Правильные 8-разрядные данные с клавишного устройства поступают на входы данных DI_0—DI_7 элемента 8212. В этой системе *линия строба* (*STB*) управляет защелками данных, тогда как входы выбора устройства $\overline{DS1}$ и *DS2* элемента 8212 управляют буферами выхода.

Клавишное устройство должно поместить 8-разрядное слово на входы данных элемента 8212 и выдать импульс

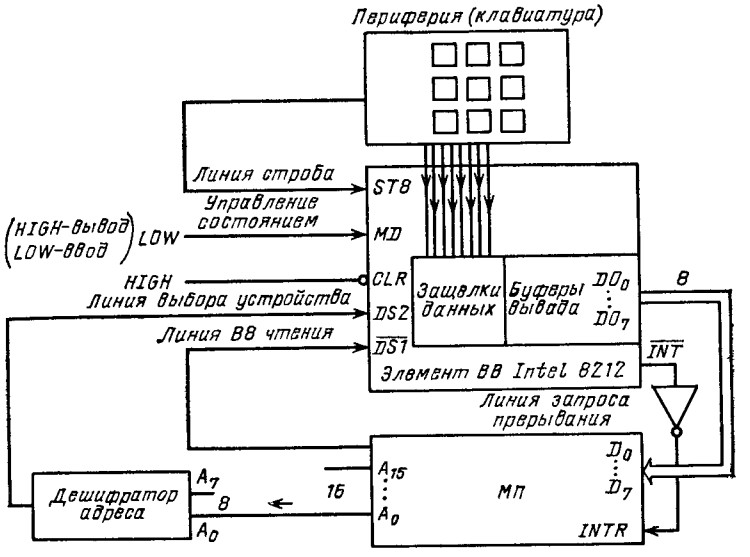


Рис. 7.14. Интерфейс клавишного устройства, построенный на элементе ВВ Intel 8212 с простым прерыванием

HIGH (строб), чтобы удержать эти данные во внутренних защелках порта ВВ. Эти события представлены на рис. 7.15 тремя верхними графиками. Строб HIGH клавишного устройства снова захватывает данные в элемент ВВ Intel 8212.

Согласно нижним графикам на рис. 7.15 после того, как данные стробированы в защелках, контуром элемента 8212 выдается импульс прерывания INT, который поступает в МП по линии требования прерывания; линии управления буфером вывода ($\overline{DS1}$ и *DS2*) активизированы. Кривые *3* и *6* иллюстрируют размещение захваченных данных командами $\overline{DS1}$ и *DS2* элемента Intel 8212 на шине данных в течение очень короткого промежутка времени. Заметим, что выводы элемента ВВ Intel 8212 возвращаются в свое

третье состояние сразу после того, как команды буферов $\overline{DS1}$ и $DS2$ признаются недействительными.

Рассмотрим снова интерфейс клавишного устройства на рис. 7.14, когда система имеет изолированный ВВ. Специальный сигнал управления ВВ считывания *I/OR* активизирует вход $\overline{DS1}$ устройства Intel 8212. Восемь адресных линий младших разрядов (A_0—A_7) полностью декодируются дешифратором адреса, выводы которого активизируют

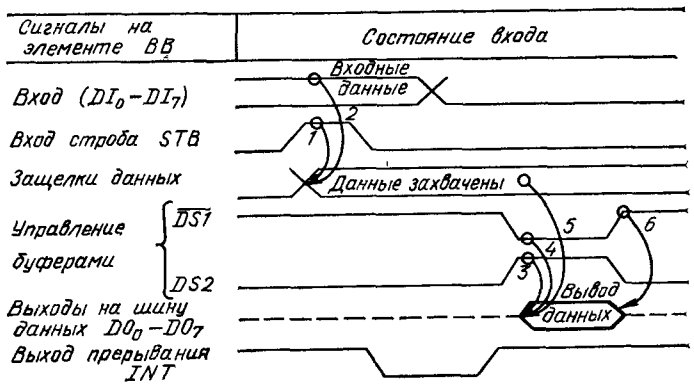


Рис. 7.15. Временна́я диаграмма. Сигналы порта ВВ в состоянии ввода по простому прерыванию

вход *DS2* по линии выбора устройства. Не забудем, что в этой системе входы $\overline{DS1}$ и *DS2* управляют буферами вывода элемента порта ВВ 8212. Заметим, что в линию требования прерывания включен инвертор. Инвертор согласовывает L-активный выход $\overline{INT}$ Intel 8212 с H-активным входом INTR МП.

Прерывание информирует МП о том, что УВВ готово и надо действовать. Реализует эту операцию специальная подпрограмма обслуживания прерывания. Если несколько устройств вызывают прерывание МП, линии вызова прерываний подвергаются операциям ИЛИ, и МП в таком случае необходимо определить ответственное за прерывание УВВ. Этот процесс выбора относится к способу опроса и составляет схему *опроса-прерывания*. В таком случае каждому порту ВВ соответствует одно состояние, код которого указывает МП — готов ли порт ввести или вывести данные.

Существуют МП, снабженные несколькими входами прерывания и командами признания или непризнания по крайней мере некоторого числа этих входов. Многие МП снабжены *векторным прерыванием*, когда МП знает устройство, вызвавшее прерывание, и ветвится на подпрограмму обслуживания соответствующего прерывания. Многие МП снабжены также устройствами определения приоритета обслуживания (аппаратно или программно) в случае двух одновременных прерываний (*арбитраж приоритета прерывания*). Эта система решает, согласиться ли с приоритетом обслуживания одного из прерываний, она обычно содержится в специальной ИС (например, устройство Intel 8259).

Преимущество опроса над процедурой прерывания состоит в том, что первый требует меньше аппаратных средств и, находясь под программным контролем, является синхронным. Недостатком является потребность в очень развитых программах, занимающих время МП в случае большого числа устройств опроса, и необходимость довольно значительного промежутка времени для ответа на один запрос.

Преимуществом прерывания над опросом является быстродействие ответа, лучшее использование МП и потребность во много раз меньших программных средств. Недостаток состоит в том, что функционирование МП асинхронно и требует сложного аппаратного интерфейса.

Упражнения

7.32. Запрос на прерывание поступает _____ (из УВВ, по команде программы).

7.33. Запрос на прерывание поступает в МП, когда порт ВВ _____ (готов, не готов) к обслуживанию.

7.34. См. рис. 7.14. Порт ВВ Intel 8212 находится в состоянии _____ (ввода, вывода).

7.35. См. рис. 7.14. Для того чтобы данные, выданные клавишным устройством, были задержаны во внутренних защелках элемента Intel 8212, вход _____ (*CLR, STB*) должен получить _____ (H-, L-) импульс. Импульс выдается _____ (клавишным устройством, МП).

7.36. См. рис. 7.15. После того как данные захвачены в

элементе Intel 8212, _____ (вывод $\overline{INT}$ принимает L-состояние, буферы вывода активизируются).

7.37. См. рис. 7.14. После выполнения команды IN МП переводит линию считывания в _____ (H-, L-) состояние, а линию выбора устройства — в _____ (H-, L-) состояние, т. е. активизируются _____ (защелки данных, буферы вывода).

7.38. Другим названием опроса является ВВ _____.

7.39. Перечислить достоинства и недостатки ввода данных прерыванием над способом опроса.

Решения

7.32. Из УВВ. **7.33.** Готов. **7.34.** Ввода. **7.35.** *STB*; H-; клавиатурой. **7.36.** Вывод $\overline{INT}$ принимает L-состояние. **7.37.** L-; H-; буферы вывода. **7.38.** Программируемый. **7.39.** Достоинства: большее быстродействие, лучшее использование микропроцессора, меньше программиых средств. Недостаткн: асинхронизм, более сложный аппаратиый интерфейс.

7.6. ДЕКОДИРОВАНИЕ АДРЕСОВ

На рис. 7.16 представлена функциональная микропроцессорная схема информационной системы. Данные могут быть введены с клавишного устройства и выведены на отдельный индикатор. Линии управления и выбора данных представлены полностью. Адаптеры интерфейса ввода и вывода являются элементами Intel 8212. Синхронизация передачи данных из интерфейса выполняется простым прерыванием. В системе используется ВВ по принципу доступа в память (порты обрабатываются как ячейки памяти). Каждый из интерфейсов был рассмотрен в § 7.5, здесь же мы остановимся на анализе цепей дешифратора адреса, которые могли бы быть использованы в такой системе.

На рис. 7.16 приведен вход дешифратора адреса, состоящий из четырех адресных линий старших разрядов (A_{12}—A_{15}). Назначением дешифратора адреса является выдача:

1. L-сигнала на линию выбора устройства ПЗУ, когда на четырехадресных линиях старших разрядов содержится 0000_2.
2. L-сигнала на линию выбора ОЗУ, когда на четырех адресных линиях старших разрядов содержится 0010_2.
3. Выходного H-сигнала на выход выбора устройства интерфейса, когда на четырех адресных линиях старших разрядов содержится 1000_2.

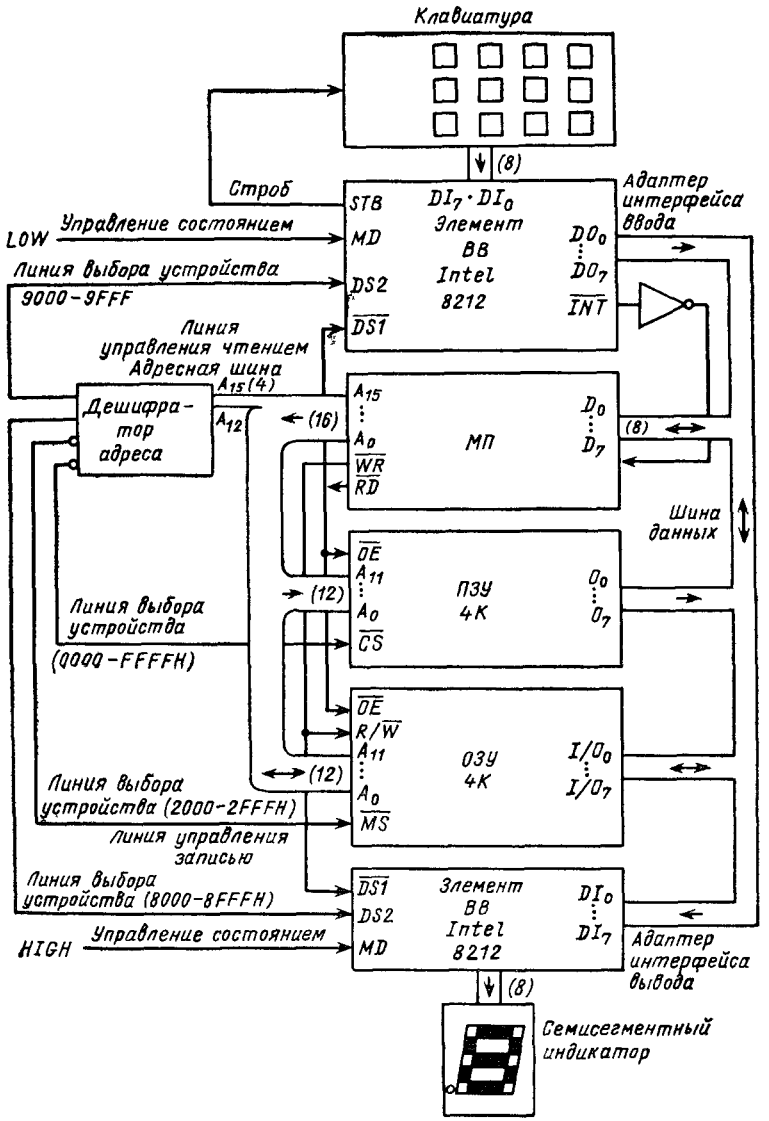


Рис. 7.16. Функциональная схема микропроцессорной информационной системы

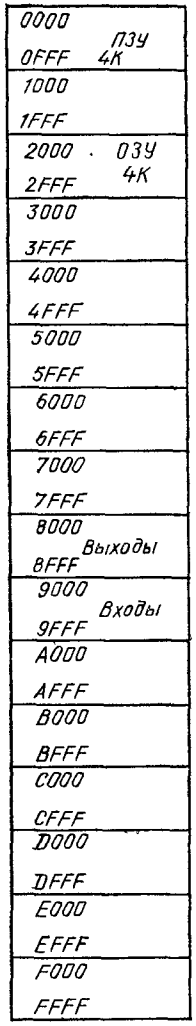


Рис. 7.17. Воображаемая память системы, приведенной на рис. 7.16

4. Выходного H-сигнала на выход выбора устройства интерфейса, когда на четырех адресных линиях старших разрядов содержится 1001_2.

На рис. 7.7 приведена воображаемая память системы. В нулевом сегменте (адреса 0000—0FFFH) находятся 4096 ячеек ПЗУ. Номер сегмента здесь эквивалентен 0000_2, согласно которому выдается сигнал выбора ПЗУ. Во втором сегменте содержатся 4096 других ячеек ОЗУ. Адреса, относящиеся к порту вывода, находятся в восьмом сегменте, а относящиеся к порту ввода — в девятом. Отметим, что любой из этих 4096 адресов (начиная с 8000H) будет активизировать один соответствующий выход интерфейса. Это связано с тем, что декодируются только 4 из 16 адресных линий: в этом состоит процедура частичного декодирования адреса, которая выполняется очень хорошо, если не добавлять другие устройства. Таким же образом любой из 4096 адресов 9000—9FFFH будет активизировать вход интерфейса, здесь это также связано с тем, что адреса частично декодированы.

На рис. 7.18 показано простое решение декодирования адреса. Для декодирования четырех адресных линий старших разрядов (A_{12}—A_{15}) используются ИС дешифратора 1 из 16 и два инвертора. Напомним, что дешифратор (или мультиплексор) иногда действует как вращающийся коммутатор, активизируя в заданный момент времени только один выход (*0*—*15*). Выход дешифратора 1 из 16 зависит от состояния линий выбора (S_0—S_3). Следовательно, если на линиях выбора имеется сигнал 0000, L-сигналом активизируется выход *0*. Из рисунка видно, что выход дешифратора выдает *0* на линию выбора нулевого сегмента памяти. Если дешифратор выбирает линии 0010 (S_3=0; S_2=0 ; S_1=1; S_0=0), выход *2* дешифратора 1 из 16 активизируется L-сигналом. Выход *2* используется для признания выбора ОЗУ.

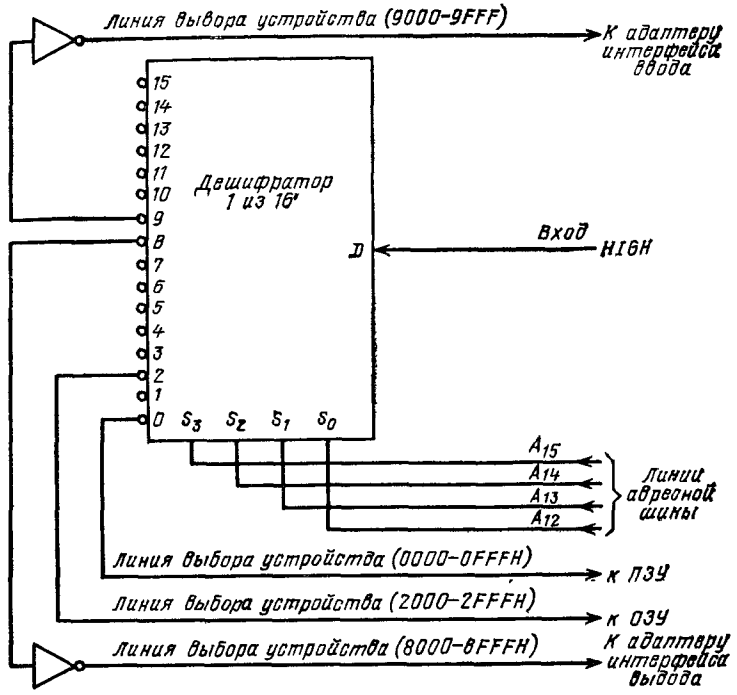


Рис. 7.18. Использование дешифратора 1 из 16 в качестве дешифратора адреса

Если четыре адресные линии старших разрядов содержат 1000_2, выход *8* дешифратора активизируется L-сигналом. Инвертор инвертирует этот выход для того, чтобы он был совместимым с импульсом HIGH, необходимым для активизации интерфейса вывода. Сигнал 1001_2 на выходах выбора дешифратора 1 из 16 также признает интерфейс ввода сигналом HIGH. Не использованные здесь выводы могут быть использованы в дальнейшем для развития периферии или памяти.

Адреса ПЗУ и ОЗУ на рис. 7.16 полностью декодированы, т. е. декодированы 16 линий и, следовательно все воз-

можные адреса доступны. Например, команда LDA, 9000H ввела бы данные, исходящие из рассматриваемого как адреса памяти клавишного устройства. Рассмотрим также команду LDA, 9FFFH, т. е. ЗАГРУЗИТЬ данные в аккумулятор, в нашем случае она будет связана с теми же поступающими с клавишного устройства данными. В случае ВВ по принципу доступа в память для того, чтобы порт ВВ был идентифицирован единственным шестнадцатеричным числом, нужно, чтобы 16 линий были декодированы или, если угодно, чтобы были полностью декодированы адресные шины.

Рисунок 7.19 представляет собой схему полного декодирования. На рис. 7.19, *а* показан ввод 16 адресных линий в дешифратор. Только в случае, когда адрес порта будет

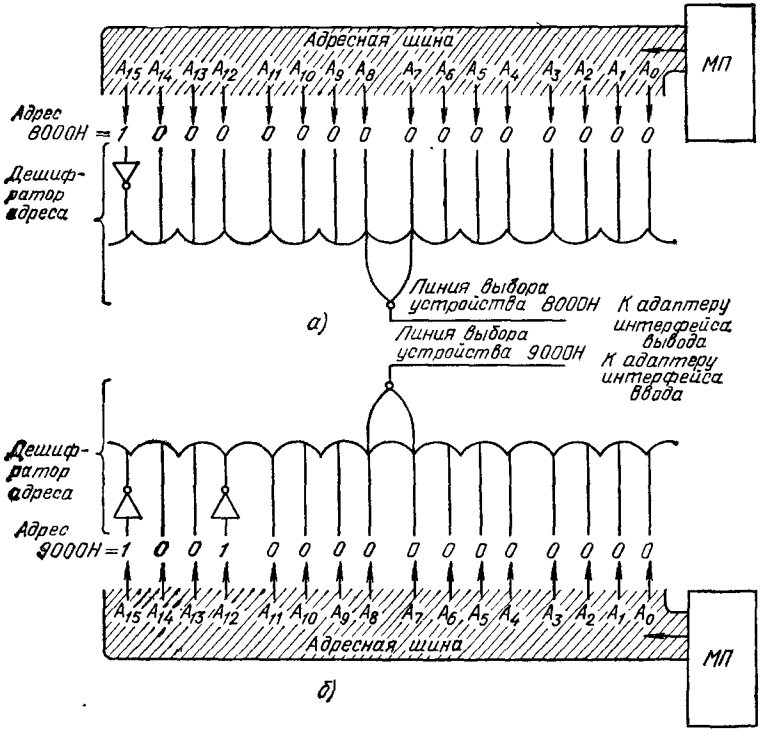


Рис. 7.19. Логическая схема полного декодирования:
а — устройства вывода по адресу 8000H; *б* — устройства входа по адресу 9000H

8000H, элемент НЕ-ИЛИ выдает Н-сигнал выбора устройства в интерфейс вывода. Все другие возможные сочетания 16 вводов дадут на выходе L-состояние, которое отменит линию выбора устройства и интерфейс вывода.

Рисунок 7.19, *б* показывает полную декодировку интерфейса ввода. Здесь 9000H является единственным адресом, который приведет выход элемента НЕ-ИЛИ в Н-состояние, активизируя интерфейс вывода по линии выбора данных. Все другие сочетания 0 и 1 повлекут за собой на выходе дешифратора L-состояние, при котором интерфейс не признается.

Упражнения

7.40. См. рис. 7.16 и 7.17. Если МП помещает 0300H на адресную шину, _____ (ПЗУ, ОЗУ) адресуется активизацией своей линии выбора _____ (Н-, L-) сигналом.

7.41. См. рис. 7.16 и 7.17. Если МП помещает 8300H на адресную шину, _____ (вход, выход) интерфейса адресуется подачей на свою линию выбора _____ (Н-, L-) сигнала.

7.42. См. рис. 7.16. Постоянное запоминающее устройство с 4096 адресами (0000H—0FFFH) _____ (полностью, частично) декодируемо.

7.43. См. рис. 7.16. 4096 адресов 8000H—8FFFH декодируемы _____ (полностью, частично).

7.44. См. рис. 7.16. Если входы выбора дешифратора 1 из 16 $S_3=1$; $S_2=1$; $S_1=1$; $S_0=0$, выход _____ разрешается и выдает _____ (Н-, L-сигнал).

7.45. См. рис. 7.16. Предположить, что в систему добавлено другое ОЗУ на 4К с адресами 1000H—1FFFH. Выход _____ дешифратора 1 из 16 на рис. 7.18 мог бы быть использован для управления линией выбора этого модуля.

Решения

7.40. ПЗУ; L-. **7.41.** Выход; Н-. **7.42.** ПЗУ имеет 4096 полностью декодируемых адресов (0000H—0FFFH), четыре адресные линии старших разрядов (A_{12}—A_{15}) декодируются кроме него и устройством вывода. **7.43.** Адреса 8000H—8FFFH декодируемы частично. Линии младших разрядов (A_0—A_{11}) декодируются дешифраторами адреса или устройством вывода. **7.44.** 1110, сегмент 1; L-сигнал. **7.45.** Выход *1*.

Дополнительные упражнения к гл. 7

7.46. Взаимные соединения элементов системы составляют _____.

7.47. Интерфейс является границей между двумя устройствами, которые должны разделить _____.

7.48. Среди прочих процессов, характеризующих интерфейс, имеется _____ (построение матриц, синхронизация).

7.49. Три взаимосвязанные системы или системы, обеспечивающие поток данных в ЭВМ, составляют _____.

7.50. Когда МП прекратит управление адресными шинами и шинами данных для того, чтобы периферия могла получить доступ в память, эта операция сокращенно называется _____.

7.51. См. рис. 7.2. Адресная шина _____ (полностью, частично) декодируема.

7.52. См. рис. 7.2. После выдачи _____ (H-, L-) сигнала на входы $\overline{CS}$ и $\overline{OE}$ ПЗУ будет _____ (записывать, считывать).

7.53. См. рис. 7.4. Кружки и стрелки указывают на временной диаграмме на соотношения причины и _____.

7.54. См. рис. 7.2. Выходы (O_0—O_7) ПЗУ находятся в состоянии _____ (HIGH, высокого сопротивления), когда память недоступна.

7.55. См. рис. 7.5. Без цепи регенерации памяти ОЗУ является _____ (статическим, динамическим).

7.56. См. рис. 7.5. Оперативное запоминающее устройство составлено из _____ (одной, многих) ИС.

7.57. См. рис. 7.5. Если линия управления записью содержит L-сигнал, ОЗУ находится в состоянии _____ (записи, считывания).

7.58. Управление считыванием на рис. 7.5 признает вход _____ ОЗУ (H-, L-) сигналом.

7.59. Полупроводниковая оперативная память называется _____.

7.60. См. рис. 7.5. Во время считывания выводы D_0—D_7 являются _____ (входами, выходами) и _____ (получают, выдают) данные.

7.61. Специальные команды IN и OUT используются при _____ (изолированном ВВ, программируемом ВВ).

7.62. _____ (Изолированные, Программируемые) ВВ используются наиболее часто.

7.63. См. рис. 7.10, *б*. Интерфейс вывода содержит один порт и _____ для помещения данных.

7.64. См. рис. 7.10, *а*. Периферия является _____ (триггером, фотодиодом).

7.65. См. рис. 7.11, *а*. Треугольное устройство внутри интерфейса ввода является _____ вывода.

7.66. См. рис. 7.12. Система использует _____ (программируемый, изолированный) ВВ.

7.67. См. рис. 7.12. Буферы вывода интерфейса ВВ Intel 8212 признаются _____ (сигналом на выводе $\overline{DS1}$, постоянно).

7.68. См. рис. 7.14. Запись готовых данных осуществляется _____.

7.69. См. рис. 7.14. Прерывание запрашивается _____ (интерфейсом ввода, МП) _____ (до, после) того, как данные захватятся стробом интерфейса ВВ Intel 8212.

7.70. См. рис. 7.14. Получив запрос на прерывание, МП завершает выполнение текущей команды, помещает в стек текущий регистр и счетчик команд и ветвится в подпрограмму, называемую _____.

7.71. _____ является способом, согласно которому периодически выбирается каждое устройство ВВ для того, чтобы знать, запрашивает ли оно прерывание.

7.72. См. рис. 7.16. _____ (Интерфейс вывода, ОЗУ) захватывает данные и удерживает их на входах индикатора.

7.73. См. рис. 7.16. Периферийным устройством ввода является _____.

7.74. См. рис. 7.16. Синхронизация передачи данных осуществляется системой _____ (опроса-прерывания, простого прерывания).

7.75. См. рис. 7.16. Адреса ОЗУ и ПЗУ декодируемы _____ (частично, полностью).

7.76. См. рис. 7.16. Адреса порта ввода и вывода декодируемы _____ (полностью, частично).

Решения

7.46. Интерфейс. **7.47.** Информацню. **7.48.** Снихронизация. **7.49.** Шину данных, адресную и управляющую шины. **7.50.** ПДП. **7.51.** Полностью. **7.52.** L-; считывать. **7.53.** Следствия. **7.54.** Высокого сопротивления. **7.55.** Статическим. **7.56.** Многих. **7.57.** Записи. **7.58.** $\overline{OE}$; L-. **7.59.** ОЗУ. **7.60.** Входами; получают. **7.61.** Изолированном ВВ. **7.62.** Программируемые. **7.63.** Защелку. **7.64.** Фотодиодом (указателем бит). **7.65.** Бу-

фером **7.66.** Изолированный. **7.67.** Постоянно. **7.68.** Прерыванием. **7.69.** Интерфейсом ввода; после. **7.70.** Подпрограммой обслуживания прерывания. **7.71.** Опрос. **7.72.** Интерфейс вывода. **7.73.** Клавишное устройство. **7.74.** Простого прерывания. **7.75.** Полностью. **7.76.** Частично.

Глава 8
МИКРОПРОЦЕССОРЫ INTEL 8080/8085

Первый МП был выпущен фирмой Intel в 1971 г. Тогда фирма изготовила и реализовала 4-разрядные МП 4004 и 8-разрядные МП 8008. В 1974 г. пришла очередь МП Intel 8080, который обрабатывает 8-разрядные слова и имеет 16-разрядные адресную шину и указатель стека. Его улучшенным вариантом является МП Intel 8085, в котором содержатся генератор тактовых импульсов, система управления и устройство определения приоритета прерываний, интеграция которых снижает число составляющих микропроцессорную систему ИС. Микропроцессор Intel 8085 работает также с единственным уровнем питающего напряжения +5 В. Он использует те же команды, что и МП Intel 8080, что делает оба устройства совместимыми. Наконец, Intel 8085 имеет две дополнительные команды, располагая, таким образом, большими возможностями благодаря содержащимся в нем дополнительным аппаратным средствам.

Типовой МП (см. гл. 5—7), был упрощенной версией МП Intel 8080/8085, поэтому довольно просто понять его действия.

Более глубокая функциональная интеграция с меньшим числом ИС является эволюционным свойством МП. Совсем недавно простая система могла содержать до 20—30 ИС. Представленная на рис. 8.1 система содержит их только три. В ней использован МП Intel 8085, который управляет шиной системы и двумя другими специальными ИС интерфейса с периферией.

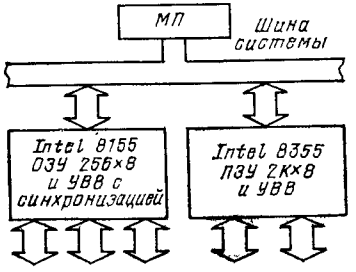


Рис. 8.1. Функциональная схема простой мнкропроцессорной системы с микросхемами интерфейса

Составляющие интерфейса, представленные на рис. 8.1, являются ИС Intel 8155 и 8355. Микросхема 8155 содержит 2048 бит памяти статического ОЗУ, организованного в память 256×8 бит; она содержит также три порта ВВ и синхронизатор. Два порта ВВ являются универсальными по 8 бит каждый. Третий (6 бит) может быть использован как порт ввода, вывода или в качестве системы сигналов управления для двух других 8-разрядных портов. Схема 8155 программируема и содержит регистр состояния и 14-разрядный счетчик-синхронизатор.

Другое устройство, представленное на рис. 8.1, является ИС интерфейса периферии 8355. Она содержит ПЗУ емкостью 16 384 бит, организованное в память 2048×6 бит, и два универсальных порта ВВ по 8 бит каждый.

8.1. СХЕМА И НАЗНАЧЕНИЕ ВЫВОДОВ

Восьмиразрядный МП Intel 8085 заключен в корпус типа DIP (с двусторонней упаковкой выводов) с 40 выводами, расположение которых приведено на рис. 8.2; в табл. 8.1 приведено название выводов и их назначение. Постав-

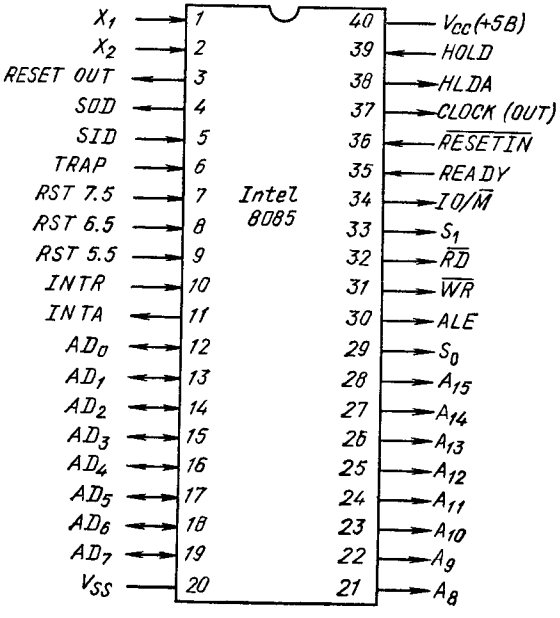


Рис. 8.2. Диаграмма выводов МП Intel 3085

Таблица 8.1. *Названия и назначение выходов МП Intel 8080/8085*

Выводы	Описание	Тип
AD_0—AD_7	Шина адреса/данных	Двунаправленная, три состояния
A_8—A_{15}	Шина адреса	Выход, три состояния
ALE	Разрешение захвата адреса	Выход[1]
$\overline{RD}$	Управление считыванием	Выход, три состояния
$\overline{WR}$	Управленне записью	Выход, три состояния
IO/$\overline{M}$	Указатель ВВ илн памяти	Выход, три состояния
S_0, S_1	Указатель состояния шины	Выход
READY	Вызов состояния ожидания	Вход
SID	Ввод последовательных данных	Вход
SOD	Вывод последовательных данных	Выход
HOLD	Требованне захвата	Вход
HLDA	Подтверждение состояиия захвата	Выход
INTR	Запрос прерывания	Вход
TRAP	Запрос немаскированного прерываиия	Вход
RST 5.5 *RST 6.5* *RST 7.5*	Запрос аппаратного векторного прерывания	Вход
$\overline{INTA}$	Подтверждение запроса на прерывание	Выход
$\overline{RESET\ IN}$	Сброс системы	Выход
RESET OUT	Сброс периферин	Выход
X_1, X_2	Соединения кристалла илн внешнего ГТИ	Вход
CLK	Сигнал внутреннего ГТИ	Выход
V_{CC}, V_{SS}	Питание, земля	

[1] Этот выход имеет три состояния в 8085, но не в 8080.

ляемая разработчиками документация уточняет, идет ли речь об Intel 8085 или 8085A (мало измененная версия 8085).

Наш типовой МП имел 16 выводов адресных линий и восемь для подсоединения шины данных. Располагая дополнительными возможностями Intel 8085 в DIP-корпусе с 40 выводами не требует дополнительных выводов для обеспечения всех входов и выходов; по этой причине выводы *12—19* использованы как равноценные линии шины адреса/данных (AD_0—AD_7). Поэтому этот микропроцессор называется устройством с мультиплексированной шиной данных/адреса. Адресные линии восьми младших разрядов разделяют выводы с линиями шины данных. Мультиплексировать — значит выбирать линии поочередно.

При таких определениях мультиплексировать шину адреса/данных означает использовать сначала шину для передачи адреса, затем использовать ее же для выдачи или получения данных. Микропроцессор Intel 8085 снабжен специальным сигналом для того, чтобы информировать периферийные устройства, производит ли мультиплексированная шина операции на адресной шине или на шине данных. Это специальный сигнал, называемый сигналом разрешения захвата адреса (*ALE*). Заметим, что выводы мультиплексированной шины двунаправлены или могут быть в положении трех состояний. Вывод управления *ALE* является выходным.

Микропроцессор Intel 8085 (как и типовой процессор) имеет 16 адресных линий. Восемь старших разрядов выведены на выводы A_8—A_{15} (см. рис. 8.2). Как и в случае типового МП, подсоединение к шинам прямое. Эти выводы являются выходами или могут быть в состоянии высокого сопротивления (в третьем состоянии). Другие выводы, идентичные выводам типового МП, являются выводами питания V_{CC} и V_{SS}, подсоединенными к источнику +5 В. Микропроцессор Intel 8085 снабжен внутренним генератором тактовых импульсов, входы которого X_1 и X_2 обычно соединены с кристаллом. Внутренняя частота МП является половиной частоты кристалла.

Многие выводы МП Intel 8085, показанные на рис. 8.2, выполняют функции управления. Аналогичные рассмотренным для типового МП $\overline{RD}$ и $\overline{WR}$ используются для информации устройства памяти или УВВ, т. е. определяют, наступило ли время послать или принять данные по шине данных (в этом случае — по мультиплексированной шине). Вход сброса ($\overline{RESET\ IN}$) действует так же, как это было в типовом МП при сбросе в 0000H счетчика команд. Шины адреса, данных и линии управления находятся в состоянии высокого сопротивления в ходе сброса. Когда МП сбрасывается, вывод *RESET OUT* (относится к операции сброса) выдает сигнал в периферийные устройства, информируя их, что операция сброса закончена.

Выход генератора тактовых импульсов *CLK* МП Intel 8085 функционирует, как и в типовом МП. Вход запроса

прерывания INTR в МП Intel 8085 является универсальным прерыванием (как в типовом МП), однако существует различие в том смысле, что прерывание INTR в МП Intel 8085 может быть разрешено или запрещено командами программы. Кроме входа нормального запроса на прерывание *(INTR)* МП Intel 8085 снабжен четырьмя другими входами прерывания: *TRAP, RST7.5, RST6.5, RST5.5.* Вход *TRAP* является входом прерывания наивысшего приоритета; следующими по порядку являются *RST7.5, RST6.5, RST5.5* и, наконец, *INTR* — самый низкий приоритет. Сигнал *TRAP* или один из трех сигналов *(RST7.5, RST6.5, RST5.5)* влечет за собой ветвление МП по вызываемому специальному адресу. Команды рестартов *RST* могут быть разрешены или запрещены программно, но прерывания по входу *TRAP* таким образом запрещены быть не могут. Запрос на прерывание *INTR* вызывает переход к новому адресу, указанному специальной командой, выданной периферией, когда активизируется выход, подтверждающий получение запроса на прерывание ($\overline{INTA}$).

Микропроцессор Intel 8085 снабжен слаборазвитыми вводом и выводом последовательных данных — *SID* (ввод последовательных данных) и *SOD* (вывод последовательных данных) (см. рис. 8.2). Отдельный бит данных на выводах *SID* загружается в наиболее значимый разряд (бит 7) аккумулятора командой RIM в МП Intel 8085. Вывод выхода *SOD* активизируется или сбрасывается командой SIM в МП.

Рассмотрим вход *READY* на рис. 8.2. Этот вход информирует МП, что периферия готова выдать или принять данные. Если *READY* имеет L-уровень в цикле считывания или записи, МП его интерпретирует как требование перейти в состояние ожидания. В этих условиях МП будет ждать до тех пор, пока периферия не просигнализирует, что она готова передать или получить данные. Затем будет продолжаться выполнение цикла записи или считывания. Вход *READY* удобен при использовании очень медленных по сравнению со скоростью обработки данных в МП устройств памяти или периферии.

Рассмотрим теперь вход *HOLD* (входной сигнал требования захвата) и выход *HLDA* (подтверждение состояния захвата), выводы которых показаны на рис. 8.2. Вход *HOLD* оповещает МП, что другое устройство хочет использовать шины адреса и данных (это может производиться в ходе ПДП). По получении сигнала *HOLD* МП заверша-

ет текущую операцию, затем выводы данных и адреса $\overline{RD}$, $\overline{WR}$ и $IO/\overline{W}$ переводятся в третье состояние, т. е. исключается взаимодействие с передачами данных на шинах. Выход *HLDA* указывает периферии, что запрос *HOLD* был получен и микропроцессор не будет управлять шинами в следующем цикле тактовых импульсов.

Выходы $IO/\overline{M}$, S_0 и S_1 являются сигналами управления, которые информируют периферию о типе машинного цикла, выполняемого МП. Эти типы записаны в табл. 8.2, соответствующие сочетания выходных сигналов на выводах $IO/\overline{M}$, S_0 и S_1 подробно показаны в левых колонках.

Таблица 8.2. *Машинные циклы МП Intel 8085*

Сигналы управления МП 8085			Состояние машинного цикла
$IO/\overline{M}$	S_1	S_0	
0	0	1	Запись в память
0	1	0	Считывание из памяти
1	0	1	Запись в УВВ
1	1	0	Считывание из УВВ
0	1	1	Извлечение КОП
1	1	1	Подтверждение запроса на прерывание
*	0	0	Останов
*	*/	*/	Ожидание
*	*	*	Сброс

Примечание. * — третье состояние (высокое сопротивление); */ — не оговаривается.

Упражнения

8.1. Intel 8085 представляет собой микропроцессор на _____ бит, потому что таков размер слов, которыми он манипулирует.

8.2. Восемь линий _____ (старших, младших) разрядов используют адресную шину, тогда как восемь линий _____ (старших, младших) разрядов используют шину _____.

8.3. Шина адреса/данных называется _____ (декодированной мультиплексированной), потому что она функционирует попеременно как шина адреса и как шина данных.

8.4. См. рис. 8.2. Выход _____ служит для оповещения

периферии, что мультиплексированные шины функционируют как адресные.

8.5. См. рис. 8.2. Intel 8085 питается напряжением _____.

8.6. См. рис. 8.2. Если входы IO/M и $\overline{WR}$ оба в L-состоянии, МП занят выполнением _____ (считывания, записи) в _____ (порт УВВ, память).

8.7. См. рис. 8.2. Если МП находится в цикле записи (вход *READY* в L-состоянии), то он _____ (завершает сразу, переходит в состояние ожидания до тех пор, пока периферия станет готова, и затем завершает) цикл считывания.

8.8. См. табл. 8.2. В ходе машинного цикла извлечения КОП сигналы управления S_0= _____ (0,1), S_1= _____ (0,1) и $IO/\overline{M}$= _____ (0,1).

8.9. См. рис. 8.2. Перечислить выводы входов прерывания МП Intel 8085.

8.10. См. рис. 8.2. Выводы ВВ последовательных данных обозначены _____.

Решения

8.1. 8 бит. 8.2. Старшнх; младших; мультиплекснрованную (данных/адреса). 8.3. Мультиплексированной. 8.4. *ALE*. 8.5. +5 В. 8.6. Выходы $IO/\overline{M}$ н $\overline{WR}$ имеют L-уровень, МП записывает в память. Черта сверху букв означает, что L-сигнал на выходе $IO/\overline{M}$ активизирует память *M*, а на выходе $\overline{WR}$ — запись. 8.7. Переходит в состояние ожидания и т. д. 8.8. S_0=1, S_1=1, $IO/\overline{M}$=0. 8.9. *INTR, TRAP, RST7.5, RST6.5, RST5.5*. Но можно рассматривать также сигнал *RESET IN* как прерывание, так как он является причиной, вызванной внешним устройством, и ветвит счетчик команд на новый адрес программы. 8.10. *SID* и *SOD*.

8.2. АРХИТЕКТУРА МП INTEL 8085

На рис. 8.3 показана архитектура МП Intel 8085. Он имеет 16-разрядный счетчик команд и защелку адреса, которая загружает специализированную адресную (A_{15}—A_{18}) и мультиплексированную шины (AD_7—AD_0). Параллельные данные входят в МП и покидают его через AD_7—AD_0. Эта шина передает адрес, когда линия управления *ALE* получает Н-сигнал, и данные, когда L-сигнал.

По 8-разрядной внутренней шине входящие и выходящие данные вводятся внутрь устройства. Они могут посту-

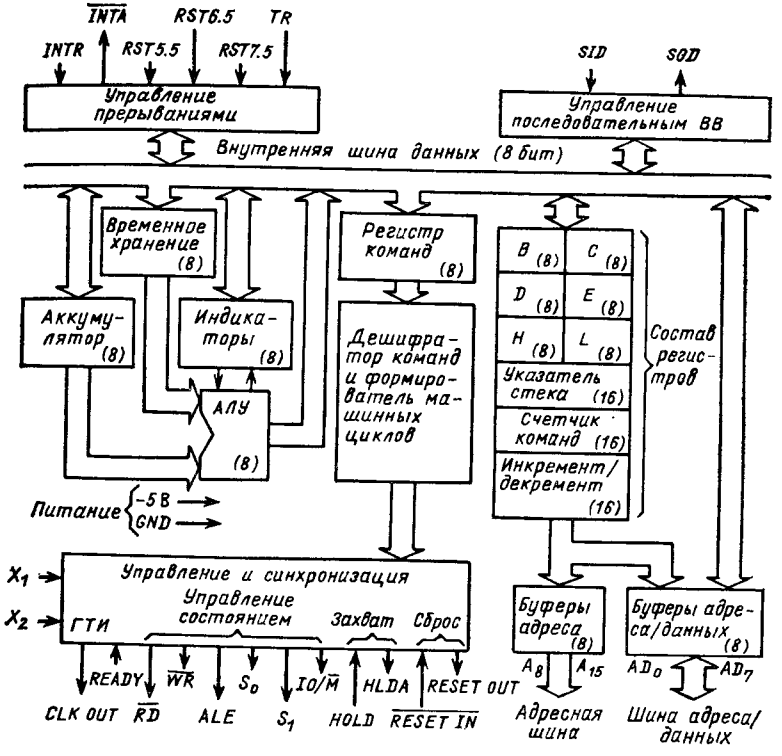


Рис. 8.3. Функциональная схема МП Intel 8085 (архитектура)

пать с внутренней шины данных в 8-разрядный аккумулятор или регистр временного хранения, в индикаторы, регистр команд, устройство управления, в какой-либо из регистров общего назначения (*B, C, D, E, H, L*), 16-разрядный указатель стека, 16-разрядный счетчик команд или 8-разрядный буфер адреса/данных. Выводы *SID* и *SOD* ввода и вывода последовательных данных приведены справа вверху на рис. 8.3, входы прерывания (*INTR, RST5.5, RST6.5, RST7.5* и *TRAP*) — вверху слева вместе с выходом $\overline{INTA}$ (подтверждение запроса на прерывание). Арифметико-логическое устройство загружается двумя 8-разрядными регистрами (аккумулятором и регистром временного хранения), как в типовом МП. Регистр состояний содержит пять индикаторов состояния вместо двух, как это было в типовом МП.

Регистр команд связан с дешифратором. Последний определяет текущую команду, требуемую микропрограмму или следующий машинный цикл. Он информирует затем схему управления и синхронизации о последовательности действий. Эта схема координирует действия МП и периферии. Если линии управления не показаны на рис. 8.3, выходы управления и состояний здесь приведены. Входы *RESET*, *HOLD* и *READY* представлены также в части, относящейся к управлению и синхронизации МП.

Регистры

Как и в случае типового МП в состав МП Intel 8085 входят 8- и 16-разрядные регистры. Адресуемых 8-разрядных регистров здесь восемь, шесть из которых (регистры общего назначения) могут быть использованы или как 8-разрядные, или могут объединяться в три 16-разрядные пары. Кроме того, МП Intel 8085 содержит два 16-разрядных регистра.

1. Аккумулятор (или регистр *A*) является ядром всех операций МП, к которым относятся арифметические, логические, загрузки или размещения данных и ВВ. Это 8-разрядный регистр.

2. Регистры общего назначения *BC*, *DE* и *HL* могут быть использованы как шесть 8-разрядных или три 16-разрядные пары регистров в зависимости от текущей выполняемой команды. Как и в типовом МП, пара *HL* (фирмой Intel названа указателем данных) может быть использована для указания адреса. Несколько команд используют пары *BC* и *DE* в качестве указателя адреса, но обычно они являются регистрами хранения данных.

3. Счетчик команд *PC* всегда указывает на ячейку памяти следующей для выполнения команды.

4. Указатель стека *SP* является специальным регистром — указателем адреса (или данных), который всегда указывает на вершину стека в ОЗУ. Это 16-разрядный регистр.

5. Регистр состояния (или индикаторов) содержит пять одноразрядных индикаторов, в которых содержится информация, относящаяся к состоянию МП. Эти указатели используются условными ветвлениями программы, вызовами подпрограмм и возвратами из подпрограмм.

Индикаторы

На рис. 8.4 представлены пять индикаторов МП 8085. Индикатор переноса *CY* устанавливается или сбрасывается в результате выполнения арифметических операций. Его состояние проверяется командами программы. Как и в типовом МП, переполнение 8 бит при сложении устанавливает 1 в *CY*; в случае вычитания, когда *CY* установлен, это указывает, что вычитаемое больше уменьшаемого.

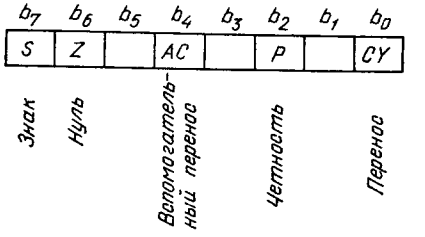


Рис. 8.4. Регистр состояний МП Intel 8085

Индикатор нуля *Z* устанавливается, когда результатом некоторых операций является 0, в противном случае он сбрасывается. Мы изучали его функционирование в типовом МП.

Индикатор знака *S* устанавливается в зависимости от состояния наиболее значимого бита после выполнения арифметических или логических команд. Эти команды используют самый старший бит данных для того, чтобы представить знак числа, содержащегося в аккумуляторе. Установленный индикатор соответствует отрицательной величине, сброшенный — положительной.

Индикатор вспомогательного переноса *AC* показывает переполнение или перенос в третьем разряде аккумулятора таким же образом, как индикатор переноса показывает переполнение или перенос в седьмом разряде. Этот индикатор используется в ходе выполнения операций двоично-десятичной арифметики.

Индикатор четности *P* проверяет число бит единиц в аккумуляторе. Если это число четное, он показывает, что паритет четный, и тогда в индикаторе паритета устанавливается 1; если число нечетное, паритет нечетный, и индикатор сбрасывается в 0. Например, если команда ADD дает результат в аккумуляторе $0011\ 0011_2$, в индикаторе паритета будет установлена 1, так как число единиц (4) четно. Если в аккумуляторе — $1010\ 1110_2$, индикатор *P* будет сброшен в 0, потому что число бит единиц (5) нечетно.

Так как все эти индикаторы содержат только единственный бит, их иногда называют не индикаторами, а битами; мы часто будем встречать выражения: бит переноса, нуля, вспомогательного переноса, паритета (в регистре состояния).

Указатель стека

Указатель стека *SP* указывает на адрес последнего помещенного в стек байта. Он может использовать в качестве стека любую часть ОЗУ. Как и в типовом МП, указатель стека декрементируется при каждом помещении в стек и инкрементируется при каждом извлечении из него.

Арифметико-логическое устройство

Арифметико-логическое устройство (АЛУ) тесно связано с аккумулятором, регистром состояний и несколькими регистрами временного хранения, недоступными программисту. Арифметические, логические операции и операции передачи данных выполняются в АЛУ, а результаты обычно помещаются в аккумулятор.

Регистр команд и дешифратор

В ходе извлечения команды ее первый байт (КОП) передается в 8-разрядный регистр команд. Содержимое регистра доступно тогда дешифратору команд. Выход этого дешифратора, переданный портами по сигналам синхронизации, управляет регистрами, АЛУ, буферами адреса и данных.

Внутренний генератор тактовых импульсов

Кристалл МП Intel 8085 содержит законченный генератор тактовых импульсов. Он требует только присоединения кварцевого кристалла для синхронизации своей работы. Микропроцессор Intel 8085 использует кварц с частотой 6,25 МГц, его версия — Intel 8085А-2 работает с кристаллом на частоте до 10 МГц. Вывод *CLK* является буферным выходом генератора тактовых импульсов, частота которого равна половине частоты кварца.

Прерывания

Пять входов аппаратного прерывания записаны в порядке приоритета в левой колонке табл. 8.3. Входом наивысшего приоритета является *TRAP*. Сигнал Н-уровня на

Таблица 83. *Прерывания в МП Intel 8085*

Выводы	Приоритет	Адрес[1] ветвления в момент прерывания	Тип прерывания
TRAP	1	24Н	Положительный фронт и высокий уровень до выбора
RST 7.5	2	3СН	Положительный фронт (захват)
RST6 5	3	34Н	Высокий уровень до выбора
RST5 5	4	2СН	
INTR	5	(*)[2]	

[1] В случае *TRAP* и *RST 5 5—7 5* содержимое счетчика команд помещается в стек перед выполнением ветвления

[2] Согласно команде, выданной в МП Intel 8085А устройством 8259 или другой схемой, когда прерывание признано

входе *TRAP* заставляет МП сохранить содержимое счетчика команд в стеке и ответвиться на ячейку памяти 0024Н. Вход *TRAP* не может быть непризнанным и составляет, следовательно, немаскируемое прерывание.

Три следующих аппаратных прерывания в табл. 8.3 являются так называемыми повторными запусками — рестартами (новый запуск программы по новому адресу в памяти). Прерывание *RST7.5* заставляет МП поместить в стек содержимое счетчика команд и ответвиться на ячейку памяти 003СН. Прерывание *RST6.5* более низкого приоритета, чем предыдущее, заставляет МП поместить в стек содержимое счетчика команд и ответвиться в память по адресу 0034Н. Прерывание *RST5.5* еще более низкого уровня заставляет МП поместить в стек содержимое счетчика команд и ответвиться в память по адресу 002СН.

Прерывание самого низкого приоритета выполняется по входу *INTR*, что заставляет процессор извлечь команду из специального внешнего источника.

Все четыре последних прерывания могут быть разрешены или не разрешены программно и составляют поэтому маскируемые прерывания.

Ввод и вывод последовательных данных

Выводы, предназначенные для *ввода и вывода последовательных* данных в МП Intel 8085, способствуют минимизации числа кристаллов в малой системе, составляя интерфейс последовательного порта. По специальной команде

RIM данные передаются с вывода последовательного входа *SID* в бит 7 (b_7) аккумулятора (см. рис. 8.5, *а*, где в качестве примера Н-сигнал передается по линии *SID* в наиболее значимый бит аккумулятора).

Отдельный последовательный бит может быть выведен через выход *SOD*, используя специальную команду SIM

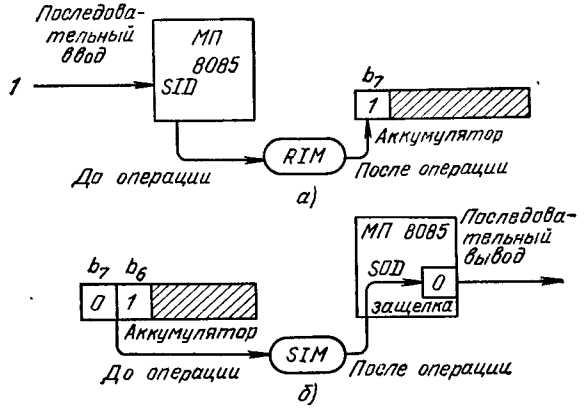


Рис. 8.5. Схемы выполнения команд:
а — последовательного ввода RIM; *б* — последовательного вывода SIM

(см. рис. 8.5, *б*, где в качестве примера L-сигнал выводится по линии *SOD* через защелку последовательного выхода). Заметим на рис. 8.5, что источником данных является наиболее значимый бит 7 (b_7) аккумулятора. Бит 6 (b_6) аккумулятора должен быть установлен в 1, чтобы мог осуществляться последовательный вывод данных.

Рис. 8.6. Доступные для программиста регистры МП Intel 8085

Последовательный вход *SID* может быть использован так же, как универсальный вход TEST, тогда как вывод выхода *SOD* может служить выходом однобитовой команды. В следующем параграфе мы подробно остановимся на многоцелевых командах SIM и RIM.

Мнемоника RIM означает *считывать маску прерывания* (*Read Interrupt Mask*), SIM — *установить маску прерывания* (*Set Interrupt Mask*).

На рис. 8.6 представлены программируемые регистры МП Intel 8085. Эти регистры являются для программиста основными, так как они доступны, а этот тип схемы составляет модель программирования МП Intel 8085.

Первичный 8-разрядный аккумулятор обозначен *A*, другие 8-разрядные регистры общего назначения (*B*, *C*, *D*, *E*, *H* и *L*) все вместе называются *вторичными аккумуляторами/счетчиками данных*. На рис. 8.6 приведены также 16-разрядные указатель стека *SP* и счетчик команд *PC*.

Упражнения

8.11. См. рис. 8.3. Intel 8085 снабжен счетчиком команд емкостью _____ бит, который связан с адресной защелкой. От нее адрес разделен на два и передается в периферию по шине _____ и _____ шине.

8.12. См. рис. 8.3. Выходная линия управления *ALE* имеет _____ (H-, L-) уровень, когда МП передает адрес на мультиплексированную шину.

8.13. См. рис. 8.3. Выводы *SID* и *SOD* жестко связаны с секцией управления _____ (прерываниями, последовательным ВВ).

8.14. Перечислить шесть регистров общего назначения емкостью 8 бит каждый, кроме аккумулятора.

8.15. Какой 16-разрядный регистр всегда содержит адреса и указывает на вершину стека в ОЗУ?

8.16. Перечислить пять индикаторов регистра состояния МП Intel 8085.

8.17. См. рис. 8.7. Каково содержимое аккумулятора после выполнения операции сложения?

8.18. См. рис. 8.7. После сложения бит знака (S) будет равен _____ (0, 1).

8.19. См. рис. 8.7. После сложения бит нуля (*Z*) будет равен _____ (0, 1).

8.20. См. рис. 8.7. После сложения бит вспомогательного переноса (*AC*) будет равен _____ (0, 1).

8.21. См. рис. 8.7. Каким будет бит четности (*P*) после сложения?

8.22. См. рис. 8.7. Каким будет состояние бита переноса (*CY*) после сложения?

8.23. Аппаратное прерывание высшего немаскированного приоритета выполняется по линии _____ (*INTR*, *TRAP*).

8.24. См. рис. 8.4. Когда активизировано аппаратное прерывание *RST7.5*, МП сохранит содержимое _____ в стеке и ответвится по адресу _____ в памяти.

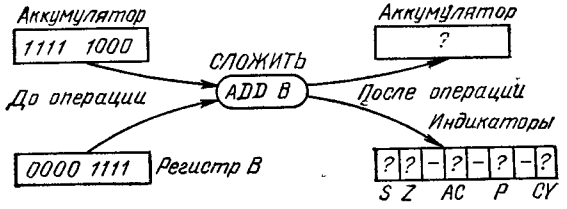


Рис. 8.7. К упражнениям 8.17—8.22

8.25. Когда шина данных передает 8 бит (параллельно), вывод *SID* управляет сразу _____ бит.

Решения

8.11. 16; адреса; мультиплексированной. **8.12.** Когда на *ALE* L-уровень, мультиплексированная шина становится двунаправленной шиной данных. **8.13.** Последовательным ВВ. **8.14.** *B*, *C*, *D*, *E*, *H*, *L*. Они могут для некоторых операций использоваться парами регистров *BC*, *DE*, *HL*. **8.15.** Указатель стека (*SP*). **8.16.** Индикаторы или биты: знака, нуля, переноса, вспомогательного переноса, четности. **8.17.** 1111 1000+ +0000 1111=1 0000 0111_2. Таким образом (A)=0000 0111 (8 младших бит суммы). **8.18.** (A)=0000 0111, следовательно, (S)=0. **8.19.** (A)= =0000 0111≠0, следовательно, (Z)=0. **8.20.** Позиция b_3 в аккумуляторе, как и в регистре *B*, содержит 1 в обоих случаях ($1_2+1_2=10_2$), таким образом, будет иметь место перенос из бита b_3 в бит b_4 (*AC*)= =1. **8.21.** Аккумулятор содержит 0000 0111, следовательно, имеется нечетный паритет, (*P*)=0. **8.22.** 1111 1000+0000 1111=1 0000 0111, имеет место переполнение в b_7, таким образом, (*CY*)=1. **8.23.** *TRAP*. **8.24.** Счетчика команд (*PC*); 3CH. **8.25.** Одиим.

8.3. СПОСОБЫ АДРЕСАЦИИ

Как и типовой МП, Intel 8085 использует пять способов адресации, а именно: неявную, регистровую, непосредственную, прямую, косвенную регистровую.

Неявная адресация. Команда STC (восстановить индикатор переноса) относится только к индикатору переноса и никакому другому регистру или памяти.

Регистровая адресация. Когда используются команды с этим способом адресации, операция и источник операнда точно определены (см. рис. 8.8, где показано выполнение

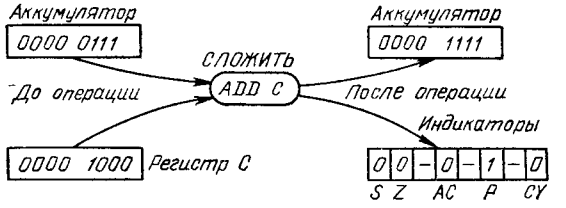


Рис. 8.8. Команда ADD *C*

команды ADD *C*). В этом примере операнд в регистре источника (регистр *C*) складывается с операндом, содержащимся в аккумуляторе. После того как команда ADD *C* выполнена, сумма ($0000\ 1111_2$) помещается в аккумулятор. Обычно результат влияет на индикаторы регистра состояния.

Два операнда являются содержимым внутренних регистров МП (*A* и *C*). Команды регистровой адресации очень эффективны в том смысле, что используют только один байт памяти программы. Они быстро выполнимы, так как не используют операцию извлечения данных из памяти.

Непосредственная адресация. Команды непосредственной адресации являются командами, по которым данные следуют непосредственно за КОП. В качестве примера на рис. 8.9 показана команда ADI (непосредственное СЛОЖЕНИЕ). Микропроцессор извлекает КОП (C6H в этом случае) из памяти программы. После декодирования МП определяет, что речь идет о команде с непосредственной адресацией. Он находит данные в ячейке памяти, следующей непосредственно за ячейкой, содержащей КОП. Эти следующие непосредственно за КОП данные ($0000\ 1000_2$) складываются с содержимым аккумулятора ($0000\ 1100_2$), где и помещается сумма ($0001\ 0100_2$).

Все команды непосредственной адресации МП Intel 8085, за исключением двух, используют аккумулятор как неявный операнд (например, команда ADI на рис. 8.9). Две

команды являются необычными — MVI (передать непосредственно), которая может передать данные, следующие непосредственно за КОП, в любой регистр или память, и команда LXI (загрузить непосредственно пару регистров), которая загружает 16-разрядное число в пару регистров. Эти команды, следовательно, не связаны с аккумулятором.

Прямая адресация. Операции, использующие прямую адресацию, точно описываются трехбайтовым форматом команд (рис. 8.10, *а*). Первый байт содержит КОП команды прямой адресации, второй — МБ и третий — СБ адреса операнда.

На рис. 8.10, *б* представлено выполнение команды LDA (загрузить *А* прямо) микропроцессором Intel 8085. Три байта команды представлены внизу слева. Код операции LDA — 3АН, следующих 2 байта составляют 16-разрядный адрес (0200Н), и ячейка памяти данных с адресом (0200Н) загружается МП. Содержимое ($1111\ 1111_2$) этого адреса загружается в аккумулятор.

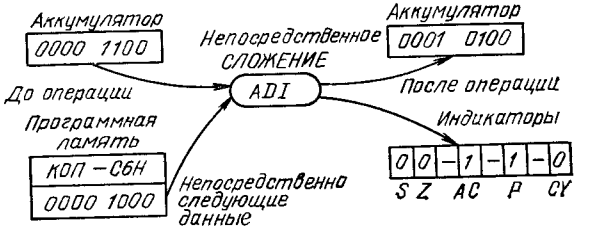


Рис. 8.9. Команда ADI

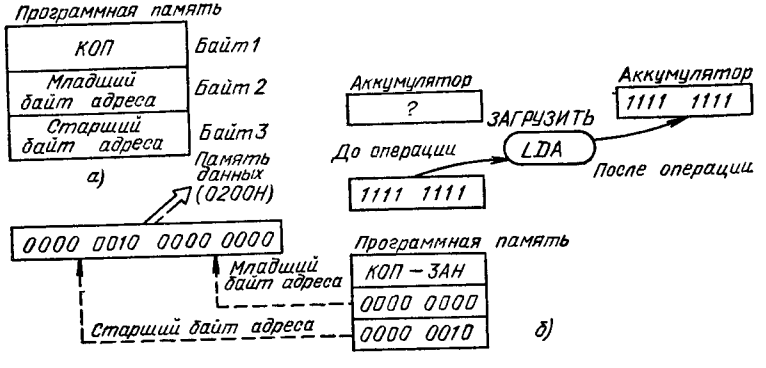


Рис. 8.10. Команда прямой адресацнн:
а — формат команды; *б* — команда загрузки аккумулятора

Косвенная регистровая адресация. Команды с такой адресацией обращаются в память, используя содержимое пары регистров для указания на адрес операнда. На рис. 8.11 приведен пример этого типа команд. Команда ADD *M* (сложить память) складывает содержимое ячейки памяти

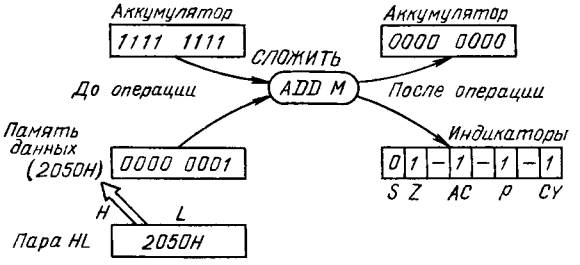


Рис. 8.11. Команда косвенной адресации сложения содержимого регистра

по адресу, на который указывает пара регистров *HL* МП, с содержимым аккумулятора. В этом случае *HL* указывает на ячейку памяти 2050Н, операнд ($0000\ 0001_2$) в этой ячейке памяти складывается с содержимым аккумулятора ($1111\ 1111_2$), т. е. $0000\ 0001 + 1111\ 1111 = 1\ 0000\ 0000$. Восемь наименее значимых бит суммы помещаются в аккумулятор после выполнения команды ADD *M*, а соответствующие индикаторы в зависимости от результата устанавливаются или сбрасываются.

Комбинированные способы адресации. Несколько команд МП Intel 8085 используют сочетание различных способов адресации. Вызов подпрограммы CALL, например, сочетает прямую и косвенную регистровую адресации. Прямой адрес в CALL точно описывает адрес вызываемой подпрограммы, адрес косвенного регистра является адресом указателя стека. Команда CALL помещает в стек прежде всего текущее содержимое счетчика команд по точно описанному указателем стека адресу, затем процессор загружает адрес прямо в счетчик команд. Наконец МП ветвится на подпрограмму, адрес которой является теперь содержимым счетчика команд.

Упражнения

8.26. Перечислить пять способов адресации МП Intel 8085.

8.27. Команда CMC (инвертировать перенос) МП Intel 8085 воздействует только на индикатор переноса, но не на другие регистры или память. Она использует, следовательно, _____ (прямую, неявную) адресацию.

8.28. Команда MOV *B, A* (передает *A* в *B*) передает содержимое одного регистра в другой. Команда MOV *B, A* использует, таким образом, _____ (непосредственную, регистровую) адресацию.

8.29. См. рис. 8.12. У команды MVI *A* _____ (неявная, непосредственная) адресация.

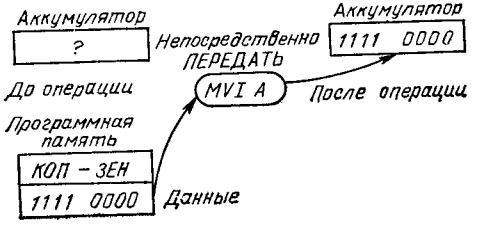


Рис. 8.12. Команда непосредственной адресации передачи данных

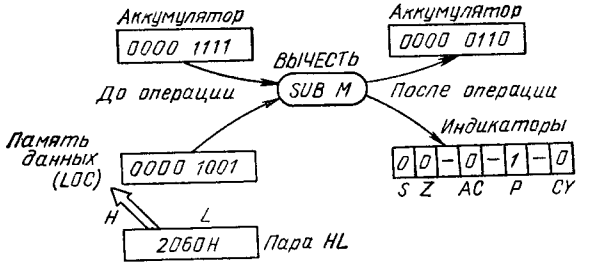


Рис. 8.13. Команда SUB *M*

8.30. См. рис. 8.13. У команды SUB *M* _____ (регистровая, косвенная регистровая) адресация.

8.31. См. рис. 8.13. Первый член находится в аккумуляторе, второй — в ячейке памяти _____.

Решения

8.26. Неявная, регистровая, непосредственная, прямая, косвенная регистровая. **8.27.** Неявную. **8.28.** Регистровую. **8.29.** Непосредственная. **8.30.** Косвенная регистровая. **8.31.** 2060H.

8.4. СОСТАВ КОМАНД МП INTEL 8080/8085

Микропроцессоры Intel 8080 и 8085 используются для создания микро-ЭВМ с загружаемыми программами. Команды программ помещаются байтами в одной или нескольких областях памяти, называемых программной памятью. Микропроцессоры Intel 8080/8085 используют одно-, двух- и трехбайтовые команды. Первым байт всегда является КОП, который уточняет характер выполняемой МП операции (их более 200). Процессор узнает КОП, когда они состоят из 8 бит двоичной системы. Состав команд, на которые МП Intel 8080/8085 реагируют, постоянно определен свойствами и структурой кристалла. Микропроцессоры Intel 8080/8085 используют одни и те же КОП. Кроме того, состав команд МП Intel 8085 содержит на две команды больше, чем МП Intel 8080.

Фирма разделяет команды МП Intel 8080/8085 по следующим группам: передача данных; арифметическая; логическая; ветвления; стека; ВВ и машинного управления.

Группа передачи предназначена для передачи данных между регистрами или ячейками памяти и регистрами. Она содержит передачи, загрузку, размещения и обмены данных.

Арифметическая группа выполняет операции сложения, вычитания, инкремента и декремента над данными в регистрах или в памяти.

Логическая группа выполняет операции И, ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ, сравнений, перемещений и инвертирования данных в регистрах или между данными в памяти и регистре.

Группа ветвления вызывает ветвления (переходы) условные или безусловные, вызовы, возвраты и повторные запуски.

Группа стека, ВВ и машинного управления понимает команды операций со стеком, считывания в портах ввода, записи в порты вывода, инициализации и считывания маскированных прерываний и установки и сброса индикаторов.

В табл. 8.4 приведен состав команд МП Intel 8080/8085. Коды операций приводятся в шестнадцатеричной записи. Здесь описаны 239 команд. Две команды отмечены как используемые только микропроцессором Intel 8085: RIM и SIM.

Команды типового МП (см. гл. 6) были частью состава команд МП Intel 8080/8085, и поэтому мы использовали

Таблица 8.4. *Состав команд микропроцессора Intel 8080/8085*

Мнемоника	КОП	Описание
ADD *A*	87	Сложить *A* с *A* (удвоение *A*)
ADD *B*	80	Сложить *B* с *A*
ADD *C*	81	Сложить *C* с *A*
ADD *D*	82	Сложить *D* с *A*
ADD *E*	83	Сложить *E* с *A*
ADD *H*	84	Сложить *H* с *A*
ADD *L*	85	Сложить *L* с *A*
ADD *M*	86	Сложить содержимое памяти LOC (HL) с *A*
ADI *v*	C6	Сложить непосредственно следующие данные *v* с *A*
ADC *A*	8F	Сложить *A* с *A* с переносом (удвоение *A*)
ADC *B*	88	Сложить *B* с *A* с переносом
ADC *C*	89	Сложить *C* с *A* с переносом
ADC *D*	8A	Сложить *D* с *A* с переносом
ADC *E*	8B	Сложить *E* с *A* с переносом
ADC *H*	8C	Сложить *H* с *A* с переносом
ADC *L*	8D	Сложить *L* с *A* с переносом
ADC *M*	8E	Сложить память LOC (HL) с *A* с переносом
ACI *v*	CE	Сложить непосредственно следующие данные *v* с *A* с переносом
ANA *A*	A7	Тест *A* *A* И *A*
ANA *B*	AO	Логическая операция И *B* И *A*
ANA *C*	A1	Логическая операция И *C* И *A*
ANA *D*	A2	Логическая операция И *D* И *A*
ANA *E*	A3	Логическая операция И *E* И *A*
ANA *H*	A4	Логическая операция И *H* И *A*
ANA *L*	A5	Логическая операция И *L* И *A*
ANA *M*	A6	Память LOC (HL) И *A*
ANI *v*	E6	Непосредственно следующие данные *v* и *A*
CALL *aa*	CD	Вызвать подпрограмму по адресу *aa*

Мнемоника	КОП	Описание
CZ *aa*	CC	Вызвать подпрограмму по адресу *aa*, если нуль
CNZ *aa*	C4	Вызвать подпрограмму по адресу *aa*, если не нуль
CP *aa*	F4	Вызвать подпрограмму по адресу *aa*, если плюс
CM *aa*	FC	Вызвать подпрограмму по адресу *aa*, если минус
CC *aa*	DD	Вызвать подпрограмму по адресу *aa*, если перенос
CNC *aa*	D4	Вызвать подпрограмму по адресу *aa*, если не перенос
CPE *aa*	EC	Вызвать подпрограмму по адресу *aa*, если четно
CPO *aa*	E4	Вызвать подпрограмму по адресу, *aa*, если нечетно
CMA	2F	Инвертировать *A*
CMC	3F	Инвертировать перенос
CMP *A*	BF	Установить индикатор нуля операцией (A)—(A)
CMP *B*	B8	Сравнить *A* с *B*
CMP *C*	B9	Сравнить *A* с *C*
CMP *D*	BA	Сравнить *A* с D
CMP *E*	BB	Сравнить *A* с *E*
CMP *H*	BC	Сравнить *A* с *H*
CMP *L*	BD	Сравнить *A* с *L*
CMP *M*	BE	Сравнить *A* с содержимым памяти LOC (HL)
CPI *v*	FC	Сравнить *A* с непосредственно следующими данными
DAA	27	Десятичная коррекция аккумулятора
DAD *B*	09	Сложить *BC* с *HL*
DAD *D*	19	Сложить *DE* с *HL*
DAD *H*	29	Сложить *HL* с *HL* (удвоение H)
DAD *SP*	39	Сложить *SP* с *H*
DCR A	3D	Декрементировать *A*
DCR B	05	Декрементировать *B*
DCR C	0D	Декрементировать *C*
DCR D	15	Декрементировать *D*

Продолжение табл. 8.4

Мнемоника	КОП	Описание
DCR *E*	1D	Декрементировать *E*
DCR *H*	25	Декрементировать *H*
DCR *L*	2D	Декрементировать *L*
DCR *M*	35	Декрементировать содержимое памяти LOC (HL)
DCX *B*	0B	Декрементировать *BC*
DCX *D*	1B	Декрементировать *DE*
DCX *H*	2B	Декрементировать *HL*
DCX *SP*	3B	Декрементировать *SP*
DI	F3	Не признать прерывание
EI	FB	Признать прерывание
HLT	76	Остановить микропроцессор
IN *v*	DB	Ввести данные с устройства *v*
INR *A*	3C	Инкрементировать *A*
INR *B*	04	Инкрементировать *B*
INR *C*	0C	Инкрементировать *C*
INR *D*	14	Инкрементировать *D*
INR *E*	1C	Инкрементировать *E*
INR *H*	24	Инкрементировать *H*
INR *L*	2C	Инкрементировать *L*
INR *M*	34	Инкрементировать содержимое памяти LOC(HL)
INX *B*	03	Инкрементировать *BC*
INX *D*	13	Инкрементировать DE
INX *H*	23	Инкрементировать HL
INX *SP*	33	Инкрементировать SP
JMP *aa*	C3	Перейти по адресу *aa*

Продолжение табл. 8.4

Мнемоника	КОП	Описание
JZ *aa*	CA	Перейти по адресу *aa*, если нуль
JNZ *aa*	C2	Перейти по адресу *aa*, если не нуль
JP *aa*	F2	Перейти по адресу *aa*, если плюс
JM *aa*	FA	Перейти по адресу *aa*, если минус
JC *aa*	DA	Перейти по адресу *aa*, если перенос
JNC *aa*	D2	Перейти по адресу *aa*, если не перенос
JPE *aa*	EA	Перейти по адресу *aa*, если паритет четный
JPO *aa*	E2	Перейти по адресу *aa*, если паритет нечетный
LDA *aa*	3A	Загрузить *A* из источника с адресом *aa*
LDAX *B*	0A	Загрузить *A* из ячейки памяти LOC (BC)
LDAX *D*	1A	Загрузить *A* из ячейки памяти LOC (DE)
LHLD *aa*	2A	Загрузить *HL* из источника с адресом *aa*
LXI *B*, *vv*	01	Загрузить *BC* непосредственно следующими данными
LXI *H*, *vv*	21	Загрузить *HL* непосредственно следующими данными
LXI *SP*, *vv*	31	Загрузить *SP* непосредственно следующими данными
MOV *A*, *B*	78	Передать данные из *B* в *A*
MOV *A*, *C*	79	Передать данные из *C* в *A*
MOV *A*, *D*	7A	Передать данные из *D* в *A*
MOV *A*, *E*	7B	Передать данные из *E* в *A*
MOV *A*, *H*	7C	Передать данные из *H* в *A*
MOV *A*, *L*	7D	Передать данные из *L* в *A*
MOV *A*, *M*	7E	Передать данные из ячейки памяти LOC (HL) в *A*
MOV *B*, *A*	47	Передать данные из *A* в *B*
MOV *B*, *C*	41	Передать данные из *C* в *B*
MOV *B*, *D*	42	Передать данные из *D* в *B*
MOV *B*, *E*	43	Передать данные из *E* в *B*
MOV *B*, *H*	44	Передать данные из *H* в *B*
MOV *B*, *L*	45	Передать данные из *L* в *B*

Продолжение табл. 8.4

Мнемонка	КОП	Описание
MOV *B*, *M*	46	Передать данные из ячейкн памятн LOC (HL) в *B*
MOV *C*, *A*	4F	Передать данные из *A* в *C*
MOV *C*, *B*	48	Передать данные из *B* в *C*
MOV *C*, *D*	4A	Передать данные из *D* в *C*
MOV *C*, *E*	4B	Передать данные нз *E* в *C*
MOV *C*, *H*	4C	Передать данные из *H* в *C*
MOV *C*, *L*	4D	Передать данные из *L* в *C*
MOV *C*, *M*	4E	Передать данные нз ячейки памяти LOC (HL) в *C*
MOV *D*, *A*	57	Передать данные из *A* в *D*
MOV *D*, *B*	50	Передать данные из *B* в *D*
MOV *D*, *C*	51	Передать данные из *C* в *D*
MOV *D*, *E*	53	Передать данные из *E* в *D*
MOV *D*, *H*	54	Передать данные из *H* в *D*
MOV *D*, *L*	55	Передать данные нз *L* в *D*
MOV *D*, *M*	56	Передать данные нз ячейкн памяти LOC (HL) в *D*
MOV *E*, *A*	5F	Передать данные из *A* в *E*
MOV *E*, *B*	58	Передать данные из *B* в *E*
MOV *E*, *C*	59	Передать данные нз *C* в *E*
MOV *E*, *D*	5A	Передать данные из *D* в *E*
MOV *E*, *H*	5C	Передать данные из *H* в *E*
MOV *E*, *L*	5D	Передать данные нз *L* в *E*
MOV *E*, *M*	5E	Передать данные нз ячейкн памяти LOC (HL) в *E*
MOV *H*, *A*	67	Передать данные из *A* в *H*
MOV *H*, *B*	60	Передать данные из *B* в *H*
MOV *H*, *C*	61	Передать данные из *C* в *H*
MOV *H*, *D*	62	Передать данные из *D* в *H*
MOV *H*, *E*	63	Передать данные нз *E* в *H*
MOV *H*, *L*	65	Передать данные нз *L* в *H*
MOV *H*, *M*	66	Передать данные нз ячейкн памятн LOC (HL) в *H*
MOV *L*, *A*	6F	Передать данные из *A* в *L*
MOV *L*, *B*	68	Передать данные из *B* в *L*

Продолжение табл. 8.4

Мнемоника	КОП	Описание
MOV *L*, *C*	69	Передать данные из *C* в *L*
MOV *L*, *D*	6A	Передать данные нз *D* в *L*
MOV *L*, *E*	6B	Передать данные нз *E* в *L*
MOV *L*, *H*	6C	Передать данные из *H* в *L*
MOV *L*, *M*	6E	Передать данные из ячейки памятн LOC (HL) в *L*
MOV *M*, *A*	77	Передать данные в ячейку памяти LOC (HL) из *A*
MOV *M*, *B*	70	Передать данные в ячейку памяти LOC (HL) из *B*
MOV *M*, *C*	71	Передать данные в ячейку памяти LOC (HL) из *C*
MOV *M*, *D*	72	Передать данные в ячейку памяти LOC (HL) нз *D*
MOV *M*, *E*	73	Передать данные в ячейку памятн LOC (HL) из *E*
MOV *M*, *H*	74	Передать данные в ячейку памятн LOC (HL) нз *H*
MOV *M*, *L*	75	Передать данные в ячейку памяти LOC (HL) из *L*
MVI *A*, *v*	3E	Передать непосредственно следующие данные *v* в *A*
MVI *B*, *v*	06	Передать непосредственно следующне данные *v* в *B*
MVI *C*, *v*	0E	Передать непосредственно следующие данные *v* в *C*
MVI *D*, *v*	16	Передать непосредственно следующие данные *v* в *D*
MVI *E*, *v*	1E	Передать непосредственно следующие данные *v* в *E*
MVI *H*, *v*	26	Передать непосредственно следующие данные *v* в *H*
MVI *L*, *v*	2E	Передать непосредственно следующие данные *v* в *L*
MVI *M*, *v*	36	Передать непосредственно следующие данные *v* в LOC (HL)
NOP	00	Нет операций
ORA *A*	B7	Проверить *A* и сбросить перенос
ORA *B*	B0	Логическая операция *B* ИЛИ *A*
ORA *C*	B1	Логическая операция *C* ИЛИ *A*

Продолжение табл. 8.4

Мнемоника	КОП	Описание
ORA *D*	B2	Логическая операция *D* ИЛИ *A*
ORA *E*	B3	Логическая операция *E* ИЛИ *A*
ORA *H*	B4	Логическая операция *H* ИЛИ *A*
ORA *L*	B5	Логическая операция *L* ИЛИ *A*
ORA *M*	B6	Содержимое ячейки памяти LOC (HL) ИЛИ *A*
ORI *v*	F6	Непосредственно следующие данные *v* или *A*
OUT *v*	D3	Вывести содержимое *A* по адресу *v*
PCHL	E9	Передать содержимое *H* и *L* в *PC* (счетчик команд)
POP *B*	C1	Извлечь из стека содержимое пары регистров *BC*
POP *D*	D1	Извлечь из стека содержимое пары регистров *DE*
POP *H*	E1	Извлечь из стека содержимое пары регистров *HL*
POP *PSW*	F1	Извлечь из стека слово состояния процессора *PSW*
PUSH *B*	C5	Загрузить в стек содержимое пары регистров *BC*
PUSH *D*	D5	Загрузить в стек содержимое пары регистров *DE*
PUSH *H*	E5	Загрузить в стек содержимое пары регистров *HL*
PUSH *PSW*	F5	Загрузить в стек слово состояния процессора *PSW*
RAL	17	Переместить циклически *CY*+*A* влево
RAR	1F	Переместить циклически *CY*+*A* вправо
RLC	07	Переместить *A* влево с переносом
RRC	0F	Переместить *A* вправо с переносом
RIM	20	Читать маску прерывания (только Intel 8085)
RET	C9	Возврат из подпрограммы

Продолжение табл. 8.4

Мнемоника	КОП	Описание
RZ	C8	Возврат из подпрограммы, если нуль
RNZ	C0	Возврат из подпрограммы, если не нуль
RP	F0	Возврат из подпрограммы, если плюс
RM	F8	Возврат из подпрограммы, если минус
RC	D8	Возврат из подпрограммы, если перенос
RNC	D0	Возврат из подпрограммы, если нет переноса
RPE	E8	Возврат из подпрограммы, если четный паритет
RPO	E0	Возврат из подпрограммы, если нечетный паритет
RST 0	C7	Повторный запуск программы с адреса 00H
RST 1	CF	Повторный запуск программы с адреса 08H
RST 2	D7	Повторный запуск программы с адреса 10H
RST 3	DF	Повторный запуск программы с адреса 18H
RST 4	E7	Повторный запуск программы с адреса 20H
RST 5	EF	Повторный запуск программы с адреса 28H
RST 6	F7	Повторный запуск программы с адреса 30H
RST 7	FF	Повторный запуск программы с адреса 38H
SIM	30	Установить маску прерывания (только для 8085)
SPHL	F9	Загрузить *SP* из *HL*
SHLD *aa*	22	Поместить *HL* в память по адресу *aa*
STA *aa*	32	Поместить *A* в память LOC по адресу *aa*
STAX *B*	02	Поместить *A* в память LOC (BC)
STAX *D*	12	Поместить *A* в память LOC (DE)
STC	37	Установить индикатор переноса
SUB *A*	97	Вычесть *A* из *A* (очистить аккумулятор)
SUB *B*	90	Вычесть *B* из *A*
SUB *C*	91	Вычесть *C* из *A*
SUB *D*	92	Вычесть *D* из *A*
SUB *E*	93	Вычесть *E* из *A*
SUB *H*	94	Вычесть *H* из *A*
SUB *L*	95	Вычесть *L* из *A*

Продолжение табл. 8.4

Мнемоника	КОП	Описание
SUB *M*	96	Вычесть содержнмое памятн LOC (HL) из *A*
SUI *v*	D6	Вычесть непосредственно следующне данные *v* из *A*
SBB *A*	9F	Вычесть *A* из *A*, очистнть аккумулятор
SBB *B*	98	Вычесть с заемом *B* нз *A*
SBB *C*	99	Вычесть с заемом *C* из *A*
SBB *D*	9A	Вычесть с заемом *D* из *A*
SBB *E*	9B	Вычесть с заемом *E* нз *A*
SBB *H*	9C	Вычесть с заемом *H* из *A*
SBB *L*	9D	Вычесть с заемом *L* из *A*
SBB *M*	9E	Вычесть с заемом содержимое памяти LOC (HL) из *A*
SBI *v*	DE	Вычесть с заемом непосредственные данные *v* из *A*
XCHG	EB	Обмен содержнмых пар регистров *DE* и *HL*
XTHL	E3	Обмен вершины стека с содержимым пары регистров *HL*
XRA *A*	AF	Логическая операция *A* ИЛИ ИСКЛЮЧАЮЩЕЕ *A* (очистка *A*)
XRA *B*	A8	Логические операции *B* ИЛИ ИСКЛЮЧАЮЩЕЕ *A*
XRA *C*	A9	Логические операцнн *C* ИЛИ ИСКЛЮЧАЮЩЕЕ *A*
XRA *D*	AA	Логнческне операции *D* ИЛИ ИСКЛЮЧАЮЩЕЕ *A*
XRA *E*	AB	Логнческне операции *E* ИЛИ ИСКЛЮЧАЮЩЕЕ *A*
XRA *H*	AC	Логнческне операцнн *H* ИЛИ ИСКЛЮЧАЮЩЕЕ *A*
XRA *L*	AD	Логнческие операции *L* ИЛИ ИСКЛЮЧАЮЩЕЕ *A*
XRA *M*	AE	Содержимое памяти LOC (HL) ИЛИ ИСКЛЮЧАЮЩЕЕ *A*
XRI *v*	EE	Непосредственные данные *v* ИЛИ ИСКЛЮЧАЮЩЕЕ *A*

те же мнемоники и КОП. В связи с тем что МП Intel 8080/8085 обладает большим числом регистров и выполняемых функций, ему свойственно большее число команд. Например, наш типовой МП имел четыре команды сложения (см. рис. 7.4), тогда как для МП Intel 8080/8085 командами сложения являются 18 первыми записанных команд. Микропроцессоры Intel 8080/8085 снабжены также четырьмя дополнительными командами двойного сложения (DAD *B*, DAD *D* и т. д.), т. е. всего 22 команды сложения. Однако опыт, приобретенный при работе с составом команд типового МП, будет полезен для освоения МП Intel 8080/8085.

Упражнения

Следующие задачи связаны с табл. 8.4, обращаться к которой читателю будет всегда интересно.

8.32. Сколько логических команд имеют МП Intel 8080/8085?

8.33. Типовой МП имеет только один вызов подпрограммы, тогда как МП Intel 8080/8085 могут выполнять _____ различных операций вызова.

8.34. Состав команд МП Intel 8080/8085 содержит_____ операций вычитания.

8.35. См. табл. 8.4. Некоторые мнемоники МП Intel 8080/8085 содержат X, например в командах DCX *B*, INX *B*, LXI *B*, STAX *B*. Каждый раз, когда появляется X в этих командах, в операции участвует_____ (пара регистров, один регистр).

Решения

8.32. 9, начиная с ANA *A*. **8.33.** 9. **8.34.** 18, начиная с SUB *A* в табл. 8.4. Сюда не включены команды сравнения, которые соответствуют внутренним вычитанням. **8.35.** Пара регистров.

8.5. КОМАНДЫ ПЕРЕДАЧИ ДАННЫХ МП INTEL 8080/8085

Такие команды обеспечивают выполнение операций размещения обмена, загрузки и перемещения данных. Здесь в собственном формате фирмы Intel приводятся только мнемоники и шестнадцатеричные КОП 84 команд. Среди всех команд микропроцессора эти используются наиболее часто.

Команды передачи данных

MOV A, A	7FH	MOV E, A	5FH	MOV H, A	67H	MOV L, A	6H
MOV A, B	78H	MOV E, B	58H	MOV H, B	60H	MOV L, B	68H
MOV A, C	79H	MOV E, C	59H	MOV H, C	61H	MOV L, C	69H
MOV A, D	7AH	MOV E, D	5AH	MOV H, D	62H	MOV L, D	6AH
MOV A, E	7BH	MOV E, E	59H	MOV H, E	63H	MOV L, E	6BH
MOV A, H	7CH	MOV E, H	5CH	MOV H, H	64H	MOV L, H	6CH
MOV A, L	7DH	MOV E, L	5DH	MOV H, L	65H	MOV L, L	6H
MOV A, M	7EH	MOV E, M	5EH	MOV H, M	66H	MOV L, M	6EH

MOV B, A	47H	MOV C, A	4FH	MOV D, A	57H	MOV M, A	77H
MOV B, B	40H	MOV C, B	48H	MOV D, B	50H	MOV M, B	70H
MOV B, C	41H	MOV C, C	49H	MOV D, C	51H	MOV M, C	71H
MOV B, D	42H	MOV C, D	4AH	MOV D, D	52H	MOV M, D	72H
MOV B, E	43H	MOV C, E	4BH	MOV D, E	53H	MOV M, E	73H
MOV B, H	44H	MOV C, H	4CH	MOV D, H	54H	MOV M, H	74H
MOV B, L	45H	MOV C, L	4DH	MOV D, L	55H	MOV M, L	75H
MOV B, M	46H	MOV C, M	4EH	MOV D, M	56H		

Непосредственная передача

MVI A,	байт 3EH	MVI E,	байт 1EH
MVI B,	байт 06H	MVI H,	байт 26H
MVI C,	байт 06H	MVI L,	байт 2EH
MVI D,	байт 16H	MVI M,	байт 36H

Непосредственная загрузка

LXI B,	2 байт	01H
LXI D,	2 байт	11H
LXI H,	2 байт	21H
LXI SP,	2 байт	31H

Загрузка/размещение данных

LDAX	B	0AH	STAX	B	02H
LDAX	D	1AH	STAX	D	12H
LHLD	адр.	2AH	SHLD	адр.	22H
LDA	адр.	3AH	STA	адр.	32H

Здесь адр. — 16-разрядные адрес (2-й и 3-й байт трехбайтовых команд).

Индикаторы условия не устанавливаются командами этой группы. Приведем обозначения сокращений (например, r1, DDD, SSS и т. д.), используемых в следующих описаниях команд пользователя, составленных фирмой Intel.

Рассмотрим команду MOV *A*, *B* (КОП—78H) МП Intel 8080/8085, которая передаст данные регистра *B* в аккумулятор (регистр *A*). Этот тип команды был уже изучен в § 6.5. Заметим, что первый регистр (*A* в этом примере) является назначением, а другой (*B*) источником данных. Общий состав КОП этого типа команд будет 01DDDSSS, его можно составить исходя из информации, приведенной выше. Назначением является аккумулятор, чему соответствует состав бит 111, источником — регистр *B* с составом

Символ	Значение
A	Аккумулятор (регистр *A*)
addr	Значение адреса — 16 бит
data	8-разрядное число (данные)
data 16	16-разрядное число (данные)
byte 2	Второй байт команды
byte 3	Третий байт команды
port	8-разрядный адрес УВВ
r, r1, r2	Один из регистров *A, B, C, D, E, H, L*
DDD, SSS	Набор битов, представляющий один из регистров *A, B, C, D, E, H, L* (DDD — назначение, SSS — источник) DDD или Регистр SSS 111 *A* 000 *B* 001 *C* 010 *D* 011 *E* 100 *H* 101 *L*
rp	Одна из пар регистров: *B* представляет пару *BC*, где *B* — старший регистр, *C* — младшнй регистр; *D* представляет пару *DE*, где *D* — старший регистр, *E* — младшнй регистр; *H* — представляет пару *HL*, где *H* — старший регистр, *L* — младший регистр; *SP* — 16-разрядный указатель стека
RP	Набор бит, представляющий одну нз пар регистров *B, D, H, SP* RP Пара регистров 00 *BC* 01 *DE* 10 *HL* 11 *SP*
rh	Первый (старший) регистр даиной пары
rl	Второй (младшнй) регистр данной пары
PC	16-разрядиый счетчик команд (PCH и PC представляют соответственно 8 старших и 8 младших бит)
SR	16-разрядный указатель стека (SPH и SPL представляют соответственно старшнх 8 и младших 8 бнт)
r_m	*m*-й бит регистра (биты нумеруются от 0 до 7 справа налево)
ETIQETTE (LABEL)	16-разрядный адрес подпрограммы
	Индикаторы
Z	Нуля
S	Знака
P	Четностн

Продолжение вывода

CV	Переноса
AC	Вспомогательного переноса
( )	Содержимое ячейки памяти или регистра, имя которого или адрес заключены в скобки
←	Передать
∧	Логическое И
∀	ИЛИ ИСКЛЮЧАЮЩЕЕ
∨	ИЛИ
+	Сложение
—	Вычитание дополнительного кода
*	Умножение
↔	Обмен
—	Обратный код [например, (A)]
n	Номер позиции (от 0 до 7)
NNN	Двоичное представленне (000—111) номера повторного запуска (рестарта) от 0 до 7

бит 000. Тогда КОП будет $0111\ 1000_2$ или 78H. По данным фирмы Intel, команда MOV *A*, *B* использует только один машинный цикл (четыре периода тактовых импульсов *T*) для передачи данных и выполняется с помощью регистровой адресации. Она не устанавливает никаких индикаторов в ходе своего выполнения. Каждая из команд передачи соответствует той или другой более общей категории и может быть анализирована подобным образом. Отметим, что в результате команд передачи данных никакие индикаторы не изменяются.

MOV r1, r2 (*Move register*). Передача между регистрами. (r1)←(r2). Содержимое регистра 2 передается в регистр 1.

0 1 D D D S S S

Циклов — 1; периодов *T* — 4 (8085), 5 (8080); адресация — регистровая.

MOV r, M (*Move from memory*). Передача из памяти (r)←((H)(L)). Содержимое памяти, адрес которой находится в паре *HL*, передается в регистр *r*.

0 1 D D D 1 1 0

Циклов — 2; периодов *T* — 7; адресация — косвенная регистровая.

MOV M, r (*Move to memory*). Передача в память. ((H)(L))←(r). Содержимое регистра *r* передается в память, адрес которой находится в паре *HL*.

0 1 1 1 0 S S S

Циклов — 2; периодов *T* — 7; адресация — косвенная регистровая.

MOV r, data (*Move immediate*). Передача непосредственная. (r)←(байт 2). Содержимое байта 2 команды передается в регистр *r*.

0 0 D D D 1 1 0
данные

Циклов — 2; периодов *T* — 7; адресация — непосредственная.

MVI M, data (*Move to memory immediate*). Передача в память непосредственная. ((H)(L))←(байт 2). Содержимое байта 2 команды передается в память, адрес которой указан парой *HL*.

0 0 1 1 0 1 1 0
данные

Циклов — 3; периодов *T* — 10; адресация — непосредственная/косвенная регистровая.

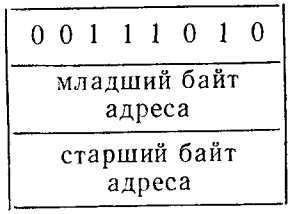
LXI rp, data 16 (*Load register pair immediate*). Загрузить непосредственно пару регистров. (rh)←(байт 3); (rl)←(байт 2). Байт 3 команды передается в старший регистр (rh), пары *rp*, а байт 2 — в младший регистр (rl) той же пары.

0 0 R P 0 0 0 1
младший байт данных
старший байт данных

Циклов — 3; периодов *T* — 10; адресация — непосредственная.

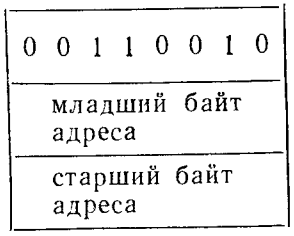
LDA addr (*Load accumulator direct*). Прямая загрузка

аккумулятора (A)←((байт 3) (байт 2)). Содержимое памяти, адрес которой определен байтами 2 и 3 команды, передается в аккумулятор (регистр *A*).



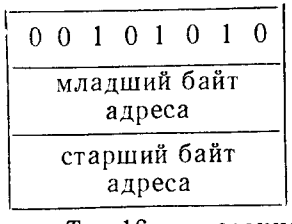
Циклов — 4; периодов *T* — 13; адресация — прямая.

STA addr (*Store accumulator direct*). Прямое размещение содержимого аккумулятора. ((байт 3) (байт 2))←(A). Содержимое аккумулятора передается в память, адрес которой указан байтами 2 и 3 команды.



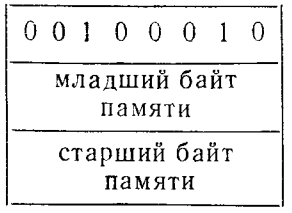
Циклов — 4; периодов *T* — 13; адресация — прямая.

LHLD addr (*Load H and L direct*). Загрузить H и L прямо. (L)←((байт 3) (байт 2)); (H)←((байт 3) (байт 2)+1). Содержимое ячейки памяти, адрес которой определяется байтами 2 и 3 команды, передается в регистр *L*. Содержимое следующего байта памяти передается в регистр *H*.



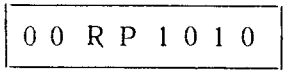
Циклов — 5; периодов *T* — 16; адресация — прямая.

SHLD addr (*Store H and L direct*). Поместить H и L прямо. ((байт 3) (байт 2))←(L); ((байт 3) (байт 2)+1)←←(H). Содержимое регистра *L* передается в ячейку памяти, адрес которой определяется байтами 2 и 3 команды. Содержимое регистра *H* передается в следующий байт памяти.



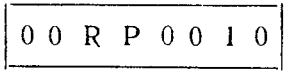
Циклов — 5; периодов *T* — 16; адресация — прямая.

LDAX гр (*Load accumulator indirect*). Загрузить аккумулятор косвенно. (A)←((гр)). Содержимое ячейки памяти, адрес которой определяется парой регистров *rp*, передается в регистр *A*. Могут использоваться только пары гр=B (регистры *B* и *C*) или гр=D (регистры *D* и *E*).



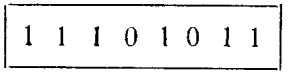
Циклов — 2; периодов *T* — 7; адресация — косвенная регистровая.

STAX гр (*Store accumulator indirect*). Загрузить аккумулятор косвенно. ((гр))←(A). Содержимое аккумулятора *A* передается в ячейку памяти, адрес которой содержится в паре регистров *rp*. Могут использоваться только пары гр=B (регистры *B* и *C*) или гр=D (регистры *D* и *E*).



Циклов — 2, периодов *T* — 7; адресация — косвенная регистровая.

XCHG (*Exchange H and with D and E*). Обмен H и L с D и E. (H)↔(D); (L)↔(E). Содержимое регистров *H* и *L* обменивается с содержимым регистров *D* и *E*.



Циклов — 1; периодов *T* — 4; адресация — регистровая.

Упражнения

8.36. Код операции команды MOV *A*, *M* _____ (шестнадцатеричный). Она передает данные, содержащиеся в _____ (ячейке памяти, регистре *A*), в _____ (ячейку

памяти, регистр A). Память определена _____ (парой HL, вторым и третьим байтами команды). Команда MOV A, M использует _____ машинных цикла и _____ периодов T. Команда MOV A, M имеет адресацию _____ (прямую, косвенную, регистровую).

8.37. Двоичный КОП MOV M, C _____. Эта команда передает данные _____ (из ячейки памяти, из регистра C) в _____ (ячейку памяти, регистр C). Ячейка памяти определена _____ (парой HL, вторым и третьим байтами команды). MOV M, C использует _____ машинных циклов и _____ периодов T. Команда MOV M, C имеет _____ (прямую, косвенную) адресацию.

8.38. См. рис. 8.14. Код операции MVI H _____ (двоичный).

8.39. См. рис. 8.14. Содержимое регистра H после команды MVI H (дать 8 бит).

8.40. См. рис. 8.14. Какое число машинных циклов и периодов T использует команда MVI H для выполнения передачи?

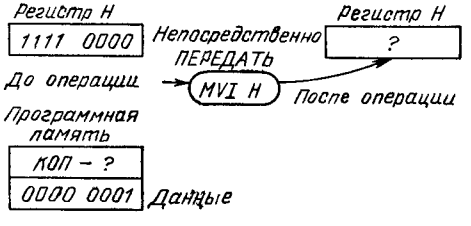


Рис. 8.14. К упражнениям 8.38—8.40

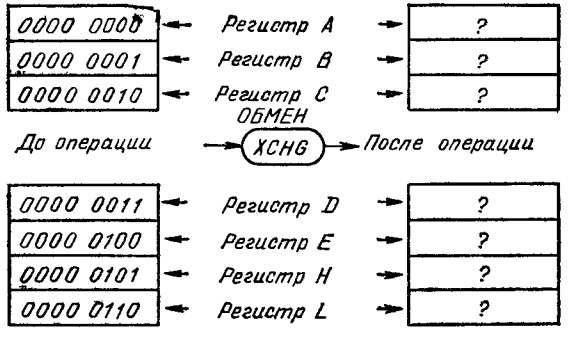


Рис. 8.15. К упражнениям 8.41—8.43

8.41. См. рис. 8.15. Дать список содержимого семи регистров МП Intel 8085 после операции обмена.

8.42. См. рис. 8.15. Команда XCHG имеет _____ (прямую, регистровую) адресацию.

8.43. См. рис. 8.15. Какие числа машинных циклов и периодов T необходимы для выполнения команды XCHG?

8.44. См. рис. 8.16. Команда SHLD имеет _____ (прямую, непосредственную) адресацию.

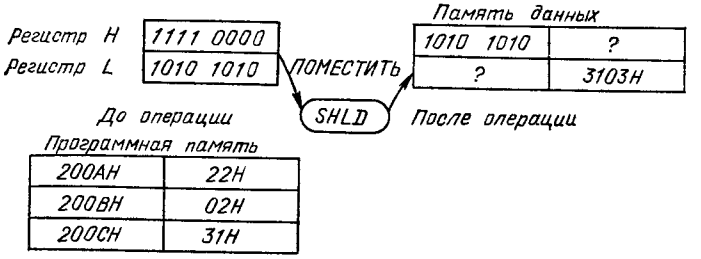


Рис 8 16. К упражнениям 8.44 и 8.45

8.45. См. рис. 8.16. После выполнения размещения данных, ячейка памяти _____ (шестнадцатеричное) содержит 1010 1010_2, а ячейка памяти 3102H — _____ (дать 8 бит).

Решения

8.36. 7EH; в ячейке памяти; в регистр A; парой HL; 2; 7; косвенную регистровую **8.37.** 0111 0001; из регистра C; память M; парой HL; два; семь; адресация косвенная регистровая. **8.38.** 0010 0110_2. **8.39.** 0000 0001_2. **8.40.** Два машнных цикла; семь пернодов T. **8.41.** Регистр A — 0000 0000 (без изменений); регистр B — 0000 0001 (без изменений); регистр C — 0000 0010 (без измененнй); регистр D — 0000 0101 (обмен с H); регнстр E — 0000 0110 (обмен с L); регистр H—0000 0011 (обмен с D); регнстр L — 0000 0100 (обмен с E). **8.42.** Регистровую (передача между регистрамн МП). **8.43.** Однн машинный цикл; четыре пернода T. **8.44.** Прямую. **8.45.** 3103H содержнт 1111 0000_2; 3102H содержит 1010 1010_2.

8.6. АРИФМЕТИЧЕСКИЕ КОМАНДЫ МП INTEL 8080/8085

Приведем кратқое описание арифметических команд МП Intel 8080/8085. Они предназначены для выполнения операций сложения, сложения с переносом, вычитания, вы-

читания с заемом, инкрементирования, декрементирования, десятичной коррекции аккумулятора.

Сложение*		Вычитание*		Удвоенная точность+		Декремент**	
ADD	A 87H	SUB	A 97H	DAD	B 09H	DCR	A 3DH
	B 80H		B 90H		D 19H		B 05H
	C 81H		C 91H		H 29H		C 0DH
	D 82H		D 92H		SP 39H		D 15H
	E 83H		E 93H				E 1DH
	H 84H		H 94H	Инкремент*			H 25H
	L 85H		L 95H				L 2DH
	M 86H		M 96H	INR	A 3CH		M 35H
ADC	A 8FH	SBB	A 9FH		B 04H	DCX	B 0BH
	B 88H		B 98H		C 0CH		D 1BH
	C 89H		C 99H		D 14H		H 2BH
	D 8AH		D 9AH		E 1CH		SP 3BH
	E 8BH		E 9BH		H 24H	Специальная*	
	H 8CH		H 9CH		L 2CH	DAA 27H	
	L 8DH		L 9DH		M 34H	Непосредственная адресация	
	M 8EH		M 9EH	INX	B 03H	ADI байт C6H	
					D 13H	ACI байт CEH	
					H 23H	SUI байт D6H	
					SP 33H	SBI байт DEH	

Здесь представлены мнемоники и КОП; + — устанавливается только (перенос), * — устанавливаются все индикаторы (*CY, Z, S, P, AC*); ** — устанавливаются все индикаторы, кроме переноса *CY* (исключение — INX и DCX, не устанавливаются никакие).

Как и в п. 8.5, группу арифметических команд будем изучать в собственном формате фирмы Intel. Эти команды оперируют с данными в памяти и регистрах. Во всех случаях, кроме указанных исключений, устанавливаются индикаторы нуля *Z*, знака *S*, четности *P*, переноса *CY* и вспомогательного переноса *AC*. Все операции вычитания проводятся с использованием дополнительного кода, устанавливают 1 в индикаторе переноса для указания переноса и сбрасывают его для указания отсутствия переноса. Как и в предыдущем случае, для понимания смысла сокращений следует обращаться к табл. 8.4.

ADD r (*Add register*). Сложение содержимого регистра. (A)←(A)+(r). Содержимое регистра *r* складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

1 0 0 0 0 S S S

Циклов — 1; периодов *T* — 4; адресация — регистровая; индикаторы — *Z, S, P, CY, AC*.

ADD M (*Add memory*). Сложение данных памяти. (A)←(A)+((H) (L)). Содержимое памяти, адрес которой содержится в регистрах *H* и *L*, складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

1 0 0 0 0 1 1 0

Циклов — 2; периодов *T* — 7; адресация — косвенная регистровая; индикаторы — *Z, S, P, CY, AC*.

ADI data (*Add immediate*). Непосредственное сложение. (A)←(A)+(байт 2). Содержимое байта команды 2 складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

1 1 0 0 0 1 1 0

Циклов — 2; периодов *T* — 7; адресация — непосредственная; индикаторы — *Z, S, P, CY, AC*.

ADC r (*Add register with carry*). Прибавление содержимого регистра и переноса. (A)←(A)+(r)+(CY). Содержимое регистра *r* и индикатора переноса (бит переполнения) складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

1 0 0 0 1 S S S

Циклов — 1; периодов *T* — 4; адресация — регистровая; индикаторы — *Z, S, P, CY, AC*.

ADC M (*Add memory with carry*). Прибавление содержимого памяти и переноса. (A)←(A)+((H) (L))+(CY). Содержимое памяти, адресом которой является содержимое пары регистров *HL*, и индикатора переноса складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

1 0 0 0 1 1 1 0

Циклов — 2; периодов *T* — 7; адресация — косвенная регистровая; индикаторы — *Z, S, P, CY, AC*.

ACI data (*Add immediate with carry*). Непосредственное сложение с учетом переноса. (A)←(A)+(байт 2)+(CY).

Содержимое 2 байт команды и индикатора переноса складывается с содержимым аккумулятора. Результат помещается в аккумулятор.

1 1 0 0 1 1 1 0

Циклов — 2; периодов T — 7; адресация — непосредственная; индикаторы — *Z, S, P, CY, AC.*

SUB r (*Subtract register*). Вычитание содержимого регистра. (A)←(A) — (r). Содержимое регистра *r* вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

1 0 0 1 0 S S S

Циклов — 1; периодов T — 5; адресация — регистровая; индикаторы — *Z, S, P, CY, AC.*

SUB M (*Subtract memory*). Вычитание содержимого памяти. (A)←(A) — ((H)(L)). Содержимое памяти, адрес которой является содержимым пары регистров *HL*, вычитается из аккумулятора. Результат помещается в аккумулятор.

1 0 0 1 0 1 1 0

Циклов — 2; периодов T — 7; адресация — косвенная регистровая; индикаторы — *Z, S, P, CY, AC.*

SUI data (*Subtract immediate*). Непосредственное вычитание. (A)←(A) — (байт 2). Содержимое 2 байт команды вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

1 1 0 1 0 1 1 0
данные

Циклов — 2; периодов T — 7; адресация — непосредственная; индикаторы — *Z, S, P, CY, AC.*

SBB r (*Subtract register with borrow*). Вычитание содержимого регистра и переноса. (A)←(A) — (r) — (CY). Содержимое регистра *r* и индикатора *CY* вычитается из содержимого аккумулятора. Результат помещается в аккумулятор.

1 0 0 1 1 S S S

Циклов — 1; периодов T — 4; адресация — регистровая; индикаторы — *Z, S, P, CY, AC.*

SBB M (*Subtract memory with borrow*). Вычитание содержимого памяти и переноса. (A)→(A) — ((H)(L)) — —(CY). Содержимое памяти, адресом которой является содержимое пары регистров *HL*, и индикатора *CY* вычитается из аккумулятора. Результат помещается в аккумулятор.

1 0 0 1 1 1 1 0

Циклов — 2; периодов T — 7; адресация — косвенная регистровая; индикаторы — *Z, S, P, CY, AC.*

SBB data (*Subtract immediate with borrow*). Непосредственное вычитание данных и переноса. (A)←(A) — — (байт 2) — (CY). Содержимое 2 байт команды и индикатора CY вычитаются из аккумулятора. Результат помещается в аккумулятор.

1 1 0 1 1 1 1 0
данные

Циклов — 2; периодов T — 7; адресация — непосредственная; индикаторы — *Z, S, P, CY, AC.*

INR r (*Increment register*). Инкремент содержимого регистра. (r)←(r)+1. Содержимое регистра увеличивается на 1. Устанавливаются все индикаторы состояния, за исключением *CY.*

0 0 D D D 1 0 0

Циклов — 1; периодов T — 4 (8085), 5 (8080); адресация — регистровая; индикаторы — *Z, S, P, AC.*

INR M (*Increment memory*). Инкремент содержимого памяти. ((H)(L))←((H)(L))+1. Содержимое памяти, адрес которой содержится в паре регистров *HL*, увеличивается на 1. Устанавливаются все индикаторы, кроме *CY.*

0 0 1 1 0 1 0 0

Циклов — 3; периодов T — 10; адресация — косвенная регистровая; индикаторы — *Z, S, P, AC.*

DCR r (*Decrement regisrer*). Декремент содержимого регистра. (r)←(r) — 1. Содержимое регистра *r* уменьшается на 1. Устанавливаются все индикаторы, кроме *CY*.

0 0 D D D 1 0 1

Циклов — 1; периодов *T* — 4 (8085), 5 (8080); индикаторы — *Z, S, P, AC*.

DCR M (*Decrement memory*). Декремент содержимого памяти. ((H) (L))←((H) (L)) — 1. Содержимое памяти, адрес которой содержится в паре регистров *HL*, уменьшается на 1. Устанавливаются все индикаторы, кроме *CY*.

0 0 1 1 0 1 0 1

Циклов — 3; периодов *T* — 10; адресация — косвенная регистровая; индикаторы — *Z, S, P, AC*.

INX rp (*Increment register pair*). Инкремент содержимого пары регистров (rh) (rl)←(rh) (rl)+1. Содержимое пары регистров *rp* увеличивается на 1. Не устанавливаются никакие индикаторы.

0 0 R P 0 0 1 1

Циклов — 1; периодов *T* — 6 (8085), 5 (8080); адресация — регистровая, индикаторы не изменяются.

DCX rp (*Decrement register pair*). Декремент пары регистров. (rh) (rl)←(rh) (rl) — 1. Содержимое пары регистров *rp* уменьшается на 1. Не устанавливаются никакие индикаторы.

0 0 R P 1 0 1 1

Циклов — 1; периодов *T* — 6 (8085), 5 (8080); адресация — регистровая; индикаторы не изменяются.

DAD rp (*Add register pair to H and L*). Сложить содержимое пары регистров с содержимым пары HL. (H) (L)← ←(H) (L)+(rh) (rl). Содержимое пары регистров *rp* складывается с содержимым пары *HL*. Устанавливается только индикатор *CY*. Он устанавливается в 1, если есть перенос при сложении с удвоенной точностью, если нет — сбрасывается.

0 0 R P 1 1 0 1

Циклов — 3; периодов *T* — 10; адресация — регистровая; индикаторы — *CY*.

DAA (*Decimal adjust accumulator*). Десятичная коррекция аккумулятора. 8-разрядное число в аккумуляторе разбивается на два 4-разрядных двоично-десятичных. Далее выполняются следующие действия: 1) если значение младшей тетрады акумулятора меньше 9 или установлен индикатор *AC*, к содержимому аккумулятора добавляется 6; 2) если значение старшей тетрады аккумулятора больше 9 или установлен индикатор переноса *CY*, 6 добавляется к значению старшей тетрады аккумулятора.

0 0 1 0 0 1 1 1

Циклов—1; периодов *T*—4; индикаторы — *Z, S, P, CY, AC*.

Рассмотрим команду ADC *B* (сложить содержимое регистра и индикатора переноса). Ее выполнение было описано выше как ADC r в формате Intel 8085, записывают ее в символической форме как

$$(A) \leftarrow (A) + (r) + (CY).$$

Это означает, что содержимое *r* (регистр *B* в нашем примере) и содержимое бита переноса *CY* складываются с содержимым аккумулятора (регистр *A*). Сумма помещается в аккумулятор. Описание Intel показывает, что это однобайтовая команда и она выполняется за один машинный цикл — четыре периода *T*. Способ адресации регистровый, и все индикаторы (*Z, S, P, CY, AC*) устанавливаются. На рис. 8.17 показано выполнение команды ADC *B*. Содержимое регистра *A* (здесь 1000 0000) складывается с содержимым регистра *B* (1000 0000) с установленным здесь в еди-

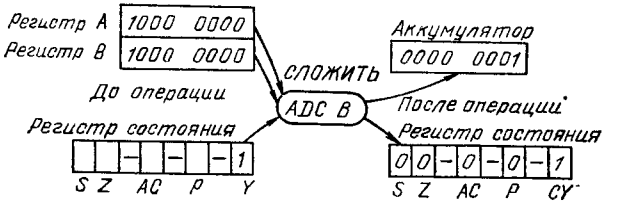


Рис. 8.17. Комаида сложения с переносом ADC *B*

ницу битом переноса. Задача могла бы быть представлена следующим образом:

	1000 0000	Регистр *A*
	1000 0000	Регистр *B*
+	1	Бит переноса
	1 0000 0001	Сумма

Восемь МБ (0000 0001) помещаются в аккумулятор (см. рис. 8.17). Переполнение аккумулятора (наиболее значимый бит суммы) устанавливает в 1 индикатор переноса *CY* после операции. Другие индикаторы сброшены в 0 после выполнения команды ADC *B* (рис. 8.17).

Рассмотрим теперь команду SBB *M* (вычесть содержимое памяти и переноса). Символическая запись фирмы Intel следующая:

$$(A) \leftarrow (A) - ((H)(L)) - (CY).$$

Это означает, что содержимое ячейки памяти, указанной парой *HL*, вычитается из содержимого аккумулятора. Бит переноса (*CY*) регистра состояния также вычитается из содержимого аккумулятора, и разность помещается в аккумулятор после выполнения команды. SBB *M* является однобайтовой командой, которая занимает два машинных цикла — семь периодов *T*.

На рис. 8.18 показано выполнение этой команды. Регистр *A* содержит 0000 0100, бит переноса — 1, и ячейка памяти, указанная парой *HL*, содержит 0000 0010. Рассмотрим действие команды SBB *M* над данными. Содержимое ячейки памяти, указанной парой *HL*, и бит переноса прежде всего складываются (0000 0010 + 1) = 0000 0011,

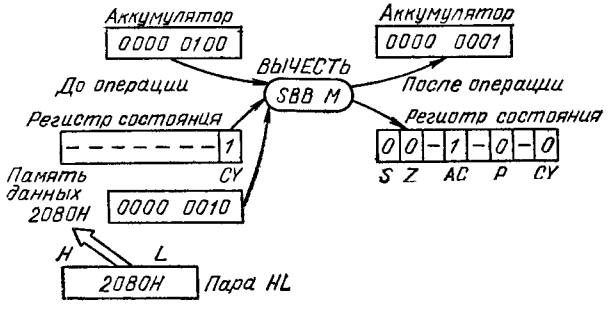


Рис. 8.18. Команда вычитания с заемом SBB *M*

новый член берется в дополнительном коде (1111 1101) и складывается с содержимым аккумулятора, что дает 1 0000 0001. Младших 8 бит (в нашем примере 0000 0001) составляют разность, тогда как переполнение 1 инвертируется МП, сбрасывая индикатор переноса *CY* в 0, т. е. переноса нет, первый член (содержимое *A*) был больше суммы двух других членов:

+	00000010	Память данных (2080H)
	1	Бит переноса
	00000011	Новый второй член
	00000100	Аккумулятор (первый член)
+	11111101	Дополнительный код второго члена
	1 00000001	

0
CY
Нет переноса — Переполнение — Разность

Команды сложения и вычитания с переносом не используются в случае вычитания или регулярного сложения с единственным байтом. Это особые команды, которые используют, чтобы сложить или вычесть два или несколько байт данных. Использование этих команд при реальном программировании будет показано в главе, посвященной программированию (см. гл. 9).

Упражнения

8.46. Код операции ADD *C*_____ ; по этой команде содержимое регистра _____ складывается с содержимым аккумулятора, и выполняется это с _____ (регистровой, косвенной регистровой) адресацией.

8.47. Код операции команды INX *H*_____ ; по этой команде _____ (инкрементируется, декрементируется) пара _____ (*BC*, *HL*), она_____ (устанавливает все, не устанавливает никаких) индикаторов.

8.48. См. рис. 8.19. Код операции команды DAD *B*_____.

8.49. См. рис. 8.19. По команде DAD *B* складывается содержимое пары HL емкостью _____ (8, 16) бит с содержимым пары _____.

8.50. См. рис. 8.19. После операции DAD *B* пара *HL* содержит _____.

8.51. См. рис. 8.19. Записать значения индикаторов регистра состояния после команды DAD *B*.

8.52. См. рис. 8.20. Является ли содержимое аккумулятора до операции DAA правильным двоично-десятичным (двоично-десятичное 8421)?

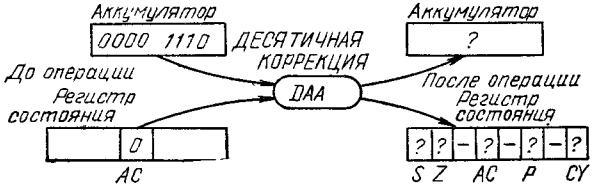


Рис. 8.19. К упражнениям 8.41—8.43

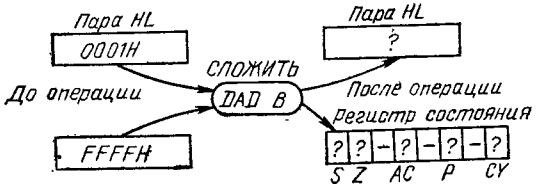
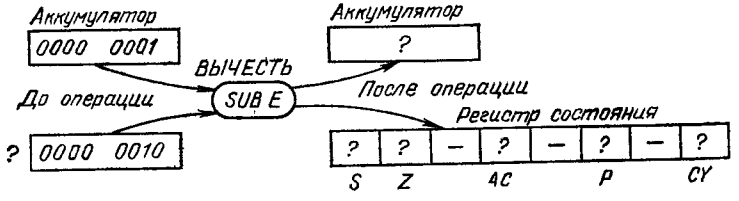


Рис. 8.20. К упражнениям 8.52—8.54



Рнс. 8.21. К упражнениям 8.55—8.57

8.53. См. рис. 8.20. Содержимое аккумулятора после операции DAA будет двоично-десятичным, десятичный эквивалент которого_____.

8.54. См. рис. 8.20. Записать состояние индикаторов в регистре состояния после операции DAA.

8.55. См. рис. 8.21. Команда SUB *E* вычитает содержимое регистра _____ из содержимого аккумулятора.

8.56. См. рис. 8.21. После выполнения команды SUB *E* содержимое аккумулятора _____.

8.57. См. рис. 8.21. Записать состояние индикаторов после выполнения команды SUB *E*.

Решения

8.46. 81H; *C*; регистровой. **8.47.** 23H; инкрементируется *HL*; не устанавливает никаких. **8.48.** 09H. **8.49.** 16; *BC*. **8.50.** Операцией является FFFFH+0001H=10000H; ((H)(L))=0000H. **8.51.** Устанавливается только индикатор переноса *CY*, состояние других индикаторов непредсказуемо. Так как имеется переполнение или перенос в ходе операции FFFFH+0001H=10000H, (CY)=1. **8.52.** 0000 1110 не является правильным двоично-десятичным, так как 1110 не имеет двоично-десятичного эквивалента. **8.53.** $0001\ 0100_{ДДК}$, т. е. 14_{10}. Так как 1110_2 больше 9, 0110 (6_{10}) было прибавлено к «неправильному» двоично-десятичному, что дает $0001\ 0100_{ДДК}=14_{10}$. **8.54.** DAA осуществляет сложение 0110_2 с $0000\ 1110_2$, что дает 0001 0100. Тогда индикаторы устанавливаются следующим образом: (Z)=0, (S)=0, (AC)=1, (P)=1 и (CY)=0. **8.55.** *E*. **8.56.** Микропроцессор преобразует содержимое *E* (0000 0010 в регистре *E*) в дополнительный код (1111 1110) и производит следующую операцию:

```
  0000 0001  Первый член
+
  1111 1110  Дополнительный код второго члена
0 1111 1111  Разность.
```

Восемь младших бит разности (1111 1111) помещены в аккумулятор. А в десятичной форме задача записывается как $1-2=-1_{10}=1111\ 1111_2$ в дополнительном коде. **8.57.** (S)=1, (Z)=0, (AC)=1, (P)=1, (CY)=1. Микропроцессор преобразует второй член (0000 0010 в регистре *E*) в дополнительный код (1111 1110) и складывает:

```
                0000 0001   Первый член
              +
Инверсия        1111 1110   Дополнительный ход
1<-------0 1111 1111        второго члена
CY                          Разность.
```

Заметим, что переполнение (старший бит) в разности — 0, он инвертируется и помещается в *CY*. Индикатор *CY*, установленный после вычитания, показывает, что разность является дополнительным кодом и второй член больше первого.

8.7. ЛОГИЧЕСКИЕ КОМАНДЫ МП INTEL 8080/8085

Приведем в краткой форме запись логических команд МП Intel 8080/8085. Задачей этих команд является выполнение логических операций И, ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ, сравнения, сдвига и инвертирования. Здесь приводятся только КОП и мнемоники,

ANA	A	A7H	ORA	A	B7H
	B	A0H		B	B0H
	C	A1H		C	B1H
	D	A2H		D	B2H
	E	A3H		E	B3H
	H	A4H		H	B4H
	L	A5H		L	B5H
	M	A6H		M	B6H

Команды непосредственной адресации
ANI байт E6H
XRI байт EEH
ORI байт F6H
CPI байт FEH

Циклический сдвиг
RLC 07H

			CMP		
XRA	A	AFH		A	BFH
	B	A8H		B	B8H
	C	A9H		C	B9H
	D	AAH		D	BAH
	E	ABH		E	BBH
	H	ACH		H	BCH
	L	ADH		L	BDH
	M	AEH		M	BEH

RRC 0FH
RAL 17H
RAR 1FH

Специальные команды
CMA 2FH
STC 37H
CMC 3FH

Как и в других случаях, группа логических команд приведена в собственном формате фирмы Intel. Эти команды выполняют логические операции над данными в памяти или регистрах и индикаторах. Обозначения в сокращениях соответствуют табл. 8.4.

ANA r (*AND register*). И регистр. (A)←(A)∧(r). Содержимое регистра *r* подвержено логическому И с содержимым аккумулятора. Результат помещается в аккумулятор. Индикатор *CY* сбрасывается, *AC* — устанавливается (8085). По результату логического ИЛИ, выполненного с третьими битами операндов, индикатор *CY* сбрасывается, *AC* устанавливается (8080).

1 0 1 0 0 S S S

Циклов — 1; периодов *T* — 4; адресация — регистровая; индикаторы — *Z, S, P, CY, AC*.

ANA M (*AND memory*). И память. (A)←(A)∧((H)(L)). Содержимое памяти, адрес которой находится в регистрах *HL*, подвержено логическому И с содержимым аккумулятора. Результат помещается в аккумулятор. Индикатор *CY* сбрасывается, *AC* устанавливается (8085). Индикатор *CY* сбрасывается, а *AC* устанавливается по результату логического ИЛИ третьих бит операндов (8080).

1 0 1 0 0 1 1 0

Циклов — 2; периодов *T* — 7; адресация — косвенная регистровая; индикаторы — *Z, S, P, CY, AC*.

ANI data (*AND immediate*). Непосредственное И. (A)←←(A)←(байт 2). Содержимое 2 байт команды подвержено логическому И с содержимым аккумулятора. Индикатор *CY* сбрасывается, *AC* устанавливается (8085). *CY* сбрасывается, *AC* устанавливается по результату логического ИЛИ третьих бит операндов (8080).

1 1 1 1 0 0 1 1 0
данные

Циклов — 2; периодов *T* — 7; адресация — непосредственная; индикаторы — *Z, S, P, CY, AC*.

XRA r (*Exclusive OR register*). ИЛИ ИСКЛЮЧАЮЩЕЕ регистра. (A)←(A)∀(r). ИЛИ ИСКЛЮЧАЮЩЕЕ выполняется с содержимыми регистра *r* и аккумулятора. Индикаторы *CY* и *AC* сбрасываются.

1 0 1 0 1 S S S

Циклов — 1; периодов *T* — 4; адресация — регистровая; индикаторы — *Z, S, P, CY, AC*.

XRA M (*Exclusive OR memory*). ИЛИ ИСКЛЮЧАЮЩЕЕ памяти. (A)←(A)∀((H)(L)). ИЛИ ИСКЛЮЧАЮЩЕЕ выполняется с содержимым памяти, адрес которой указан парой *HL*, и содержимым аккумулятора. Результат помещается в аккумулятор. Индикаторы *CY* и *AC* сбрасываются.

1 0 1 0 1 1 1 0

Циклов — 2; периодов *T* — 7; адресация — косвенная регистровая; индикаторы — *Z, S, P, CY, AC*.

XPI data (*Exclusive OR immediate*). ИЛИ ИСКЛЮЧАЮЩЕЕ непосредственное. (A)←(A)∀(байт 2). ИЛИ ИСКЛЮЧАЮЩЕЕ выполняется с содержимым 2 байта команды и содержимым аккумулятора. Результат помещается в аккумулятор. Индикаторы *CY* и *AC* сбрасываются.

1 1 1 0 1 1 1 0
данные

Циклов — 2; периодов T — 7; адресация — непосредственная; индикаторы — *Z, S, P, CY, AC.*

ORA r (*OR register*). ИЛИ регистр. (A)←(A)∨(r). ИЛИ выполняется с содержимым регистра *r* и аккумулятора. Результат помещается в аккумулятор. Индикаторы *CY* и *AC* сбрасываются.

1 0 1 1 0 S S S

Циклов — 1; периодов T — 4; адресация — регистровая; индикаторы — *Z, S, P, CY, AC.*

ORA M (*OR memory*). ИЛИ память. (A)←(A)∨∨((H))(L)). ИЛИ выполняется с содержимым памяти, адрес которой содержится в регистрах *HL*, и аккумулятора. Результат помещается в аккумулятор. Индикаторы *CY* и *AC* сбрасываются.

1 0 1 1 0 1 1 0

Циклов — 2; периодов T — 7; адресация косвенная регистровая; индикаторы — *Z, S, P, CY, AC.*

ORI data (*OR immediate*). ИЛИ непосредственно. (A)←(A)∨(байт 2). ИЛИ выполняется с содержимыми 2 байта команды и аккумулятора. Результат помещается в аккумулятор. Индикаторы *CY* и *AC* сбрасываются.

1 1 1 1 0 1 1 0
данные

Циклов — 2; периодов T — 7; адресация — непосредственная; индикаторы — *Z, S, P, CY, AC.*

CMP r (*Compare register*). Сравнить регистр. (A) — (r). Содержимое регистра *r* вычитается из содержимого аккумулятора. Содержимое аккумулятора не изменяется. Индикаторы устанавливаются, как при вычитании. Устанавливается 1 в *Z*, если (A) = (r). Устанавливается 1 в *CY*, если (A) < (r).

1 0 1 1 1 S S S

Циклов — 1; периодов T — 4; адресация — регистровая; индикаторы — *Z, S, P, CY, AC.*

CMP M (*Compare memory*). Сравнить память. (A) — —((H)(L)). Содержимое памяти, адрес которой является содержимым пары *HL*, вычитается из содержимого аккумулятора, которое не изменяется. Индикаторы устанавливаются, как при вычитании. Устанавливается 1 в *Z*, если (A) = ((H)(L)). Устанавливается 1 в CY, если (A) < <((H)(L)).

1 0 1 1 1 1 1 0

Циклов — 2; периодов T — 7; адресация — косвенная регистровая; индикаторы — *Z, S, P, CY, AC.*

CPI data (*Compare immediate*). Непосредственное сравнение. (A) — (байт 2). Содержимое 2 байт команды вычитается из аккумулятора. Индикаторы устанавливаются, как при вычитании. Устанавливается 1 в *Z*, если (A) = (байт 2). Устанавливается 1 в *CY*, если (A) < (байт 2).

1 1 1 1 1 1 1 0
данные

Циклов — 2; периодов T — 7; адресация — непосредственная; индикаторы — *Z, S, P, CY, AC.*

RLC (*Rotate left*). Сдвиг влево. $(b_{n+1})\leftarrow(b_n)$; $(b_0)\leftarrow$ $\leftarrow(b_7)$; (CY)←(b_7). Содержимое аккумулятора сдвигается на одну позицию влево. Младший бит b_0 и *CY* принимают значения вытесненного бита, т. е. бывшего старшего бита b_7 (рис. 8.22, *в*). Устанавливается только *CY*.

0 0 0 0 0 1 1 1

Циклов — 1; периодов T — 4; индикатор — *CY*.

RRC (*Rotate right*). Сдвиг вправо. $(b_n)\leftarrow(b_{n+1})$; $(b_7)\leftarrow$ $\leftarrow(b_0)$; (CY)←(b_0). Содержимое аккумулятора сдвигается на одну позицию вправо. Старший бит b_7 и *CY* принимают значения вытесненного бита, т. е. бывшего младшего бита b_0 (см. рис. 8.22, *г*). Устанавливается только *CY*

0 0 0 0 1 1 1 1

Циклов — 1; периодов T — 4; индикатор — *CY*.

RAL (*Rotate left trough carry*). Циклический сдвиг вле-

во. $(b_{n+1}) \leftarrow (b_n)$; $(CY) \leftarrow (b_7)$; $(b_0) \leftarrow (CY)$. Содержимое аккумулятора сдвигается влево циклически вместе с *CY*. В младшем бите b_0 устанавливается содержимое *CY*, а в

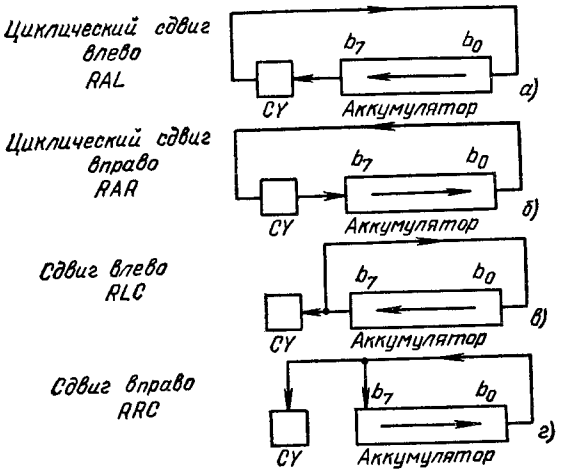


Рис 8.22. Команды сдвига:
а — циклического влево; *б* — циклического вправо; *в* — влево; *г* — вправо

CY — вытесненное содержимое бывшего старшего бита b_7 (рис. 8.22, *а*). Устанавливается только *CY*.

0 0 0 1 0 1 1 1

Циклов — 1; периодов *T* — 4; индикатор — *CY*.

RAR (*Rotate right trough carry*). Циклический сдвиг вправо. $(b_n) \leftarrow (b_{n+1})$; $(CY) \leftarrow (b_0)$; $(b_7) \leftarrow (CY)$. Содержимое аккумулятора сдвигается вправо циклически вместе с *CY*. В старшем бите b_7 устанавливается содержимое *CY*, а в *CY* — вытесненное содержимое бывшего младшего бита b_0 (рис. 8.22, *б*). Устанавливается только *CY*.

0 0 0 1 1 1 1 1

Циклов — 1; периодов *T* — 4; индикатор — *CY*.

CMA (*Complement accumulator*). Инвертировать аккумулятор. $(\overline{A}) \leftarrow (A)$. Содержимое аккумулятора инвертируется (все бит 0 становятся 1, все бит 1 становятся 0). Индикаторы не устанавливаются.

0 0 1 0 1 1 1 1

Циклов — 1; периодов *T* — 4; индикаторы — никакие.

CMC (*Complement carry*). Инвертировать перенос. $(CY) \leftarrow (\overline{CY})$. Инвертируется содержимое индикатора *CY*. Никакие прочие индикаторы не устанавливаются.

0 0 1 1 1 1 1 1

Циклов — 1; периодов *T* — 4; индикаторы — никакие.

STC (*Set carry*). Установить перенос. $(CY) \leftarrow 1$. В индикаторе *CY* устанавливается 1. Никакие прочие индикаторы не устанавливаются.

0 0 1 1 0 1 1 1

Циклов — 1; периодов *T* — 4; индикаторы — никакие.

Отметим употребление фирмой Intel знаков: $\wedge$ — для обозначения операции логического И в описаниях, относящихся к данным в командах ANA r, ANA M и ANI; $\vee$ — для представления операции логического ИЛИ в описаниях, относящихся к данным в командах ORA r, ORA M и ORI; $\forall$ — для логической операции ИЛИ ИСКЛЮЧАЮЩЕЕ в описаниях, относящихся к командам XRA r, XRA M и XRI[1].

Рассмотрим четыре команды сдвига в МП Intel 8080/8085, представленные на рис. 8.22. Эффект команд сдвига вправо и влево, включающих перенос, поясняется на рис. 8.22, *а, б* (RAL и RAR). Мы уже встречали эти действия, изучая команды сдвига типового МП (см. § 6.4).

На рис. 8.22, *в, г* приведены действия МП по командам сдвига аккумулятора вправо и влево. Эти действия несколько отличны от предыдущих, потому что не включают переноса [2].

Упражнения

8.58. Код операции команды ANA M_____ . По этой команде выполняется операция И с содержимым аккумуля-

[1] Эти математические символы являются общепринятыми в теории дискретных структур. — *Прим ред*

[2] Их называют операциями арифметического сдвига. — *Прим. ред.*

тора и памяти _____ (программной, данных), указанной парой регистров _____ . Это _____ байтовая команда.

8.59. Код операции команды ORI _____. Это _____ байтовая команда _____ (прямой, непосредственной) адресации. ИЛИ выполняется с содержимым аккумулятора и памяти _____ (программы, данных).

8.60. См. рис. 8.23. Каким будет содержимое аккумулятора после выполнения команды RLC?

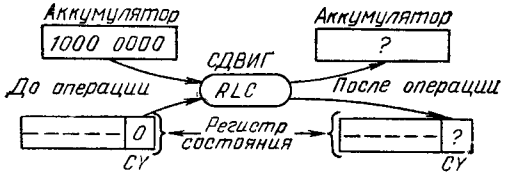


Рис. 8.23. К упражненям 8.60 и 8.61

Рис. 8.24. К упражнению 8.62

8.61. См. рис. 8.23. Индикатор переноса будет _____ (установлен в 1, сброшен в 0) после команды сдвига влево RLC.

8.62. См. рис. 8.24. После выполнения команды CMC индикатор переноса будет _____ (установлен в 1, сброшен в 0).

Решения

8.58. A6H; данных; HL; одно-. **8.59.** 6H; двух-; непосредственной; программы. **8.60.** 0000 0001. **8.61.** Установлен в 1. **8.62.** Сброшен в 0.

8.8. КОМАНДЫ ВЕТВЛЕНИЙ И ПЕРЕХОДОВ МП INTEL 8080/8085

Рассмотрим краткое изложение команд ветвлений и переходов МП Intel 8080/8085. Они записаны только мнемоникой фирмы Intel и шестнадцатеричным КОП каждой команды. Здесь адр. — 16-разрядный адрес (байт 2 и 3 трехбайтовой команды).

Переход		Возврат		Вызов		Повторный пуск (рестарт)	
JMP	адр. C3H	RET	C9H	CALL	адр. CDH	RST	0 C7H
JNZ	адр. C2H	RNZ	C0H	CNZ	адр. C4H		1 CFH
JZ	адр. CAH	RZ	C8H	CZ	адр. CCH		2 D7H
JNC	адр. D2H	RNC	D0H	CNC	адр. D4H		3 DFH
JC	адр. DAH	RC	D8H	CC	адр. DCH		4 E7H
JPO	адр. E2H	RPO	E0H	CPO	адр. E4H		5 EFH
JPE	адр. EAH	RPE	E8H	CPE	адр. ECH		6 F7H
JP	адр. F2H	RP	F0H	CP	адр. F4H		7 FFH
JM	адр. FAH	RM	F8H	CM	адр. FCH		
PCHL	адр. E9H						

Как и ранее, группа команд ветвления будет показана в собственном формате фирмы Intel. Она содержит команды перехода, вызова, возврата и повторного запуска (рестарта). Эта группа изменяет последовательный нормальный ход программы. Команды здесь двух типов: условного и безусловного переходов. Безусловные переходы просто выполняют операцию, определенную счетчиком команд; условные — проверяют состояние одного из индикаторов МП для определения целесообразности задаваемого ветвления. Условия, которые могут быть уточнены, задаются в технической документации пользователя фирмой Intel в следующей форме:

Условия	CCC	Условия	CCC
NZ — не нуль (Z=0)	000	PO — нечетность (P=0)	100
Z — нуль (Z=1)	001	PE — четность (P=1)	101
NC — нет переноса (CY=0)	010	P — плюс (S=0)	110
C — перенос (CY=1)	011	M — минус (S=1)	111

Смысл сокращений, используемых в следующих описаниях, приведен в § 8.5.

JMP addr (*Jump*). Переход или ветвление. (PC)←←(байт 3)(байт 2). Управление передается команде, адрес которой установлен в байт 3 и 2 текущей команды.

1 1 0 0 0 0 1 1
младший адрес
старший адрес

Циклов — 3; периодов T — 10; адресация — непосредственная; индикаторы — никакие.

Jcondition addr (*Conditionel jump*). Условное ветвление. Если (CCC), то (PC)←(байт 3) (байт 2). Если установленное условие выполняется, управление передается команде, адрес которой установлен в байт 2 и 3 текущей команды; если нет — управление продолжается последовательно.

1 1 C C C 0 1 0
младший адрес
старший адрес

Циклов — 2/3 (8085), 3 (8080); периодов *T* — 7/10 (8085), 10 (8080); адресация — непосредственная; индикаторы — никакие.

CALL addr (*Call*). Вызов. ((SP)—1)←(PCH); ((SP)—2)←(PCL); (SP)←(SP)—2; (PC)←(байт 3) (байт 2). Старших 8 бит следующей команды передаются в память, адрес которой на 1 меньше содержимого регистра *SP*. Восемь бит следующей команды передаются в память, адрес которой на 2 меньше содержимого регистра *SP*. Содержимое регистра *SP* дважды декрементируется. Управление передается команде, адрес которой определяется байт 3 и 2 текущей команды.

1 1 0 0 1 1 0 1
младший байт адреса
старший байт адреса

Циклов — 5; периодов *T* — 18 (8085), 17 (8080); адресация — непосредственная, косвенная регистровая; индикаторы — никакие.

Condition addr (*Conditionel call*). Условный вызов. Если (CCC), то ((SP)—1)←(PCH); ((SP)—2)←(PCL); (SP)←(SP)—2; (PC)←(байт 3) (байт 2). Если заданные условия выполняются, выполняются действия, определенные в команде CALL (см. выше); если нет — управление продолжается последовательно.

1 1 C C C 1 0 0
младший байт адреса
старший байт адреса

Циклов — 2/5 (8085), 3/5 (8080); периодов *T* — 9/18 (8085), 11/17 (8080); адресация — непосредственная, косвенно регистровая; индикаторы — никакие.

RET (*Return*). Возврат. (PCL)←((SP)); (PCH)←←((SP))+1); (SP)←(SP)+2. Содержимое памяти, адрес которой определен в регистре *SP*, передается в младших 8 бит регистра *PC*. Содержимое памяти, адрес которой на 1 старше содержимого регистра *SP*, передается в старших 8 бит регистра *PC*. Содержимое регистра *SP* дважды инкрементируется.

1 1 0 0 1 0 0 1

Циклы — 3; периодов *T* — 10; адресация — косвенная регистровая; индикаторы — никакие.

Rcondition (*Conditional return*). Возврат условный. Если (CCC), то (PCL)←((SP)); (PCH)←((SP)+1); (SP)←←(SP)+2. Если данное условие удовлетворено, выполняются действия, определенные командой возврата (см. выше), если нет — управление следует последовательно.

1 1 C C C 0 0 0

Циклы — 1/3; периодов *T* — 6/12 (8085), 5/11 (8080); адресация — косвенная регистровая; индикаторы — никакие.

RST (*Restart*). Повторный пуск (рестарт). ((SP)—1←←(PCH); ((SP)—2) ←(PCL); (SP)←(SP)—2; (PC)←←8×(NNN). Старших 8 бит адреса следующей команды передаются в память, адрес которой на 1 меньше содержимого *SP*. Младших 8 бит адреса следующей команды передаются в память, адрес которой меньше на 2 единицы содержимого регистра *SP*. Содержимое регистра *SP* декрементируется дважды. Управление передается команде, адресом которой является 8 раз содержимое NNN (8×××NNN)..

1 1 N N N 1 1 1

Циклов — 3; периодов T — 12 (8085); 11 (8080); адресация — косвенная регистровая; индикаторы — никакие.

15	14	13	12	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

Счетчик команд после команды рестарта.

PCHL (*Jump H and L indirect — move H and L to PC*). Перейти *H* и *L* косвенно — передать *H* и *L* в *PC*. (PCH)←←(H); (PCL)←(L). Содержимое регистра *H* передается в старших 8 бит регистра *PC*, а регистра *L* — в младших 8 бит регистра *PC*.

1 1 1 0 1 0 0 1

Циклов — 1; периодов T — 6 (8085), 5 (8080); адресация — регистровая; индикаторы — никакие.

Рассмотрим на рис. 8.25 использование команды условного перехода. Команда JZ имеет КОП 1100 1010_2. Часть 001 КОП (биты 5, 4 и 3) определяет, что условием перехода является (Z) = 1. На рис. 8.25 — это повод передать новый 16-битовый адрес в счетчик команд *PC*. Младший адрес (0000 0000) передается сначала, затем — старший адрес (0010 0001_2). Новый адрес, помещенный в счетчик команд, побуждает МП перейти в новую область программы.

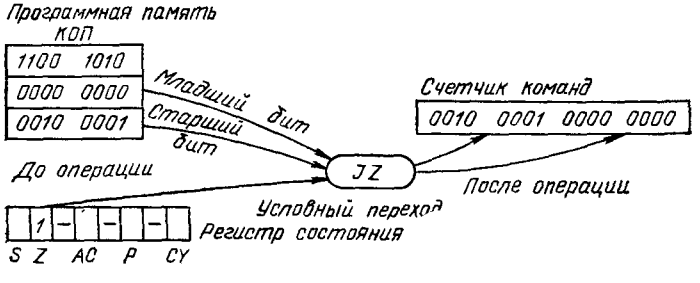


Рис. 8.25. Команда перехода (ветвления), если нулевой результат

Команда специального ветвления CALL используется для перехода от основной программы к подпрограмме. Она сочетает переход и помещение в стек данных. Как и в типовом МП, помещается сначала 16-разрядное содержимое счетчика команд *PC*. Затем осуществляется передача в счетчик команд 16-разрядного адреса подпрограммы. После этого МП переходит к подпрограмме.

В отличие от типового МП Intel 8080/8085 снабжен командами условного вызова. В этом случае только что описанная команда вызова выполняется, если только выполнено особое заданное условие.

Как и в типовом микропроцессоре, в Intel 8080/8085 используют команды возврата, позволяющие ему возвращаться в строго определенное место памяти основной программы после окончания подпрограммы. Кроме того, МП Intel 8080/8085 снабжены командами условного возврата, которые выполняются, если только выполняется особое заданное условие.

Команды рестарта (повторного пуска) являются специальными и специфическими командами вызова, предназначенными для прерываний. Как и CALL, команда рестарта помещает в стек содержимое счетчика команд и переходит по одному из восьми предопределенных адресов. Разница состоит только в том, что CALL непосредственно определяет адрес перехода. В формате команды RST адрес перехода определяется 3-разрядным кодом (бит 5, 4 и 3). Например, КОП команды RST 2—1101 0111 (D7H). Адресный код 010 (2_{10}), умножается на 8, что дает 16_{10}, или 10000_2, т.е. адрес перехода в этом примере составляет 0000 0000 0001 0000_2.

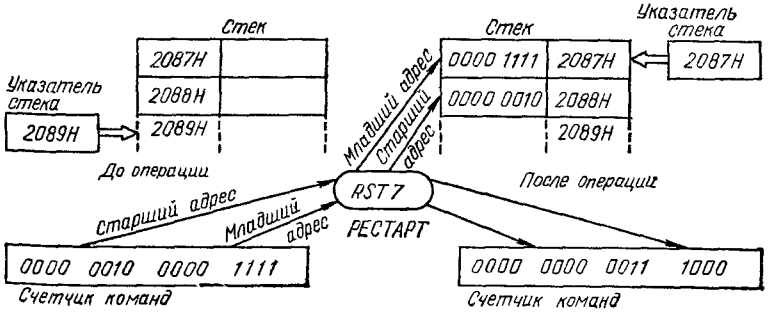


Рис. 8.26. Команда повториого пуска (рестарта) RST 7

Рассмотрим пример, показанный на рис. 8.26, относящийся к команде RST7. Заметим, что указатель стека установлен в 2089H до операции RST7. Эта команда помещает в стек текущее содержимое счетчика команд. Затем предопределенный адрес ($8\times7=56_{10}=38$H) передается в счетчик команд *PC*, что вызывает переход или рестарт МП по новому адресу 0000 0000 0011 1000_2 (0038H).

Команда рестарта редко появляется в коде источника в прикладных программах. Наиболее часто периферии, требующие обслуживания прерывания, посылают эту однобайтовую команду процессору. Поэтому иногда рестарты называют программными прерываниями.

Упражнения

8.63. Команда JMP является_____ байтовой и _____ (прямой, непосредственной) адресации. В ходе ее выполнения 2 байт программной памяти передаются в _____ (счетчик команд, указатель стека).

8.64. Команды вызова и _____ (возврата, повторного пуска) используются парно.

8.65. Операция _____ (возврата, рестарта) является особой формой команды вызова, используемой прерываниями.

8.66. Команда JNC является командой _____ (условного, безусловного) перехода, двоичный КОП которой _____ . Операция определена одной командой из_____ байт _____ (прямой, непосредственной) адресации.

8.67. Мнемоника однобайтовой команды, которая обращает МП Intel 8080/8085 в память 0000H, есть_____.

8.68. Мнемоника команды группы передачи данных, которая обращает МП Intel 8080/8085 по адресу, содержащемуся в паре *HL*, есть_____ .

8.69. См. рис. 8.27. Мнемоникой условного перехода является _____ .

8.70. См. рис. 8.27. Согласно имеющемуся содержимому в регистре состояния условия условного перехода _____ (выполнены, не выполнены)

8.71. См. рис. 8.27. Каково 16-разрядное содержимое счетчика команд после выполнения условного перехода.

8.72. См. рис. 8.28. Двоичный КОП команды рестарта _____ .

8.73. См. рис. 8.28. Перечислить содержимое ячеек 2183H и 2184H стека после выполнения операции рестарта.

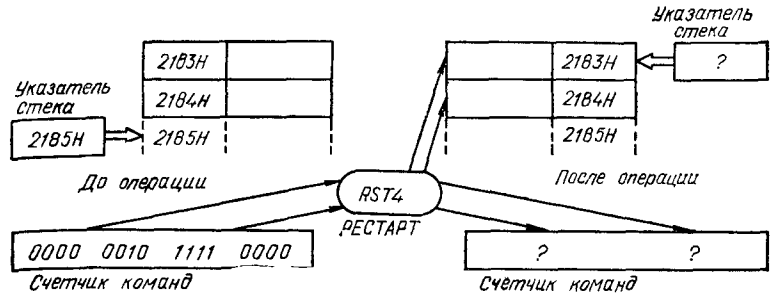


Рис 8.27. К упражнениям 8.69—8.71

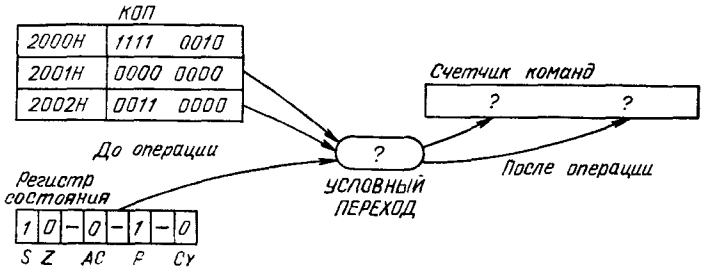


Рис. 8.28. К упражнениям 8.72—8.75

8.74. Перечислить содержимое счетчика команд *PC* после операции рестарта на рис. 8.28.

8.75. См. рис. 8.28. Каково шестнадцатеричное содержимое указателя стека после выполнения команды RST4?

Решения

8.63. Трех-; непосредственной; счетчик команд. **8.64.** Возврата. **8.65.** Рестарта. **8.66.** Условного; 1101 0010; 3; непосредственной. **8.67.** RSTO. **8.68.** PCHL. **8.69.** JP (перейти, если плюс). **8.70.** Индикатор знака (S)= =1; число отрицательно; перехода нет, значнт — не выполнены. **8.71.** (S)=1, перехода нет, значнт (PC)=2003H илн 0010 0000 0000 0011_2. **8.72.** 1110 0111 или E7H. **8.73.** (PC) загружается в стек следующим образом: сначала старшнй байт загружается в ячейку памятн 2184H, содержимое которой становится 0000 0010, затем младший байт — в ячейку памяти 2183H, которая будет содержать 1111 0000. **8.74.** Микропроцессор умножает номер рестарта (здесь 4_{10}) на 8, что дает 32_{10} (или 20H). Новый двончный адрес в PC — 0000 0000 0010 0000 (0020H), по которому переходит процессор. **8.75.** 2183H.

8.9. КОМАНДЫ СТЕКА, ВВ и УПРАВЛЕНИЯ МП INTEL 8080/8085

Рассмотрим краткое описание этих команд. Они выполняют операции помещения в стек и извлечения из него, ввода и вывода данных, обмена данными, подтверждения и неподтверждения прерываний, содержат команды отсутствия операций и останова, универсального считывания и установления маски прерывания.

Операции со стеком

PUSH	B	C5H
	D	D5H
	H	E5H
	PSW	F5H
POP	B	C1H
	D	D1H
	H	E1H
	PSW	F1H

Операции ВВ

OUT	байт	D3H
IN	байт	DBH

Операции управления

DI	F3H
EI	FBH
NOP	00H
HLT	76H

Новые команды. Только Intel 8085

XTHL	E3H
SPHL	F9H
RIM	20H
SUB	30H

Как и ранее, группа команд стека, ВВ и машинного управления показана в собственном формате фирмы Intel.

PUSH rp (*Push*). Поместить в стек. ((SP)—1)←(rh); ((SP)—2)←(rl); ((SP)←(SP)—2). Содержимое старшего регистра пары *rp* передается в память, адрес которой меньше на 1 содержимого регистра *SP*. Содержимое младшего регистра пары *rp* передается в память, адрес которой меньше на 2 содержимого регистра *SP*. Содержимое регистра *SP* декрементируется дважды. Пара регистров *rp*=*SP* не оговаривается.

1 1 R P 0 1 0 1

Циклов — 3; периодов *T* — 12 (8085); адресация — косвенная регистровая; индикаторы — не определены.

PUSH PSW (*Push processor status word*). Поместить в стек слово состояния процессора. ((SP)—1)←(A); $((SP)-2)_0$←(CY); $((SP)-2)_1$←X; $((SP)-2)_2$←(P); $((SP)-2)_3$←X; $((SP)-2)_4$←(AC); $((SP)-2)_5$←X; $((SP)-2)_6$←(Z); $((SP)-2)_7$←(S); (SP)←(SP)—2. X не оговаривается. Содержимое регистра *A* передается в память, адрес которой меньше на 1 содержимого регистра *SP*. Содержимое индикаторов составляет PSW, и это слово передается побитно (см. индексы в записи операций) в память, адрес которой меньше на 2 содержимого регистра *SP*, которое декрементируется дважды ((SP)—2).

1 1 1 1 0 1 0 1

Циклов — 3; периодов *T* — 12 (8085), 11 (8080); адресация — косвенная регистровая; индикаторы — никаких.

Слово индикаторов

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

X не оговаривается.

POP rp (*Pop*). Извлечь из стека. (rl)←((SP)); (rh)←←((SP)+1); (SP)←(SP)+2. Содержимое памяти, адрес которой определяется регистром *SP*, передается в младший регистр пары *rp*. Содержимое памяти, адрес которой больше на 1 содержимого регистра *SP*, передается в старший регистр пары *rp*. Содержимое регистра инкрементируется дважды. Пара *rp*=*SP* не оговаривается.

1 1 R P 0 0 0 1

Циклов — 3; периодов *T* — 10; адресация — косвенная регистровая; индикаторы — никакие.

POP PSW (*Pop processor status word*). Извлечь из стека слово состояния процессора. (CY)←$((SP))_0$; (P)←←$((SP))_2$; (AC)←$((SP))_4$; (Z)←$((SP))_6$; (S)←$((SP))_7$; (A)←((SP)+1); (SP)←(SP)+2. Содержимое памяти, адрес которой определен содержимым регистра *SP*, побитно (в соответствии с индексами в записи операций) используется для восстановления индикаторов условия. Содержимое памяти, адрес которой больше на 1 содержимого *SP*, передается в регистр *A*. Содержимое *SP* инкрементируется дважды.

1 1 1 1 0 0 0 1

Циклов — 3; периодов T — 10; адресация — косвенная регистровая; индикаторы — Z, S, P, AC, CY.

XTHL (*Exchange stack top with H and L*). Обменять вершину стека с H и L. (L)↔((SP)); (H)↔((SP)+1). Содержимое регистра H обменивается с содержимым памяти, адрес которой определен содержимым регистра SP. Содержимое регистра H обменивается с содержимым памяти, адрес которой больше на 1 содержимого SP.

1 1 1 0 0 0 1 1

Циклов — 5; периодов T — 16 (8085), 18 (8080); адресация — косвенная регистровая; индикаторы — никакие.

SPHL (*Move HL to SP*). Передать HL в SP. (SP)←←(H)(L). Содержимое регистров H и L (16 бит) передается в регистр SP.

1 1 1 1 1 0 0 1

Циклов — 1; периодов T — 6 (8085), 5 (8080); адресация — регистровая; индикаторы — никакие.

IN port (*Input*). Ввести данные. (A)←(данные). Данные, помещенные на 8-разрядную двунаправленную шину данных определенным портом, передаются в регистр A.

1 1 0 1 1 0 1 1
порт

Циклов — 3; периодов T — 10; адресация — прямая; индикаторы — никакие.

OUT port (*Output*). Вывести данные. (Данные)←(A). Содержимое регистра A помещается на 8-разрядную двунаправленную шину данных для передачи в определенный порт.

1 1 0 1 0 0 1 1
порт

Циклов — 3; периодов T — 10; адресация — прямая; индикаторы — никакие.

EI (*Enable interrupts*). Разрешение прерываний. Система прерываний разрешается тотчас после выполнения следующей команды. Прерывания не признаются во время выполнения команды EI.

1 1 1 1 1 0 1 1

Циклов — 1; периодов T — 4; индикаторы — никакие.

Примечание. Запрещается помещать команду EI на шину в ответ на INTA, протекающую в цикле INA (8085).

DI (*Disable interrupts*). Запрещение прерывания. Система прерывания не разрешается сразу после выполнения команды DI. Прерывания не признаются во время выполнения команды DI.

1 1 1 1 0 0 1 1

Циклов — 1; периодов T — 4; индикаторы — никакие. Запрещено помещать команду DI на шину в ответ на $\overline{\text{INTA}}$ во время цикла INA (8085).

HLT (*Halt*). Останов. Процессор останавливается. Регистры и индикаторы не устанавливаются (8085). Второй ALE выдается во время выполнения HLT для дестробирования информации о состоянии цикла HLT (8085).

0 1 1 1 0 1 1 0

Циклов — 1+ (8085), 1 (8080); периодов T — 5 (8085), 7 (8080); индикаторы — никакие.

NOP (*No op*). Нет операции. Не выполняется никакая операция. Регистры и индикаторы не устанавливаются.

0 0 0 0 0 0 0 0

Циклов — 1; периодов T — 4; индикаторы — никакие.

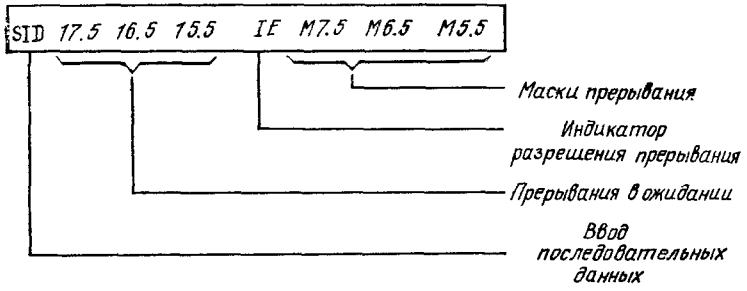
RIM (*Read interrupt mask*). Читать маску прерывания (только 8085). Команда RIM загружает в аккумулятор данные, относящиеся к прерываниям и последовательным вводам. Эти данные содержат следующую информацию: состояние маски текущего прерывания для аппаратных прерываний *RST5.5*, *6.5* и *7.5* (1 — маска запрещена); состояние индикатора текущего разрешения (1 — прерывание разрешено), за исключением случаев непосредственного

следования прерывания *TRAP* (см. ниже); аппаратные прерывания в ожидании (т.е. сигналы получены, но не обслуживаются) на линиях *RST5.5*, *6.5* и *7.5*; с последовательно вводимых данных.

Непосредственно после прерывания *TRAP* должна быть выполнена команда RIM как часть служебной подпрограммы, если позже необходимо разыскать состояние текущего прерывания. Индикатор состояния разрешения прерывания (EI), который был до прерывания *TRAP*, загружается (только в этом случае) в 3-й бит аккумулятора. После прерываний *RST5.5*, *6.5* и *7.5* или *INTR* триггер индикатора прерывания отражает состояние текущего разрешения. Состояние триггера *RST7.5* (всегда активизировано положительным фронтом ввода на линию *RST7.5*, даже когда это прерывание в предыдущем было маскировано), загружается в 6-й бит аккумулятора (17.5) (см. команду SIM).

КОП	00100000

Содержимое аккумулятора после команды RIM

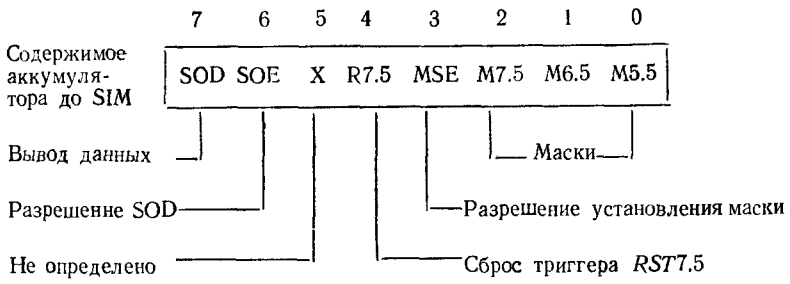


Циклов — 1; периодов T — 4; индикаторы — никакие.

SIM (*Set interrupt mask*). Установить маску прерывания (только 8085). Выполнение команды SIM использует содержимое аккумулятора (который должен быть предварительно загружен) для выполнения следующих действий: программировать маски аппаратных прерываний *RST5.5*, *6.5* и *7.5*; сбросить защелки для разъединения фронтом сигнала *RST7.5*; загрузить защелку выхода *SOD*. Для программирования маски прерываний: 1) Установить 3-й бит аккумулятора в 1, как и один из бит 0, 1, 2, которые не разрешают прерывания соответственно *RST5.5*, *6.5* и *7.5*. 2) Выполнить команду SIM. Если 3-й бит аккумулятора 0 в момент выполнения SIM, регистр маски прерывания не изменяется. Если 4-й бит аккумулятора 1, защелка *RST7.5* характеризуется тем, что она устанавливается положительным фронтом импульса на выводе входа *RST7.5* даже в случае, когда переход в подпрограмму обслуживания заторможен маскированием. Этот захват остается в Н-состоянии до тех пор, пока имеется состояние сброса, вызванное сигналом $\overline{RESET\ IN}$, командой SIM с 4-м битом Н-уровня аккумулятора или подтверждением получения внутри процессора прерывания *RST7.5*, следующим за снятием маски (командой SIM). Сигнал $\overline{RESET\ IN}$ активизирует всегда 3 бит маски *RST*.

Если 6-й бит аккумулятора устанавливается снова в 1, когда команда SIM выполнена, состояние 7-го бита аккумулятора загружается в защелку *SOD* и поступает в распоряжение интерфейса внешнего устройства. Защелка *SOD* не устанавливается командой STM, если 6-й бит 0. *SOD* всегда сброшена сигналом $\overline{RESET\ NI}$.

КОП	00110000



Циклов — 1; периодов T — 4; индикаторы — никакие.

Эта группа содержит операции помещения в стек и извлечения из стека данных. Как и в типовом МП, стек Intel 8080/8085 находится в ОЗУ. Вершина его определяется с помощью указателя стека по адресу, определяемому программистом. Все команды PUSH (поместить в стек) и POP (ИЗВЛЕЧЬ из стека) МП Intel 8080/8085 передают 2 байт данных только в одной команде. Команды PUSH PSW и POP PSW помещают в стек и извлекают из него данные слова состояния микропроцессора, которое соответствует сочетанию регистра состояния (индикаторов) и аккумулятора МП.

Команды IN (ввести) и OUT (вывести) действуют аналогично таким же командам в типовом МП и используются с изолированнымн ВВ.

Операция XTHL выполняется по однобайтовой команде обмена вершины стека с содержимым пары *HL*. Операция SPHL выполняется по команде, которая передает содержимое пары *HL* в указатель стека *SP*. Эта команда может быть использована для установления указателя стека, определяя, следовательно, вершину стека в ОЗУ.

Команда NOP не выполняет никакой операции (как и в типовом микропроцессоре). Она бывает полезна каждый раз, когда нужна задержка в цикле синхронизации.

Команда HLT останавливает МП. Счетчик команд содержит адрес следующей команды. Остальные регистры и индикаторы не изменяются. Войдя в это состояние, МП не может быть запущен кроме как внешним событием, обычно прерыванием. Эта команда используется для остановки МП, когда он ожидает, чтобы периферия завершила свою задачу и прервала действия МП.

Запуск МП может быть вызван одним из двух следующих способов: командой в памяти программы или внешним прерыванием. Аппаратные прерывания МП Intel 8080/8085 управляются двумя командами: разрешением прерывания (EI) и запрещением прерывания (DI). Когда команда EI выполнена, она устанавливает триггер разрешения прерывания внутри МП. Если индикатор установлен в 1 (или разрешен), МП Intel 8085 согласится и ответит одному из пяти возможных аппаратных прерываний.

Команда DI (запретить прерывание) сбрасывает в 0 индикатор разрешения прерывания (триггер), после этого МП игнорирует все прерывания, за исключением прерывания по входу *TRAP*, которое не может быть запрещено никакой из имеющихся в программе команд.

Команда EI часто используется как один из этапов пуска. При подаче напряжения МП пытается выполнить команды, расположенные по неопределенным адресам. Тогда используется обычно вход $\overline{RESET}$, который загружает счетчик команды адресом 0000H. Вход $\overline{RESET}$ осуществляет также сброс в 0 индикатора разрешения прерывания, в результате чего запрещаются все прерывания, за исключением *TRAP*. Одной из первых после этого сброса будет команда разрешения прерываний (EI).

Команды часто квалифицируются на маскируемые и немаскируемые. Немаскируемое прерывание — это такое,

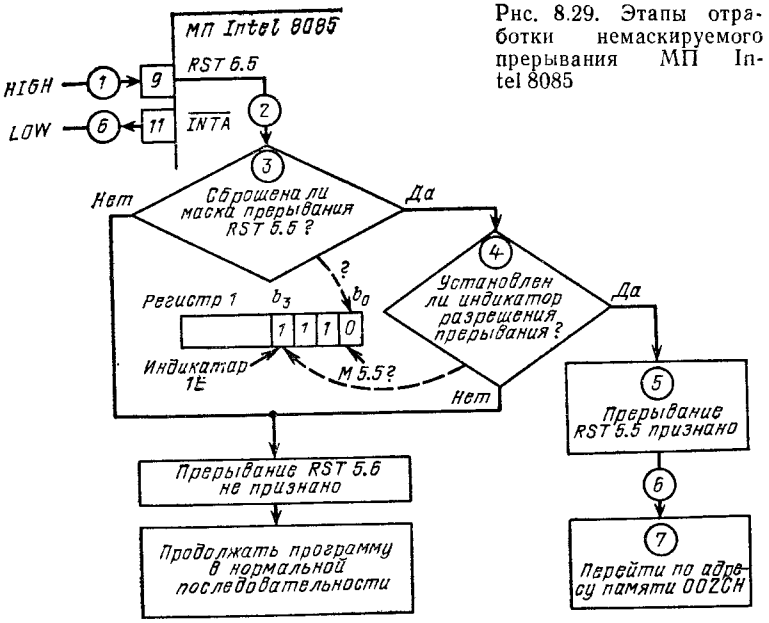


Рис. 8.29. Этапы отработки немаскируемого прерывания МП Intel 8085

которое система не может запретить, маскируемое прерывание может быть запрещено.

В микропроцессоре Intel 8085 прерывание *TRAP* рассматривается, как немаскируемое, все прочие — маскируемые.

Команды EI и DI разрешают и запрещают все прочие прерывания, взятые группой. Прерывания *RST7.5*, *RST6.5* и *RST5.5* могут быть разрешены или запрещены индивидуально. Действующий механизм, согласно которому можно разрешить или запретить прерывания избирательно, составляет маскирование прерывания. Когда прерывание маскировано, оно не разрешается МП. Если, напротив, маска снята, оно разрешается. Рассмотрим пример простого прерывания, представленного на рис. 8.29, и проследим операции МП, обращаясь к номерам в кружках.

1 — вход *RST5.5* (вывод *9*) кристалла МП Intel 8085 активизирован Н-сигналом;

2 — для упрощения примера принимаются три гипотезы:
   а) не активизируется прерывание приоритета *TRAP*;
   б) не активизируется или маскируется прерывание более высокого приоритета *RST7.5*;

в) не активизируется или маскируется прерывание более высокого приоритета *RST6.5*;

3 — бит маски прерывания (b_0) регистра состояния прерывания проверяется. Он сброшен в 0 или немаскирован, что удовлетворяет первому условию признания прерывания *RST5.5* МП.

4 — индикатор разрешения прерывания (b_3) проверяется. Он установлен в 1, что значит — все прерывания разрешены.

5 — прерывание *RST5.5* признается МП. Процессор заканчивает выполнение текущей операции и помещает в стек содержимое счетчика команд.

6 — кристалл МП Intel 8085 подтверждает получение запроса на прерывание — сигнал $\overline{\text{INTA}}$ в направлении периферии.

7 — микропроцессор переходит в память по адресу 002CH.

На рис. 8.29 МП решил проверить 2 бит регистра состояния прерывания. Сначала в ходе этапа 3: проверка бита маски *RST5.5* (*M5.5*). Бит *M5.5* был 0 или немаскирован, что означает, что МП мог признать прерывание. Затем на этапе 4 индикатор признания прерывания (*IE*) был проверен, в свою очередь. На рис. 8.29 индикатор *IE* имеет 1 (разрешен), что означает, что это прерывание может быть признано.

Некоторые биты регистра состояния прерывания могут быть изменены специальной командой (Intel 8085). Команда SIM (УСТАНОВИТЬ маску прерывания) помещает содержимое аккумулятора в регистр *I*. Мы уже видели, что SIM используется также на выходе бита данных в защелке *SOD* (вывод последовательных данных) МП Intel 8085 (см. рис. 8.5, *б*).

На рис. 8.30 показано использование регистра маскирования прерывания командой SIM. Биты маски прерываний *RST5.5*, *RST6.5* и *RST7.5* могут быть установлены (маскированы) или сброшены (демаскированы), если b_3 (разрешение установления маски) признан логической 1. Логическая 1 в позиции b_4 сбросит в 0 триггер *RST7.5*. На рис. 8.30 b_5 не определен. Биты b_6 и b_7 регистра маски прерывания объединены на последовательном выходе процессора (вывод *SOD* МП Intel 8085). Признавая b_6 единицей, мы позволяем передачу данных b_7 регистра 1 на выход *4* кристалла МП Intel 8085 (защелка *SOD*).

Действие команды RIM на регистр состояния прерывания показано на рис. 8.31. Команда RIM передает содер-

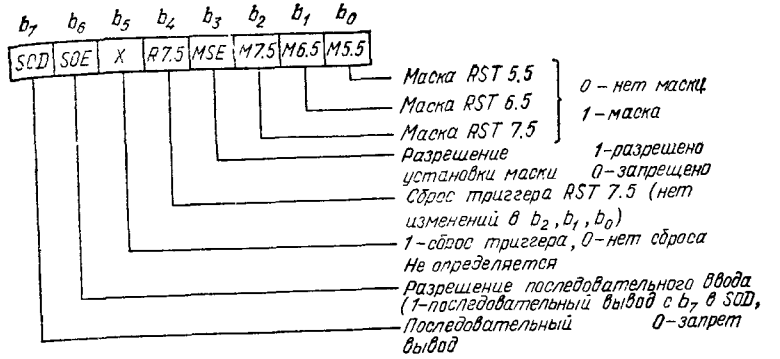


Рис. 8.30. Состав регистра прерываний при команде SIM

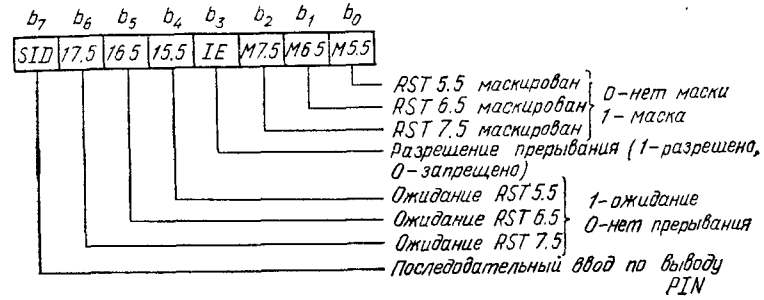


Рис. 8.31. Состав регистра прерываний при команде RIM

жимое внутренних защелок в аккумулятор. Мы уже использовали команду RIM (см. рис. 8.5, *а*) для ввода последовательных данных на вывод *SID* МП Intel 8085 в b_7 аккумулятора. Состояние других защелок, как и ввод последовательных данных, представлено в аккумуляторе.

Упражнения

8.76. См. § 8.9. Записать команды стека.

8.77. См.§ 8.9. Какие команды используются только в МП Intel 8085?

8.78. См. рис. 8.32. Каково содержимое (шестнадцатеричное) пары *HL* после выполнения команды XTHL?

8.79. См. рис. 8.32. Каково содержимое адресов стека 2080H после выполнения команды XTHL?

8.80. См. рис. 8.31. Каким будет бит регистра состояния прерывания, установленный командой EI?

8.81. Шестнадцатеричный КОП команды HLT _____. Эта команда _____ (устанавливает, не устанавливает) индикаторы. Будучи остановлен, процессор может быть пущен только таким событием, как _____ (прерывание, команда NOP).

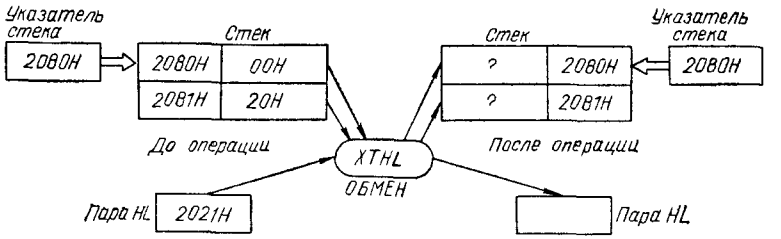


Рис. 8.32. К упражнениям 8.78 и 8.79

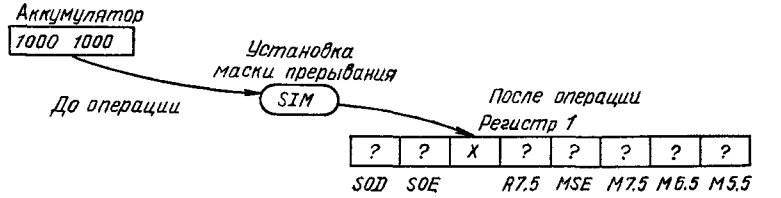


Рис. 8.33. К упражнениям 8.85—8.87

8.82. Команда OUT имеет _____ (прямую, регистровую) адресацию, и адрес порта находится в памяти _____ (данных, программы).

8.83. Команда RIM может быть использована для _____ (выдачи, получения) последовательных данных через вывод _____ (*SID*, *SOD*) МП Intel 8085. Этот бит данных будет в _____ бите аккумулятора.

8.84. Когда маскируется такое прерывание, как *RST6.5*, оно _____ (разрешается, запрещается).

8.85. См. рис. 8.33. Записать все биты регистра *I* после выполнения команды.

8.86. См. рис. 8.33. Будут ли последовательные выходящие данные захвачены в защелке *SOD* МП Intel 8085?

8.87. См. рис. 8.33. Записать немаскируемые команды после команды SIM.

Решения

8.76. PUSH и POP. Команды XTHL и SPHL также связаны со стеком тогда, когда они могут изменить содержимое указателя стека. **8.77.** RIM и SIM. **8.78.** 2000H. **8.79.** (2080H) = 21H и (2081H) = 20H. **8.80.** Бит b_3 устанавлнвается в 1. **8.81.** 76H; не устанавливает никакие; прерывание. **8.82.** Прямую; программы. **8.83.** Получения; *SID*; седьмом. **8.84.** Запрещается. **8.85.** *SOD* (последовательный ввод данных) — 1; *SOE* (разрешение последовательного ввода) — 0; *R7.5* (сброс триггера *RST7.5*) — 0; *MSE* (разрешение установить маску) — 1; *M7.5* (маска *RST7.5*) — 0; *M6.5* (маска *RST6.5*) — 0; *M5.5* (маска *RST5.5*) — 0. **8.86.** Бит разрешения последовательного вывода регистра *I* запрещен, следовательно, последовательных данных и их захвата не будет. **8.87.** *RST7.5*, *RST6.5*, *RST5.5*.

Дополнительные упражнения к гл. 8

8.88. Первый микропроцессор 4004 на 4 бит был выпущен фирмой Intel в_____.

8.89. Микропроцессор Intel 8085 является улучшенным вариантом _____.

8.90. Микропроцессор Intel 8085 используется обычно с двумя специальными периферийными кристаллами _____ которые содержат ОЗУ, ПЗУ и порты ВВ.

8.91. См. рис. 8.2. Выводы *12—19* являются цепями двунаправленной шины _____.

8.92. См. рис. 8.2. Вход *READY* тесно связан с состоянием _____ (останова, ожидания) процессора.

8.93. См. рис. 8.2. Какие два вывода (входной и выходной) тесно связаны с ПДП?

8.94. См. табл. 8.2. Сигналы управления МП Intel 8085 $IO/\overline{M}=0$, $S_1=1$ и $S_2=0$ показывают, что имеет место операция, выполняемая процессором. Какая?

8.95. Первичный аккумулятор является регистром _____.

8.96. Какой 16-разрядный регистр содержит всегда адрес и указывает на команду, выполняемую последовательно?

8.97. Индикатор знака, сброшенный в 0, означает, что результат выполненной арифметической операции _____ (положителен, отрицателен).

8.98. Индикатор нуля, сброшенный в 0 после арифметической операции, означает, что результатом в аккумуляторе является _____ (нуль, не нуль).

8.99. Вход *TRAP* является _____ (маскированным, немаскированным) прерыванием и не может быть запрещен.

8.100. См. табл. 8.3. Когда аппаратное прерывание *RST6.5* активизировано, МП сохранит содержимое _____ в стеке и ответвится по шестнадцатеричному адресу памяти _____.

8.101. Вторичные аккумуляторы (*BC*, *DE* и *HL*) могут быть аккумуляторами или _____.

8.102. Микропроцессор Intel 8085 снабжен _____ индикаторами, которые содержат информацию о состоянии.

8.103. Команда MOV *A*, *L* передает данные из одного регистра в другой, следовательно, имеет место _____ способ адресации.

8.104. Команда MVI *L* передает данные из второго байта команды в регистр _____, адресация будет _____.

8.105. Каждая команда косвенной регистровой адресации использует _____ (1, 3) байт программной памяти.

8.106. Команды косвенной адресации используют _____ (1, 2 или 3) байт программной памяти.

8.107. Состав команд МП Intel 8080 и Intel 8085 _____ (идентичен, почти идентичен).

8.108. Для Intel 8080/8085 команда поместить *A* в память является командой _____ (ветвления, передачи данных).

8.109. Для Intel 8080/8085 команда сдвига является командой _____ (передачи данных, логической).

8.110. Для Intel 8080/8085 команда CALL будет классифицирована как команда _____ (ветвления, логическая).

8.111. См. табл. 8.4. Каждая команда МП Intel 8080/8085 имеет мнемонику и _____.

8.112. См. табл. 8.4. Некоторые мнемоники содержат букву *I* — ADI *v*, ANI *v*, MVI *v*, например. Когда в командах появляется буква *I*, это означает, что адресация _____.

8.113. См. табл. 8.4. Каковы мнемоники двух команд, используемых МП Intel 8085 и не используемых МП Intel 8080?

8.114. В команде MOV *D*, *E* источником является регистр _____, назначением — регистр _____.

8.115. Источником данных команды MVI *A* является _____.

8.116. Команда MOV *B*, *M* имеет _____ адресацию.

8.117. Команда STA имеет _____ адресацию.

8.118. Команда MOV *A*, *L* имеет _____ адресацию.

8.119. Команда ADD *M* складывает содержимое памяти, указанное парой _____, с содержимым регистра _____. Сумма помещается в регистр _____.

8.120. См. рис. 8.34. Шестнадцатеричный КОП команды INX *H* _____.

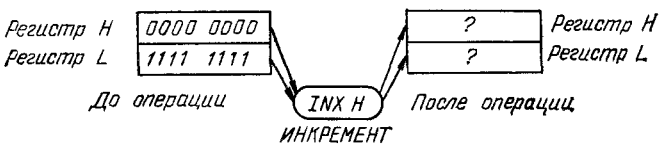


Рис. 8.34. К упражнениям 8.120—8.122

8.121. См. рис. 8.34. Команда INX *H* имеет _____ адресацию.

8.122. См. рис. 8.34. Записать содержимое регистров *H* и *L* после инкрементирования.

8.123. Знак $\vee$, используемый фирмой Intel, описывает логическую операцию _____ (И, ИЛИ).

8.124. Команда XRA *A* выполняет операцию ИЛИ ИСКЛЮЧАЮЩЕЕ содержания регистра _____ с самим собой. Результат _____ (сбрасывает, проверяет) аккумулятор.

8.125. См. рис. 8.35. Код операции команды CPI _____

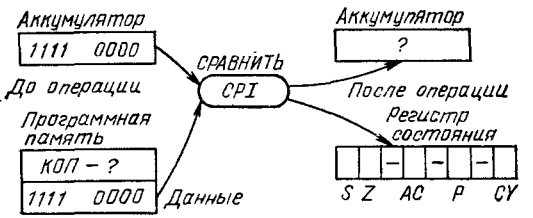


Рис. 8.35. К упражнениям 8.125—8.127

8.126. См. рис. 8.35. Каково содержимое аккумулятора после выполнения команды CPI?

8.127. См. рис. 8.35. Перечислить состояния индикаторов после команды CPI.

8.128. Команды CALL тесно связаны с подпрограммами, тогда как команды рестарта с _____.

8.129. Команда JPE относится к операции _____ (условного, безусловного) перехода.

8.130. Команда RNZ проверяет состояние индикатора _____ перед выполнением операции возврата.

8.131. См. рис. 8.36. Команда RET является _____ байтовой, КОП которой _____ .

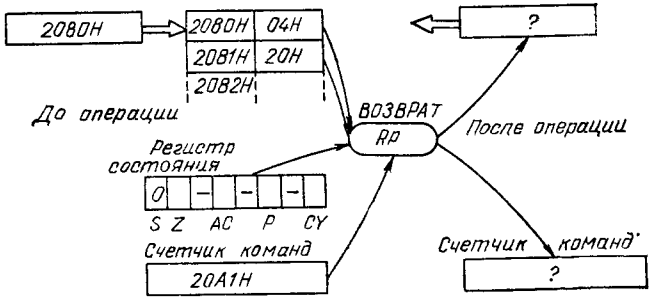


Рис. 8.36. К упражнениям 8.131—8.133

8.132. См. рис. 8.36. Каким будет содержимое счетчика команд после возврата из подпрограммы?

8.133. См. рис. 8.36. Каким будет содержимое указателя стека после выполнения команды RET?

8.134. Операции PUSH и POP выполняются по командам из _____ байт с _____ адресацией.

8.135. Операции IN и OUT выполняются по командам, состоящим из _____ байт с _____ адресацией.

8.136. Выполнение команды DI не будет признавать прерывания. Какие?

8.137. Каковы действия команды SIM?

Решения

8.88. 1971 г. **8.89.** Intel 8080. **8.90.** Intel 8155, Intel 8355. **8.91.** Адреса/данных (шина мультиплексирована). **8.92.** Ожидания. **8.93.** Входной *HOLD*, выходной *HLDA* (подтверждение состояния захвата). **8.94.** Считывание из памяти. **8.95.** *A*. **8.96.** Счетчик команд. **8.97.** Положителен. **8.98.** Не нуль. **8.99.** Немаскированным. **8.100.** Счетчика команд (*PC*); 34H. **8.101.** Указателями стека. **8.102.** Пятью. **8.103.** Регистровый. **8.104.** *L*; непосредственная. **8.105.** 1. **8.106.** 1. **8.107.** Почти идентичен. **8.108.** Передачи данных. **8.109.** Логический. **8.110.** Ветвления. **8.111.** КОП. **8.112.** Непосредственная. **8.113.** RIM, SIM. **8.114.** *E*; *D*. **8.115.** Второй байт команды в памятн программы. **8.116.** Косвенную регистровую. **8.117.** Прямую. **8.118.** Регистровую. **8.119.** *HL*; *A*; *A*. **8.120.** 23H. **8.121.** Регистровую. **8.122.** Регистр (H)=0000 0001, регистр (L)=0000 0000. **8.123.** ИЛИ. **8.124.** *A*; сбрасывает. **8.125.** FEH. **8.126.** 1111 0000 (без изменения). **8.127.** (S)=0, (AC)=0, (CY)=0, (Z)=1, (P)=1. **8.128.** Прерываниями. **8.129.** Условного. **8.130.** Нуля (Z). **8.131.** Одно-; F0H. **8.132.** 2004H. **8.133.** 2082H. **8.134.** 1; косвенной регистровой. **8.135.** 2; прямой. **8.136.** *INTR*; *RST5.5*; *RST6.5*; *RST7.5*. **8.137.** Программирование маски прерываний для аппаратных прерываний *RST5.5, 6.5* и *7.5*; сброс защелок прерываний *RST7.5* для прерывания фронтом нмпульсов и захвата выхода *SOD*.

Глава 9

ПРОГРАММИРОВАНИЕ МП INTEL 8080/8085

Для программиста микро-ЭВМ, построенные на МП Intel 8080/8085, представлены составом команд, памятью, регистрами общего назначения, счетчиком команд, портами ВВ, индикаторами, указателем стека и стеком. Все эти элементы, за исключением памяти и УВВ, составляют часть МП.

В дальнейшем мы будем использовать состав команд МП Intel 8080/8085, данный в табл. 8.4 и изученный в гл. 8.

На рис. 9.1, *б* приведена часть регистра состояния, а также краткое описание каждого из пяти индикаторов.

Программы будут записаны на ассемблере. Используемый фирмой Intel формат делит каждую линию ассемблера на следующие поля:

МЕТКА КОП ОПЕРАНД КОММЕНТАРИЙ

Элемент в поле метки решающий, он составляет «название линии». Обычно метки будут иметь линии, отмеченные переходом. Поле КОП содержит мнемонику, относящуюся к выполняемой операции. Поле операнда (иногда оно называется аргументом) содержит данные, которые будут обработаны по указанному КОП. Поле комментариев может содержать полезную информацию описания функций рассматриваемых команд.

Пусть имеем линию программы на ассемблере:

DATA MOV *A*, *M*; Ввод данных в аккумулятор. Каждый из элементов этого примера принадлежит одному из полей. Вот их описание:

Рис. 9.1. Регистры, доступные программисту (*а*), и состав регистра состояний (*б*)

1. Поле МЕТКА содержит DATA (данные). Это имя линии, и, вероятно, цель прошедшей (или будущей) команды перехода.

2. Поле КОП: MOV. Это указывает МП, что речь идет о команде передачи данных. Микропроцессор ищет следующее поле (ОПЕРАНД), чтобы определить источник и назначение передачи данных.

3. Поле ОПЕРАНД: *A*, *M*. Аккумулятор *A* является назначением данных, память *M* — их источником. Адрес ячейки памяти указан парой регистров *HL*; в этом случае имеем команду косвенной регистровой адресации.

4. Поле КОММЕНТАРИЙ: *Ввод данных в аккумулятор*. Это то, что появляется, когда операция выполнена. В этом случае данные передаются в аккумулятор из ячейки памяти, указанной парой *HL*. Точка с запятой (;) используется как разделитель между полем операнда и комментариями. Поля КОП и операнда должны быть заполненными, поля метки и комментариев заполняются по необходимости. Некоторые неявные команды требуют заполнения только поля КОП.

Напомним (см. гл. 6), что структурная схема, исходные программы на ассемблере и сам ассемблер оказывают помощь при программировании микропроцессорных систем. Объектная программа в форме листинга на ассемблере является конечным этапом разработки для микро-ЭВМ.

9.1. ПОСЛЕДОВАТЕЛЬНЫЕ ПРОГРАММЫ

Рассмотрим функциональную структурную схему на рис. 9.2, *а*. Она является основой для составления последовательной программы, названной так, потому что в ней нет

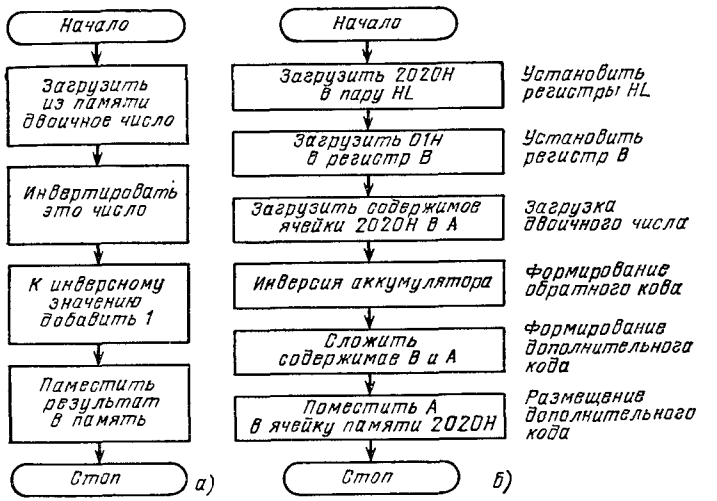


Рис. 9.2. Структурная схема программы формирования дополнительного кода (*а*) и ее развитие (*б*)

ветвей. Эта программа загружает двоичное число, переводит его в дополнительный код и помещает, наконец, его представления в дополнительном коде в память. Вспомним, что такая функциональная структурная схема является общим описанием решения поставленной задачи.

На рис. 9.2, *б* представлена более подробная структурная схема этой же задачи для того, чтобы отразить в ней специфические действия микропроцессора — в данном случае МП Intel 8080/8085. Согласно этой схеме можно записать программу на ассемблере, как это показано в табл. 9.1. Заметим, что каждой команде этой программы соответствует элемент в форме прямоугольной ячейки на структур-

Таблица 9.1. *Программа формирования дополнительного кода двоичного числа на ассемблере*

Метка	КОП	Операнд	Комментарий
	LXI	*H*, 2020H	Установить в паре *HL* 2020H
	MVI	*B*, 01*H*	Установить в регистре *B* 01H
	MOV	*A*, *M*	Передать двоичное число из ячейки памяти 2020H в аккумулятор
	CMA		Инвертировать содержимое аккумулятора
	ADD	*B*	Сложить *B* с *A* для формирования дополнительного кода
	MOV	*M*, *A*	Передать число в дополнительном коде из аккумулятора в ячейку памяти 2020H
	HLT		Стоп МП

ной схеме (рис. 9.2, *б*). Так, например, первая ячейка содержит: *загрузить 2020H в пару HL*, что соответствует команде LXI *H*, 2020H программы на ассемблере в табл. 9.1.

Следующим этапом должен быть перевод программы-источника на ассемблере в его эквивалент на машинном

Таблица 9.2. *Объектная программа формирования дополнительного кода двоичного числа*

Шестнадцатеричные		Метка	КОП	Операнд	Комментарий
адрес	содержимое				
2000	21		LXI	*H*, 2020H	Установить 2020H в паре *HL*
2001	20				
2002	20				
2003	06		MVI	*B*, 01H	Установить 01H в регистре *B*
2004	01				
2005	7E		MOV	*A*, *M*	Передать в аккумулятор двоичное число из памяти 2020H
2006	2F		CMA		Инвертировать аккумулятор
2007	80		ADD		Сложить *B* с *A* для формирования дополнительного кода
2008	77		MOV	*M*, *A*	Передать в память 2020H число аккумулятора в дополнительном коде
2009	76		HLT		Стоп МП

языке. Этот процесс можно выполнить, используя специальную программу вычислительной машины — асссемблер или при необходимости вручную. Состав команд табл. 8.4 может быть использован для учебного перевода. Программа, представленная в табл. 9.2, содержит список адресов программной памяти и шестнадцатеричные машинные коды для каждого из операндов и каждой команды.

Заметим, что программа в табл. 9.2 не содержит элементов в поле метки. Это связано с тем, что метки обычно используются для обозначения назначения команд ветвления. Программы в табл. 9.1, 9.2 являются последовательными, т. е. не содержат ветвлений.

Упражнения

9.1. Список в табл. 9.1 является примером программы _____ (с ветвлением, последовательной).

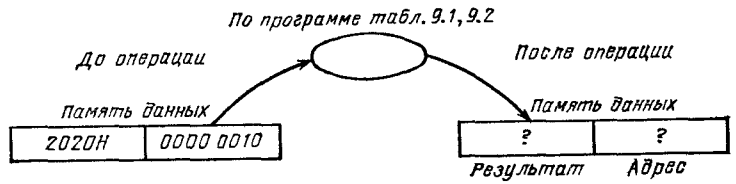


Рис. 9.3. К упражнениям 9.2 и 9.3

9.2. См. рис. 9.3. Результатом в памяти после выполнения программы табл. 9.2 будет _____ (дать 8 бит). Это представление двоичного числа в _____ (инверсном, дополнительном) коде.

9.3. См. рис. 9.3. Адресом ячейки памяти, в которую будет помещен результат после выполнения программы, станет _____ (дать шестнадцатеричное значение).

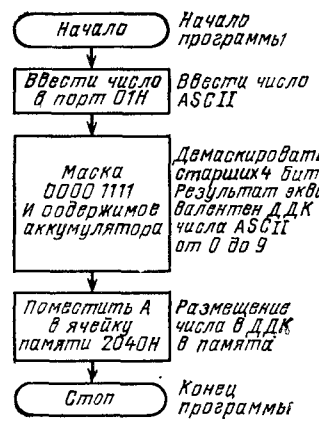


Рис. 9.4. К упражнениям 9.4—9.6

9.4. См. рис. 9.4. Здесь представлена структурная схема сегмента программы, которая вводит число в ASCII и демаскирует 4 _____ (старших, младших) бит. Результат эквивалентен _____ (двоично-десятичному, двоичному) числу.

9.5. См. рис. 9.4. Дополнить вторую и третью линии следующего сегмента программы;

МЕТКА	КОП	ОПЕРАНД	КОММЕНТАРИЙ
Линия 1	IN	01H	Ввести числ ASCII в порт 01H
Линия 2			И содержимого аккумулятора с 0000 1111 (OFH)
Линия 3			Поместить содержимое аккумулятора в ячейку памяти 2040H
Линия 4	HLT		Остановить МП

9.6. Записать шестнадцатеричный машинный код для выполнения сегмента задачи 9.5. Программа начнется в ячейке памяти 2000H в последовательности:

Память (шестнадцатеричный адрес)	Содержимое (шестнадцатеричное)	
2000	BD	} Команда ввода
2001	01	}
2002	—	
2003	—	
2004	—	
—	—	
—	—	

Решения

9.1. Последовательной. **9.2.** 1111 1110; дополнительном. **9.3.** После выполнения пара HL будет всегда указывать на ячейку памяти 2020H, где будет помещен результат. **9.4.** Старших; двоично-десятичному.

9.5.

МЕТКА	КОП	ОПЕРАНД	КОММЕНТАРИЙ
Линия 1	IN	01H	Ввестн число ASCII в порт 01H И
Линия 2	ANI	0FH	И содержимого аккумулятора с 0000 1111 (0FH)
Линия 3	STA	2040H	Поместить содержимое аккумулятора в ячейку памяти 2040H
Линия 4	HLT		Остановить микропроцессор

9.6.

Память (адрес)	Содержимое
2000H	BDH
2001H	01H
2002H	E6H
2003H	0FH
2004H	32H

Продолжение

2005H	40H
2006H	20H
2007H	76H

9.2. ЦИКЛИЧЕСКИЕ ПРОГРАММЫ

Рассмотрим функциональную схему системы, построенной на МП Intel 8080/8085, представленной на рис. 9.5, *а*. Предположим, что двоичный 8-разрядный видеотерминал

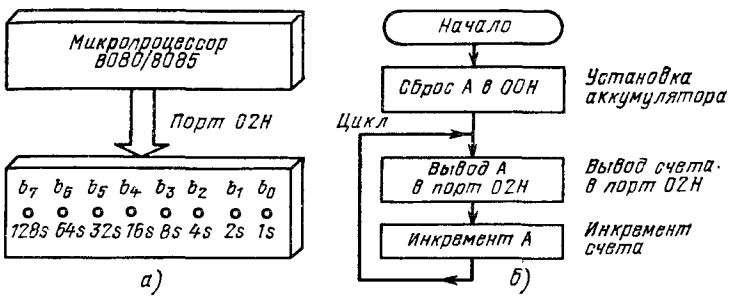


Рис. 9.5. Упрощенная функциональная схема индикацин двоичного счета (*а*) н соответствующая ему более подробная структурная схема программы (*б*)

будет вести прямой счет, начиная с $0000\ 0000_2$. На рис. 9.5, *б* представлена подробная структурная схема, которая решает эту задачу.

Первая прямоугольная ячейка на рис. 9.5, *б* соответствует установке аккумулятора в 00Н, вторая — передаче счета в порт вывода 02Н, третья — инкрементированию счета в аккумуляторе. Эта программа позволит вывести на видеотерминал результат счета с очень большой скоростью.

Таблица 9.3. *Программа двоичного счета на ассемблере*

Метка	КОП	Операнд	Комментарий
LOOP	XRA	*A*	Сбросить аккумулятор
	OUT	02H	Вывести содержимое аккумулятора в порт 02H
	INR	*A*	Инкрементировать аккумулятор
	JMP	LOOP	Переход назад в начало цикла (символический адрес — LOOP)

Программа в табл. 9.3 осуществит двоичный счет, соответствующий структурной схеме. Команда XRA *A* сбрасывает аккумулятор. Команда OUT передает его содержимое в порт вывода 02H. Счет в аккумуляторе повышается на 1 по команде INR*A*. Команда JMP вызывает переход МП (возврат) к команде с символической меткой LOOP. Циклическая программа продолжается в последовательности OUT, INR*A*, JMP.

В действительности программа, приведенная в табл. 9.3, будет считать настолько быстро, что лампы видеотерминала будут казаться горящими постоянно. С целью замедления счета в программу обычно вводится задержка. На рис. 9.6, *а* представлена функциональная схема программы двоичного счета, содержащая временну́ю задержку.

Более подробная структурная схема на рис. 9.6, *б* является развитием предшествующей функциональной схемы. Пять основных ячеек ее относятся к подпрограмме задержки, которая производится повторением цикла задержки 65 535 (FFFFH) раз.

Соответствующая программа на ассемблере приведена в табл. 9.4. Отметим, что каждая прямоугольная ячейка

Таблица 9.4. *Программа счета в цикле с задержкой на ассемблере*

Метка	КОП	Операнд	Комментарий
COUNT	XRA	*A*	Сброс аккумуляторов в 00H
	OUT	02H	Вывести содержимое аккумулятора в порт 02H
	INR	*A*	Инкрементировать содержимое аккумулятора
	MOV	*B*, *A*	Сохранить содержимое аккумулятора в регистре B
DELAY	LXI	H, FFFFH	Загрузить FFFFH в пару HL
	DCX	*H*	Декрементировать пару
	MOV	*A*, *L*	Передать содержимое регистра для проверки в аккумулятор
	ORA	*H*	Выполнить или с содержимым регистра *H* и аккумулятора — установить *Z* в 1, если регистры *H* и *A* оба содержат 0
	JNZ	DELAY	Если *A*=0, переход к циклу задержки; если нет — продолжать последовательно
	MOV	*A*, *B*	Восстановить счет (передать *B* в аккумулятор)
	JMP	COUNT	Переход к COUNT (цикл счета)

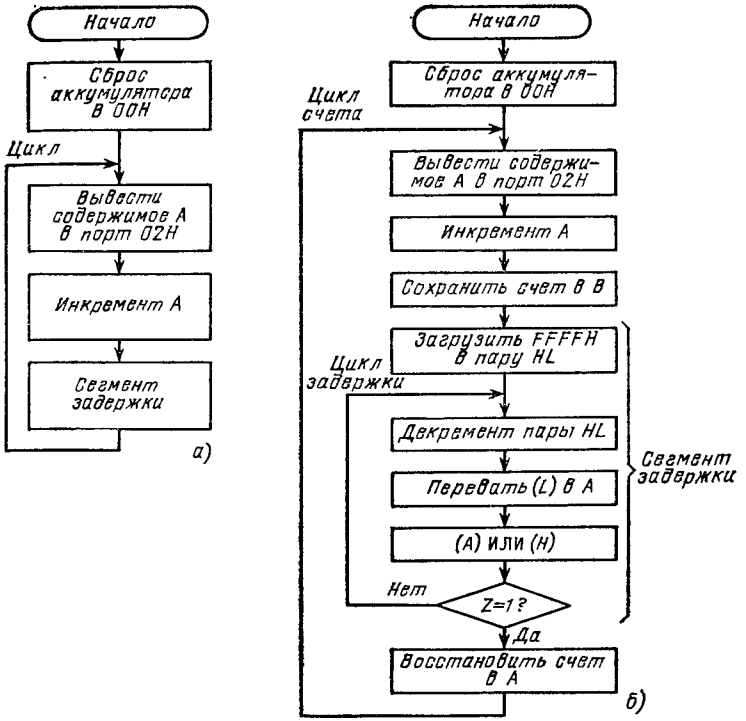


Рис. 9.6. Функциональная схема программы с временно́й задержкой (*а*) и ее развитие (*б*)

на рис. 9.6, *б* эквивалентна одной команде на ассемблере. Цикл временно́й задержки имеет символический адрес или метку DELAY (см. табл. 9.4). Более долгий цикл счета имеет символический адрес COUNT. Программа выполнится в цикле 65 536 раз, проходя через цикл временно́й задержки каждый раз, когда она входит в цикл счета. В МП Intel 8080/8085 состав команд DCX *H*; MOV *A*, *L*; ORA *H* и JNZ занимает 24 мкс. В цикле, повторенном 65 536 раз, общий интервал времени будет: $24 \times 65\,536 = 1\,572\,884$ мкс, т. е. около 1,6 с. Будет объявлено первое число, затем после интервала в 1,6 с — второе и т. д.

Цель команд MOV *A*, *L* и ORA *H* программы в табл. 9.4 не может быть неочевидной — эти две команды находятся здесь для установления индикатора *Z*, когда пара *HL* окончательно декрементирована до 0000H. К сожалению, пре-

дыдущая команда ДЕКРЕМЕНТИРОВАТЬ пару *HL* (команда DCX *H*) никак не воздействует на индикатор. Поэтому были добавлены команды MOV *A*, *L* и ORA *H*.

Подпрограммы временно́й задержки с программами микро-ЭВМ используются совместно. Они основаны на том, что каждая команда процессора требует для выполнения некоторое время. Существуют методы программирования, позволяющие выдавать более долгие интервалы, чем те, что мы только что изучили.

Отметим использование в программе на рис. 9.6, *б* двух циклов: один из них находится внутри другого. Этот способ программирования называется вложением циклов. Цикл задержки называется внутренним циклом.

Упражнения

9.7. Программы в табл. 9.3 и 9.4 идентичны, за исключением того, что последняя содержит подпрограмму _____.

9.8. См. табл. 9.3. Последней командой программы является JMP. Куда перейдет программа после этой команды?

9.9. См. табл. 9.4. Записать пять команд, формирующих подпрограмму временно́й задержки.

9.10. См. рис. 9.6, *б*. Цикл временно́й задержки, представленный на структурной схеме, называется программистами _____ (бесконечным, вложенным).

9.11. См. табл. 9.4. Первый раз, когда МП встретит команду JNZ, индикатор _____ (сброшен в 0, установлен в 1). Выполнив ее, МП перейдет к выполнению следующей команды.

9.12. См. табл. 9.4 Записать необходимый шестнадцатеричный машинный код для выполнения программы, начинающейся с адреса 2000H в памяти.

9.13. См. табл. 9.4. Записать шестнадцатеричный машинный код, нужный для выполнения программы, начинающейся в памяти с адреса 2000H:

Память	Содержимое
2000	AF Сброс аккумулятора
2001	—
2002	—
—	—
—	—

Решения

9.7. Временно́й задержки. **9.8.** Команда JMP LOOP побудит МП перейти к программе по символическому адресу LOOP, в этом случае к

команде OUT. **9.9.** LXI *H*; DCX *H*; MOV *A*, *L*; ORA *H* и JNZ. **9.10.** Вложенным. **9.11.** (*Z*)=0; DCX *H* (символнческий адрес DELAY). **9.12.** Установлен в 1; MOV *A*, *B*. **9.13.** Обратиться к табл. 8.4.

9.3. МАТЕМАТИЧЕСКИЕ ПРОГРАММЫ

Пусть требуется сложить два десятичных числа 1 110 527 и 192 514 в системе, построенной на МП Intel 8080/8085. Проведем сложение обычным методом после преобразования десятичных чисел в шестнадцатеричные:

	1 11	Переносы
1110527_{10}	+ 10F1FF	Первое число
192514_{10}	02F002	Второе число
	13E201H	Сумма

Таким образом, получаем 10 F1FF+02F002=13E201H.

Аккумулятор МП Intel 8080/8085 может выдать только 8 бит (1 байт) данных; процессор мог бы решить эту задачу, если бы программист разделил шестнадцатеричные числа на однобайтовые группы, а каждая из этих групп стала бы слагаемым. Процесс сложения выполняется по следующей схеме:

Старшие байты	Средние байты	Младшие байты	
1←	1←	1	
+ 10	+ F1	+ FF	Первое число
02	F0	02	Второе число
13	E2	01H	Сумма

Сначала складываются младшие байты (FFH+02H==01H плюс перенос), затем средние и предыдущий перенос (1+F1H+F0H=E2H плюс перенос), наконец, старшие и предыдущий перенос (1+10H+02H=13H). Заметим, что в двух последних операциях сложения при вычислении окончательной суммы следует учитывать переносы из младших байтов. Микропроцессоры Intel 8080/8085 снабжены специальными командами СЛОЖИТЬ с переносом для решения подобных задач.

Последовательность действий для выполнения сложения 24 бит (3 байт) могла бы быть следующей:

1. Сложить младший байт первого числа с МБ второго.
2. Записать частичную сумму младших байтов.
3. Сложить перенос предыдущего сложения, промежуточный байт первого числа и промежуточный байт второго числа.

Таблица 9.5. *Программа сложения 24-разрядных двоичных чисел на ассемблере*

Метка	КОП	Операнд	Комментарий
	LXI	*H*, 2020*H*	Установить пару *HL* (которая будет использована как указатель) в 2020H
	MOV	*A*, *M*	Загрузить младший байт первого числа в ячейке памяти LOC 2020H в аккумулятор
	INX	*H*	Инкрементировать указатель *HL* до 2021H
	ADD	*M*	Сложить содержимое ячейки памяти LOC 2021H с содержимым аккумулятора (сложить младший байт)
	STA	2026*H*	Поместнть частичную сумму младших байтов в ячейку памяти LOC 2026H
	INX	*H*	Инкрементировать указатель *HL* до 2022H
	MOV	*A*, *M*	Загрузить промежуточный байт чнсла в ячейке памятн LOC 2022H в аккумулятор
	INX	*H*	Инкрементировать указатель *HL* до 2033H
	ADC	*M*	Сложить содержимое ячейки памяти LOC 2023H, аккумулятор н перенос (сложить байты с переносом)
	STA	2027*H*	Поместить частичную промежуточную сумму в LOC 2027H
	INX	*H*	Инкрементировать указатель *HL* до 2024H
	MOV	*A*, *M*	Загрузить старший байт первого числа в памяти LOC 2024H в аккумулятор
	INX	*H*	Инкрементировать указатель *HL* до 2025H
	ADC	*M*	Сложить содержимое памятн LOC 2025H, аккумулятор и перенос (сложить старшие байты с переносом)
	STA	2028*H*	Поместить частичную сумму старшнх байтов в ячейку памяти LOC 2026H
	HLT		Остановить МП

4. Записать вторую частичную сумму.

5. Сложить перенос предыдущего сложения, СБ первого числа и старший байт второго числа.

6. Записать частичную сумму СБ.

Предположим для данного случая, что адреса памяти соответствуют воображаемой памяти данных, представленной на рис. 9.7. Мы найдем здесь также позиции 24-разрядного результата в памяти. С использованием этих данных адресов была подготовлена программа на ассемблере, по-

Рис. 9.7. Воображаемая часть программы сложения 24-разрядных чисел

казанная в табл. 9.5 и осуществляющая выполнение рассматриваемой операции. Заметим, что первой командой сложения ADD *M* является операция сложения без переноса. Две следующие, однако, ADC *M* являются операциями сложения с переносом.

Сложение больших чисел требует введения сложения нескольких байт. В приведенном нами примере перерабатываются числа длиной 3 байт — говорят, что выполняется сложение с тройной точностью.

Упражнения

9.14. См. табл. 9.5. Записанная программа является примером сложения с _____ (двойной, тройной) точностью.

9.15. Пусть надо сложить 10F1FFH и 02F002H (см. § 9.3). Записать содержимое памяти 2020—2025H в памяти данных (см. рис. 9.7) до выполнения программы сложения с тройной точностью.

9.16. См. рис. 9.7. Записать содержимое ячеек памяти 2026H, 2027H и 2028H после выполнения сложения чисел 10F1FFH и 02F002H.

9.17. См. табл. 9.5. Пара *HL* используется как регистр _____ (общего назначения, указатель).

9.18. См. табл. 9.5. Команда ADD *M* указывает на операцию сложения _____ (с переносом, без переноса).

Таблица 9.6. *Программа сложения 32-разрядных чисел*

Метка	КОП	Операнд	Комментарий
	MVI	C, 04H	Установнть в регистре *C* число слагаемых байт
	LXI	*D*, 2020*H*	Установить в указателе *DE* адрес старшего байта для сложення
	LXI	*H*, 2030*H*	Установить в указателе *HL* адрес младшего байта второго числа
	XRA	*A*	Сброс индикатора переноса
LOOP	LDAX	*D*	Загрузить байт первого числа в аккумулятор
	ADC	*M*	Сложить байт второго числа, аккумулятор и бит переноса
	STAX	*D*	Поместнть частичную сумму в ячейку памяти LOC, указанную парой *DE*
	DCR	*C*	Декрементировать регистр *C*
	JZ	*END*	Если $Z=1$, перейти по снмволическому адресу, помеченному *END*
	INX	*D*	Инкрементировать указатель *DE*
	INX	*H*	Инкрементировать указатель *HL*
	JMP	LOOP	Безусловный переход по снмволическому адресу, помеченному LOOP
END	HLT		Остановить МП

9.19. См. табл. 9.5. По команде STA 2027H содержимое регистра _____ (*A*, *B*) помещается в память по адресу _____.

9.20. См. рис. 9.8. Где в памяти помещается первое число для сложения?

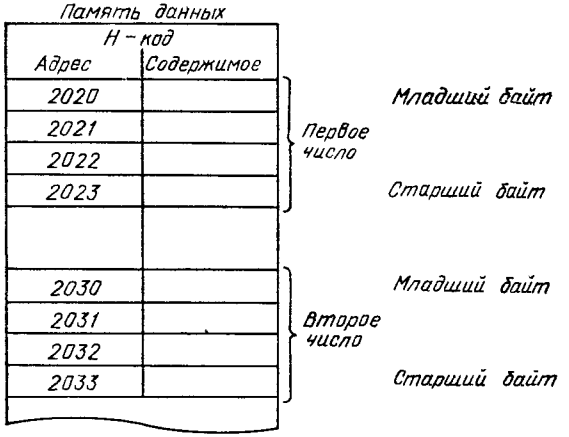


Рис. 9.8. К упражненням 9.20—9.25

9.21. См. рис. 9.8. Где помещается сумма двух чисел после выполнения сложения?

9.22. Как нужно изменить команду MVI *C* в табл. 9.6 для сложения 24 бит вместо 32?

9.23. См. рис. 9.8. Записать содержимое следующих ячеек памяти для сложения 12F0C3FFH с 01B4 D503H: 2020, 2021, 2022, 2023, 2030, 2031, 2032, 2033.

9.24. См. рис. 9.8. Чему равна сумма 12F0C3FFH+ +01B4D503H? Дать воображаемую память (адреса и содержимое), относящуюся к размещению суммы до выполнения сложения.

9.25. См. табл. 9.6. Если индикатор $(Z)=0$, команда _____ будет выполняться после команды JZ.

Решения

9.14. Тройной. **9.15.** До сложения 10F1FFH+02F002H содержимое памяти данных следующее (H-код):

Адрес	Содержнмое
2020	FF
2021	02
2022	F1
2023	0
2024	10
2025	02

9.16. В H-коде

Адрес	Содержимое	
2026	01	младшнй байт суммы
2027	E2	промежуточный байт суммы
2028	13	старшнй байт суммы

9.17. Указатель. **9.18.** Без переноса. **9.19.** *A* (аккумулятор); 2027H. **9.20.** Первое число (4 байт) помещается в памяти по адресам 2020H—2023H. **9.21.** В памяти, указанной парой *DE*, т. е. по адресам 2020H — 2023H. **9.22.** MVI *C*, 03H. Операнд команды MVI *C* (03H в данном случае) должен представлять число байтов слагаемых. **9.23.** В H-коде:

Память	2010	2021	2022	2023	2030	2031	2032	2033
Данные	FF	C3	F0	12	03	D5	B4	01

9.24. 12F0C3FFH+01B4D503H=14A59902H. Воображаемая память приведена на рис. 9.8.

Программа в H-коде:

Память	Содержимое
2020	02 младший байт суммы
2021	99
2022	A5
2023	14

9.25. INX *D*

Дополнительные упражнения к гл. 9

9.26. Для программиста микро-ЭВМ, построенная на МП Intel 8080/8085, состоит из памяти, портов ВВ, индикатора, стека, указателя стека, счетчика _____, состава _____ и регистров общего назначения.

9.27. Записать четыре поля одной типовой строки на ассемблере.

9.28. Операнд поля иногда еще называют _____.

9.29. Команда на ассемблере содержит обычно элементы полной _____ и _____ , тогда как метка и комментарии вводятся по мере необходимости.

9.30. См. рис. 9.9. Эта программа загружает числа ASCII, взятие в памяти _____и_____ , маскирует 4 бит _____ (старших, младших), складывает числа, производит десятеричную коррекцию суммы и помещает _____ (двоичную, двоично-десятичную) сумму в память 2032H.

Память данных

H - код	
Адрес	Содержимое
2030	B5
2031	B9
2032	

Представление ДДК 5 в ASCII

Представление ДДК 9 в ASCII

Сумма ДДК

Рис. 9.9. К упражнениям 9 30—9.41

Задачи 9.29—9.31 читателю следует решать, обращаясь к рис. 9.10.

9.31. Какая команда соответствует кадру *1*?

9.32. Какая команда соответствует кадру *2*?

9.33. Какая команда соответствует кадру *3*?

9.34. Какая команда соответствует кадру *4*?

9.35. Какая команда соответствует кадру *5*?

9.36. Какая команда соответствует кадру *6*?

9.37. Какая команда соответствует кадру *7*?

9.38. Какая команда соответствует кадру *11*?

9.39. Какая команда соответствует кадру *12*?

9.40. См. рис. 9.10. Записать программу на ассемблере, решающую эту задачу.

9.41. См. рис. 9.10. После выполнения программы двоично-десятичная сумма_____ будет помещена в память _____.

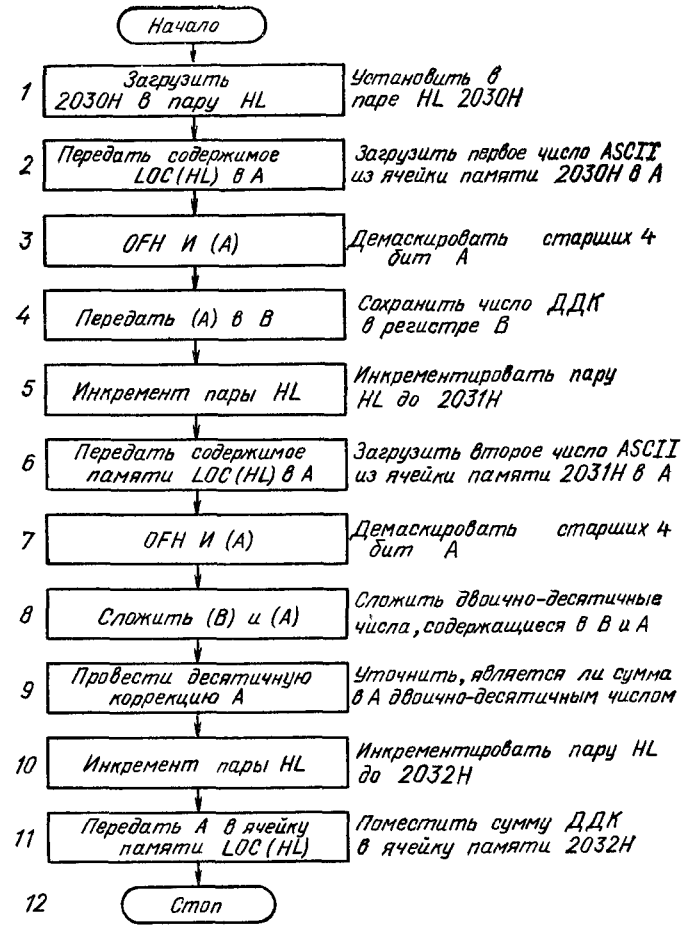


Рис. 9.10 К упражнениям 9.31—9.41

Решения

9.26. Команд; команд. **9.27.** Метка, КОП, операнд, комментарий. **9.28.** Аргументом. **9.29.** КОП, операнда. **9.30.** 2030H, 2031H; старших; двоично-десятичную. **9.31.** LXI *H*, 2030H. **9.32.** MOV *A*, *M*. **9.33.** ANI OFH. **9.34.** MOV *B*, *A*. **9.35.** INX *H*. **9.36.** ADD *B*. **9.37.** DAA. **9.38.** MOV *M*, *A*. **9.39.** HLT.

9.40.

КОП	ОПЕРАНД	Комментарий
LXI	*H*, 2030H	Установить регистры
MOV	*A*, *M*	Загрузнть первое число в *A*
ANI	OFH	Демаскировать старших 4 бит *A*
MOV	*B*, *A*	Сохраннть двоично десятичное в *B*
INX	*H*	Инкрементировать регистры до 2031H
MOV	*A*, *M*	Загрузить второе число в *A*
ANI	OFH	Демаскировать старших 4 бит *A*
ADD	*B*	Сложить двоично-десятичное в *A* и *B*
DAA		Уточнить сумму в старших 4 бнт иа равенство двоично-десятнчиому
INX	*H*	Инкрементировать пару до 2032H
MOV	*M*, *A*	Поместнть двоично-десятичную сумму в ячейку памяти по адресу 2032H
HLT		Остановнть МП

9.41. 0001 0100 ДДК; 2032H.

СПИСОК ЛИТЕРАТУРЫ

1. **Краткий** терминологический словарь по мнкропроцессорной технике/А. Ф. Барышев, Е. Е. Дудников, Й. Лоозе и др. М.: Международный центр НТИ. Международный НИИ проблем управления, 1984.

2. **Фудзисава Т., Касами Т.** Математика для радиоинженеров. Теория дискретных структур/Пер. с япон. А. В. Кузнецова. М.: Радио и связь, 1984.

3. **Вуд А.** Мнкропроцессоры в вопросах и ответах: Пер. с англ./Под ред. Д. А. Поспелова. М.: Энергоатомиздат, 1985. С. 184.

4. **Уокерли Дж.** Архитектура и программнрование микро-ЭВМ. В 2 книгах: Пер. с англ./Под ред. А. Г. Филиппова. М.: Мир, 1984. Кн. 1; Кн. 2.

5. **Ч. Гилмор.** Введение в микропроцессорную технику/Пер. с англ. В. М. Кнсельннкова, В. К. Потоцкого, Л. В. Шабанова. М.: Мир, 1984.

6. **Балашов Е. П., Григорьев В. Л., Петров Г. А.** Мнкро- и мини-ЭВМ. Л.: Энергоатомиздат, 1984.

7. **Горбунов В. Л., Панфилов Д. И.** Микропроцессоры. Лабораторный практнкум. М.: Высшая школа, 1984.

8. **Григорьев В. Л.** Программное обеспечение микропроцессорных систем. М.: Энергоатомиздат, 1983.

ОГЛАВЛЕНИЕ

Производственное издание

Токхайм Роджер Л.

МИКРОПРОЦЕССОРЫ. КУРС И УПРАЖНЕНИЯ

Редактор издательства *А. А. Устинов*
Художественный редактор *А. Т. Кирьянов*
Технический редактор *Г. В. Преображенская*
Корректор *Л. С. Тимохова*

ИБ № 1908

Сдано в набор 30.01.87. Подписано в печать 05.02.88. Формат 84×108¹/₃₂. Бумага типографская № 2. Гарнитура литературная. Печать высокая. Усл печ л. 17,64. Усл. кр.-отт. 17,64. Уч.-изд. л. 18,3. Доп. тираж 80 000 экз. Заказ 784. Цена 1 р. 50 к.

Энергоатомиздат, 113114, Москва, М-114, Шлюзовая наб., 10

Владимирская типография Союзполиграфпрома при Госкомиздате СССР 600000, г. Владимир, Октябрьский проспект, д. 7