Таблица 6.8

Последовательность команд обработки прерывания

Банк	Адрес ПЗУ (страница)	Слово	Команда	Комментарии
0	6	82	SRC 4	Содержимое индексных регистров *IR* 8, 9 передается в качестве адреса в ПЗУ и ЗУПВ и загружается в регистр команды SRC
0	6	83	INC 9	Возникновение прерывания
0	6	84	(JMS 0)	Подтверждение прерывания. Адрес 6 84 сохраняется в стеке, команда по адресу 6 84 игнорируется (переход по адресу 003)
0	6	84	(3)	
0	0	3	SB 1	Выбор банка индексных регистров с номером 1
0	0	4	XCH 7	(ACC)→IR_7*—Содержимое аккумулятора сохраняется в регистре IR_7
0	0	5	LCR	(CR)→ACC
0	0	6	RAL	(CY)→ACC_0, ACC_0→ACC_1 . . . ACC_3→CY
0	0	7	XCH 6	(ACC)→IR_6^*, *CY* и содержимое регистра команд CR сохраняется в IR_6^*
0	0	8		
		9		Подпрограмма распознавания и обслуживания прерывания
0	P	*n*	XCH 6	(IR6*)→ACC
		n+1	RAR	ACC_0→CY. Восстановление CY
		n+2	DCL	ACC_0→CR_0, ACC_1→CR_1, ACC_2→CR_2. Восстановление *CR*
		n+3	XCH 7	IR_7^*→ACC
		n+4	BBS	Адрес 684 загружается в программный счетчик, содержимое индексного регистра по команде SRC передается на выход
0	6	84	WRM	Возврат к программе

СПИСОК ЛИТЕРАТУРЫ

1. **MCS-4** Microcomputer Set, Users Manual, Intel Corp., Santa Clara, Ca., 1974.
2. **MCS-40** Users Manual For Logic Designers, Intel Corp., Santa Clara, Ca., 1975.
3. **Brewer M.** The Designers Guide to Programmed Logic, for PLS 400 Systems, Pro-log Corp., Monterey, Cal., 1973.
4. **Collins D. C., Garen E. R., Lemas M.** Software Development for Microcomputers, Integrated Computer Systems, Culver City, Ca., 1975.

Глава 7

МИКРОПРОЦЕССОРНЫЕ НАБОРЫ 8008/8080 И МСОМ-8

В гл. 7 описываются микропроцессорные наборы 8008/8080 фирмы Intel и МСОМ-8 фирмы Nippon, совместимые программно и по логическим уровням. Рассмотрены базовая микросхема процессора и ее соединения в микропроцессорных системах. Процессор 8080 имеет 8-битовую шину данных и 16-битовую адресную шину. Описаны четыре вида адресации к памяти: прямая, косвенная (через регистры), непосредственная и через стек. Набор команд состоит из команд пересылок, условных переходов, операций со стеком, команд вызова подпрограмм, команд арифметических, логических и ввода—вывода. Приведены примеры программирования, включая описание отдельных подпрограмм и организацию прерывания. Рассмотрены приоритетная система прерываний и линии ввода—вывода.

Материал[1] этой главы представляет собой частично переработанные публикации фирмы Intel Corporation. Всю ответственность за представленный в таком виде материал несет автор.

7.1. МИКРОПРОЦЕССОР 8080

Структурная схема микропроцессора представлена на рис. 7.1, а расположение выводов — на рис. 7.2. Назначение выводов, показанных на рис. 7.1, приведено ниже[2].

Φ_1, Φ_2 — входы синхросигналов генератора;

RESET (*сброс*) — вход, по которому осуществляется сброс содержимого программного счетчика, в результате чего выполнение программы начинается с нулевой ячейки памяти;

SYNC (*синхронизация*) — выход синхросигнала, указывающего начало каждого машинного цикла;

READY (*готовность*) — вход сигнала, поступающего на процессор от внешних устройств и указывающего о готовности данных к вводу. Используется для синхронизации обмена между центральным процессором (ЦП) и внешними устройствами. При отсутствии сигнала после обращения к внешнему устройству ЦП переходит в состояние ожидания;

WAIT (*ожидание*) — выход сигнала подтверждения того, что ЦП находится в состоянии ожидания;

HOLD (*захват*) — вход сигнала запроса внешней адресной шины и шины данных ЦП. Запрос удовлетворяется после завершения очередного цикла обращения к памяти. После удовлетворения запроса шины переводятся в высокоимпедансное состояние, а сам процессор завершает выполнение текущего цикла;

HLDA (*HOLD ACKNOWLEDGE* — подтверждение захвата) выход подтверждения о получении сигнала запроса шин. После

[1]) Программы, рисунки и таблицы заимствованы из проспекта фирмы Intel Corporation. Все права сохранены.

[2]) Добавлено при переводе.

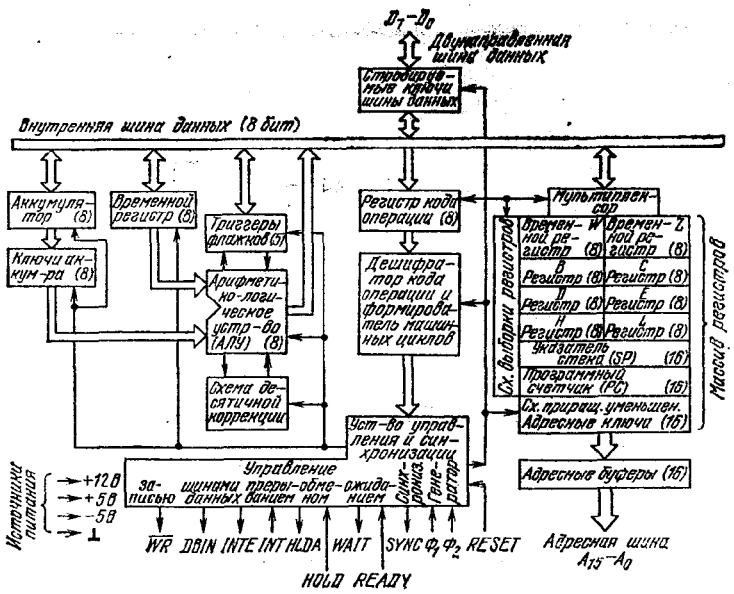


Рис. 7.1. Структурная схема ЦП 8080

окончания текущего машинного цикла ЦП приостанавливает выполнение своих операций, пока действует сигнал запроса. Сигналы *захват шин* и *подтверждение захвата* можно использовать для организации прямого доступа к памяти, а также в многомикропроцессорных системах;

INT (INTERRUPT — запрос прерывания) — входной сигнал от внешнего устройства на прерывание работы ЦП и обслуживание внешнего устройства. Запрос воспринимается в конце цикла выполнения текущей команды. Если триггер разрешения прерывания установлен, то ЦП переходит к выполнению программы прерывания;

INTE (INTERRUPT ENABLE — разрешение прерывания) — выходной сигнал, указывающий, что триггер разрешения прерывания установлен;

DBIN (DATA BUS INPUT — прием с шины данных) — выход сигнала, указывающего, что шина данных находится в режиме приема, т. е. ЦП ожидает поступления данных по шине;

$\overline{WR}$ *(WRITE* — запись) — выход сигнала, использующегося для записи данных в память или для управления вводом — выводом. Указывает, что на шине данных находятся данные, поступающие из ЦП.

Общее описание. Микропроцессор 8080 — полный 8-битовый параллельный центральный процессор (ЦП), предназначенный для применения в цифровых вычислительных системах общего назначения. Он выполнен на одном кристалле БИС с высокой степенью интеграции по *n*-канальной МОП-технологии фирмы Intel, что позволяет получить значительно более высокие характеристики, чем у аналогичных микропроцессоров с двумикросекундным командным циклом. Полная микропроцессорная система (микроЭВМ) организуется путем соединения микросхемы 8080 с микросхемами (портами) ввода — вывода (до 256 входных и 256 выходных портов) и микросхемами полупроводниковой памяти любого типа и быстродействия.

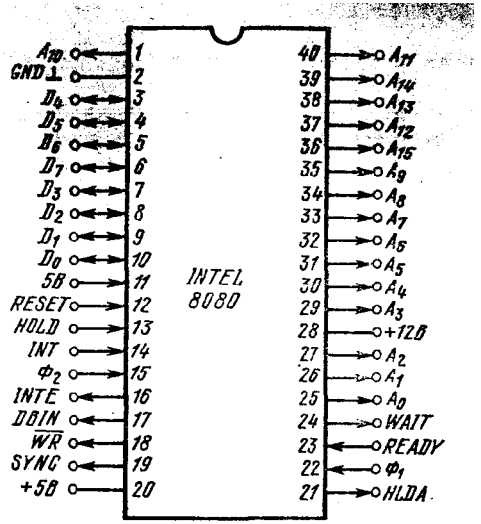


Рис. 7.2. Расположение выводов ЦП 8080

Хотя микропроцессор 8080 значительно превосходит по характеристикам существующие микропроцессоры, он проектировался программно совместимым на уровне машинных кодов с микропроцессором 8008 фирмы Intel. Как и 8008, ЦП 8080 содержит шесть 8-битовых регистров данных, 8-битовый аккумулятор, четыре 8-битовых регистра для временного хранения, пять анализируемых триггеров-флажков и 8-битовое арифметико-логическое устройство (АЛУ) параллельного действия. Микропроцессор 8080 может также работать в десятичной арифметике и использовать 16-битовые арифметические команды с непосредственной адресацией, значительно упрощающие вычисление адреса памяти и обеспечивающие быстрое выполнение арифметических операций.

ЦП 8080 имеет стековую архитектуру, где любая часть внешней памяти может быть использована как стек (вошедший первым выходит последним) для хранения и восстановления содержимого аккумулятора, триггеров флажков или любого регистра данных.

Для указания местоположения стека в памяти имеется указатель стека. Одно из преимуществ стековой архитектуры состоит в возможности многоуровневых (вложенных) прерываний, так как состояние системы может быть легко сохранено при возникновении прерывания и восстановлено после его обработки. Другим преимуществом являются практически неограниченные возможности обращения к подпрограммам из подпрограмм (вложение подпрограмм).

Микропроцессор создавался с учетом простоты разработки систем. Отдельные 16-битовая шина адреса и 8-битовая двунаправленная шина данных допускают непосредственное подсоединение к микросхемам памяти и портам ввода—вывода. Сигналы управления не требуют декодирования и формируются самим процессором.

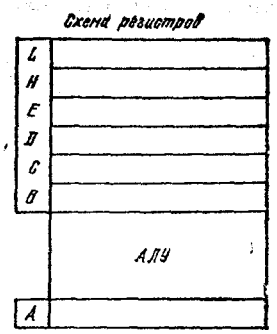


Рис. 7.3. Схема регистров ЦП 8080

Все шины, включая шину управления, совместимы по уровням с ИС ТТЛ.

Для программиста микро-ЭВМ состоит из следующих частей:

1. Семи рабочих регистров, используемых при выполнении всех операций над данными и для хранения адресов ячеек памяти.

2. Памяти, в которой хранятся команды программы и данные и к которой необходимо адресоваться для получения требуемой информации.

3. Программного счетчика, указывающего адрес следующей команды.

4. Указателя стека — регистра, позволяющего использовать различные области памяти в качестве стека. Указатель стека упрощает выполнение подпрограмм и обслуживание прерываний, как показано ниже.

5. Схемы ввода—вывода, осуществляющей связь между ЦП и внешними устройствами.

Описание регистров. ЦП 8080 имеет программно-доступные 8-битовый аккумулятор и шесть дополнительных 8-битовых рабочих регистров, образующих сверхоперативную память (СОЗУ) (рис. 7.3.) Все семь регистров нумеруются и вызываются по цифрам 0, 1, 2, 3, 4, 5 и 7. Кроме того, как обычно, к ним можно обращаться по буквам B, C, D, E, H, L и A (аккумулятор) соответственно.

Некоторые операции ЦП 8080 выполняются с парами регистров, обозначаемыми буквами B, D, H и PSW (слово состояния программы). Обращение к парам регистров осуществляется при работе с 16-битовыми словами следующим образом:

Вызываемая пара	Регистры
B	B, C
D	D, E
H	H, L

Обычно пара регистров содержит адрес ЗУПВ или ПЗУ.

Память. ЦП 8080 может работать с ПЗУ, ППЗУ (программируемое ПЗУ), ЗУПВ. Программно можно считывать данные из любого типа памяти, но записывать только в ЗУПВ. Для программиста память является последовательностью байтов, представляемых двумя шестнадцатеричными цифрами. Можно адресоваться к 65535 байтам памяти, причем каждый байт адресуется числом от 0 до $65535_{10} = FFFF_{16}$ — наибольшего числа, которое может быть записано в 16 битах.

Программный счетчик является программно доступным регистром, содержимое которого указывает адрес следующей команды, подлежащей выполнению.

Указатель стека. Стек — это область памяти, выделяемая программистом для хранения адресов и данных при стековых операциях. Стековые операции выполняются рядом команд ЦП 8080 и облегчают выполнение подпрограмм и обслуживание прерываний. Программист определяет адреса, с которыми оперируют стековые операции, с помощью программно доступного 16-битового регистра, называемого указателем стека.

Ввод — вывод. Для ЦП 8080 «внешний мир» состоит из 256 устройств ввода и 256 устройств вывода. Каждое устройство обменивается с ЦП путем посылки байта информации в аккумулятор или приема из него. Каждое устройство снабжается программно неизменяемым номером от 0 до 255.

Синхронизация. Формат команд ЦП 8080 содержит от одного до трех байт. Каждая команда требует для выборки и выполнения от одного до пяти машинных циклов. Машинные циклы именуются M_1, M_2, M_3, M_4 и M_5. Каждый машинный цикл включает от трех до пяти тактов: T_1, T_2, T_3, T_4, T_5. Каждый такт длится в течение одного периода синхросигнала (длительность такта при частоте 2 МГц 0,5 мкс). Имеются три состояния WAIT (ОЖИДАНИЕ), HOLD (ЗАХВАТ), HALT (ОСТАНОВ), которые могут длиться неограниченное число тактов. Цикл M_1 — это всегда цикл выборки команды, и длится он 4 или 5 тактов. Циклы M_2, M_3, M_4 и M_5 обычно состоят из 3 тактов каждый. Для лучшего понимания работы ЦП 8080 обратимся к рис. 7.4.

В течение такта T_1 содержимое программного счетчика выдается на адресную шину, на выходе *SYNC* устанавливается высокий потенциал, а на шину данных поступает информация о состоянии, относящаяся к выполняемому циклу. За тактом T_1 всегда следует такт T_2, в течение которого проверяется наличие сигналов подтверждения состояний ОСТАНОВ, ГОТОВНОСТЬ и ЗАХВАТ. Если на входе *READY* имеется сигнал готовности (высокий уровень), то ЦП переходит к такту T_3, в противном случае — в состояние ОЖИДАНИЕ (такт T_w) и находится в нем до тех пор, пока не появится сигнал готовности.

Таким образом, сигнал готовности позволяет синхронизовать ЦП с памятью с любым временем доступа или с любым внешним устройством. Более того, управляя соответствующим образом входом готовности, можно обеспечить пошаговое выполнение программы.

В течение такта T_3 цикла M_1 по шине данных поступают данные из памяти в регистр кода операции, как показано на рис. 7.4. Дешифратор кода операции и устройство управления формируют сигналы управления и синхронизации для внутренних пересылок данных, а также соответствующие дешифрируемой команде машинные циклы.

После такта T_4, если он завершающий, или такта T_5 ЦП переходит к такту T_1 цикла M_2, если для выполнения команды требуется более одного цикла, или к выполнению цикла M_1 в противном слу-

чае. Рассмотренная последовательность повторяется требуемое для выполнения команды число раз.

Только в последнем такте последнего машинного цикла проверяется наличие запроса прерывания на входе *INT*, и если таковой имеется, то ЦП переходит к выполнению цикла M_1 специального вида, в течение которого содержимое программного счетчика не увеличивается на 1, а выдается сигнал подтверждения прерывания на выход *INTE*. В этом случае из устройства, вызвавшего прерывание в ЦП, должна быть послана одна из восьми возможных команд, организующих прерывание.

В общем для выполнения команд требуется от 4 (для команд без обращения к памяти типа арифметических операций с аккумулятором или с регистрами) до 18 (для наиболее сложных операций — обмен содержимым регистров *H*, *L* и двух верхних ячеек стека) машинных тактов. При максимальной частоте синхросигнала 2 МГц это означает, что для выполнения команды требуется от 2 до 9 мкс. При выполнении команды HALT (ОСТАНОВ) ЦП переходит в состояние ОЖИДАНИЕ и остается в нем до поступления сигнала прерывания.

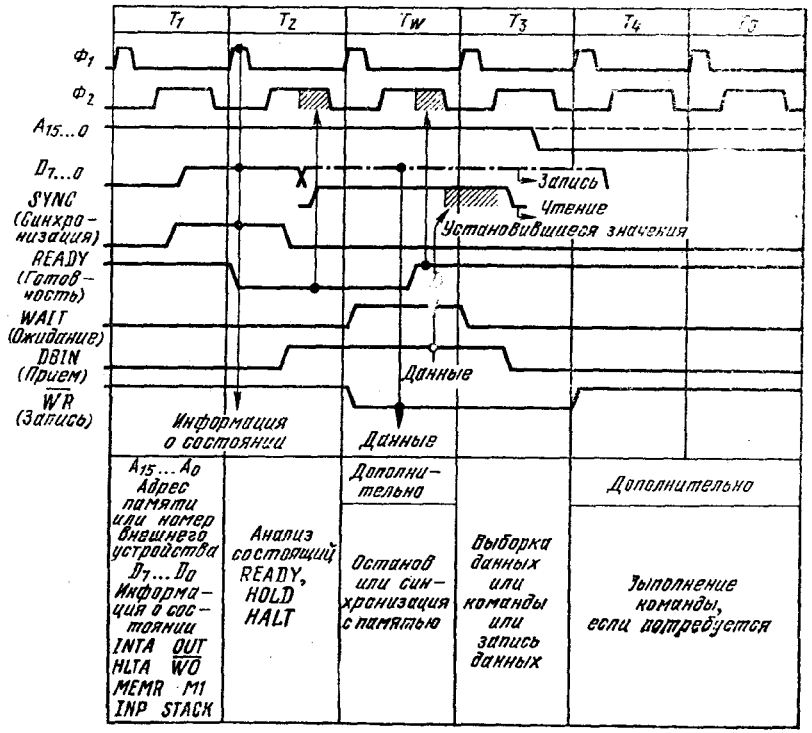


Рис. 7.4. Временные диаграммы функционирования ЦП 8080

Информация о состоянии. Как уже указывалось, для выполнения команд ЦП 8080 требуется от одного до пяти машинных циклов. В начале каждого цикла на шину данных выдается байт состояния (в течение действия сигнала *SYNC*). Назначение каждого разряда байта состояния расшифровывается в приведенной ниже таблице.

Описание информации о состоянии

Обозначение	Разряд шины данных	Описание
INTA [1] (INTERRUPT ACKNOWLEDGE — ПОДТВЕРЖДЕНИЕ ПРЕРЫВАНИЯ)	D_0	Сигнал подтверждения прерывания. Может быть использован для стробирования сигнала, организующего прерывание во время действия сигнала на выводе *DBIN*
$\overline{WO}$ (WRITE-OUTPUT — ЗАПИСЬ—ВЫВОД)	D_1	Указывает, что в данном машинном цикле будет производиться запись в память или вывод ($\overline{WO}=0$). В противном случае будет выполняться ввод или чтение из памяти
STACK (СТЕК)	D_2	Указывает, что на шине адреса находится содержимое указателя стека, адресующегося к верхней ячейке стека
HLTA (HALT ACKNOWLEDGE — ПОДТВЕРЖДЕНИЕ ОСТАНОВА)	D_3	Подтверждение выполнения процессором операции останова (HALT)
OUT (OUTPUT CYCLE — ВЫВОД)	D_4	Указывает, что на адресной шине находится адрес устройства вывода, причем шина данных будет содержать выводимую информацию, когда на выходе $\overline{WR}$ появится сигнал
M1	D_5	Указывает о нахождении ЦП в цикле извлечения первого байта команды
INP[1] (INPUT CYCLE — ВВОД)	D_6	Сигнализирует, что на адресной шине находится адрес устройства ввода, а ввод будет производиться на шину данных по сигналу ПРИЕМ на выводе *DBIN*
MEMR[1] (MEMORY READ — ЧТЕНИЕ)	D_7	Указывает, что шина данных будет использована (в текущем цикле) для чтения данных из памяти

[1] Эти три состояния могут использоваться для контроля прохождения информации по шине данных в ЦП 8080.

Так как информация о состоянии выдается из ЦП на шину данных в течение коротких интервалов времени, то необходимо позаботиться о ее хранении. Например, на рис. 7.5 показано использование для хранения информации о состоянии микросхемы ввода — вывода 8212. В типичной структуре системы 8080, состоящей из микросхем ЦП, ЗУПВ, ПЗУ и ввода-вывода, показанной на рис. 7.6, для хранения байта состояния и его дешифрации используется спе-

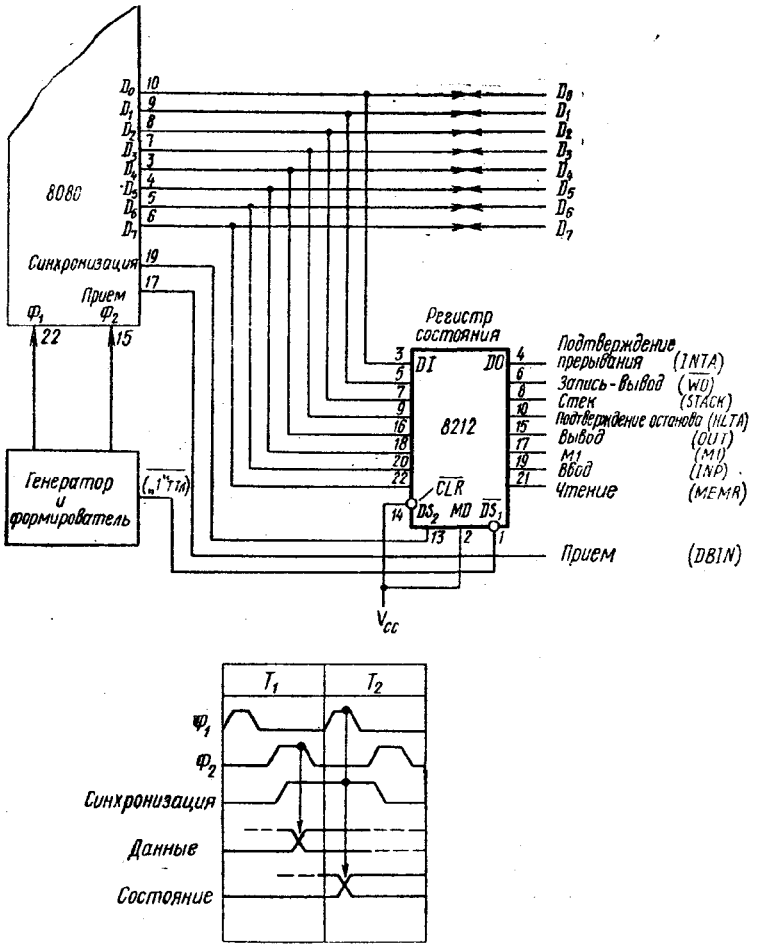


Рис. 7.5. Схема соединения ЦП 8080 с внешним регистром состояния

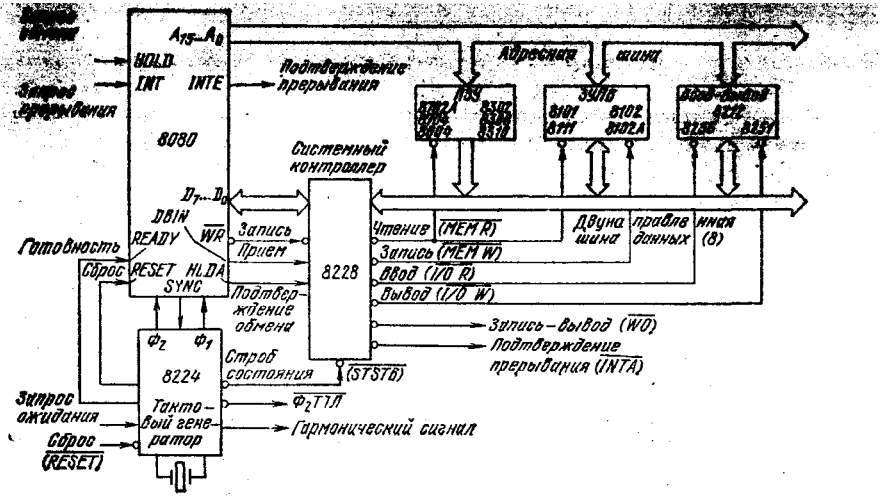


Рис. 7.6. Структурная схема микропроцессорной системы 8080

циальный *системный контроллер* — микросхема 8228. Байт состояния используется для управления направлением передачи по шине данных.

7.2. АДРЕСАЦИЯ ПАМЯТИ

Прямая адресация. При прямой адресации команда содержит адрес памяти. Например, команда ЗАГРУЗИТЬ СОДЕРЖИМОЕ ЯЧЕЙКИ 1F2A В АККУМУЛЯТОР является командой с прямой адресацией, где 1F2A — прямой адрес.

Команда должна быть записана в памяти в виде, показанном на рис. 7.7. Команда занимает три байта памяти, причем второй и третий байты содержат адрес.

Косвенная адресация через пару регистров. Адрес ячейки памяти может быть определен с помощью содержимого пары регистров. Для большинства команд ЦП 8080 в этих целях используются регистры *H* и *L*. Регистр *H* содержит старший байт адреса, регистр *L* — младший. Структура однобайтовой команды загрузки аккумулятора содержимым ячейки 1F2A памяти показана на рис. 7.8.

Кроме того, есть две команды, использующие для адресации регистры *B*, *C* или *D*, *E*. Как и ранее, первый регистр пары содержит старший байт адреса, а второй — младший. Эти команды (STAX и LDAX) будут описаны ниже.

Непосредственная адресация. Команда с непосредственной адресацией содержит в одном из своих полей сам операнд. Например, команда ЗАГРУЗИТЬ В АККУМУЛЯТОР ВЕЛИЧИНУ $2A_{16}$ располагается в памяти, как показано на рис. 7.9. Команды с не-

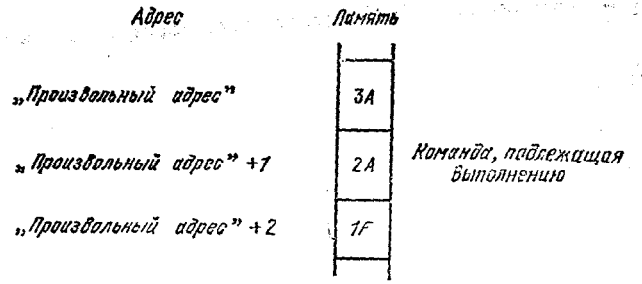


Рис. 7.7. Прямая адресация

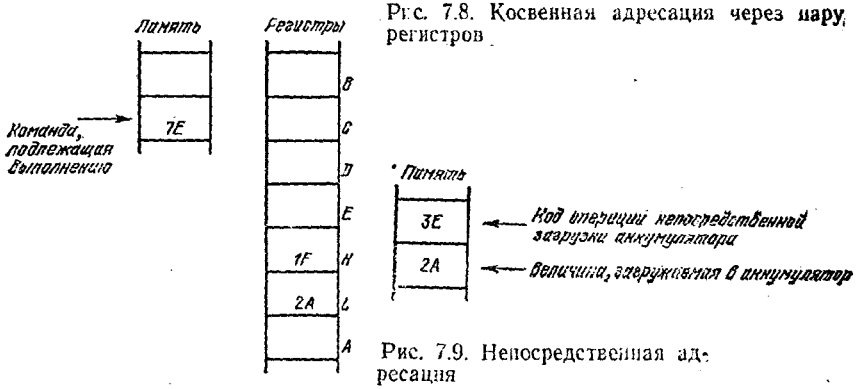


Рис. 7.8. Косвенная адресация через пару регистров

Рис. 7.9. Непосредственная адресация

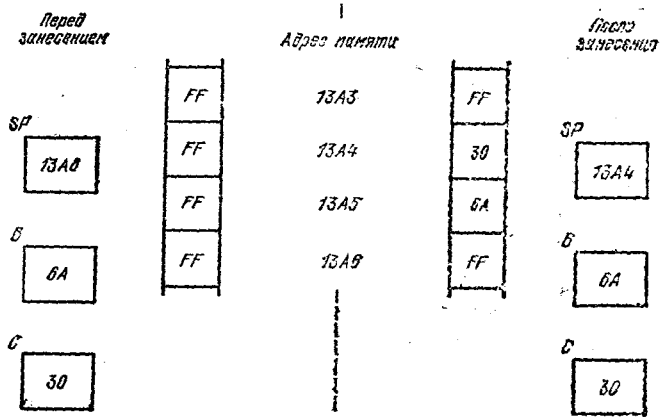


Рис. 7.10. Адресация через указатель стека (операция занесения в стек)

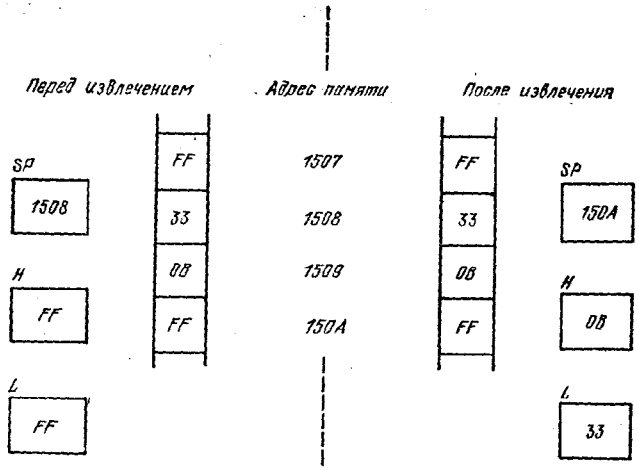


Рис. 7.11. Операция извлечения из стека

посредственной адресацией не обращаются в памяти для извлечения операнда — они сами содержат операнд.

Адресация через указатель стека. Адресация может быть осуществлена через 16-разрядный регистр— указатель стека. Существуют только две операции со стеком: запись данных в стек, называемая PUSH — ЗАНЕСЕНИЕ В СТЕК, и выборка данных из стека, называемая POP — ИЗВЛЕЧЕНИЕ ИЗ СТЕКА.

Операция занесения в стек используется для пересылки 16 бит данных из пары регистров или из программного счетчика в область памяти, отведенную под стек. Адрес памяти, по которому должно произойти обращение к памяти при выполнении операции занесения в стек с помощью указателя стека, определяется так:

— 8 старших бит данных запоминаются по адресу «содержимое указателя стека — 1»;

— 8 младших бит запоминаются по адресу «содержимое указателя стека — 2»;

— содержимое указателя стека автоматически уменьшается на 2.

К примеру, предположим, что указатель стека содержит адрес $13A6_{16}$, регистр B — $6A_{16}$, регистр C — 30_{16}.

В результате выполнения операции занесения в стек из пары регистров B возникает ситуация, показанная на рис. 7.10.

Операция извлечения из стека используется для пересылки 16 бит из области памяти, отведенной под стек, в любую пару регистров или в программный счетчик. Адрес памяти, по которому осуществляется обращение при выполнении операции, определяется следующим образом:

— второй регистр пары или младшие 8 бит программного счетчика загружаются из памяти по адресу, хранимому [illegible] стека;

Таблица 7.1

Выполнение команды CALL (ВЫЗОВ ПОДПРОГРАММЫ)

Адрес памяти	Команда	Комментарии
0C02		
0C03	ВЫЗОВ ПОДПРОГРАММЫ (CALL)	Занесение в стек адреса следующей команды ($0C06_{16}$) и переход к началу подпрограммы по адресу $0F02_{16}$
0C04	02	
0C05	0F	
0C06	СЛЕДУЮЩАЯ КОМАНДА	
0F00		
0F01		
0F02	ПЕРВАЯ КОМАНДА ПОДПРОГРАММЫ	
0F03		
—	Тело подпрограммы	
—		
—		
0F4E		
0F4F	ВОЗВРАТ (RET)	Извлечение адреса возврата ($0C06_{16}$) и переход к следующей команде основной программы

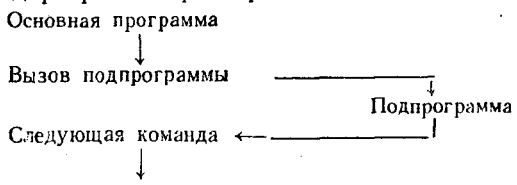
— первый регистр пары или старшие 8 бит программного счетчика загружаются из памяти по адресу «содержимое указателя стека+1»;

— содержимое указателя стека автоматически увеличивается на 2.

К примеру, пусть указатель стека содержит адрес 1508_{16}, по адресу 1508_{16} в памяти записано число 33_{16}, а по адресу 1509_{16} — число $0B_{16}$. Результат извлечения из стека в пару регистров *H* показан на рис. 7.11.

Перед выполнением стековых операций в указатель стека необходимо загрузить требуемое значение адреса. Для загрузки указателя стека любой величиной используется команда LXISP (LOAD IMMEDIATE STACK POINTER — ЗАГРУЗИТЬ НЕПОСРЕДСТВЕННО УКАЗАТЕЛЬ СТЕКА).

Подпрограммы и использование стека для адресации. При выполнении подпрограмм характерна последовательность событий:



где стрелки указывают последовательность выполнения.

Когда выполняется команда CALL (ВЫЗОВ ПОДПРОГРАММЫ), содержимое программного счетчика (адрес следующей команды) заносится в стек и начинается выполнение подпрограммы. Последней выполняемой командой подпрограммы должна быть команда возврата, по которой из стека в программный счетчик извлекается этот адрес, и, таким образом, инициируется выполнение следующей команды программы (см. табл. 7.1).

Любая подпрограмма может сама обратиться к подпрограмме и т. д. Количество вложенных подпрограмм ограничивается только объемом памяти, отведенным под стек. Анализ последовательности занесений и извлечений из стека указывает, что порядок извлечений обратен порядку вызовов подпрограмм, даже если одна и та же подпрограмма вызывалась несколько раз.

7.3. БИТЫ УСЛОВИЙ

ЦП 8080 выдает 5 бит условий, отображающих результат выполнения операций. Все биты, за исключением бита дополнительного переноса, могут быть проанализированы командами управления последовательностью выполнения программы (командами условного перехода). Ниже рассмотрено, в каких командах и какие биты анализируются, а также как выполняются команды в зависимости от битов состояния.

Далее предполагается, что бит состояния равен 1, если о нем говорится, что он «установлен», и равен 0, если «сброшен».

Бит переноса (*Carry*) устанавливается и сбрасывается командами сложения, вычитания, сдвига и логическими командами, выполненными над данными, и может быть программно проанализирован. Например, при сложении двух однобайтных слов может появиться перенос из старшего разряда:

Бит	7	6	5	4	3	2	1	0
AE_{16} =	1	0	1	0	1	1	1	0
+ 74_{16} =	0	1	1	1	0	1	0	0
122_{16}	0	0	1	0	0	0	1	0

Перенос = 1, устанавливается бит переноса = 1

Если в результате выполнения операции сложения появляется единица переноса из старшего разряда, бит переноса устанавливается, если же перенос не возникает, бит переноса сбрасывается.

Замечание. Операции сложения, вычитания, сдвига и логические устанавливают и сбрасывают бит переноса по разным правилам. ЦП 8080 имеет несколько видов операций сложения: ADD, ADC, ADI, ACI и DAD; вычитания: SUB, SBB, SUI, SBI, CMP и CPI; циклического сдвига: RAL, RAR, RLC, RRC; логических операций: ANA, ORA, XRA, ANI, ORI, XRI.

Дополнительный бит переноса (Auxiliary carry) устанавливается при появлении переноса из бита 3. Состояние дополнительного бита переноса не может быть проанализировано непосредственно программой, за исключением команды десятичной коррекции (DAA). Ниже приведен пример, в котором сбрасывается бит переноса и устанавливается дополнительный бит переноса:

Бит		7	6	5	4	3	2	1	0
	$2E_{16}=$	0	0	1	0	1	1	1	0
+	$74_{16}=$	0	1	1	1	0	1	0	0
	$A2_{16}$	1	0	1	0	0	0	1	0
		↓				↓			
		Перенос = 0				Дополнительный перенос = 1			

Бит дополнительного переноса может быть установлен всеми операциями сложения, вычитания, приращения (+1), уменьшения (—1) и операциями сравнения.

Бит знака (Sign). В байте можно представить числа от -128_{10} до $+127_{10}$. При этом, как обычно, седьмой бит представляет знак. Если он равен 1, байт содержит числа от -128_{10} до -1_{10}, если 0 — от 0 до $+127_{10}$.

В конце выполнения некоторых операций бит знака устанавливается по седьмому биту результата.

Бит нулевого признака (Zero) устанавливается, если результат определенных операций равен 0. Бит нуля сбрасывается, если результат ненулевой.

Если операция дает нулевой байт результата и единицу переноса, как показано ниже, бит нуля также устанавливается.

Бит		7	6	5	4	3	2	1	0
		1	0	1	0	0	1	1	1
	+	0	1	0	1	1	0	0	1
1		0	0	0	0	0	0	0	0
↑									
Перенос из 7-го бита					Нулевой результат Нулевой бит устанавливается в 1				

Бит четности (Parity) устанавливается при выполнении некоторых операций путем подсчета числа бит в байте, равных единице.

Если суммарное число равно четной величине, то индицируется сигнал четности, в противном случае — сигнал нечетности. Бит четности устанавливается сигналом четности и сбрасывается сигналом нечетности.

7.4. СИСТЕМА КОМАНД ЦП 8080

Детальное описание команд[1] ЦП 8080 представлено в табл. 7.2... 7.11.

Основные обозначения

Обозначение	Содержание
$<B_2>$	Второй байт команды
$<B_3>$	Третий байт команды
r	Обозначение одного из рабочих регистров: *A, B, C, D, E, H, L*
c	Обозначение состояния одного из следующих триггеров флажка:
	Триггеры флажка — Условие истинности
	C (перенос) — наличие переноса или заема
	Z (нуль) — нулевой результат
	S (знак) — старший разряд байта результата равен единице
	P (четность) — результат содержит четное число единиц
M	Адрес ячейки памяти, хранящийся в регистрах *H* и *L*
( )	Содержимое ячейки памяти или регистра
∧	Логическое И
∀	Исключающее ИЛИ
∨	ИЛИ
r_m	*m*-й бит регистра *r*
SP	Указатель стека
PC	Счетчик команд
←	Пересылка
XXX	Не имеющие значения биты
SSS	Регистр (источник)
DDD	Регистр назначения (приемник)

Номер регистра (SSS или DDD)	Наименование регистра
000	*B*
001	*C*
010	*D*
011	*E*
100	*H*
101	*L*
110	Память
111	Аккумулятор (*A*)

[1] Коды операций команд и число периодов тактовых импульсов, требуемых для выполнения команд, приведены в книге Дж. Хилбурн, П. Джулич. Микро-ЭВМ и микропроцессоры. М., Мир, 1979, стр. 414—420.

Таблица 7.2

Команды прямой адресации

Мнемокод	Длина (байт)	Число циклов	Описание операции
STA $<B_2>$ $<B_3>$	3	4	$[<B_3><B_2>]\leftarrow(A)$ Запомнить содержимое аккумулятора по адресу, указываемому байтами B_2 и B_3 команды
LDA $<B_2>$ $<B_3>$	3	4	$(A)\leftarrow[<B_3><B_2>]$ Загрузить аккумулятор содержимым ячейки, адресуемой байтами B_2 и B_3 команды

Таблица 7.3

Команды с адресацией через пару регистров *H*, *L*

Мнемокод	Длина (байт)	Число циклов	Описание операции
ADD M	1	2	$(A)\leftarrow(A)+(M)$. Сложить
ADC M	1	2	$(A)\leftarrow(A)+(M)+$(перенос). Сложить с переносом
SUB M	1	2	$(A)\leftarrow(A)-(M)$. Вычесть
SBB M	1	2	$(A)\leftarrow(A)-(M)-$(заем) Вычесть с заемом
ANA M	1	2	$(A)\leftarrow(A)\wedge(M)$. Логическое И
XRA M	1	2	$(A)\leftarrow(A)\vee(M)$. Исключающее ИЛИ
ORA M	1	2	$(A)\leftarrow(A)\vee(M)$. Логическое ИЛИ
CMP M	1	2	$(A)-(M)$. Сравнение
INR M	1	3	$[M]\leftarrow[M]+1$. Увеличить на 1 содержимое ячейки, адресуемой регистрами *H* и *L*. Все биты состояния, за исключением переноса, могут измениться
DCR M	1	3	$[M]\leftarrow[M]-1$. Содержимое ячейки памяти, указываемой парой регистров *H*, *L*, уменьшить на 1. Все биты состояния, кроме переноса, могут измениться

Примечание. Память (*M*) адресуется содержимым регистров *H*, *L*. Флажки выставляются точно так же, как и в безадресных командах (см. табл. 7.5).

Таблица 7.4

Команды с непосредственной адресацией

Мнемокод	Длина (байт)	Число циклов	Описание операции
ADI $<B_2>$	2	2	$(A)\leftarrow(A)+<B_2>$. Сложить
ACI $<B_2>$	2	2	$(A)\leftarrow(A)+<B_2>+$(перенос). Сложить с переносом

Мнемокод	Длина (байт)	Число циклов	Описание операций
SUI $<B_2>$	2	2	$(A)\leftarrow(A)-<B_2>$. Вычесть
SBI $<B_2>$	2	2	$(A)\leftarrow(A)-<B_2>-$(заем). Вычесть с заемом
ANI $<B_2>$	2	2	$(A)\leftarrow(A)\wedge<B_2>$. Логическое И
XRI $<B_2>$	2	2	$(A)\leftarrow(A)\vee<B_2>$. Исключающее ИЛИ
ORI $<B_2>$	2	2	$(A)\leftarrow(A)\vee<B_2>$. Логическое ИЛИ
CPI $<B_2>$	2	2	$(A)-<B_2>$. Сравнить
LXI B $<B_2>$ $<B_3>$	3	3	$(C)\leftarrow<B_2>$; $(B)\leftarrow<B_3>$. Загрузить B_2 в регистр *C*, а B_3 — в регистр *B*
LXI D $<B_2>$ $<B_3>$	3	3	$(E)\leftarrow<B_2>$; $(D)\leftarrow<B_3>$. Загрузить второй байт команды в регистр *E*, третий — в регистр *D*
LXI H $<B_2>$ $<B_3>$	3	3	$(L)\leftarrow<B_2>$; $(H)\leftarrow<B_3>$. Загрузить второй байт команды в регистр *L*, третий — в регистр *H*

Таблица 7.5

Команды обращения к регистрам

Мнемокод	Длина (байт)	Число циклов	Описание операций
INR r	1	1	$(r)\leftarrow(r)+1$. Увеличить на единицу содержимое регистра *r*. В результате могут поменяться все биты состояния, кроме переноса
DCR r	1	1	$(r)\leftarrow(r)-1$. Уменьшить на единицу содержимое регистра *r*. В результате могут поменяться все биты состояния, кроме переноса
ADD r	1	1	$(A)\leftarrow(A)+(r)$. Сложить содержимое регистра *A* с содержимым регистра *r* и заслать результат в *A*. (Все биты состояния подвержены изменению)

Продолжение табл. 7.5

Мнемокод	Длина (байт)	Число циклов	Описание операции
ADC r	1	1	(A)←(A)+(r)+(перенос). Сложить содержимое регистра *r* и триггера переноса с содержимым регистра *A*, результат оставить в *A*. (Все биты состояния подвержены изменению)
SUB r	1	1	(A)←(A)−(r). Вычесть содержимое регистра *r* из содержимого регистра *A*, результат занести в *A*. Используется вычитание в дополнительном коде. (Все биты состояния подвержены изменению)
SBB r	1	1	(A)←(A)−(r)−(заем). Вычесть из содержимого регистра *A* содержимое регистра *r* и триггера переноса. Результат оставить в *A*. (Все биты состояния подвержены изменению)
ANA r	1	1	(A)←(A)∧(r) Поместить в регистр *A* результат логической операции И над содержимым регистров *A* и *r*. (Сбрасывается перенос).
XRA r	1	1	(A)←(A)∀(r) Поместить в регистр *A* результат Исключающего ИЛИ над содержимым регистров *A* и *r*. (Сбрасывается перенос)
ORA r	1	1	(A)←(A)∨(r) Поместить в регистр *A* результат логического ИЛИ над содержимым регистров *A* и *r*. (Сбрасывается перенос)
CMP r	1	1	(A)−(r). Сравнить содержимое регистра *A* с содержимым регистра *r*. Содержимое регистра *A* не изменяется. Триггеры состояния устанавливаются по результату вычитания. Равенство (A=r) индицируется установкой триггера нуля в *1*. Неравенство (A<r) индицируется установкой триггера переноса в 1
RAL	1	1	$A_{m+1}\leftarrow A_m$, $A_0\leftarrow$(C), (C)←A_7 Циклический сдвиг *A* влево на 1 бит. Бит переноса сдвигается в A_0, A_7 сдвигается в триггер переноса
RAR	1	1	$A_m\leftarrow A_{m+1}$, $A_7\leftarrow$(C), (C)←A_0 Циклический сдвиг *A* вправо на 1 бит. A_0 сдвигается в триггер переноса, а бит переноса сдвигается в A_7
RLC	1	1	$A_{m+1}\leftarrow A_m$, $A_0\leftarrow A_7$, (C)←A_7 Циклический сдвиг *A* влево на 1 бит. A_7 сдвигается в A_0 и в триггер переноса
RRC	1	1	$A_m\rightarrow A_{m+1}$, $A_7\leftarrow A_0$, (C)←A_0 Циклический сдвиг *A* вправо на 1 бит. A_0 сдвигается в A_7 и в триггер переноса

Продолжение табл. 7.5

Мнемокод	Длина (байт)	Число циклов	Описание операции
CMA	1	1	(A)←$\overline{(A)}$ Взять обратный код *A*. Триггер состояния не изменяется
STC	1	1	(C)←1 Установить триггер переноса в 1. Другие триггеры состояния не изменяются
CMC	1	1	(C)←$\overline{(C)}$ Обращение бита переноса. Другие триггеры состояния не затрагиваются

Таблица 7.6

Команды пересылок

Мнемокод	Длина (байт)	Число циклов	Описание операции
MOV r_1, r_2	1	1	(r_1)←(r_2). Загрузить регистр r_1 из регистра r_2. Содержимое r_2 не изменяется
MOV r, M	1	2	Загрузить регистр *r* из ячейки памяти с адресом (*H*, *L*)
MOV M, r	1	2	(M)←(r). Записать в ячейку памяти с адресом (*H*, *L*) содержимое регистра *r*
MVI r <B_2>	2	2	(r)←<B_2> Загрузить второй байт команды в регистр *r*
MVI M <B_2>	2	3	<M>←<B_2>. Загрузить второй байт команды по адресу (*H*, *L*)

Таблица 7.7

Условные переходы

Мнемокод	Длина (байт)	Число циклов	Описание операции
JMP <B_2> <B_3>	3	3	(PC)←<B_3><B_2>. Безусловный переход по адресу, указанному в байтах B_2 и B_3 команды
JC <B_2> <B_3>	3	3	ЕСЛИ (C)=1, то (PC)←<B_3><B_2>, ИНАЧЕ (PC)←(PC)+3

Продолжение табл. 7.7

Мнемокод	Длина (байт)	Число циклов	Описание операции
JNC <B_2> <B_3>	3	3	ЕСЛИ (C)=0, (PC)←<B_3><B_2> ИНАЧЕ (PC)←(PC)+3
JZ <B_2> <B_3>	3	3	ЕСЛИ (Z)=1, (PC)←<B_3><B_2>, ИНАЧЕ (PC)←(PC)+3
JNZ <B_2> <B_3>	3	3	ЕСЛИ (Z)=0, (PC)←<B_3><B_2>, ИНАЧЕ (PC)←(PC)+3
JP <B_2> <B_3>	3	3	ЕСЛИ (S)=0, (PC)←<B_3><B_2>, ИНАЧЕ (PC)←(PC)+3
JM <B_2> <B_3>	3	3	ЕСЛИ (S)=1, (PC)←<B_3><B_2>, ИНАЧЕ (PC)←(PC)+3
JPE <B_2> <B_3>	3	3	ЕСЛИ (P)=1, (PC)←<B_3><B_2>, ИНАЧЕ (PC)←(PC)+3
JPO <B_2> <B_3>	3	3	ЕСЛИ (P)=0, (PC)←<B_3><B_2>, ИНАЧЕ (PC)←(PC)+3

Таблица 7.8

Операции со стеком

Мнемокод	Длина (байт)	Число циклов	Описание операции
LXI SP <B_2> <B_3>	3	3	$(SP)_L$←<B_2>, $(SP)_H$←<B_3> Загрузить байт B_2 в младшие разряды указателя стека, а B_3 — в старшие.
PUSH PSW	1	3	[SP−1]←(A), [SP−2]←(F), (SP)=(SP)−2 Сохранить содержимое аккумулятора *A* и битов условий *F* (5 флажков) путем внесения в стек по адресу указателя стека. Содержимое регистра *SP* уменьшается на 2. Слово состояния флажков запоминается в следующем виде: D_0:CY_2 (перенос *C*) D_1:1 D_2:*P* D_3:0 D_4:CY_1 (дополнительный перенос) D_5:0 D_6:*Z*

Продолжение табл. 7.8

Мнемокод	Длина (байт)	Число циклов	Описание операции
PUSH B	1	3	D_7:*S* (старший значащий бит)
PUSH D	1	3	[SP−1]←(B), [SP−2]←(C), (SP)=(SP)−2
PUSH H	1	3	[SP−1]←(D), [SP−2]←(E), (SP)=(SP)−2
POP PSW	1	3	[SP−1]←(H), [SP−2]←(L), (SP)=(SP)−2 (F)←[SP], (A)←[SP+1], (SP)=(SP)+2 Восстановить содержимое *A* и *F* из двух последних ячеек стека, увеличить *SP* на 2
POP B	1	3	(C)←[SP], (B)←[SP+1], (SP)=(SP)+2
POP D	1	3	(E)←[SP], [D]←[SP+1], (SP)=(SP)+2
POP H	1	3	(L)←[SP], (H)←[SP+1], (SP)=(SP)+2
XTH L	1	5	(L)↔[SP], (H)↔[SP+1] Обменять содержимым регистры *H*, *L* и верхнюю пару ячеек стека, адресуемого указателем стека *SP*. Содержимое *SP* не изменяется
SPHL	1	1	(SP)←(H)(L) Переслать содержимое регистров *H* и *L* в регистр *SP*
PCHL	1	1	(PC)←(H)(*L*). КОСВЕННЫЙ ПЕРЕХОД — перейти по адресу, указанному в регистрах *H*, *L*
DAD SP	1	3	(H)(L)←(H)(L)+(SP). Сложить содержимое регистров *H*, *L* и *SP* и заслать результат в регистры *H* и *L*. Если возникло переполнение, то устанавливается бит переноса, в противном случае он сбрасывается. Другие биты состояния не затрагиваются. Это удобно для адресации через стек
INX SP	1	1	(SP)←(SP)+1
DCX SP	1	1	(SP)←(SP)−1

Таблица 7.9

Условные команды перехода к подпрограммам и возврата из подпрограмм

Мнемокод	Длина (байт)	Число циклов	Описание операции
CALL <B_2> <B_3>	3	5	[SP−1][SP−2]←(PC), (SP)=(SP)−2 (PC)←<B_3><B_2>, Занести содержимое счетчика команд *PC* в стек по адресу указателя стека *SP*. Содержимое регистра *SP* уменьшить на 2. Безусловный переход к команде, записанной по адресу, указываемому байтами <B_2> и <B_3> команды
CC <B_2> <B_3>	3	3/5	ЕСЛИ (C)=1, то [SP−1][SP−2]←PC (SP)=(SP)−2, (PC)←<B_3><B_2>, ИНАЧЕ (PC)=(PC)+3

Продолжение табл. 7.9

Мнемокод	Длина (байт)	Число циклов	Описание операции
CNC <B_2> <B_3>	3	3/5	ЕСЛИ (C)=0, то [SP−1][SP−2]←PC (SP)=(SP)−2, (PC)←<B_3><B_2>, ИНАЧЕ (PC)=(PC)+3
CZ <B_2> <B_3>	3	3/5	ЕСЛИ (Z)=1, то [SP−1][SP−2]←PC, (SP)=(SP)−2, (PC)←<B_3><B_2>, ИНАЧЕ (PC)=(PC)+3
CNZ <B_2> <B_3>	3	3/5	ЕСЛИ (Z)=0, то [SP−1][SP−2]←PC (SP)=(SP)−2, PC←<B_3><B_2>, ИНАЧЕ (PC)=(PC)+3
CP <B_2> <B_3>	3	3/5	ЕСЛИ (S)=0, то [SP−1][SP−2]←PC, (SP)=(SP)−2, (PC)←<B_3><B_2> ИНАЧЕ (PC)=(PC)+3
CM <B_2> <B_3>	3	3/5	ЕСЛИ (S)=1,то [SP−1][SP−2]←PC, (SP)=(SP)−2, (PC)←<B_3><B_2>, ИНАЧЕ (PC)=(PC)+3
CPE <B_2> <B_3>	3	3/5	ЕСЛИ (P)=1, то [SP−1][SP−2]←PC, (SP)=(SP)−2, (PC)←<B_3><B_2>, ИНАЧЕ (PC)=(PC)+3
CPO <B_2> <B_3>	3	3/5	ЕСЛИ (P)=0, то [SP−1][SP−2]←PC (SP)=(SP)−2, (PC)←<B_3><B_2>, ИНАЧЕ (PC)=(PC)+3
RET	1	3	(PC)←[SP][SP+1], (SP)=(SP)+2 Возвращение к команде, адрес которой записан в верхней паре ячеек стека. Увеличение содержимого указателя стека на 2
RC	1	1/3	ЕСЛИ (C)=1, то (PC)←[SP], [SP+1], (SP)=(SP)+2, ИНАЧЕ (PC)=(PC)+1
RNC	1	1/3	ЕСЛИ (C)=0, то (PC)←[SP], [SP+1], (SP)=(SP)+2, ИНАЧЕ (PC)=(PC)+1
RZ	1	1/3	ЕСЛИ (Z)=1, то (PC)←[SP], [SP+1], (SP)=(SP)+2, ИНАЧЕ (PC)=(PC)+1
RNZ	1	1/3	ЕСЛИ (Z)=0, то (PC)←[SP], [SP+1], (SP)=(SP)+2, ИНАЧЕ (PC)=(PC)+1
RP	1	1/3	ЕСЛИ (S)=0, то (PC)←[SP], [SP+1], (SP)=(SP)+2, ИНАЧЕ (PC)=(PC)+1
RM	1	1/3	ЕСЛИ (S)=1, то (PC)←[SP], [SP+1], (SP)=(SP)+2, ИНАЧЕ (PC)=(PC)+1

Продолжение табл. 7.9

Мнемокод	Длина (байт)	Число циклов	Описание операции
RPE	1	1/3	ЕСЛИ (P)=1, то (PC)←[SP], [SP+1], (SP)=(SP)+2, ИНАЧЕ (SC)=(PC)+1
RPO	1	1/3	ЕСЛИ (P)=0, то (PC)←[SP][SP+1], (SP)=(SP)+2, ИНАЧЕ (PC)=(PC)+1
RST	1	3	[SP−1][SP−2]←(PC), (SP)=(SP)−2 (PC)←(00000000 00 AAA 000) Вектор прерывания, выдаваемый внешним устройством (см. рис. 7.17)

Таблица 7.10

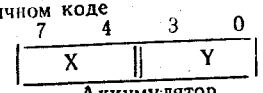
Операции ввода — вывода

Мнемокод	Длина (байт)	Число циклов	Описание операции
IN <B_2>	2	3	(A)←(вводимые данные) Во время такта T_1 третьего цикла байт B_2 команды, содержащий адрес УВВ, выдается на адресную шину, а во время действия сигнала *SYNC* (синхронизация) на линию *INP* (ввод) (а не *MEMR* (чтение)) поступает высокий потенциал. Данные загружаются в аккумулятор с шины данных по сигналу *DBIN* (прием), выдаваемому ЦП. Триггеры состояния не затрагиваются
OUT <B_2>	2	3	(Выводимые данные)←(A) Во время такта T_1 третьего цикла байт B_2 команды, содержащий адрес УВВ, выдается на адресную шину, а во время действия сигнала *SYNC* (синхронизация) на контакт *OUT* (вывод) подается высокий потенциал. Содержимое аккумулятора выдается на шину данных, когда сигнал на выводе $\overline{WR}$ (запись) становится равным 0
EI	1	1	Разрешить прерывание
DI	1	1	Запретить прерывание С помощью указанных команд может быть установлен или сброшен триггер разрешения прерывания (*INTE*). Сигнал запроса прерывания *INT* может быть воспринят ЦП, только если триггер разрешения прерывания *INTE* установлен. Сразу после приема сигнала в ЦП триггер *INTE* сбрасывается. Во время выполнения команд EI и DI сигнал запроса прерывания не воспринимается

Таблица 7.11

Другие команды

Мнемокод	Длина (байт)	Число циклов	Описание операции
LDAX D	1	2	(A)←[(D)(E)]. Загрузить *A* из ячейки с адресом, записанным в регистрах *D, E*
INX B	1	1	(B)(C)←(B)(C)+1. Увеличить содержимое пары регистров *B, C* на 1. Триггеры состояния не изменять
INX H	1	1	(H)(L)←(H)(L)+1. Увеличить на 1 содержимое пары регистров *H, L*. Триггеры состояния не изменять
INX D	1	1	(D)(E)←(D)(E)+1
DAD B	1	3	(H)(L)←(H)(L)+(B)(C)
DAD H	1	3	(H)(L)←(H)(L)+(H)(L) (сдвиг влево удвоенной точности содержимого регистров *H* и *L*)
DAD D	1	3	(H)(L)←(H)(L)+(D)(E)
STAX B	1	2	[(B)(C)]←(A). Запомнить содержимое *A* по адресу, записанному в паре регистров *B* и *C*
STAX D	1	2	[(D)(E)]←(A). Запомнить содержимое аккумулятора по адресу, записанному в паре регистров *D* и *E*
LDAX B	1	2	(A)←[(B)(C)]. Загрузить аккумулятор из ячейки с адресом, записанным в регистрах *B* и *C*
DCX B	1	1	(B)(C)←(B)(C)−1
DCX H	1	1	(H)(L)←(H)(L)−1
DCX D	1	1	(D)(E)←(D)(E)−1
XCHG	1	1	(H)↔(D)(E)↔(L) Поменять содержимое пар регисторов *H, L* и *D, E*
DAA	1	1	Десятичная коррекция аккумулятора. Его содержимое (8 бит) — скорректированный результат арифметической операции над 2-разрядными десятичными операндами, представленными в двоично-десятичном коде



Аккумулятор

Корректировка заключается в следующем: если $Y \geqslant 10$ или есть перенос из 4-го разряда (бит 3), то Y=Y+6 и перенос из старшего разряда Y в X (из бита 3 в 4).
Если $X \geqslant 10$ или есть перенос из 8-го разряда (бит 7), или $Y \geqslant 10$ и X=9, то X=X+6 (и устанавливается триггер переноса). В данной команде анализируются два триггера переноса: CY_1—перенос из 3-го бита (4-й разряд), который доступен для анализа данной команды как пятый флажок, и CY_2 — перенос из 7-го бита (8-й разряд), как обычно. Команда может изменить состояние любого триггера флажков

Продолжение табл. 7.11

Мнемокод	Длина (байт)	Число циклов	Описание операции
SHLD $<B_2>$ $<B_3>$	3	5	[$<B_3><B_2>$]←(L), [$<B_3><B_2>+1$]←(H) Запомнить содержимое регистров *H* и *L* по адресу, указанному во втором и третьем байтах команды
LHLD $<B_2>$ $<B_3>$	3	5	(L)←[$<B_3><B_2>$], (H)←[$<B_3><B_2>+1$] Загрузить регистры *H* и *L* из ячейки с адресом, указанным в байтах B_2 и B_3 команды
HLT	1	1	По приходу команды HALT ЦП переходит в состояние останов. Содержимое всех регистров и памяти не изменяется, а содержимое счетчика команд *PC* модифицируется

7.5. ПРИМЕРЫ ПРОГРАММИРОВАНИЯ

Загрузка аккумуляторов. Одна из наиболее часто встречающихся задач при программировании — засылка некоторого числа в аккумулятор. В табл. 7.12 приведены 8 различных вариантов загрузки аккумулятора. Каждая строчка представляет собой отдельный вариант.

MOV A, B — однобайтовая команда (см. табл. 7.6). Регистр *A* используется как регистр назначения (приема), регистр *B* — как регистр — источник данных.

Таблица 7.12

Команды загрузки аккумулятора

Мнемокод	Операнд	Комментарии	Длина (байт)
MOV	A, B	Загрузить *A* из регистра *B*	1
MVI	A, 23	Загрузить в *A* непосредственно число 23_{16}	2
LDA	4098	Загрузить *A* из ячейки памяти № 4098	3
MOV	A, M	Загрузить *A* из ячейки с адресом, содержащимся в регистрах *H* и *L*	1
LDAX	B	Загрузить *A* из ячейки с адресом, содержащимся в регистрах *B, C*	1
LDAX	D	Загрузить *A* из ячейки с адресом, содержащимся в регистрах *D, E*	1
LHLD	4098	Загрузить аккумулятор из ячейки, адрес которой записан в ячейке 4098 (косвенная адресация)	4
MOV	A, M		1
POP	A	Загрузить *A* из стека	2
IN	10	Загрузить *A* из УВВ № 10	