В. В. Долгий
И. А. Шалатонин

ПРОГРАММИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

ПРАКТИКУМ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ РАДИОФИЗИКИ И ЭЛЕКТРОНИКИ
Кафедра кибернетики

Программирование микропроцессорных систем

Методические указания
к учебному практикуму

Для студентов II курса специальностей:
G 31 04 02 «Радиофизика»,
G 31 04 03 «Физическая электроника»,
E 25 01 10 «Коммерческая деятельность на рынке радиоэлектронных средств и информационных услуг»

МИНСК
2003

Содержание:

ВВЕДЕНИЕ

Развитие и распространение микропроцессоров, а также микроконтроллеров и микро-ЭВМ на их основе стимулирует постоянно возрастающий интерес специалистов к их изучению и применению. В связи с этим является актуальной проблема обучения студентов основам функционирования микропроцессоров, их программирования и выработки навыков по их применению в собственных разработках. Решение этой проблемы предполагает создание лабораторных комплексов, оснащенных специальным оборудованием и соответствующим методическим обеспечением.

Описанный в данной работе комплекс предназначен для начального знакомства студентов с принципами функционирования восьмиразрядных микропроцессоров и выработка у них навыков программирования на языке ассемблер микропроцессорных систем. В качестве аппаратных средств, на которых базируется учебный практикум, используется выпущенный Рижским ПО «ВЭФ» учебный микропроцессорный комплект (УМК).

УМК имеет встроенный сетевой источник питания, средства отображения информации на магистралях и клавиатуру для ввода данных и программ. УМК базируется на микропроцессоре КР580ВМ80 (аналог-I8080).

Среди первых разработок микропроцессоров в фирме INTEL наиболее удачным оказался МП I8080. При разработке новых типов МП фирма INTEL всегда стремилась соблюдать принцип архитектурной совместимости. Новый кристалл МП I8086 был разработан с учетом программной совместимости с I8080. Многие блоки в его структуре были введены с целью обеспечить эту совместимость. Последующие микропроцессоры I80x86 имели архитектурную совместимость “снизу-вверх”.

Изучение на первом этапе микропроцессора I8080 позволяет в дальнейшем понять архитектурные особенности, принципы функционирования, и программирования микропроцессорных систем на базе I80x86.

1. Микропроцессорные системы и их программирование

1.1. Микропроцессоры-история,основные термины и определения

Одним из крупнейших достижений интегральной электроники явилось появление в начале 70-х годов микропроцессоров (МП). Микропроцессором (МП) называется программно-управляемое устройство, осуществляющее прием, обработку и выдачу цифровой информации, построенное на основе одной или нескольких БИС (больших интегральных схем). МП нельзя рассматривать как результат революционного открытия. Это скорее естественный этап в эволюции микроэлектронной технологии. Главная причина столь быстрого вторжения МП в жизнь состоит в том, что они объединили в себе:

- универсальные возможности программируемого вычислительного средства;
- преимущества технологии БИС.

Первые БИС появились в конце 60-х – начале 70-х годов и были в основном ориентированы на выполнение определенных специальных функций (БИС для микрокалькуляторов, БИС памяти и т.д.) Достижения в производстве БИС в последующем сделали возможным реализацию традиционной архитектуры ЭВМ на одном или нескольких кристаллах БИС. Таким образом, можно дать другое определение микропроцессора: МП - это процессор ЭВМ, выполненный средствами интегральной технологии.

Первые МП - это результат предельного упрощения структуры процессора ЭВМ до уровня, который мог быть реализован средствами микроэлектроники при доступном к тому времени уровне интеграции. Современные МП по своей вычислительной мощности не уступают процессорам больших ЭВМ и служат элементной базой для построения ЭВМ IV и последующих поколений.

В основе построения МП лежит традиционная фоннеймановская архитектура ЭВМ (Джон фон Нейман (1903-1957), математик; с 1930г. работал в США). Архитектура фоннеймановского типа базируется на двух основных идеях:

1. Программа вычислений вводится в ЭВМ и хранится в той же памяти, что и обрабатываемые данные;

1. Команда представляется в виде числового кода и по форме ничем не отличается от обрабатываемых данных (основное отличие - место расположения в памяти).

Выборка и выполнение команды в ЭВМ фоннеймановского типа описывается с помощью так называемого цикла управления фон Неймана (рис.1).

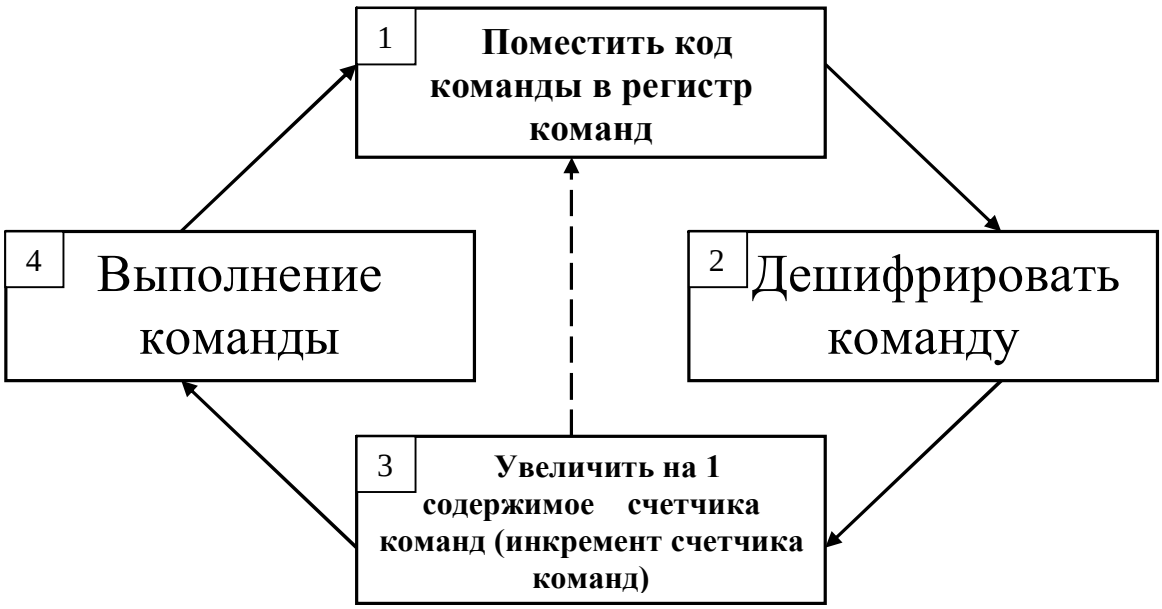


Рис. 1. Цикл управления фон Неймана.

В фазе выборки команды 1 содержимое ячейки памяти (ЯП) выбирается из памяти и помещается в регистр команд (РК). В фазе 2 код команды поступает из РК в дешифратор команд, где он преобразуется в систему управляющих сигналов. После этого счетчик команд инкрементируется в фазе 3, по его содержимому будет адресоваться уже следующая ячейка памяти. Если команда состоит из нескольких байтов, то фазы 1 - 3 повторяются до полного извлечения команды из памяти. После этого процессор переходит к фазе 4 выполнения команды. Когда команда выполнена, происходит возврат к фазе 1 и осуществляется выборка следующей команды.

Таким образом, процессор ЭВМ работает циклично, то есть непрерывно выполняет одни и те же действия: извлекает команды из последовательно расположенных ячеек памяти и выполняет их.

В следствии выше сказанного любой процессор фоннеймановского типа должен содержать счетчик команд. Адресация к ячейкам памяти, в которых находятся коды команд,

осуществляется через счетчик команд, который в структуре МП часто называют PC (PC – program counter). При извлечении каждого байта из памяти PC автоматически инкрементируется.

Таким образом, функционирование процессора ЭВМ, а следовательно, и микропроцессора, опирается на принципы программного управления и хранения данных и программ в общей памяти.

Термин «микропроцессор» впервые был употреблен в 1972 г., хотя годом рождения этого прибора следует считать 1971 г., когда фирма Intel выпустила МП серии 4004 – «интегральное микропрограммируемое вычислительное устройство», представляющее собой однокристальный центральный процессор, имеющий в своем составе 4-разрядный параллельный сумматор, 16 4-разрядных регистров, накапливающий сумматор и стек. МП 4004 был реализован на 2300 транзисторах и мог выполнять 45 различных команд. Последующие поколения МП, представляющие собой 8-, 16- и 32-разрядные приборы, появились соответственно в 1972, 1978 и 1985 гг.

Динамика развития и параметры первых МП от Intel представлена в таблице 1.

Таблица 1.

Процессор	Проектные нормы, мкм	Число транз. на крист., тыс.	Разрядность шины данных	Рабочая частота, МГц	Адресуемая память	Виртуальная память, ТБайт
I4004	10	2.3	4	0.75	4 Кбайт	-
I8008	10	3.5	8	0.8	16 Кбайт	-
I8080	6	6	8	2	64 Кбайт	-
I8086	3	29	16	5;8;10	1 Мбайт	-
I8088	3	29	8/16	5;8	1 Мбайт	-
I80286	1,5	135	16	8;10;12;16	16 Мбайт	0,001
I80386DX	1	275	32	20;25;33;40	4 Гбайт	64
I80386SX	1	275	16/32	20;25;33	4 Гбайт	64
I80486DX	0,8-1,0	1200	32	22;33;50;66;100	4 Гбайт	64
I80486SX	0,8	1185	32	16;20;25;33;50	4 Гбайт	64
Pentium	0,8-0,6	3100	32	60;66;75;90;100; 120;133;166;200	4 Гбайт	64

Используемая в практикуме учебная микро-ЭВМ УМК построена на базе микропроцессорного комплекта (МПК) КР580.
Микропроцессорным комплектом БИС называется совокупность интегральных микросхем, совместимых по конструктивно-технологическому исполнению и предназначенных для совместного использования при построении микропроцессоров, микро-ЭВМ, контроллеров и других средств вычислительной техники.
Функциональный состав, параметры и аналоги микросхем, входящих в состав МПК КР 580 представлен в таблице 2.

Таблица 2.

Функциональное назначение и тип микросхемы	Обозначение функционального аналога	Технология	Разрядность	Быстродействие
1	**2**	**3**	**4**	**5**
ЦПУ параллельной обработки данных КР580 ВМ80	8080	nМОП	8	F_T ≤2,5МГц
Программируемый синхронно-асинхронный передатчик КР580 ВВ51	8251	nМОП	5…8	F_T ≤2МГц $V_{ОБМ}$=56 Кбит/с
Программируемый параллельный интерфейс ВВ580ВВ55	8255	nМОП	8, 4, 1	F_T ≤2МГц
Программируемый контроллер прямого доступа к памяти КР580ВН57	8257	nМОП	Канал данных-8, адрес-16	F_T ≤2МГц
Программируемое многорежимное временное устройство (таймер) КР580ВИ53	8233	nМОП	8	F_T ≤2МГц
Программируемое устройство управления прерываниями КР58ОВН59	8259	nМОП	С число запросов 8…64	$t_{здр.}$=500 нс.
Генератор, тактовых импульсов КР580ГФ24	8224	ТТЛШ	—	F_T ≤27МГц

Контроллер электроннолучевой трубки КР580ВГ75	8275	nМОП	8	$F_T \leq 3{,}12$МГц
Программируемый контроллер клавиатуры и индикации КР580ВВ79	8279	nМОП	8	$F_T \leq 2$МГц
Контроллер шин КР580ВГ18	8218	ТТЛШ	—	$t_{здр.} \leq 100$ нс.
Системный контроллер и шинный формирователь КР580ВК28	8228	ТТЛШ	8	$t_{здр.} \leq 60$ нс.
Системный контроллер и шинный формирователь КР580ВК38	8238	ТТЛШ	8	$t_{здр.} \leq 60$ нс.
Буферные регистры КР580ИР82	8282	ТТЛШ	8	$t_{здр.} \leq 40$ нс.
Буферные регистры КР580ИР83	8283	ТТЛШ	8	$t_{здр.} \leq 45$ нс.
Шинные формирователи КР580ВА86	8286	ТТЛШ	8	$t_{здр.} \leq 30$ нс.

1.2.Учебная микро-ЭВМ УМК

УМК является автономной микро-ЭВМ, предназначенной для изучения особенностей построения программирования и обслуживания микропроцессорных систем.

Микро-ЭВМ УМК выполняет следующие функции:

1.Ввод информации в микро-ЭВМ и вызов директив с помощью клавиатуры;

2.Отображение вводимой и выводимой информации в шестнадцатиричном коде на семисегментном дисплее;

3.Запуск, выполнение и отладка программ пользователя.

В УМК предусмотрено пошаговое выполнение программ, при этом для отображения состояния шины адреса, данных и регистра состояния в двоичном коде используется индикация на светодиодах.

На лицевой панели УМК расположена вилка для подключения макетного ТЭЗ. Структурная схема микро-ЭВМ приведена на рис.2

Микропроцессор работает совместно с системой памяти (микросхемы ОЗУ, ПЗУ, ППЗУ), устройствами ввода-вывода (УВВ) информации. Микропроцессор выполняет программу и управляет всеми перечисленными устройствами. Таким образом, УМК построена по модульному принципу и имеет магистральную (шинную) организацию межмодульных связей. В качестве микропроцессора используется I8080 (микропроцессор с фиксированной системой команд, отечественный аналог КР580ВМ80).

В архитектуре магистрального типа важное значение приобретает интерфейс. В узком смысле интерфейсом (от англ. interfase – сопрягать, согласовывать) называют устройства сопряжения, в широком смысле под интерфейсом понимают совокупность аппаратных, программных и конструктивных средств, обеспечивающих взаимодействие функциональных модулей системы.

Внешний вид лицевой панели микро-ЭВМ УМК изображен на рис.3

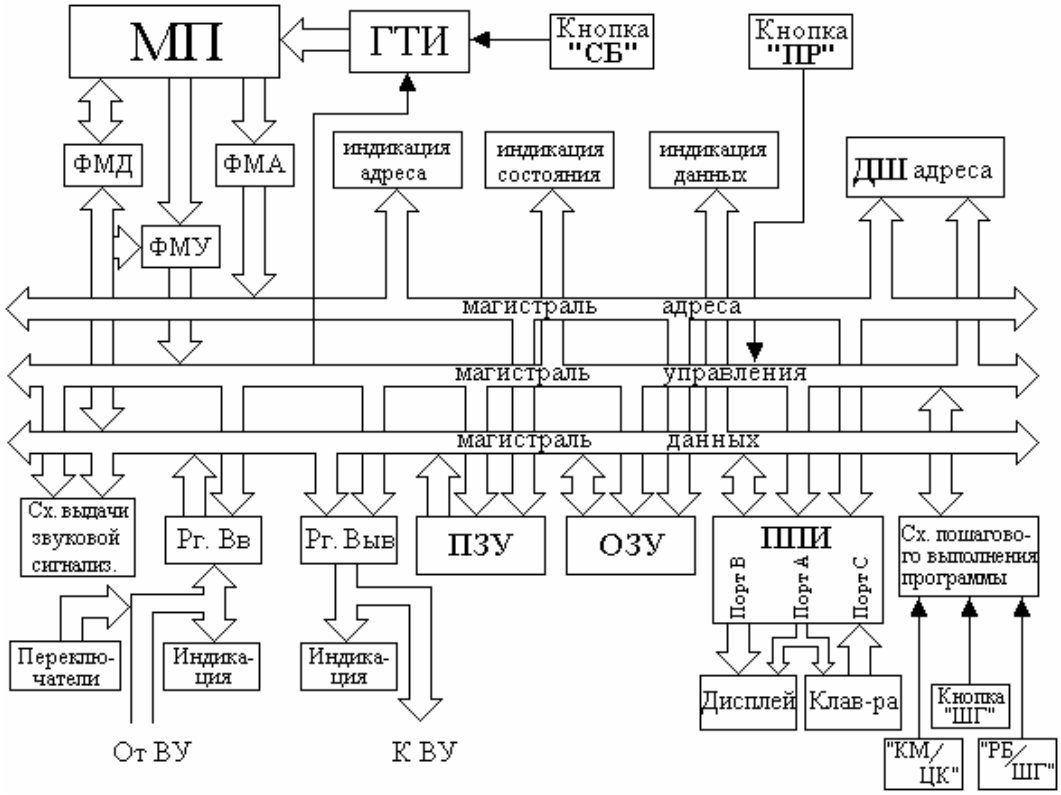


Рис. 2. Структура микро-ЭВМ

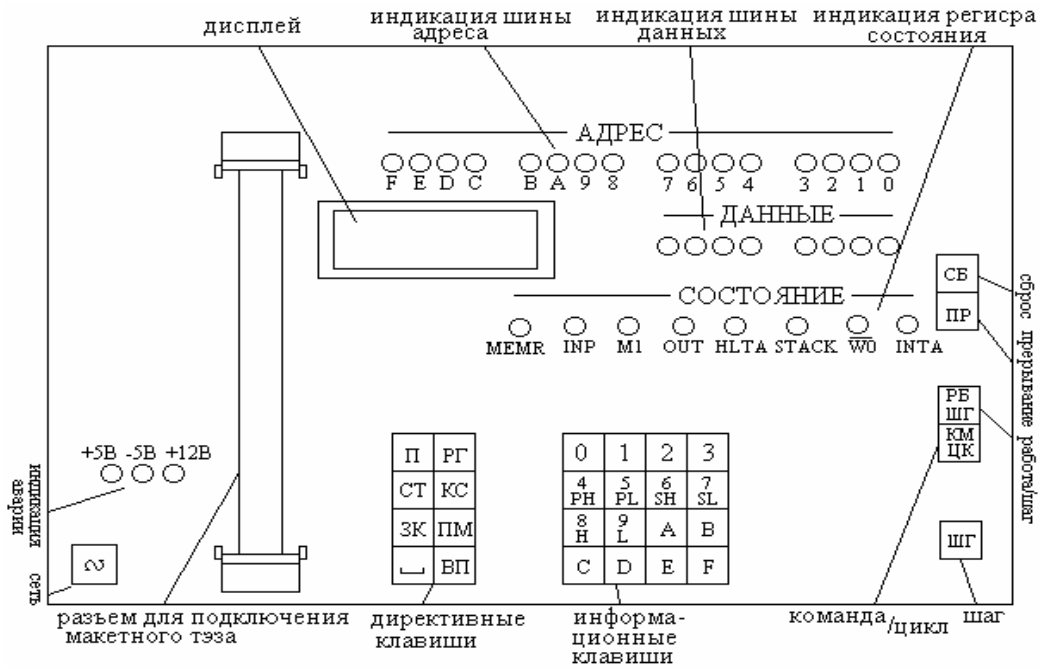


Рис 3. Лицевая панель микро-ЭВМ

Микро-ЭВМ состоит из следующих блоков: 1. блок центрального процессора. 2. блок памяти. 3. схема дешифрации. 4. схема пошагового выполнения программ. 5. клавиатура. 6. световая индикация и дисплей.

Блок памяти состоит из постоянного запоминающего устройства (ПЗУ) объемом 2Кбайт, реализованного на двух микросхемах К572РФ1 (или одной К572РФ2) с ультрафиолетовым стиранием и оперативного запоминающего устройства (ОЗУ) объемом 1Кбайт, реализованного на двух микросхемах КР541РУ2 (каждая микросхема имеет организацию 1024х4 бит). В ПЗУ хранится программа “Монитор”, занимающая 1Кбайт памяти (адреса 0000-03FF) и обеспечивающая ввод информации с клавиатуры пульта оператора, выполнения директив и вывод информации на дисплей. Еще 1Кбайт ПЗУ (адреса 0400-07FF) используется для хранения демонстрационных программ и программ организации временных задержек. ОЗУ используется для хранения изменяющихся программ и данных. ОЗУ занимает адреса 0800-0BFF и имеет емкость 1Кбайт. При работе с ОЗУ необходимо учитывать, что последние 54 ячейки ОЗУ использует при своей работе программа “Монитор”.

В процессе подготовки УМК к проведению учебного практикума он был дополнен имитаторами внешних устройств, смонтированных на ТЭЗе М1. В качестве простейших устройств ввода-вывода были использованы 8-разрядные регистры, выполненные на базе многорежимных буферных регистров (МБР) К589ИР12.

Схема подключения к микро-ЭВМ входного устройства, выполненного на базе МБР К589ИР12 (D1) и переключателей S1-S8, приведена на рис 4а. При замкнутом переключателе на вход регистра подается «О», а при разомкнутом — «1». Переключатели используются для имитации передачи данных от внешнего устройства. К регистру можно подключить светодиоды (HL_1 — HL_8) для индикации чисел, записанных в нем. На рис 4б приведена схема подключения выходного устройства к микро-ЭВМ, построенная на базе схемы К589ИР12 (D1). Светодиоды HL_1 — HL_8 указывают число, записанное в выходном устройстве.

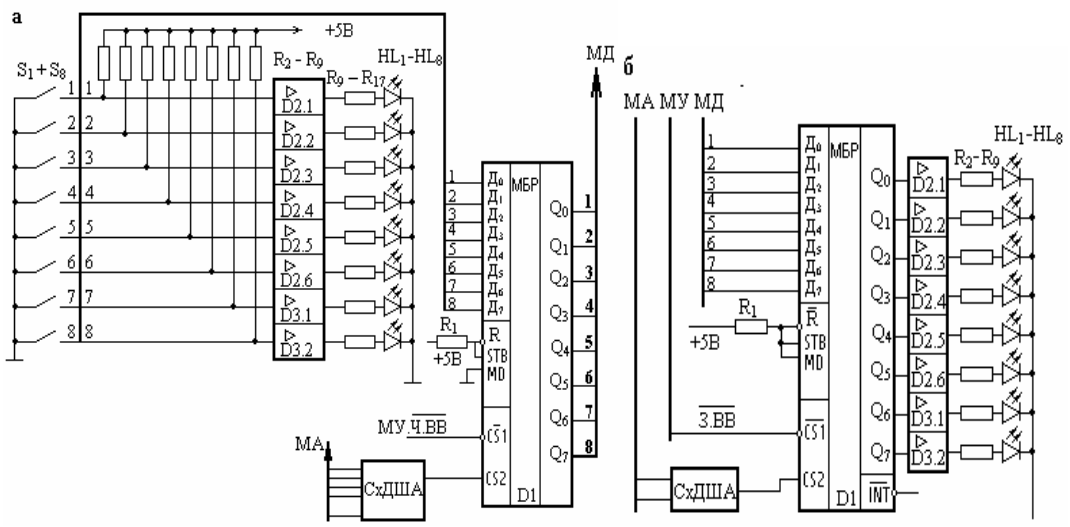


Рис.4. Схемы подключения входного (а) и выходного (б) устройств к микро-ЭВМ

Обмен данными между микро-ЭВМ и внешними устройствами может вызываться как в определенных местах в программе (программный ввод-вывод), так и по сигналам прерывания (ввод-вывод по прерываниям). В последнем случае подпрограмма обмена данными с внешним устройством будет вызываться за счет перевода микро-ЭВМ в режим обслуживания прерывания.

В качестве устройств ввода-вывода могут применяться и более сложные схемы, например программируемое устройство ввода-вывода информации в параллельном коде (КР580ВВ55). Например, в случае УМК, посредством КР580ВВ55 подключены семисегментный дисплей и клавиатура.

В микро-ЭВМ звуковые сигналы формируются простейшей схемой (рис. 5), на вход которой со звуковой частотой записываются по очереди «0» и «1». Устройство формирования звуковых сигналов имеет адрес 80.

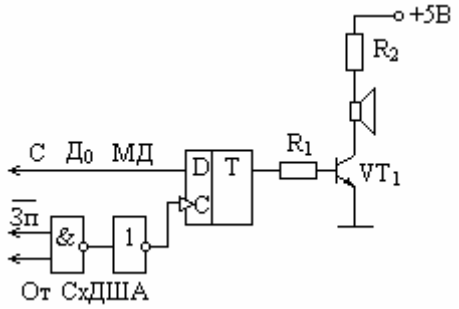


Рис. 5. Схема формирования звуковых сигналов в микро-ЭВМ

На лицевой панели УМК расположена клавиатура, дисплей и световая индикация (рис. 3). С клавиатуры пульта осуществляется вызов следующих директив: 1. чтение и модификация ячеек памяти (П); 2. чтение и модификация содержимого регистров МП (РГ); 3. вычисление контрольной суммы массива памяти (КС); 4. запоминание массива памяти константой (ЗК); 5. перемещение заданного массива памяти в адресном пространстве (ПМ); 6. выполнение программы пользователя с возможностью установки до двух точек останова (СТ). Клавиатура состоит из 24 клавиш, из них 8 клавиш директивные, а 16-информационные. Числа с информационных клавиш вводятся в шестнадцатиричном коде.

Блок питания микро-ЭВМ УМК вырабатывает стабилизированное напряжение +5В, -5В, +12В, необходимое для запитки всех энергетических частей микро-ЭВМ. Он имеет встроенную защиту от перегрузки по току, а также защиту от увеличения напряжения на выходах блока. На лицевой панели УМК загораются светодиоды индикации аварии. В этом случае УМК следует выключить и повторно включить через 10 с.

1.3. Структурная схема Функционирование Программная модель МП КР580ВМ80

Учебная микро-ЭВМ УМК построена на МП КР580ВМ80 (аналог-I8080). Структура МП КР580ВМ80 приведена на (рис.6).

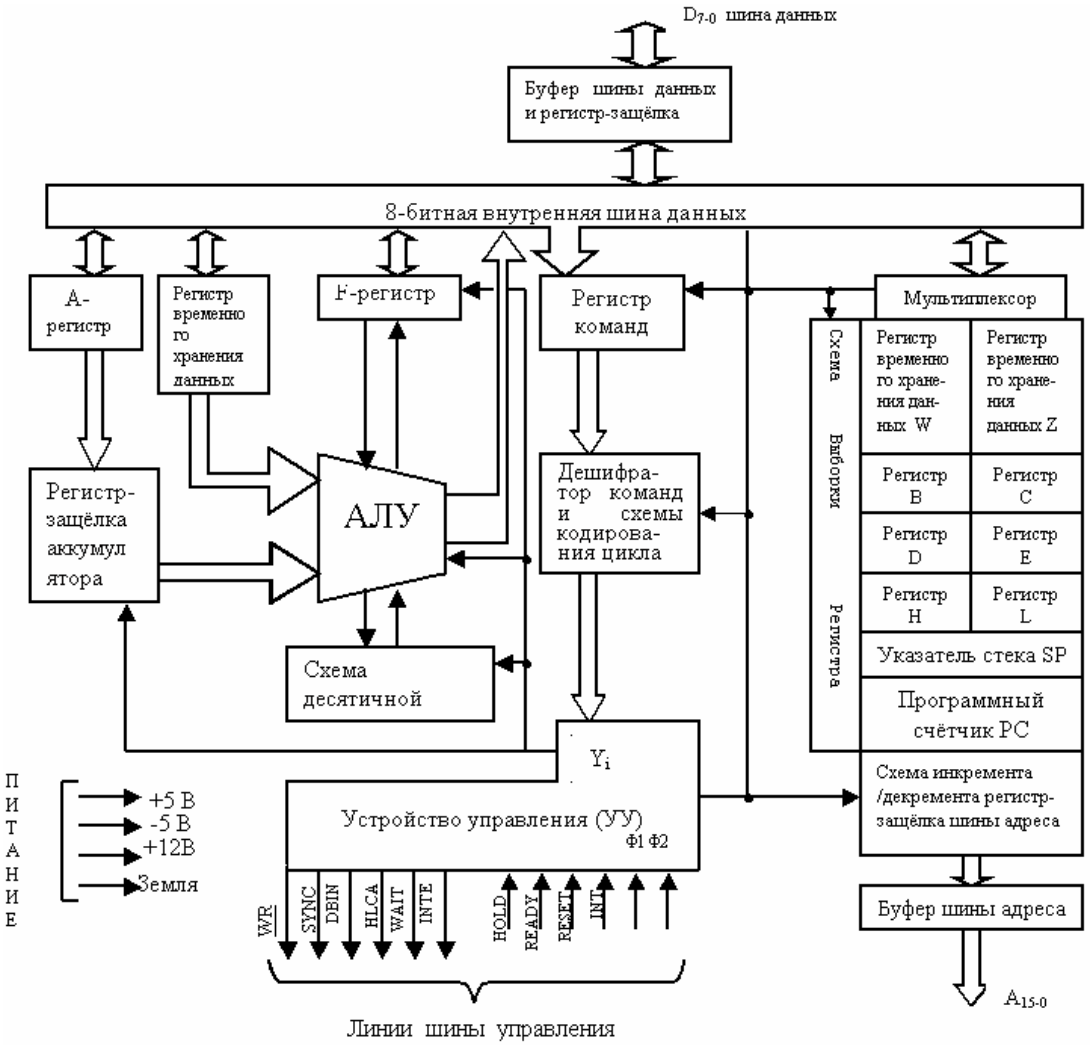


Рис. 6. Структурная схема МП КР580ВМ80

Структурной схемой микропроцессора называется совокупность составляющих его функциональных блоков и связей между ними.

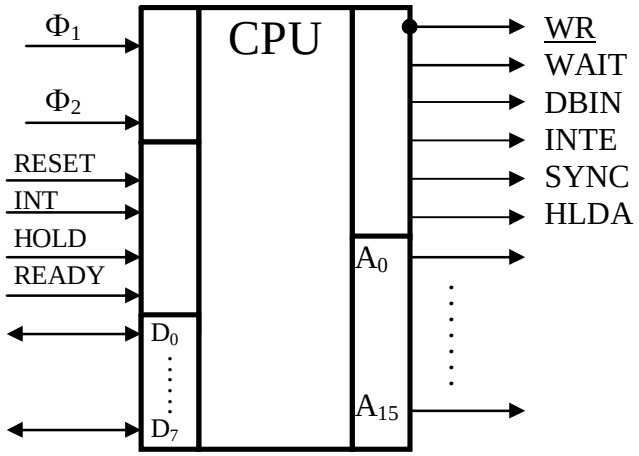


Рис.7 Условное обозначение микропроцессора I8080.

Восьмиразрядные регистры общего назначения B, C, D, E, H, L образуют внутреннюю память МП и используются для хранения промежуточных результатов. Программный счетчик PC (program counter) хранит адрес текущей адресуемой ячейки памяти (ЯП). Указатель стека *SP* (stack pointer) адресует последнюю занятую ячейку стека (стеком называется специально выделенная программистом область памяти для временного хранения данных и адресов, доступ к которой осуществляется в соответствии с принципом LIFO (Last Input First Output -последним пришел первым вышел)).

Арифметико-логическое устройство (АЛУ) предназначено для обработки двух 8-битовых операндов.

Два буферных регистра АЛУ (БР1, БР2) предназначены для приема и хранения операндов на время выполнения команд в АЛУ.

Схемы десятичной коррекции (СДК) позволяют привести результаты операции над двоично-десятичными данными, выполняемой в АЛУ, к двоично-десятичному представлению.

Аккумулятор (А) - регистр, используемый для хранения одного из операндов, передаваемых в АЛУ. Результат работы АЛУ по окончании обработки данных вновь помещается в аккумулятор.

Регистр признаков (РП) предназначен для отображения результата операции в АЛУ. Каждый бит регистра хранит какой-то один признак: бит Z - признак нуля, бит S - признак знака, бит P - признак чётности, бит C - признак переноса, бит *AC* - признак вспомогательного переноса. Формат РП представлен на рис.7.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	O	AC	O	P	1	C

РИС. 7. Формат регистра признаков.

Важнейшим из флажков является флажок С. Его наличие позволяет организовать на малоразрядном микропроцессоре обработку данных любой длины из последовательной обработки байтов операндов.

Все биты РП, за исключением бита АС, могут быть проанализированы командами условного перехода.

Наличие в системе команд микропроцессора команд условного перехода, осуществляющих выбор одного из двух направлений в зависимости от значения проверяемого флажка, придает МП-системе черты "интеллекта", адаптации к складывающейся ситуации.

Регистр команд (РК) хранит код поступившей команды и связан со сложной схемой дешифрации команды и управления машинными циклами, из которой информация о коде операции поступившей из памяти команды передается в устройство управления (УУ).

На УУ подаются сигналы синхронизации и внешние управляющие сигналы INT, HOLD, READY. УУ генерирует необходимые последовательности сигналов, коммутирующих схемы самого микропроцессора и поступающих в другие компоненты системы. Основу устройства управления МП КР580ВМ80 составляют программируемые логические матрицы (ПЛМ).

ФУНКЦИОНИРОВАНИЕ МП.

Определяющим для любого МП является состав его внешних выводов (табл. 3). Двенадцати сигналов управления, формируемых и принимаемых МП, недостаточно для эффективного управления МПС. Поэтому разработчики микропроцессора I8080 пошли по следующему пути: в начале всякого машинного цикла МП выставляет на шину данных байт слова состояния процессора (ССП), которое должно быть записано в специальный внешний регистр слова состояния (РГСС). Тем самым число управляющих сигналов, генерируемых МП,

увеличивается на 8. Основное назначение битов ССП - генерирование управляющих сигналов для внешних устройств.

Совокупность битов ССП характеризует тип начавшегося машинного цикла. Всего имеется 10 типов машинных циклов, различающихся значением байта ССП.

Объединяя определенным образом выводы $P_ГCC$ и сигналы DBIN,WR формируют в МПС магистраль управления (рис.8).

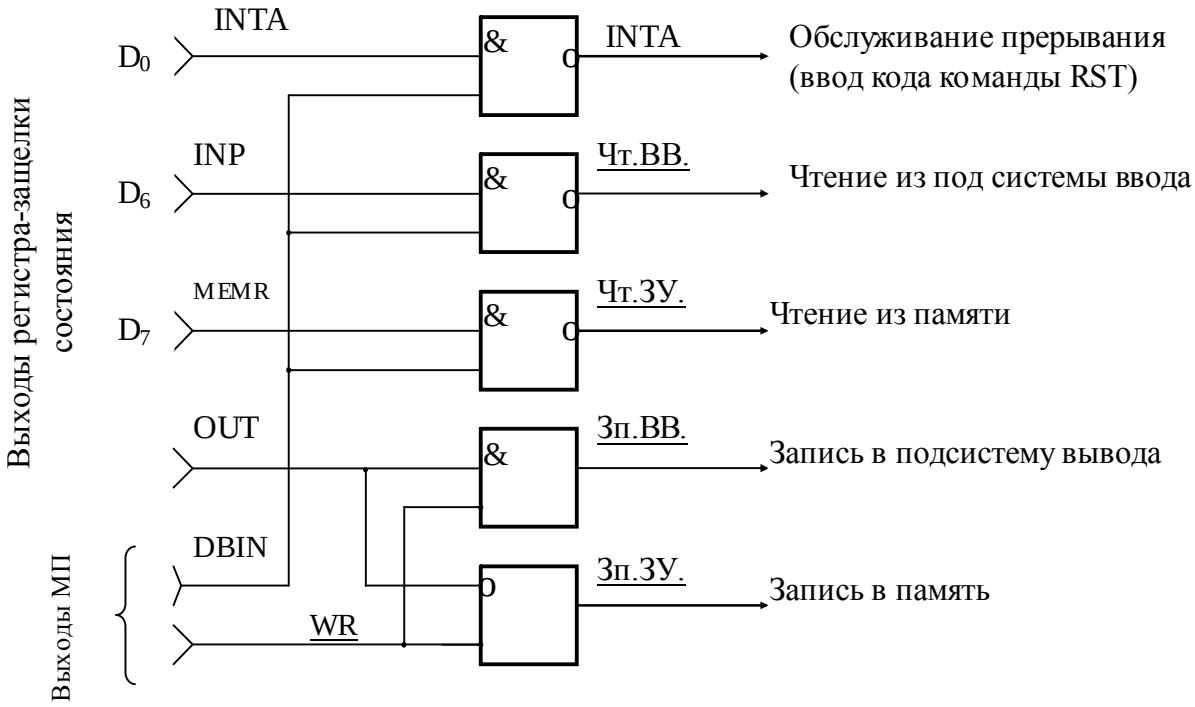


Рис.8.Схема формирования сигналов шины управления в системном контроллере.

Магистраль управления МПС обычно состоит из следующих выходов:

1. Чтение памяти Чт. ЗУ;
2. Запись в память Зп. ЗУ;
3. Чтение данных из УВВ Чт.ВВ;
4. Запись данных в УВВ ЗП. ВВ;
5. Обслуживание прерывания Об ПР.

Сигналы на указанных выходах формируются из записанного в соответствующий регистр слова состояния микропроцессора и его выходных сигналов DBIN, WR в системном контроллере (СК). Чт. ЗУ формируется аппаратно из сигналов DBIN (прием) микропроцессора и сигнала MEMR (память) слова состояния (рис. 8).

Зп. ЗУ - формируется из сигнала микропроцессора $\overline{WR}$ (выдача) и сигнала слова состояниям MEMR (память) (рис.8).

Зп. ВВ - это комбинация сигнала из слов состояния OUT и сигнала микропроцессора $\overline{WR}$ (выдача) (рис.8).

Чт. ВВ - можно сформировать из сигнала микропроцессора DBIN (прием) и разряда D6 INP (ввод) слова состояния (рис.8).

Сигнал Об. ПР формируется из сигнала DBIN микропроцессора и разряда D0 (INTA) слова состояния.

Таблица 3. Сигналы микропроцессора I8080

Обозначение	Наименование	Функциональное назначение и условия возникновения
1	2	3
Сигналы управления шинами		
DBIN	Разрешение приема информации с шины данных системы (выходной)	Шина данных ШД (0-7) МП находится в режиме приема информации
$\overline{\text{(WR)}}$	Признак выдачи информации на шину данных (выходной)	На шину данных ШД (0-7) МП выдана информация для записи в ЗУ или во внешнее устройство
Сигналы управления прямым доступом к памяти		
HOLD	Захват шин (входной)	Запрос на прямой доступ к памяти со стороны внешнего устройства
HLDA	Подтверждение захвата шин (выходной)	Признак допуска внешнего устройства к шинам данных и адреса системы. Выходы шин ШД (0-7), ША (0-15) МП в состоянии с высоким выходным сопротивлением
Сигналы управления прерываниями		
INT	Запрос прерывания (входной)	Запрос прерывания работы процессора. Проверяется в состояниях ожидания во время останова, а также в конце выполнения команды, если установлен триггер разрешения прерывания
INTA	Разрешение прерывания (выходной)	Разрешение прерывания. Соответствует состоянию внутреннего триггера разрешения прерывания (ТРП)
Сигналы управления асинхронной передачей данных		
READY	Готовность данных к передаче (входной)	Информация для обмена между ЗУ или внешним устройством и МП готова.

		При отсутствии сигнала готовности ГТ микропроцессор переходит в состояние ожидания, выходит из него при поступлении сигнала ГТ
WAIT	Ожидание (выходной)	Микропроцессор находится в состоянии ожидания
Сигналы общего назначения		
SYNC	Синхронизация (выход-ной)	Начало машинного цикла. По этому сигналу на шину данных ШД (0-7) микропроцессор выдает управляющее слово
RESET	Сброс (выходной)	Сигнал устанавливает в нуль счетчик команд, сбрасывает триггеры разрешения прерывания и захвата шин
φ1,φ2	Тактовые импульсы (входные)	Постоянно подаются на МП, обеспечивают тактирование
A I5-A0	Шина адреса	Предназначена для передачи адресов от МП в память и порты ввода-вывода
D7-D0	Шина данных	По этой шине осуществляется обмен любой информацией между компонентами МПС

Каждая команда МП выполняется в строго определенной последовательности, определяемой кодом команды, и ее выполнение синхронизируется во времени сигналами Ф1, Ф2 генератора тактовых импульсов (ГТИ). При использовании любого микропроцессора необходимо ясно представлять динамику его работы, т.е. на каких магистралях и когда МП будет выдавать ту или иную информацию. Это в дальнейшем поможет понимать работу схем, позволяющих согласовывать МП с различными внешними устройствами. Определим следующие понятия: цикл команды, машинный цикл, машинный такт.

Цикл команды (ЦК) - это время, необходимое МП для получения, декодирование и выполнение команды.

Машинный цикл (МЦ) - это время, требуемое для извлечения (записи) одного байта информации из (в) памяти или порт ввода/вывода или выполнение команды длиной в один байт.

Тактовым импульсом (ТИ) называется период синхросигналов Ф1, Ф2. Начало тактов определяется нарастающим фронтом сигнала

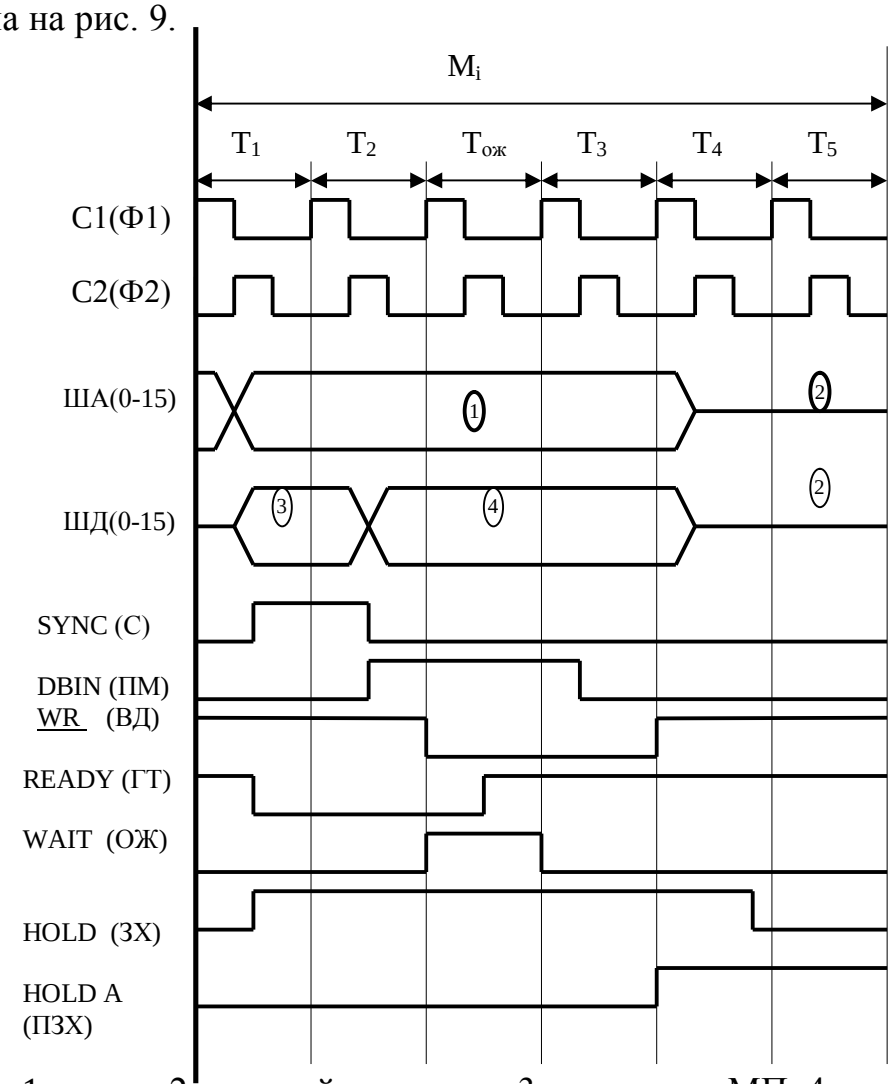
Ф1, а большинство действий в системе привязано к нарастающему фронту сигнала Ф2.

Между ЦК, МЦ, ТИ существуют следующие соотношения

ЦК = (1 + 5) МЦ

╚ =(3÷5) ТИ

Из него следует, что любая команда в зависимости от своего типа может состоять из 1 до 5 машинных циклов, а любой из 10 типов МЦ состоит из 3÷5 ТИ (причем действия МП в первых трех тактах для всех МЦ одинаковы). Временная диаграмма обобщенного МЦ приведена на рис. 9.



1- адрес; 2-высокий импеданс; 3- состояние МП; 4- данные.

Рис.9. Временная диаграмма выполнения обобщённого машинного цикла.

Во время такта T1 всякого машинного цикла содержимое PC посылается на шину адреса, вырабатывается сигнал SYNS, а на шину данных выставляется байт состояния, характеризующий начинающийся машинный цикл. В такте T2 анализируется сигнал на входах READY, HOLD , а также проверяется, не находится ли МП в режиме останова. Если сигнал READY равен "1", то МП входит в такт T3, в противном случае - в состояние ожидания, пока на вход READY не поступит сигнал готовности. В такте T3 МП принимает с ШД или выдает на ШД информацию. В тактах T4, T5 МП производит внутренние пересылки и преобразования.

Если этого не требуется, такты T4, T5, в МЦ могут отсутствовать. В конце последнего машинного цикла выполнения любой команды анализируется сигнал -INT. Если этот сигнал равен "1" и прерывание разрешено (на выходе INTE - логическая "1"), то МП начинает специальный цикл прерывания, во время которого содержимое PC не изменяется, а прерывающее внешнее устройство посылает в МП код команды RST<N> с адресом прерывающей программы.

ПРОГРАММНАЯ МОДЕЛЬ МИКРОПРОЦЕССОРА I8080.

Программист, создавая программу для микропроцессора, имеет дело лишь с его системой команд и программно доступных регистров. Поэтому упрощенное представление микропроцессора I8080 с точки зрения программиста имеет вид (рис.10). В неё включены только те регистры, к которым можно обращаться программно.

Рис.10. Программная модель микропроцессора.

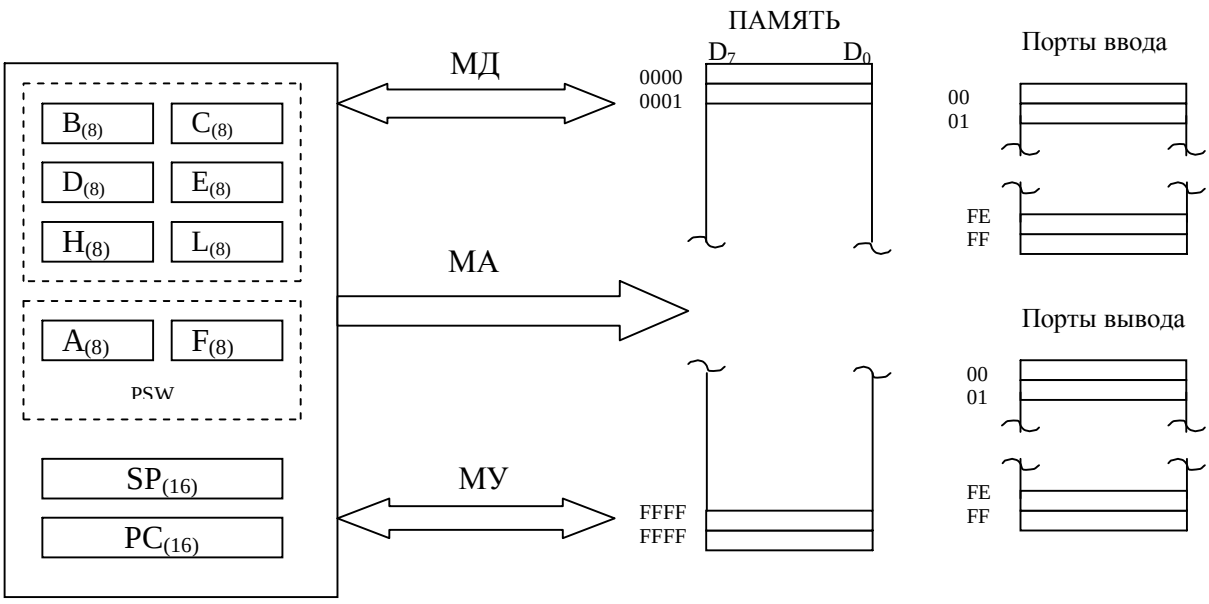


Рис.11. Объединение РОНов в регистровые пары.

Всего для программиста доступно 10 регистров: шесть 8-разрядных регистров общего назначения (B, C, D, E, H, L), 8-разрядные регистры A (аккумулятор), F (регистр признаков) и 16-разрядные регистры SP (указатель стека) и PC (счетчик команд).

Особенностью рассматриваемого микропроцессора является возможность 8-битных РОНов объединяться в 16-битные регистровые пары (рис.11). Пары B, D, H могут хранить 16-битные операнды или 16-битные адреса (т.е. выступать в виде указателей памяти). Основным указателем памяти служит H-пара. Ячейка памяти, адрес которой определяется содержимым H-пары, обозначается M (memory - память). Имея шестнадцатиразрядную магистраль адреса, микропроцессор может адресовать 64 К ячеек памяти (64 кБайт).

Карта памяти представлена на рис. 12.

Адреса	Область памяти ЗУ	
0000 ↕ 03FF	Программа “Монитор”	
0400 ↕ 07FF	ПЗУ пользователя	2К
0800 ↕ 0BCA	Область ОЗУ для хранения изменяющихся программ и данных пользователя	
0BCB	Верхушка стека	1К
↕ 0BFF	Область ОЗУ программы “Монитор”	
0C00 ↕ FFFF	Неиспользованные адреса	61К

Адрес	Порты ввода
20	Входной регистр
FA	Порт С ППИ дисплея, служащий в качестве регистра чтения клавиатуры (РгЧк)

Адрес	Порты вывода
30	Выходной регистр
80	Регистр схемы выдачи звуковой сигнализации
F8	Порт А ППИ дисплея и клавиатуры, служащий в качестве регистра сканирования
F9	Порт В ППИ дисплея и клавиатуры, служащий в качестве регистра сегментов дисплея

Рис.12 Карта памяти

Микропроцессоры от INTEL при адресации устройств памяти и устройств ввода-вывода (УВВ) используют принцип выделения подсистем. В этом случае вся микропроцессорная система разделяется на подсистемы (память, ввод, вывод). В рамках каждой подсистемы используется вся совокупность адресов.

В МПС на базе МП I8080 для адресации внешних устройств (ВУ) выделяется самостоятельное адресное пространство 00 до FF, которое не накладывается на адресное пространство памяти.

Портом будем называть адресуемый байтовый регистр. Таким образом, МП может адресовать 256 портов ввода и 256 портов вывода.

Основные элементы портов - регистры-защелки. Это обычный параллельный регистр, отслеживающий входной код при одном значении управляющего сигнала и фиксирующий его при противоположном.

Обращение к портам ввода-вывода осуществляется только 2-байтными командами IN<N> OUT<N>, где <N> адрес порта. При выполнении этих команд адрес порта, взятый со второго байта, загружается в регистры временного хранения W и Z и выдается одновременно на адресные линии AI5**…**A8 и A7**…**A0 . Такой прием разработчиками МП использовался для выравнивания нагрузки на линии адреса, так как одну часть портов можно теперь подключать к линиям AI5**…**A8, а другую - к линиям A7**…**A0.

1.4.ФОРМАТЫ КОМАНД, МЕТОДЫ АДРЕСАЦИИ И СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА I8080.

Каждый МП характеризуется прежде всего своей системой команд. Система команд - это полный перечень элементарных действий, которые способен выполнять МП. Система команд МП I8080 включает команды форматом (1**…**3) байта. Первый байт любой команды содержит код операции (КОП). Он определяет формат команды и те действия, которые должны быть произведены МП над операндами, а также метод адресации операндов. Операнды могут обозначать адреса ячеек памяти, регистры или непосредственно данные. Команда может не иметь ни одного операнда, иметь один или два операнда:

однобайтная команда (D7-D0 - код операции);

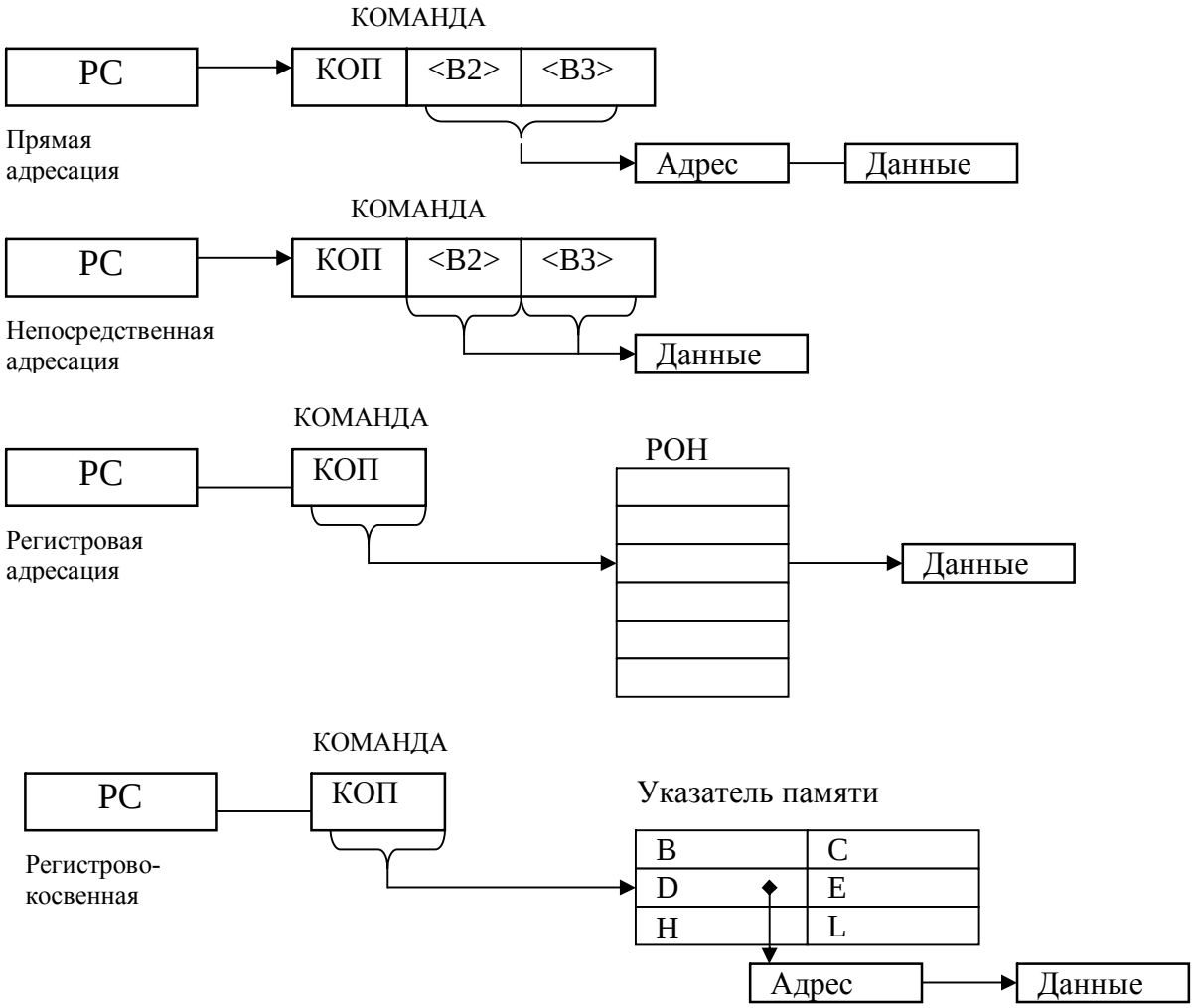
двухбайтная команда (D7-D0 – код операции; D7-D0 – данные или адрес);

трехбайтная команда (D7-D0 – код операции; (D7-D0) (D7-D0) – данные или адрес).

В поле операнда всех команд каким-либо образом определяются данные, участвующие в операции, или операнды. Способ определения операнда называется режимом адресации. МП I8080 поддерживает пять режимом адресации:

1.Прямая.
2.Непосредственная.
3.Регистровая.
4.Регистровая-косвенная
5.Стековая

Режимы адресации микропроцессора показаны на рис.13



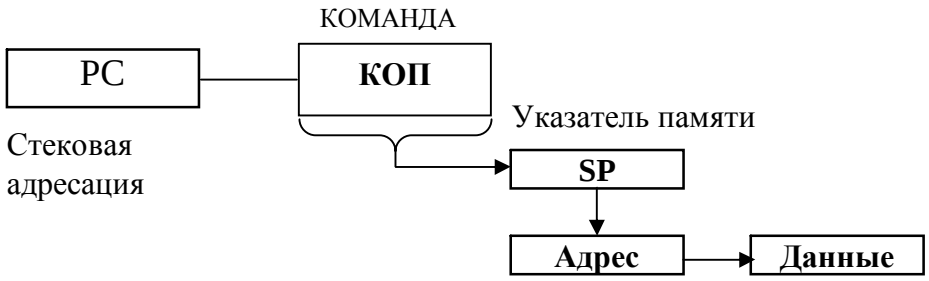


Рис.13. Режимы адресации.

Наиболее простой, но и наиболее неэкономичной, является прямая адресация (полная адресация, расширенная адресация, абсолютная адресация): в поле операнда содержится полный 16-битный адрес памяти. С помощью прямой адресации можно обращаться к любой ячейке адресного пространства.

При необходимости включения в программу фиксированных значений (кодов символов, слов-масок и др.) применяется непосредственная адресация, в которой операндом является второй байт <B2> самой команды. В микропроцессоре I8080 допускается непосредственно адресовать 16-битные слова (<B2> - младший байт, <B3> - старший байт), используемые для инициализации указателей памяти.

При использовании регистровой адресации операндом является содержимое адресуемого в команде РОНа. Команды с таким режимом адресации оказываются короткими (1-байтными) и выполняются достаточно быстро. В 1-байтных командах передач MOV r_2,r_1 адресуются два РОН, так как их адреса имеют всего по 3 бита.

При выполнении 1-байтных команд с регистровой адресацией часто подразумевается, что операнд находится в определенном внутреннем регистре микропроцессора и его специально адресовать не нужно. Например, все команды сдвига микропроцессора К580 оперируют содержимым аккумулятора. Во всех бинарных операциях преобразования данных подразумевается, что одним из операндов является содержимое аккумулятора и что результат операции загружается в аккумулятор. Регистровая адресация применяется также в унарных операциях типа инвертирования аккумулятора.

Удобным средством, позволяющим компактно адресовать все пространство памяти, является регистрово-косвенная (часто

называемая косвенная) адресация. В этом случае команда указывает на пару регистров BC, DE, HL в которых к моменту обращения должен быть записан адрес ячейки памяти, содержащий обрабатываемые данные. В микропроцессоре I8080 основным указателем памяти служит регистровая H-пара. Следует иметь ввиду, что при написании мнемоники ассемблерных команд в поле операндов часто встречается буква M под которой понимают содержимое ячейки памяти, адресуемой косвенным образом посредством регистровой пары HL В командах загрузки и запоминания аккумулятора в качестве указателей памяти могут выступать B- и D-пары.

Косвенная адресация широко применяется при обработке регулярных структур данных типа массивов. В указатель памяти сначала загружается базовый (начальный) адрес массива, а переход к последовательным элементам массива осуществляется с помощью инкремента указателя памяти.

Стековая адресация, строго говоря, является разновидностью регистрово-косвенной адресации, при которой в качестве указателя ячейки памяти используется шестнадцатиразрядный указатель стека SP.

За исключением режимов обслуживания прерываний и выполнения команд передачи управления МП выполняет программу, считывая команды последовательно из ячеек памяти. В командах передачи управления адрес, к которому будет обращаться МП, указывается с помощью прямой адресации.

Одним из недостатков микропроцессора I8080 считается отсутствие индексной адресации. В данном режиме исполнительный адрес образуется путем сложения второго байта команды, рассматриваемого как целое без знака и называемого смещением и содержимого внутреннего 16-битового индексного регистра. Индексную как и косвенную адресацию удобно применять в циклических программах. Заметим, что при необходимости индексную адресацию в микропроцессоре I8080 можно реализовать программно, разумеется, за счет непроизводительных потерь времени и увеличения длины программы.

Команды межрегистровых передач

В любой прикладной программе возникает необходимость передать содержимое регистра-источника (S-регистра) в регистр-получатель (D-регистр). Эту функцию выполняют многочисленные команды межрегистровых передач (пересылок) D←(S) . Содержимое S-регистра и состояние флажков не изменяются, поэтому иногда команды данной группы называются командами копирования содержимого S-регистра в D-регистре.

Однобайтные команды передачи имеют вид MOV D,S. В качестве D и S можно указывать любой РОН и аккумулятор. Например, команда MOV B, A с кодом 01 000 111 передает содержимое аккумулятора в регистр B. С помощью косвенной адресации команда MOV может обращаться к памяти, используя в качестве указателя памяти регистры (H, L). Например, команда MOV M,C с кодом 01 110 001 передает содержимое регистра C в ячейку памяти, адресуемую H-парой: [H,L]←(C). Заметим, что команды передачи содержимого внутреннего регистра микропроцессора в память часто называют командами запоминания. Команда MOV L,M имеет следующий смысл: L←([H,L]). Команды передачи содержимого ячейки памяти во внутренний регистр обычно называются командами загрузки.

Однобайтные команды LDAX B и LDAX D загружают в аккумулятор содержимое ячейки памяти, адресуемой соответственно парами регистров (B, C) и (D, E). Команды запоминания STAX B и STAX D*)* осуществляют передачу содержимого аккумулятора в память.

Особенностью микропроцессора I8080 являются 1-байтные команды передач 16-битных операндов. Команды SPHL и PCHL передают содержимое регистров (H, L) в указатель стека и программный счетчик PC соответственно. Интересна команда обмена XCHG , производящая обмен содержимого регистров (D , E) и (H, L) : (D, E)↔(H,L).

В 2-байтных командах передач MVI r, <B2> применяется непосредственная адресация: второй байт команды <B2> передается в адресуемый регистр: r← <B2>. В качестве r можно указывать любой РОН, аккумулятор или ячейку памяти M, адресуемую регистрами (H, L).

В 3-байтных командах передач используется прямая и непосредственная адресации. Команда LDA загружает в аккумулятор содержимое ячейки памяти, адресуемой вторым и третьим байтами команды: *A*←([<B3><B2 >]) , а команда запоминания S TA производит противоположную передачу: [<B3><B2>]←(A). Команда LHLD применяется для загрузки основного указателя памяти - регистров (H, L) из двух смежных ячеек памяти L←([<B3><B2>]), H←([<B3><B2>+1]). Команда SHLD запоминает содержимое (H, L) в двух смежных ячейках памяти.

Трехбайтные команды **LXI** rp, где rp означает регистровые пары B, D, H и указатель стека SP, применяются для загрузки начальных значений (инициализации) внутренних указателей памяти. Например, команда **LXI B** имеет следующее содержание: C← <B2>,B← <B3>.

В группу команд межрегистровых передач включаются 2-байтные команды ввода IN и вывода OUT. Второй байт <B2> этих команд представляет собой 8-битный адрес порта ввода (вывода), а получателем (источником) может быть только аккумулятор. Таким образом, микропроцессор позволяет адресовать до 256 портов ввода и до 256 портов вывода. Конечно, это не означает, что к МП-системе можно подключать по 256 устройств ввода и вывода, так как для каждого устройства кроме информационных портов, необходимы каналы обмена управляющей информацией.

Реализация ВВ с помощью специальных команд называется аккумуляторным (изолированным) ВВ. Здесь адресное пространство портов ВВ отделено от адресного пространства памяти. При организации в МП-системе ВВ, отображенного на память, порты ВВ считываются ячейками памяти с определенными адресами. Следовательно, все команды с обращением к памяти, адресующие порты ВВ, становятся командами ВВ и специальные команды не нужны.

Целесообразно обратить особое внимание на команды передач между внутренними регистрами микропроцессора, так как они наиболее часто встречаются в программах. Пусть, например, возникла необходимость передать содержимое регистров (D, E) в указатель стека SP . Специальная команда такой передачи в системе команд отсутствует. Однако нетрудно установить, что передачу SP←(D, E)- осуществляет последовательность из трех однобайтных команд: XCHG, SPHL, XCHG.

Удобным средством представления команд данной группы являются матрицы источников-получателей. Каждая строка матрицы соответствует некоторому S-регистру, а каждый столбец - некоторому D-регистру МП-системы. Элементами матрицы служат команды, осуществляющие передачи D← (S) . Рекомендуется самостоятельно построить матрицы источников-получателей для 8- и 16-битных операндов.

В группу команд межрегистровых передач следует отнести команды PUSH загрузки в стек и команды POP извлечения из стека.

Команды арифметических операций

Вычислительные возможности микропроцессора I8080 ограничены простыми командами сложения и вычитания 8-битных операндов. Операции умножения и деления, а также операции с другими форматами данных реализуются с помощь подпрограмм, что приводит к значительному уменьшению производительности МП-системы.

Операнды представляются в формате целого со знаком в дополнительном коде с диапазоном изменений от -128 до +127. Напомним, что старший бит, представляет собой знак S, и сумма дополнительных кодов операндов равна дополнительному коду суммы, если переполнение исключено, а возникающий перенос из знакового бита игнорируется. В арифметических операциях одним из неадресуемых операндов является содержимое аккумулятора, результат загружается в аккумулятор и в соответствии с полученным результатом производится модификация всех флажков.

Сложение и вычитание осуществляются однобайтными командами ADD r : A←(A)+(r) и
SUB r : A← (A)-(r). Флажок C в операции вычитания устанавливается, если уменьшаемое меньше вычитаемого, т.е. он превращается во флажок заема.

Во многих применениях базовой длины слова микропроцессора недостаточно, и приходится вводить многобайтные числа с последовательной обработкой отдельных байтов. В памяти такие числа хранятся в смежных ячейках и адресуются по младшему байту. При обработке многобайтных чисел очень удобны команды сложения с переносом ADC r : A←(A)+(r)+C и вычитания с заемом SBB r : A← (A)-(r)-C.

В системе команд имеются 2-байтные команды сложения и вычитания с непосредственной адресацией: адресуемым операндом

служит второй байт <B2> команды. Это команды ADI: A←(A)+<B2>, SUI : A←(A)-<B2>, ACI : A←<B2>FC, SBI : A ←(A)-<B2>-C.

Команды арифметического сравнения CMP r: (A)-(r) и CPI: (A)-<B2> производят вычитание из содержимого аккумулятора значения адресуемого операнда, модифицируют по результату все флажки, но не изменяют содержимое аккумулятора. Наличие команд такого неразрушающего сравнения упрощает, например, поиск в массиве заданного слова-эталона. Значение отыскиваемого слова загружается в аккумулятор, базовый адрес массива загружается в регистры (H, L), а затем с помощью команды CMP M элементы массива сравниваются со словом-эталоном. Факт равенства фиксируется по флажку Z = I .

Для несложной обработки больших массивов числовых данных удобно ввести формат двоично-кодированных десятичных чисел, или BDC - чисел. В этом формате байт содержит две десятичные цифры (тетрады) в коде 8421. Сложение BDC -чисел выполняется в два этапа:

- операнды складываются, как двоичные числа;
- производится коррекция в общем случае неправильного промежуточного результата.

Однобайтная команда десятичной коррекции аккумулятора D AA в два приема корректирует 8-битное значение в аккумуляторе на две BDC - цифры с правильной установкой флажка C:

1. Если младшая тетрада содержит число, большее 9, или если установлен флажок межтетрадного переноса AC = 1, то содержимое аккумулятора увеличивается на 6.

2. Если после этого старшая тетрада аккумулятора содержит число, большее 9, или если установлен флажок C, то в старшую тетраду прибавляется 6.

Например, промежуточный результат двоичного сложения BDC -чисел 79 и 39 имеет вид 1011 0010 (C = 0, AC = 1), а после выполнения команды D AA получаем 0001 1000 (C = 1), т.е. правильный результат 118.

Следует специально подчеркнуть, что команда D AA не корректирует результат двоичного вычитания.

При программировании часто возникает необходимость увеличения (инкремента) или уменьшения (декремента) значения операнда на единицу. Эти действия легко осуществляются командами ADI и SUI , второй байт <B2> ,который равен единице. Для сокращения длин Программ и времени их выполнения в системе команд предусмотрены

1-байтные команды инкремента INR r : r← (r)+1 и декремента DCR r : (r)-1→ r. Эти команды не изменяют значения флажка C.

Особенностью микропроцессора I8080 являются арифметические операции с 16-битными операндами. В них операнды считаются целыми без знака, и значения всех флажков не изменяются. Команды инкремента INX rp и декремента DCX rp позволяют увеличить или уменьшить содержимое регистров (B, C), (D , E), (H, L) и указателя стека SP на единицу. Команда двойного сложения DAD rp : H,L← (H,L)суммирует содержимое регистра (H, L) и адресуемого 16-битного регистра. Заметим, что команда DAD H эквивалентна удвоению содержимого регистров (H, L), т.е. сдвигу на один разряд влево.

Команды логических операций и сдвигов

Логические операции являются поразрядными, т.е. выполняются независимо для всех 8 бит операндов. Неадресуемый операнд находится в аккумуляторе, туда же загружается результат операции. По операции модифицируются состояния всех флажков, кроме флажка переноса C, который никогда не может быть установлен в единицу и поэтому принудительно сбрасывается.

При выполнении команды ANA r производится поразрядная конъюнкция операндов. Данная команда применяется для проверки значения определенного бита слова в аккумуляторе с помощью другого слова-маски. Если, например, необходимо проверить состояние второго бита A_2, то маска должна иметь вид 0000 0100. После выполнения команды ANA по флажку Z, можно судить о состоянии бита A_2 : $Z=\underline{A_2}$. Кроме того, с помощью команды ANA , используя слово-маску, можно сбрасывать определенные биты слова в аккумуляторе.

Команда ORAL r : A←(A)v(r)осуществляет поразрядную дизъюнкций операндов. Эта команда, а применяется для установки определенных битов слова в аккумуляторе с помощью слова-маски, а также упаковки слова из полей других слов. Например, результатом операции ORA с операндами 0000 $X_1\,X_2\,X_3\,X_4$ и $Y_7\,Y_6\,Y_5\,Y_4$ 0000 будет упакованное слово $Y_7\,Y_6\,Y_5\,Y_4\;X_4\;X_3\,X_2\,X_1$.

Команда ИСКЛЮЧАЮЩЕГО ИЛИ XRA r: A ←(A)+(r) производит поразрядное сложение операндов по mod 2. Команда XRA применяется для инвертирования определенных битов с помощью

слова-маски на основе тождества 1+X=$\overline{X}$. Например, для инвертирования седьмого и первого битов маска имеет вид 1000 0010. Другое применение команды XRA связано со сравнением слов на абсолютное равенство. В единственном случае, когда операнды поразрядно совпадают, результат операции содержит нули во всех разрядах (согласно тождеству X+X=0) о чем сигнализирует флажок Z =1.

Двухбайтные команды с непосредственной адресацией соответствующих логических операций имеют мнемонические обозначения ANI, ORI, XRI. Операндами этих команд являются содержимое аккумулятора и второй байт <B2> команды, а результат загружается в аккумулятор.

Во многих случаях при выполнении программ необходимо проверять или изменять (маскировать) состояние одного или нескольких разрядов числа в аккумуляторе. В МП КР580ВМ80 отсутствуют команды работы с отдельными битами операндов. Однако эту процедуру можно выполнить с помощью следующих операций:

1) логического умножения числа в аккумуляторе и маски, которое очищает разряд числа, если в соответствующем разряде маски будет записан 0, и не изменяет его, если в разряде маски записана 1;

2) логического сложения числа в аккумуляторе и маски, которое устанавливает разряд числа в 1, если в таком же разряде маски будет записана 1, и не изменяет его, если в этом разряде записан 0;

3) логического «исключающего ИЛИ» числа в аккумуляторе и маски, которое инвертирует содержание разряда числа, если в соответствующем разряде маски записана 1, и не изменяет его, если в этом разряде записан 0.

Проведение логических операций возможно также с содержимым аккумулятора и внутренними регистрами МП БИС. В этом случае команды — однобайтные. При выполнении всех логических команд задействуются разряды Z, S, P, AC регистра признаков (в разряд C записывается 0). Это позволяет проверять состояние любого разряда числа и выполнять условные переходы в программах. Условные переходы организуются в программах с помощью команд условных переходов. При выполнении этих команд МП БИС проверяет состояние соответствующего разряда регистра состояния. Если при проверке условие не подтверждается, то выполняется следующая по

порядку команда программы. Все команды условных переходов — трехбайтные: первый байт содержит код команды, второй и третий байты — адрес передачи управления. Таким образом, команды условных переходов позволяют строить ветвящиеся алгоритмы и в зависимости от текущего значения результата выполнения программы переходить на различные участки программы.

Операндом унарных 1-байтных команд сдвига является содержимое аккумулятора, в котором формируется результат. Сдвиги выполняются влево и вправо только на один разряд. В зависимости от того, что помещается в освобождающийся при сдвиге бит и как используются выдвигающийся (спадающий) бит и флажок переноса, вводится несколько команд сдвигов.

В командах циклического сдвига (ротациях) влево RLC и вправо RRC выдвигающийся бит помещается в освобождающийся и, кроме того, фиксируется во флажке переноса C. В командах сдвига через перенос влево RAL и вправо RAR выдвигающийся бит помещается во флажок переноса C, а текущее значение флажка C передается в освобождающийся бит.

К логическим операциям можно отнести 1-байтные команды инвертирования аккумулятора CMA: A←(A)установки флажка переноса STC: C←1 и инвертирования флажка переноса CMC: C←(C)

Команды передачи управления.

К окончанию текущей команды, а точнее - после выборки команды из памяти в программном счетчике PC сформирован адрес следующей по порядку команды. При естественном порядке выполнения команд, соответствующем простейшим линейным программам, производятся выборка и исполнение этой следующей команды формируется адрес следующей команды и т.д. Однако линейные прикладные программы на практике не встречаются. В разветвляющихся и циклических программах и при использовании подпрограмм приходится выполнять не следующую по порядку команду, а команду, находящуюся в другой ячейке программной памяти. Для этого достаточно загрузить в PC адрес новой ячейки, называемый адресом перехода. Такая процедура называется передачей управления, а специальные команды, которыми она реализуется,

называются командами передачи управления, или командами управления программой. В системе команд микропроцессора I8080 предусмотрен обширный набор команд передачи управления.

Команда безусловной передачи управления без возврата JMP , называемая также безусловным переходом или просто переходом, состоит из байта кода операции и двух байтов полного 16-битного адреса перехода. При ее выполнении адрес перехода загружается в PC, а текущее содержимое PC теряется: PC←<B3><B2>. Данная команда, соответствующая оператору GOTO, позволяет передать управление любой ячейке адресного пространства. Современная методика создания структурированных программ рекомендует использовать минимум команд – JMP.

Трехбайтные команды условной передачи управления без возврата, называемые также условными переходами или разветвлениями (и даже развилками!), осуществляют передачу управления только при удовлетворении некоторого условия, заданного в коде операции. Если условие не удовлетворяется, то передачи управления не происходит, а выполняется следующая по порядку команда, т.е. в этой ситуации команда разветвления эквивалентна холостой команде.

Проверяемым условием является текущее значение одного из флажков, указываемых в коде операции. Для удобства программирования предусмотрены команды разветвлений, осуществляющие передачу управления по единичному и нулевому значению каждого из флажков, кроме флажка AC. Например, команда JZ передает управление, т.е. осуществляет загрузку PC ← <B3 ><B2> , если флажок Z = 1, а команда JNZ - если флажок Z =0. Аналогичные парные команды имеются для флажков C (JC, JNC), S (JM, JP) и P (JPE, JPO). Всего, таким образом, получается восемь команд разветвлений.

Трехбайтные команды безусловной передачи управления с возвратом, называемые также вызовами CALL, состоят из байта кода операции и адреса перехода, представляющего собой начальный адрес подпрограммы. При выполнении вызова необходимо временно запомнить текущее содержимое программного счетчика (т.е. адрес команды, следующей за командой CALL, и называемый адресом возврата), загрузить адрес перехода команды CALL в PC, а после выполнения подпрограммы передать адрес возврата в PC. Последняя функция реализуемая специальной 1-байтной командой возврата

RETURN . Таким образом, вызов одной подпрограммы и возврат из нее осуществляется сравнительно простыми средствами - необходим 16-битный регистр для временного запоминания адреса возврата.

Ситуация несколько усложняется, если вызванная подпрограмма вызывает еще одну, а эта вторая - третью и т.д. (организуются вложенные подпрограммы). Фактически проблема вложения сводится к запоминанию адресов возврата в порядке вызовов подпрограмм и к возврату их в PC по мере выполнения подпрограмм. Для этого необходимо блок регистров, работающий таким образом, что последний включенный в него адрес возврата извлекается первым. Такой блок регистров называется стеком, магазином или буфером LIFO. Максимальное число запоминаемых в стек слов называется его глубиной.

В МП-системах на базе микропроцессоров I8080 для стека выделяется специальная область ЗУПВ, и его глубина считается неограниченной. Для идентификации последней занятой ячейки стека, называемой его верхушкой, в микропроцессоре имеется внутренний 16-битный регистр - указатель стека SP .

При выполнении команды PUSH rp в ячейку памяти с адресом (SP) - 1 записывается содержимое восьми старших, разрядов регистровой пары rp, а в ячейку с адресом (SP)-2 содержимое восьми младших разрядов этой регистровой пары. Содержимое регистра SP уменьшается на 2 (стек "растет вверх", в область меньших адресов). Когда выполняется команда POP rp извлечение данных из стека, содержимое верхушки стека передаётся в младшие 8 бит регистровой пары rp , а содержимое ячейки стека, адрес которой на единицу больше адреса верхушки (записанного в регистре SP), передается в старшие 8 бит регистровой пары rp . После этого содержимое регистра SP увеличивается на 2. Для правильной работы стека команды PUSH и POP обязательно должны быть парными. Иллюстрация работы стека с подвижной верхушкой показана на рис.16.

Таким образом, команда CALL фактически является командой PUSH PC с последующей загрузкой в PC второго и третьего байтов команды CALL. Выполнение ее несколько усложняется тем обстоятельством, что для запоминания в стеке 16-битного адреса возврата приходится использовать две 8-битные ячейки. Поэтому команда CALL производит следующие действия:
[(SP)-1]← (PC H), [(SP)-2]← (PC L), PC← <B3><B2>, SP← (S)-2, где (SP)- содержимое указателя стека до выполнения команды CALL , а

(PC H) и (PC L) соответственно старшие и младшие 8 бит программного счетчика. Однобайтная команда возврата RETURN эквивалентна команде POP PC: PC L← ([SP]), PC H←([SP+1]), SP← (SP)+2.

В командах условных вызовов и возвратов проверяется состояние указанного в коде операции флажка и, если оно удовлетворяет условию, осуществляются вызов и возврат, а если не удовлетворяет - выполняется следующая по порядку команда. Имеются команды, проверяющие единичное и нулевое значения каждого из четырех флажков:

- флажок Z: вызовы CZ, CNZ; возвраты RZ, RNZ,
- флажок C: вызовы CC, CNC; возвраты RC,RNC,
- флажок S: вызовы CM, CP; возвраты RM, RP;
- флажок P: вызовы CPE, CPO; возвраты RPE,RPO.

В системе команд микропроцессора имеется особая 1-байтная команда вызова, предназначенная для обработки прерываний и введения контрольных точек (разрывов) при отладке программ. Она называется рестартом (повторным запуском) и имеет мнемоническое обозначение RST . В коде операции рестарта 11AAA111 три бита AAA формируются подсистемой прерываний или указываются программистом. Выполнение команды рестарта сводится к двум действиям:

- текущее содержимое PC загружается в стек;
- в PC передается код 00000000 00AAA000.

Следовательно, в зависимости от значений AAA микропроцессор переходит к одной из восьми ячеек, находящихся в первых адресах памяти: 0000, 0008, 0010,**…**0038.

Особенной командой безусловной передачи управления без возврата оказывается PCH L: PC ←(H, L), в результате выполнения которой микропроцессор продолжает программу с адреса, загруженного в программный счетчик из регистров (H, L).

Внешний стек, служащий для временного хранения адресов возврата в командах вызовов, удобно использовать для запоминания содержимого внутренних регистров. Необходимость такого запоминания возникает в следующих ситуациях.

Ограниченного числа внутренних РОН при программировании часто "не хватает" для размещения промежуточных результатов, которые потребуются в программе несколько позже. Для освобожде-

ния внутренних регистров содержимое их загружается в стек, а по мере необходимости извлекается из стека.

В мультипрограммной среде и при обработке прерываний микропроцессор должен прерывать выполнение текущей программы и переключаться на выполнение другой программы (так называемое контекстное переключение). Разумеется, при таком переключении необходимо сохранять состояние программы, т.е. содержимое всех внутренних регистров, или, по крайней мере, тех из них, которые будут использоваться в новой программе, с последующим восстановлением состояния прерванной программы. Как минимум, при контекстном переключении следует временно запомнить слово состояния процессора PSW - содержимое аккумулятора и регистра флажков.

В системе команд микропроцессора имеется команда PUSH PSW загрузки в стек PSW, а также соответствующая команда POP PSW извлечения PSW из стека и загрузки его в регистр флажков и аккумулятор. Выполнение этих команд сопровождается автоматической модификацией SP.

В ряде случаев очень удобной оказывается специфическая 1-байтная команда XTH L: (L)↔([SP],(H)↔([SP+1])) обмена содержимого регистров (H, L) и двух верхних ячеек стека, т.е. последних загруженных в стек данных. Значение SP при выполнении команды XTH L не изменяется.

Фактическая область стека определяется программистом с помощью команды с непосредственными данными LXI SP: SP←<B3><B2>. Следует инициализировать указатель стека до использования в программе команд загрузки данных в стек и извлечения данных из стека и позаботиться о том, чтобы при выполнении программы область стека не использовалась для других целей.

Команды управления микропроцессором (специальные команды).

Однобайтные команды данной, обычно немногочисленной, группы применяются для задания режима работы микропроцессора.

Команда останова HLT вызывает прекращение выполнения программы и переводит микропроцессор в состояние останова (подробнее см. в следующем разделе).

Команда разрешения прерываний EI устанавливает внутренний триггер INTE разрешения прерываний в такое состояние (INTE = 1), когда микропроцессор реагирует на запросы прерываний, поступающие от периферийных устройств, инициирующих обмен данными.

Команда запрещения (маскирования) прерываний DI устанавливает вышеупомянутый триггер в такое состояние (INTE = 0), когда микропроцессор не воспринимает запросов прерываний от периферийных устройств.

Наконец, своеобразная холостая команда NOP не производит никаких действий, кроме инкремента программного счетчика для перехода к следующей команде. Команда NOP обычно используется в так называемых программных циклах задержки, в которых микропроцессор генерирует сигналы программируемой длительности.

1.5.Общие сведения о языке Ассемблер.

Главной задачей проектирования микропроцессорных систем становится разработка программных средств, т.е. прикладных программ.

Особое внимание уделяется при программировании МПС языку ассемблера. Главное его достоинство заключается в том, что он допускает представление всех элементов программы в символической (буквенно-цифровой) форме. В общем случае преобразование символических наименований языка ассемблера в понятные машине двоичные коды возлагается на специальную программу, называемую ассемблирующей программой или ассемблером. Таким образом, ассемблер - это программа, которая переводит программу, написанную на языке ассемблера на язык машинных кодов. Язык ассемблера ЭВМ того или иного типа нередко также называют ассемблер. Чтобы разделить эти два понятия, в случае, когда речь идет о программе будем писать ассемблер (строчные буквы), а вместо слов "язык ассемблера" - АССЕМБЛЕР (прописные буквы).

Начинать программирование на АССЕМБЛЕРе рекомендуется даже в том случае, если транслятор с этого языка отсутствует. Разумеется, в этом случае ассемблирование входной программы приходится заполнять вручную.

В общем случае в состав Ассемблера входят:

- система команд микропроцессора;

- директивы ассемблера, т.е. такие команды, которые не имеют аналогов среди команд микропроцессора. Эти команды не могут породить команд объективной программы после трансляции, поэтому их часто называют псевдокомандами. Директивы являются указаниями программе-ассемблеру о выполнении определенных действий в процессе ассемблирования. Директивы определяют:

- порядок ассемблирования;

- размещение в памяти информации;

- присваивают численные значения символическим наименованиям;

-резервируют память и выполняют другие функции.

Примеры директив: ORG, EQU, SET, END, IF, ENDIF, MACRO, ENDM, TITLE, DB, DW, DS.

Примечание. *При ассемблировании вручную директивы не используются*. В современном макроассемблере широко используются макрокоманды.

Группу команд можно определить, один раз, как большую команду - макрокоманду с уникальной мнемоникой, не входящей в систему команд МП. После определения макрокоманды в начале программы ее мнемонику можно использовать сколько угодно раз так, как будто она включена в систему команд МП. Таким образом, введение макрокоманд придает ассемблеру некоторые черты языков высокого уровня.

Примечание. *При ассемблировании вручную макрокоманды не используются*.

Ассемблерные программы записываются в виде последовательности команд, называемых также операторами или предложениями ассемблера. Каждый оператор записывается в одной строке, имеющей четыре поля:

- поле метки;

- поле мнемоники команды;

- поле операндов;

- комментарии.

Поля отделяются друг от друга, как правило, пробелами, а поле комментариев отделяется от поля операндов символом “**;**”. Для удобства чтения ассемблерных программ применяется формат с фиксированными полями, и комментарии даются на русском языке.

Примечание. *Поле операндов в случае двух операндов состоит из двух частей (операнд 1, операнд 2), разделенных запятой.*

При ассемблировании вручную удобно использовать формат бланка для записи ассемблерных программ, приведенных на рис.14.

Адрес ЯП	Содержимое ЯП	Метка	Мнемоника команды	Операнды	Комментарии
0800	DB	MAP:	IN	20	
01	20		/		
02	C6		ADI	01	
03	01		/		
04	D3		OUT	30	
05	30		/		
06	C3		JMP	MAP	
07	00		/		
08	08		/		

Рис.14. Бланк программирования для языка ассемблер.

Каждая строка стандартных ассемблерных программ соответствует одной машинной команде. Команды микропроцессора имеют длину 1-3 байта, что исключает однозначное соответствие между номерами строк и адресами ячеек памяти.

При ассемблировании вручную такое соответствие очень удобно, поэтому каждая строка бланка на рис.14 эквивалентна одной ячейке программной памяти. Адреса ячеек в виде 4-разрядных шестнадцатеричных чисел находятся в столбце "Адрес".

Столбцы метки, мнемоника операнда и комментарии представляют собой аналогичные поля строк стандартного языка ассемблера и заполняются соответствующим образом. Однако теперь нужно учитывать длину команды, и при использовании 2- и 3-байтных команд следующие одна или две строки бланка пропускаются. Обычно они отмечаются прочерками в столбце мнемоника операции.

После составления ассемблерной программы начинается собственно ассемблирование, т.е. заполнение шестнадцатиричными кодами столбца "Содержимое ЯП". Когда в строке имеется мнемоника команды, записываемое в этот столбец значение находится по таблице кодов операций микропроцессора. Если в поле операнда команды имеется числовой непосредственный операнд, в столбец записывается

его 16-ричный эквивалент. Для символьного непосредственного операнда столбец ЯП должен содержать его внутреннее машинное представление.

Символические наименования, представляющие собой адреса переходов в командах передачи управления и адреса рабочих ячеек, оформляются в виде регистровой таблицы, аналогичной таблице символических наименований ассемблера.

Метки при ассемблировании вручную заменяются адресами ячеек памяти, к которым они относятся.

Значением метки являются содержимое столбца "Адрес" отмеченной строки. Когда при просмотре программы в поле операнда встречается символический адрес, в регистровой таблице отыскивается его значение и записывается в столбце ЯП. Напомним, что в микропроцессоре I8080 адрес записывается "наоборот" - сначала младший байт, а затем старший.

После заполнения всех строк в столбце "Содержимое ЯП" будет находиться объектная программа.

2.Работа с учебной микро-ЭВМ ”УМК”

Для включения микро-ЭВМ необходимо:

1) установить кнопку " ~ " в отжатое состояние;
2) подключить микро-ЭВМ к сети переменного тока;
3) переключатель "РБ/ШГ" установить в состояние "РБ";
4) если требуется для выполнения задания макетный ТЭЗ, то следует поднять направляющие в вертикальное положение до фиксации и присоединить ТЭЗ к "УМК";
5) включить УМК, нажав кнопку " ~ ";
6) нажать управляющую кнопку "СБ" (сброс).

При этом в крайней левой позиции дисплея должен появиться знак " - ".

После указанных действий микро-ЭВМ готова к работе и находится в режиме ожидания директив оператора.

Задание I. Исследовать содержимое памяти. Нажать директивную клавишу П (чтение и модификация содержимого ячеек памяти). С помощью информационных клавиш (путем их последовательного нажатия) задать шестнадцатеричный адрес ячейки памяти (например, 0800), содержимое которой необходимо проверить. Каждая цифра адреса при этом записывается в младший разряд адресного дисплея и производится одновременный сдвиг всех знаков на один разряд влево. После ввода четвертой цифры следует нажать клавишу "ВП", инициирующую выполнение команды. После этого на дисплей данных выведется число, записанное по набранному адресу.

Нажмите на директивную клавишу " - ". Микро-ЭВМ увеличит на единицу адрес на адресном дисплее и выведет на дисплей данных

содержимое новой ячейки памяти. Последовательно нажимая на клавишу " - ", можно проверить содержимое последовательно расположенных ячеек памяти. Для выхода в режим "Монитора" следует нажать директивную клавишу "ВП" или "СБ". При этом в крайне левой позиции дисплея снова появится знак " — ".

Задание 2. Запись числа в ОЗУ. Нажать клавишу "П", набрать адрес ячейки ОЗУ (например, 0800 - начальный адрес области ОЗУ). Ввести команду, нажав клавишу "ВП". При этом на дисплей данных будет выведено содержимое выбранной ячейки памяти. С помощью информационных клавиш набирать на дисплее данных нужный новый байт данных (например FE). При этом обратить внимание на то, что всякая последующая вводимая цифра отображается в младшем разряде дисплея данных, а предыдущая смещается в старший разряд. Запись нового набранного значения в ячейку памяти осуществляется путем нажатия либо клавиши " - " либо "ВП". Следует иметь в виду, что при нажатии клавиши " - " "Монитор" перейдет к следующей ячейке памяти, а при нажатии "ВП" выполнение команды чтения и модификации ОЗУ завершится.

2.1. Загрузить в область ОЗУ, начиная с адреса 0810,последовательность чисел 00 до IA. Проверить результаты путем чтения ОЗУ.

2.2. Убедиться в невозможности записи данных в область ПЗУ.

Задание 3. Чтение и модификация программно-доступных регистров

Нажатием клавиши "РГ" и одного из идентификаторов пользователь получает доступ в один из программно-доступных регистров МП. Идентификаторами регистров служат информационные клавиши с 4/РН по F . Ответом на ввод идентификатора является индикация содержимого указанного регистра на дисплее данных в виде совокупности шестнадцатеричных цифр. Для изменения содержимого регистра набирается на дисплее данных с помощью информационных клавиш новый байт данных. После этого нажимается клавиша " - " и можно вводить идентификатор следующего регистра. При необходимости перехода к следующему регистру без изменения содержимого идентифицируемого следует нажать клавишу " - ", не набирая новых данных. Для завершения директивы нажимается клавиша "ВП".

3.1. Запишите последовательно во все 8-разрядные регистры МП, начиная с РОНа В, числа от 00 до 08, в регистр РС - число 0800, в регистр SP - OBAO.

3.2. Проверьте результат путем чтения программно доступных регистров.

Задание 4. Заполнение массива памяти константой.

4.1. Нажмите последовательно следующие клавиши: "ЗК" 0800 - 0900 - FF "ВП".

В ходе выполнения этой директивной команды массив памяти с адреса 0800 по 0900 включительно заполнится байтом данных FF. Нужно иметь в виду, что попытка заполнить информацией 54 последние ячейки ОЗУ с адресами OBCB - OBFF приводит к разрушению стека “Монитора”.

4.2. Проверить полученный результат.

Задание 5. Перемещение массива памяти в адресном пространстве.

5.1. Заполните массив ячеек ОЗУ с адресами 0800 no 0820 константой ЕЕ.

5.2. (См. Задание 4). Введите следующую директивную команду: “ПМ” 0800 - 0820 и 0900 "ВП".

5.3. Проверить полученный результат.

Задание 6. Определение контрольной суммы массива памяти.

6.1. Заполнить область ячеек ОЗУ с адресами с 0800 по 0809 последовательностью чисел 00 - 09.

6.2. Введите следующую директивную команду:
"КС" 0800 - 0809 “ВП”;

6.3. Проверьте полученный результат.

Задание 7. Передача управления программе пользователя

7.I.Загрузить в ОЗУ программу рис.15.

Адрес ЯП	Содержимое ЯП	Метка	Мнемоника команды	Операнды	Комментарии
0800	DB	MAP:	IN	20	
01	20		/		
02	C6		ADI	01	
03	01		/		
04	D3		OUT	30	
05	30		/		
06	C3		JMP	MAP	
07	00		/		
08	08		/		

Рис.15. Программа пользователя

7.2. Ввести следующую директивную команду:
"СТ" 0800 "ВП".

7.3. Проверить, что в результате выполнения программы пользователя индикация порта с номером 30 вывода отображает правильный результат.

7.4. Вызвать программу "Монитор", нажав клавишу "СБ".

7.5. Ввести следующую директивную команду:
"СТ" 0800 0804 "ВП".

Убедиться, что в результате выполнения этой директивы выполнение программы прерывается при достижении указанного адреса точки останова 0804. При этом содержимое регистров МП сохраняется, а управление передается "Монитору". За пользователем остается возможность выполнения любой директивы.

7.6. Проверьте содержимое аккумулятора и объясните полученный результат.

7.7. Продолжите выполнение программы с точки останова, введя следующую директивную команду:
"СТ" 0806 "ВП",

Убедитесь в том, что в результате выполнения этой директивы выполнение программы начинается с текущего адреса, находящегося в счетчике команд и прерывается при достижении адреса точки останова 0806.

Задание 8. Пошаговое выполнение программы.

В микро-ЭВМ имеются две разновидности пошагового выполнения программ:

1) покомандный режим;
2) поцикловый режим.

Напомним, что в поцикловом режиме "ЦК" команды программы выполняются по машинным циклам, в покомандном режиме "КМ" микро-ЭВМ переводится в состояние "Ожидание" после выполнения каждой команды.

1.I. Установите переключатель "РБ"/"ШГ" в состояние “ШГ” (при этом происходит подключение индикации на светодиодах).

8.2. Переключателем "КМ"/"ЦК" выберите один из режимов работы ("КМ").

8.3. Передайте управление программе пользователя (см. задание 7) и выполните программу последовательно по командам.

8.4. Нажмите кнопку: "СБ".

8.5. Переключателем "КМ"/"ЦК" выберите режим "ЦК".

8.6. Передайте управление программе пользователя и выполните программу последовательно по циклам. Дня двух первых команд программы укажите число и наименование машинных циклов.

Примечание. Для выхода из пошагового режима следует выполнить одно из следующих действий:

1) установить переключатель “РБ”/”ШГ” в положение "РБ" и нажать кнопку "СБ";

2) установить переключатель "РБ"/"ШГ' в состояние "РБ" и нажать кнопку "ШГ".

Задание 9. Прерывание выполнения программы пользователя.

9.1. Передайте управление программе пользователя в режиме "РБ".

9.2. Для прерывания программы пользователя нажмите управляющую кнопку "ПР". При этом формируется код команды RST7 (код FF); управление передается подпрограмме обработки прерывания. Подпрограмма сохраняет состояние всех регистров МП, передает управление "Монитору". На дисплее индуцируется содержимое счетчика команд, которое на единицу больше адреса последнего байта последней выполняемой команды. После этого пользователь может вызвать выполнение любой из существующих директив.

9.3. Продолжите выполнение прерванной программы с адреса останова.

Примечание. Команду RST7 (код FF) рекомендуется использовать для завершения выполнения программ пользователя в качестве последней команды программы.

Содержание отчета

Отчет должен содержать:
I) Структуру микро-ЭВМ "УМК".
2) Карту памяти микро-ЭВМ.
3) Описание клавиатуры "УМК" и директивных команд.
4) Блок-схему и текст исследуемой в лабораторной работе программы. В комментариях к каждой команде указать ее формат и используемый метод адресации.

3.Задачи для самостоятельного решения

Задачи, предложенные в данной главе, студент должен получить у преподавателя в качестве задач для домашней подготовки, отладить их с помощью симулятора I8080 (см. Приложение 2), а затем продемонстрировать их работоспособность преподавателю на реальной микропроцессорной системе УМК. Разработанные и отлаженные программы представить в отчете с использованием стандартного бланка программирования (см. рис.14).

Задача 1.

Загрузить в аккумулятор число FF и выполнить логические операции DAA, CMP A, ORA A, ORI FF, SUI 01, CMA, сохраняя результат в регистрах B, C, Д, E, H, L. Сохранить содержимое указанных регистров также в стеке, верхушка стека 0A00.

Задача 2.

Загрузить в аккумулятор число 00 и выполнить логические операции DAA, CMP A, ORA A, ORI 00, SUI FF, CMA, сохраняя результат в регистрах B, C, Д, E, H, L. Сохранить содержимое указанных регистров также в стеке, верхушка стека 0A00.

Задача 3.

Заполнить массив памяти 0B01 - 0B3F содержимым регистров B и C, причем байт, содержащийся в регистре B отсылать в четные ячейки памяти. Выйти на подпрограмму "задержки" на 1 секунду и осуществить зажигание-гашение светодиодов 30 порта, считывая информацию последовательно из каждой ячейки заполненного массива ОЗУ.

Задача 4.

Заполнить массив памяти 0A01 - 0A4F содержимым регистров B и C, причем байт, содержащийся в регистре B отсылать в четные ячейки памяти. Выйти на подпрограмму "задержки" на 1 секунду и осуществить зажигание(1 сек)-гашение(2 сек) светодиодов 30 порта, считывая информацию последовательно из каждой ячейки заполненного массива ОЗУ.

Задача 5.

Принять из 20 порта код 01. При получении других значений подавать звуковой сигнал в 80 порту любой тональности. Постоянно находится в цикле опроса 20 порта, до тех пор, пока не будет выставлен контрольный код 01.

Задача 6.

Принять из 20 порта код FE. При получении других значений подавать звуковой сигнал в 80 порту любой тональности. Постоянно находится в цикле опроса 20 порта, до тех пор, пока не будет выставлен контрольный код FE.

Задача 7.

Загрузить 00 в A, установить признак переноса C="1", осуществить сдвиг вправо и выполнить логические и арифметические операции над содержимым аккумулятора.

```
ADI FF
SUI 01
CPI 01
ORI FF
DCR A
INR A
```

Результаты поместить в РОНах B, C, D, E, H, L. Заполнить и объяснить таблицу результатов. Сохранить содержимое всех регистров в стеке. Верхушка стека должна быть установлена 0AFF.

Задача 8.

Загрузить 00 в A, установить признак переноса C="1", осуществить сдвиг вправо и выполнить логические и арифметические операции над содержимым аккумулятора.

```
ADI FE
SUI 01
CPI 01
ORI FF
DCR A
INR A
```

Результаты поместить в РОНах B, C, D, E, H, L.

Заполнить и объяснить таблицу результатов. Сохранить содержимое всех регистров в стеке. Верхушка стека должна быть установлена 0B80.

Задача 9.

Выйти на подпрограмму задержки по условию равенства нулю числа вводимого из порта 20 и осуществить зажигание светодиодов 30 порта с интервалом 4 секунды, задействуя младшую и старшую тетрады попарно Д0 Д1 Д2 Д3 Д4 Д5 Д6 Д7 . Повторить эту процедуру 5 раз, после чего подать звуковой сигнал (80 порт) любой тональности.

Задача 10.

Выйти на подпрограмму задержки по условию неравенства нулю числа вводимого из порта 20 и осуществить зажигание светодиодов 30 порта с интервалом 4 секунды, задействуя младшую и старшую тетрады попарно Д0 Д1 Д2 Д3 Д4 Д5 Д6 Д7. Повторить эту процедуру 5 раз, после чего подать звуковой сигнал (80 порт) любой тональности.

Задача 11.

Принять из 20 порта число BB, при получении других значений находится в цикле опроса 20 порта. Полученное число BB записать в массив памяти: начальный адрес 0A00 конечный адрес 0AFF. При этом использовать косвенную регистровую адресацию.

Задача 12.

Загрузить аккумулятор числом B1, и выполнить логические и арифметические операции над аккумулятором: CMA, DAA, ADD A, SUB A, ANA A, ORA A сохраняя результат в одном из регистров общего назначения (A, B, C, D, E, H, L соответственно). Сохранить содержимое регистров в стеке: начальный адрес стека (верхушка) должен быть установлен в 0BB0. Результат выполнения команд программы занести в таблицу

Исходное число находящееся в аккумуляторе	Команда	Результат выполнения команды

Задача 13.

Загрузить аккумулятор числом B1, и выполнить логические и арифметические операции над аккумулятором: CMA, DAA, ADD A,

SUB A, ANA A, ORA A сохраняя результат в одном из регистров общего назначения B, C, Д, E, H, L. Сохранить содержимое регистров в стеке: начальный адрес стека (верхушка) должен быть установлен в 0BB0.

Задача 14.

Выйти на подпрограмму "задержки" и осуществить зажигание-гашение светодиодов 30 порта с интервалом 3 секунды и 9 секунд, задействуя попарно разряды Д0 , Д2 ; Д1 , Д3 ; ..... Д5 , Д7 ; Д6 , Д0 . Повторить эту процедуру 3 раза, после чего подать звуковой сигнал (80 порт) любой тональности.

Задача 15.

Составить программы для вычисления величин:

а) z = (4x - 8y):2

б) z = 6x - 5

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 16.

Принять из 20 порта число AA. При получении других значений находится в цикле опроса 20 порта. Полученное число AA записать в массив памяти: начальный адрес 0A00 конечный адрес 0AFF

Задача 17.

Принять из 20 порта число 94. При получении других значений находится в цикле опроса 20 порта. Полученное число записать в массив памяти: начальный адрес 0B00 конечный адрес 0BFF

Задача 18.

Дан массив памяти объемом 20 ячеек с адреса 0B00. Определить количество чисел в массиве, больших 0A. Результат поместить в выходной порт с адресом 30.

Задача 19.

Найти минимальное число в массиве из 9 целых чисел. Массив расположен в памяти, начиная с адреса 0A00. Результат поместить в выходной порт с адресом 30.

Задача 20.

Найти максимальное число в массиве чисел, состоящем из 10 ячеек, начиная с адреса 0900 и выдать его на вход 30.

Задача 21.

МП контролирует во внешних реле, выводы которых подключены к разрядам D5 - D0 порта с адресом 20. Нуль соответствует разомкнутому реле, а единица - замкнутому. В зависимости от состояния реле МП выполняет одну из трех подпрограмм, выбор которых производится следующим образом.

- подпрограмма 1 зажигает в вых. порту 30 один светодиод (Д0) в том случае, если замкнуты 0, 1.
- подпрограмма 2 зажигает в вых. порту 30 два светодиода (Д1 ,Д0) в том случае, если 1 не выполняется и если замкнуты реле 2,3,4.
- подпрограмма 3 зажигает в вых. порту 30 три светодиода (Д2 ,Д1 ,Д0) в том случае, если 1,2 не выполняются и если замкнуты реле 4,5.

Задача 22.

Массив чисел из 20(10) записан с адреса 0B00 и дальше. Найти в массиве число 0B, адрес числа занести в регистровую пару BC.

Задача 23.

Получить из порта 20 число и подсчитать количество разрядов установленных в 0. Результат поместить в порт 30.

Задача 24.

Дан массив из 5-ти чисел, расположенных в ячейках 0A00 - 0A04. Расположить числа по возрастанию в ячейках 0B00-0B04.

Задача 25.

Получить из порта 20 число и подсчитать количество разрядов установленных в 1. Результат поместить в порт 30.

Задача 26.

Ввести с 20 порта число, проинвертировать и подсчитать число разрядов установленных в 1. Результат занести в ячейку 0B00.

Задача 27.

В ячейке ОЗУ 0B00 хранится число X1 , в ячейке 0B01 X2 . Необходимо вывести через порт 30 большее из чисел. Составить программу.

Задача 28.

В памяти МПС записана подпрограмма перемножения двух чисел. Пользуясь этой подпрограммой, составить программу для вычисления выражения Z = X + X^2 + X^3, где X - число, вводимое в МПС через порт 20. Результат следует хранить в ячейках памяти начиная с адреса 0B00.

Задача 29.

Загрузить аккумулятор числом AA, и выполнить логические операции: CMA, DAA, CMP A, ORA A, ORI 00 , SUI 01, сохраняя результат в одном из регистров общего назначения (B, C, Д, E, H, L соответственно). Сохранить содержимое регистров в стеке, верхушка стека 0BB0. Результат выполнения команд программы занести в таблицу

Исходное число находящееся в Аккумуляторе	Команда	Результат выполнения команды

Задача 30.

Загрузить аккумулятор числом AA и выполнить логические и арифметические операции над аккумулятором: CMA, DAA, ADD A, SUB A, ANA A, ORA A сохраняя результат в одном из регистров общего назначения (B, C, Д, E, H, L соответственно). Сохранить содержимое регистров в стеке: начальный адрес стека (верхушка) должен быть установлен в 0BB0.

Результат выполнения команд программы занести в таблицу

Исходное число находящееся в Аккумуляторе	Команда	Результат выполнения команды

Задача 31.

Исходное расположение чисел задано в таблице

Число	Ячейка памяти
X1	0901
X2	0902
Y1	0A00
Y2	A010

Реализовать на языке ассемблер следующие алгоритмы обработки указанных чисел

$$\frac{X1+X2}{2} \; ; \; \frac{Y1+Y2}{2}$$

Задача 32.

Найти максимальное число в массиве 8-битных целых чисел и поместить в регистр D. Длина массива хранится в ячейке с адресом 0B00 и равна 8. Обрабатываемый массив хранится в смежных ячейках памяти с начальным адресом (базовым) 0A00. В качестве счетчика элементов массива использовать регистр B.

Задача 33.

Сложить два трехбайтовых числа A и B. Число A находится в ячейка памяти 0B00-0B02, число B - 0A00-0A02. Результат поместить в ячейки памяти 0900-0902.

Задача 34.

Составить программу, определяющую наличие "1" в любом из разрядов входного порта (20) и определяющую номер первого разряда слева, в котором записана "1".

Задача 35.

Составить программу, выполняющую обмен содержимым между парами регистров (B,C) и (Д,E) микропроцессора КР 580 ИК80, пользуясь командами PUSH и POP. Программу разместить в памяти микро-ЭВМ начиная с адреса 0800. В начале программы регистровую пару D обнулить, а в регистровую пару B поместить число 0F1C.

Задача 36.

Просуммировать 8-битные элементы массива, общее число которых (20) 16. Начальный адрес массива 0B00. Программу разместить в памяти начиная с адреса 0800. Сумму разместить в регистрах (Д,Е).

Задача 37.

Составить программу вычитания двух 10-ти байтных чисел A и B.
A ---> 0900 - 0909B ---> 0A00 - 0A09. Результат побайтно инвертировать и поместить в память начиная с ячейки памяти с адресом 0B00.

Задача 38.

Найти максимальное число в массиве из 0B положительных чисел. Начальный адрес массива 0B00. Программу разместить в памяти, начиная с адреса 0800. Найденный максимальный элемент поместить в регистр Д.

Задача 39.

Переслать содержимое области памяти объемом 64 байт с начальным адресом 0B00 в другую область памяти с начальным адресом 0900 . При обнаружении кода 4B передача должна завершатся и программа переходит к другим действиям (зажигает светодиоды порта 30). Программа должна начинаться с адреса 0800.

Задача 40.

Загрузить в разряд С регистра признаков (флажок переноса) значение разряда D6 содержимого регистра.

D7	D6	D5	D4	D3	D2	D1	D0

Задача 41.

Используя подпрограмму "задержки", осуществить зажигание (1сек) - гашение (1 сек) светодиодов в следующем порядке:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	*	*	0	0	0
0	0	*	*	*	*	0	0
0	*	*	*	*	*	*	0
*	*	*	*	*	*	*	*
0	*	*	*	*	*	*	0

0	0	*	*	*	*	0	0
0	0	0	*	*	0	0	0
0	0	0	0	0	0	0	0
0	0	0	*	*	0	0	0

и т.д.

Задача 42.

Загрузить 00 в A, установить признак переноса C="1", осуществить сдвиг вправо и выполнить логические и арифметические операции над содержимым аккумулятора.

```
ADI FE
SUI 03
CPI 01
ORI FF
DCR A
INR A
```

Результаты поместить в РОНах B, C, D, E, H, L. Заполнить и объяснить таблицу результатов. Сохранить содержимое всех регистров в стеке. Верхушка стека должна быть установлена 0BB0.

Задача 43.

Заполнить массив памяти 0A00 - 0A4F содержимым регистров B и C, причем байт, содержащийся в регистре B отсылать в четные ячейки памяти. Выйти на подпрограмму "задержки" на 1 секунду и осуществить зажигание (1 сек) - гашение (1 сек) светодиодов 30 порта, считывая информацию последовательно из каждой ячейки заполненного массива ОЗУ.

Задача 44.

Принять из 20 порта код 01. При получении других значений подавать звуковой сигнал в 80 порту любой тональности. Постоянно находится в цикле опроса 20 порта, до тех пор, пока не будет выставлен контрольный код 01. При наличии контрольного кода осуществить зажигание светодиодов D0, D1 порта 30.

Задача 45.

Принять из 20 порта число 66, при получении других значений находится в цикле опроса 20 порта. Полученное число 66 записать в массив памяти: начальный адрес 0900, конечный адрес 0979. При этом использовать косвенную регистровую адресацию.

Задача 46.

Принять из 20 порта число BB, при получении других значений находится в цикле опроса 20 порта. Полученное число BB записать в массив памяти: начальный адрес 0B00, конечный адрес 0BBB.Выйти на подпрограмму "задержки" и осуществить зажигание (2 сек) - гашение (2 сек) светодиодов 30 порта (в цикле).

Задача 47.

Выйти на подпрограмму задержки при наличии числа A1 в порту 20 и осуществить зажигание (1сек) - гашение (1сек) светодиодов 30 порта, задействуя поочередно разряды Д0,Д1,Д2,Д3,Д4,Д5,Д6,Д7(режим бегущих огней). Повторить эту процедуру 3 раза, после чего подать звуковой сигнал (80 порт) любой тональности.

Задача 48.

Принять из 20 порта число 81, при получении других значений находится в цикле опроса 20 порта. Полученное число 81 записать в ячейку памяти 0800. Заполнить массив памяти 0A00-0AFB байтами в следующем порядке.

	D7	D6	D5	D4	D3	D2	D1	D0
0A00	1	0	0	0	0	0	0	0
0A01	0	1	0	0	0	0	0	0
0A02	0	0	1	0	0	0	0	0
0A03	0	0	0	1	0	0	0	0
0A04	0	0	0	0	1	0	0	0
0A05	0	0	0	0	0	1	0	0
0A06	0	0	0	0	0	0	1	0
0A07	0	0	0	0	0	0	0	1
0A08	1	0	0	0	0	0	0	0

и т.д.

Задача 49.

Используя подпрограмму "задержки", осуществить зажигание (1сек) - гашение (1 сек) светодиодов в порядке инверсном таблице:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	*	*	0	0	0
0	0	*	*	*	*	0	0
0	*	*	*	*	*	*	0
0	*	*	*	*	*	*	0
0	0	*	*	*	*	0	0
0	0	0	*	*	0	0	0
0	0	0	0	0	0	0	0
0	0	0	*	*	0	0	0

и т.д.

Задача 48.

Используя подпрограмму "задержки", осуществить зажигание (1сек) - гашение (1 сек) светодиодов в следующем порядке:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	*	*	0	0	0
0	0	*	*	*	*	0	0
0	*	*	*	*	*	*	0
0	*	*	*	*	*	*	0
0	0	*	*	*	*	0	0
0	0	0	*	*	0	0	0
0	0	0	0	0	0	0	0
0	0	0	*	*	0	0	0

и т.д.

Задача 49.

Выйти на подпрограмму "задержки" по условию равенства нулю числа вводимого из порта 20 и осуществить зажигание светодиодов 30 порта с интервалом 4 секунды, задействуя младшую и старшую тетрады поочередно (Д0 Д1; Д2 Д3 ;и Д4 Д5 ; Д6 Д7). Повторить эту процедуру 5 раз, после чего подать звуковой сигнал (80 порт) любой тональности.

Задача 50.

Дан массив 20 чисел. Подсчитать число чисел, делящихся без остатка на B. Массив расположен в ОЗУ начиная с адреса 0B00. Число B вводится с порта с адресом 20. Результат выдается в порт с адресом 30.

Задача 51.

Написать программу, выполняющую вычитание из числа 7F , хранящегося в регистре B, числа вводимого из порта с адресом 20. Результат вычитания выдать в порт с адресом 30. Примечание: организовать выполнение программы в цикле ("вечный цикл").

Задача 52.

Написать программу покомпонентного логического умножения двух векторов на языке ассемблера. Массив байтов, представляющий собой первый вектор, имеет началом ячейку с адресом 0900. Другой вектор представляет собой массив байтов, начинающийся с ячейки с адресом 0A00. Результат умножения должен быть представлен массивом с начальным адресом 0B00. Емкость каждого из массивов представляется числом, хранящимся в регистре B.

Задача 53.

Составить программы для вычисления величин:

а) z=2x+1

б) z=3x+2y

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 54.

Составить программы для вычисления величин:

а) z=(4x-8y):2

б) z=6x-5

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 55.

Составить программы для вычисления величин:

$$z=\begin{cases} x-y \text{ при } x>=8 \\ x+y \text{ при } x<8 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 56.

Составить программы для вычисления величин:

$$z=\begin{cases} 3x+y \text{ при } x>=10 \\ 5x-y \text{ при } x<10 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 57.

Составить программы для вычисления величин:

$$z=\begin{cases} x+2y+1 \text{ при } x>5 \\ 2x-3y+3 \text{ при } x<=5 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 58.

Составить программы для вычисления величин:

$$z=\begin{cases} x+1 \text{ при } x<=15 \\ x+y \text{ при } 15<x<30 \\ 0 \text{ при } x>=30 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 59.

Составить программу возведения в куб и квадрат чисел от 1 до 4, поступающих с порта 20. Результат выполнения помещать в РОНы B и C.

Задача 60.

Составить программу для вычисления z:

$$z = \begin{cases} 2x + 1 \text{ при } y=0 \\ 3x + 2y \text{ при } y>0 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 61.

Составить программу для вычисления z:

$$z = \begin{cases} x - y \text{ при } x>=8 \\ x + y \text{ при } x<8 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 62.

Составить программу для вычисления z:

$$z = \begin{cases} 3x + y \text{ при } x>=10 \\ 5x - y \text{ при } x<10 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 63.

Составить программу для вычисления z:

$$z = \begin{cases} x + 2y + 1 \text{ при } x>5 \\ 2x - 3y + 3 \text{ при } x<=5 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

Задача 64.

Составить программу для вычисления z:

$$z = \begin{cases} x + 1 \text{ при } x <= 15 \\ x + y \text{ при } 15 < x < 30 \\ 0 \text{ при } x >= 30 \end{cases}$$

Число x поступает через ячейку ОЗУ 0B00, y - через ячейку ОЗУ 0B01, число z должно хранится в ячейке ОЗУ 0B02.

4.Контрольные вопросы по темам

Тема 1. Работа с микро-ЭВМ УМК

1. Опишите основные элементы структуры микро-ЭВМ УМК.
2. Для какой цели используется работа микропроцессора в режиме ожидания
3. Опишите структуру памяти микро-ЭВМ УМК.
4. Как записать число в программно доступные регистры МП и память в учебной микро-ЭВМ УМК?
5. Что происходит в микро-ЭВМ УМК при попытке записать данные в ПЗУ?
6. Какие изменения происходят в МП в процессе выполнения сброса (подаче сигнала RESET)?
7. Сколько транзисторов содержится в кристалле МП?
8. Каково адресное пространство МП I8080, КР1810ВМ86?
9. Какие линии МП являются двунаправленными?
10. Какой адрес появляется на шине адреса после адреса FFFF?
11. Перечислите группу сигналов МП, связанных с прерыванием.
12. Какой командой можно сбросить сигнал INTE?
13. Какой командой можно установить сигнал INTE?
14. Фронтом какого сигнала определяется начало такта МП?
15. Каким сигналом определяется начало МЦ?
16. Фронтом какого сигнала определяется большинство действии в системе?
17. В каком такте МЦ МП выдает на шину данных байт состояния?
18. Какая информация содержится в байте состояния МП?
19. Укажите парные сигналы МП.
20. Какие сигналы МП имеют активный нулевой уровень?
21. Чему равно напряжение питания МП?
22. Сколько РОН имеет МП?
23. Расскажите о программной модели МП.
24. Дать определения понятий "микропроцессор" (МП), "микропроцессорная система" (МПС), "микропроцессорный комплект" (МПК), "микро-ЭВМ".

Тема 2. Запись и выполнение простых программ

1. Что такое машинный такт?

2. Опишите назначение PC, SP.
3. Что такое машинный цикл?
4. Что такое цикл команды?
5. Какой цикл является первым при выполнении любой команды?
6. Для каких целей используются в МПС разряды слова состояния?
7. Для каких целей используются в МПС буферы магистрали МД, МА?
8. Какие типы команд Вы знаете?
9. Назовите методы адресации.
10. Опишите функциональное назначение внешних выводов МП.
11. Назовите состав МПК К580.
12. Назовите назначение элементов структуры МП.
13. Каким сигналом определяется начало всякого МЦ?
14. Что понимается под символом М в мнемоническом обозначении команд?
15. Привести примеры команд с непосредственной адресацией.
16. В чем заключается регистровая косвенная адресация данных в МП КР580. Привести примеры команд.
17. Какие виды адресации данных используются в следующих командах МП I8080
а) MOV D, A; б) ADI CC; в) SUB M;
г) MOV C, M; д) ANA H; е) JMP 00BC;
ж) CALL FF00; з) MVI A, FF; и) RET;
к) OUT 00; л)CMP M?
18.Опишите алгоритм работы команд: CMP R; CPI D8.
19.Опишите алгоритм работы команд условного прохода.

Тема 3. Ввод-вывод, маскирование данных, организация условных переходов.

1. Нарисуйте схему фиксации слова состояния процессора с помощью схемы МБР.
2. Нарисуйте схему формирования МУ на базе комбинации каналов слова состояния и DBIN, WR.
3. При каких условиях устанавливаются флажки регистра признаков?
4. Перечислите виды логических операции, выполняемых МП.
5. Какие адреса имеют при прямой адресации порты МПС, ячейки памяти?

6. Дать определение понятия "шина". Какие шины имеются в микропроцессорах?
7. Дать определение понятия "порт".
8. Дать определение понятия "интерфейс".
9. Расшифровать коды состояния регистра признаков МП I8080: а) 00000010; 6) 10010111; в) 100000111.
10. Назвать и охарактеризовать три способа обмена данными между МПС и внешними устройствами (ВУ).
11.С помощью каких команд микро-ЭВМ может осуществлять ввод-вывод информации?
12. За сколько машинных тактов осуществляется ввод-вывод данных по командам IN<A1>, OUT<A1>?
13. Укажите достоинства и недостатки различных методов адресации к внешним устройствам.
14. При выполнении каких команд, приведенных в программе 8.7, задействуются разряды регистра признаков МП БИС?
15. По каким условиям записывается 1 в каждый из разрядов регистра признаков?

Тема 4.Подпрограмма и стек.

1. Максимальный объем памяти микро-ЭВМ на базе МП I8080?
2. Что такое подпрограмма?
3. Что называется вызовом подпрограммы?
4. Какие команды используются для вызова подпрограммы и возврата из неё?
5. Как работает команда CALL <A2><A1 >?
6. Как работает команда RET?
7. В какой последовательности сохраняется и извлекается содержимое регистровых пар в стеке?
8. С помощью какой команды первоначально назначается (создается) область стеке?
9. Какую регистровую пару называют словом состояния процессора и обозначают PSW?
10.Определение стека.
11.Перечислите команды условного вызова подпрограммы.
12. Перечислите команды условного возврата из подпрограммы.
13.Укажите количество машинных тактов выполнения команды CALL<A2><A1 >.

14.В какой последовательности записывается и считывается из стека содержимое аккумулятора и регистра признаков МП БИС при выполнении команды PUSH PSW и POP PSW?
15.Сравните процесс выполнения микро-ЭВМ команд CALL и RST.
16.Какое максимальное время задержки может обеспечивать подпрограмма 8.10, если длительность машинного такта T=1мкс?

Тема 5.Выполнение арифметических операций.

1. Какие арифметические операции может выполнять МП I8080?
2. На чем основаны алгоритмы умножения и деления, пригодные для реализации в МПС на базе МП I8080?
3. Сформулируйте правила выполнения процессором команды DAA.
4. Укажите возможные способы представления чисел в МП I8080.
5. Предстаете числа от 0 до 20 в BCD коде (Binary Coded Decimal).
6. После каких команд можно осуществлять десятичную коррекцию числа аккумулятора (выполнять команду DAA)?
7. Можно ли непосредственно исследовать перемножение двух чисел по подпрограмме 8.18?
8. В чем преимущество вычисления функции, заданных в виде таблиц?
9. Как представляются отрицательные числа в двоичном коде?
10.Найдите десятичные эквиваленты следующих двоичных чисел:
а) 10000001 (дополнительный код)
б) 00000001 (обратный код)
в) 10001000 (прямой двоичный код)
г) 10001001 (двоично-десятичный код)
д) 11111111 (обычный двоичный код без знака)
е) 11111101 (дополнительный код).
11.Назовите составные части языка Ассемблер.
12.Что определяют директивы Ассемблера и используются ли они при ассемблировании "вручную"?
13.Дайте определение макрокоманды.
14.Расскажите о структуре полей оператора Ассемблера.
15.Расскажите о трансляции меток.
16.Дайте определение исходного модуля, объектного и исполняемого модуля.
17.В чем сущность индексной адресации и как её реализовать программно в МП 18080?

Тема 6. Подключение дисплея и клавиатуры к микроЭВМ .

1. Какие коды необходимо записать по адресам 0900-0905 для вывода на дисплей чисел 1,2,3,4,5.
2. Как следует изменить программу 8.22 для изменения направления сканирования индикаторов дисплея
3. Какая из клавиш определится нажатой при запуске на выполнение программы 8.24 , если мы будем держать нажатыми одновременно клавиши а) 3 и 5 б) 2 и 8
4. Поясните реализацию режима статической индикациина схеме рис 8.15
5. Поясните реализацию режима динамической индикации на схеме рис.8.15
6. Поясните режим сканирования клавиатуры (см. рис. 8.16) на основании анализа работы программ 8.23 и 8.24

Литература

1. Мирский Г.Я. Микропроцессоры в измерительных приборах.-М.:Радио и связь, 1984.
2. Каган Б.М., Сташин В.В. Основы проектирования микропроцессорных устройств автоматики.-М.:Энергоатомиздат, 1987.
3. Григорьев В.Л. Программное обеспечение микропроцессорных систем.-М.:Энергоатомиздат, 1983.
4. Балашов Е.П., Пузанков Д.В. Микропроцессоры и микропроцессорные сиетемы.-М.:Радио и связь, 1981.
5. Микропроцессоры. Кн.1, 3 /Под ред. Л.П.Преснухина. Мн.: Высшая школа, 1986. . :;:
6. Микро-ЭВМ "УМК". Эксплуатационная документация. Альбом , I. -
;
7. Микро-ЭВМ./Под ред. А.Дирксена.-М.:Энергоиздат, 1982. -
8. Гуртовцев А.Л., Гудыменко Программы для микропроцессоров. Справочное пособие.-Мн.:Вышэйшая школа, 1989.
9. Майоров В.Г., Гаврилов А.И. Практический курс программирования микропроцессорных систем. -М.:Машиностроение, 1989.

Приложение 1.
Система команд микропроцессора КР580ВМ80

1.ПЕРЕСЫЛКИ

Кол-во тактов	Флаги ZS PC	Команда	Код								Комментарий
13	---	LDA адрес	3A								Загрузить “A” из ячейки памяти
13	---	STA адрес	32								Послать “A” в ячейку памяти
7	---	LDAX B	0A								Загрузить “A” из адреса(BC)
7	---	LDAX D	1A								Загрузить “A” из адреса(DE)
7	---	STAX B	02								Послать “A” по адресу (BC)
7	---	STAX D	12								Послать “A” по адресу (DE)
10	---	IN Порт	DB								Загрузить "A" из порта
10	---	OUT Порт	D3								Послать "A" в порт
			A	B	C	D	E	H	L	M	Загрузить “A” из регистра
5(7)	---	MOV A, рег.	7F	78	79	7A	7B	7C	7D	7E	
5(7)	---	MOV B, рег.	47	40	41	42	43	44	45	46	“B”
5(7)	---	MOV C, рег.	4F	48	49	4A	4B	4C	4D	4E	“C”
5(7)	---	MOV D,рег.	57	50	51	52	53	54	55	56	“D”
5(7)	---	MOV E, рег.	5F	58	59	5A	5B	5C	5D	5E	“E”
5(7)	---	MOV H, рег.	67	60	61	62	63	64	65	66	“H”
5(7)	---	MOV L, рег.	6F	68	69	6A	6B	6C	6D	6E	“L”
5(7)	---	MOV M, рег.	77	70	71	72	73	74	75	--	“M”
7(10)	---	MVI рег., число	3E	06	0E	16	1E	26	2E	36	Загрузить число в регистр

ПРИМЕЧАНИЕ. КОМАНДЫ, ОПЕРИРУЮЩИЕ С ПАМЯТЬЮ, ИМЕЮТ БОЛЬШЕЮ ДЛИТЕЛЬНОСТЬ.

Кол-во тактов	Флаги ZSPC	Команда	Код	Комментарий
10	----	LXI B, адр.	01	Загрузить “BC” адресом
10	----	LXI D, адр.	11	“DE”
10	----	LXI H, адр.	21	“HL”
10	----	LXI SP, адр.	31	“SP”
11	----	PUSH B	C5	Запомнить “BC” в стеке
11	----	PUSH D	D5	“DE”
11	----	PUSH H	E5	“HL”
11	----	PUSH PSW	F5	“PSW”
10	----	POP B	C1	Загрузить “BC” из стека
10	----	POP D	D1	“DE”
10	----	POP H	E1	“HL”
10	----	POP PSW	F1	“PSW”
16	----	LHLD адрес	2A	Загрузить “HL” из адреса
16	----	SHLD адрес	22	Послать “HL” по адресу
4	----	XCHG	EB	Обмен между “HL” и “DE”
18	----	XTHL	E3	Обмен между “HL” и верх. стека
5	----	PCHL	E9	Загрузить “PC” из “HL”
5	----	SPHL	F9	Загрузить “SP” из “HL”

2,3 АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Кол-во тактов	Флаги ZSPC	Команда	Код								Комментарий
			A	B	C	D	E	H	L	M	
4(7)	~~xxxx~~	ADD рег.	87	80	81	82	83	84	85	86	“A“+регистр
4(7)	~~xxxx~~	ADC рег.	8F	88	89	8A	8B	8C	8D	8E	“A“+рег.+перенос
4(7)	~~xxxx~~	SUB рег.	97	90	91	92	93	94	95	96	“A“-регистр
4(7)	~~xxxx~~	SBB рег.	9F	98	99	9A	9B	9C	9D	9E	“A“-рег.-перенос
4(7)	~~xxx~~0	ORA рег.	B7	B0	B1	B2	B3	B4	B5	B6	“A“ ∪ регистр
4(7)	~~xxx~~0	ANA рег.	A7	A0	A1	A2	A3	A4	A5	A6	“A“ ^ регистр

4(7)	~~xxx~~0	XRA рег.	AF	A8	A9	AA	A B	A C	AD	AE	“A“+рег. по мод.2
4(7)	~~xxxx~~	CMP рег.	BF	B8	B9	BA	BB	BC	BD	BE	сравнить “A“ и рег.
5(10)	~~xxx~~	INR рег.	3C	04	0C	14	1C	24	2C	34	регистр + 1
5(10)	~~xxx~~	DCR рег.	3D	05	0D	15	1D	25	2D	35	регистр - 1
7	~~xxxx~~	ADI данные	C6	“A”+ данные							
7	~~xxxx~~	ACI данные	CE	“A”+ данные + перенос							
7	~~xxxx~~	SUI данные	D6	“A”- данные							
7	~~xxxx~~	SBI данные	DE	“A”- данные – перенос							
7	~~xxx~~0	ORI данные	F6	“A” ∪ данные							
7	~~xxx~~0	ANI данные	E6	“A” ^ данные							
7	~~xxx~~0	XRI данные	EE	“A” + данные по мод.2							
7	~~xxxx~~	CPI данные	FE	сравнить “A” с данными							
4	---	CMA	2F	инвертировать “A”							
4	~~xxxx~~	DAA	27	десятичная коррекция “A”							
4	---~~x~~	RAL	17	сдвиг “A” влево; C─> A0, A7─>C							
4	---~~x~~	RLC	07	сдвиг “A” влево; A7─>A0, A7─>C							
4	---~~x~~	RAR	1F	сдвиг “A” вправо; C─> A7, A0─>C							
4	---~~x~~	RRC	0F	сдвиг “A” вправо; C─> A7, A0─>C							
5	---	INX B	03	“BC” +1							
5	---	INX D	13	“DE” +1							
5	---	INX H	23	“HL” +1							
5	---	INX SP	33	“SP” +1							
5	---	DCX B	0B	“BC” ─1							
5	---	DCX D	1B	“DE” ─1							
5	---	DCX H	2B	“HL” ─1							
5	---	DCX SP	3B	“ SP” ─1							
10	---~~x~~	DAD B	09	“BC” + “HL”, результат в “HL”							
10	---~~x~~	DAD D	19	“DE” + “HL”							
10	---~~x~~	DAD H	29	“HL” + “HL”							
10	---~~x~~	DAD SP	39	“SP” + “HL”							

4 КОМАНДЫ ВЕТВЛЕНИЯ

Кол-во тактов	Флаги ZSPC	Команда	Код	Комментарий
10	---	JMP адрес	C3	Безусловный переход на адрес
10	---	JZ-JNZ адрес	CA-C2	Переход по флангу “Z”
10	---	JM-JP адрес	FA-F2	“S”(минус-плюс)
10	---	JPE-JPO адрес	EA-E2	“P”(чет-нечет)
10	---	JC-JNC адрес	DA-D2	“C”
17	---	CALL адрес	CD	Вызов подпрограммы из адреса
11(17)	---	CZ-CNZ адрес	CC-C4	Вызов п/п по флангу “Z”
11(17)	---	CM-CP адрес	FC-F4	“S”
11(17)	---	CPE-CPO адрес	EC-E4	“P”
11(17)	---	CC-CNC адрес	DC-D4	“C”
5	---	RET	C9	Возврат из п/п по адресу, взятому из стека
5(11)	---	RZ-RNZ	C8-C0	Возврат из п/п по флангу “Z”
5(11)	---	RM-RP	F8-F0	“S”
5(11)	---	RPE-RPO	E8-E0	“P”
5(11)	---	RC-RNC	D8-D0	“C”

ПРИМЕЧАНИЕ. ПРИ ВЫПОЛНЕНИИ УСЛОВИЯ ПЕРЕХОДА, ВЫЗОВА ИЛИ ВОЗВРАТА
КОМАНДА ИМЕЕТ БОЛЬШУЮ ДЛИТЕЛЬНОСТЬ.

5. СПЕЦИАЛЬНЫЕ КОМАНДЫ

Кол-во тактов	Флаги ZSPC	Команда	Код	Комментарий
11	---	RST 0	C7	Переход по прерыванию на адрес 0000

11	---	RST 1	CF	0008
11	---	RST 2	D7	0010
11	---	RST 3	DF	0018
11	---	RST 4	E7	0020
11	---	RST 5	EF	0028
11	---	RST 6	F7	0030
11	---	RST 7	FF	0038

Кол-во тактов	Флаги ZSPC	Команда	Код	Комментарий
4	----	E1	FB	Разрешить прерывание
4	----	D1	F3	Запретить прерывание
4	---1	STC	37	Установить флаг переноса "C"
4	---x	CMC	3F	Инвертировать флаг переноса
4	----	NOP	00	Нет операции (холостая команда)
7	----	HLT	76	Останов до появления прерывания

Приложение 2. Описание симулятора МП КР580ВМ80

Данная программа предназначена для изучения функционирования МП КР 580ВМ80 (импортный аналог I8080), программирования на языке Ассемблер.

Интерфейс программы приведен на рис. 1. В нем можно выделить следующие блоки :

- блок состояния памяти;
- блок состояния регистров;
- блок битов состояний;
- управляющую панель с дисплеем;
- указатель стека;
- порты ввода и вывода.

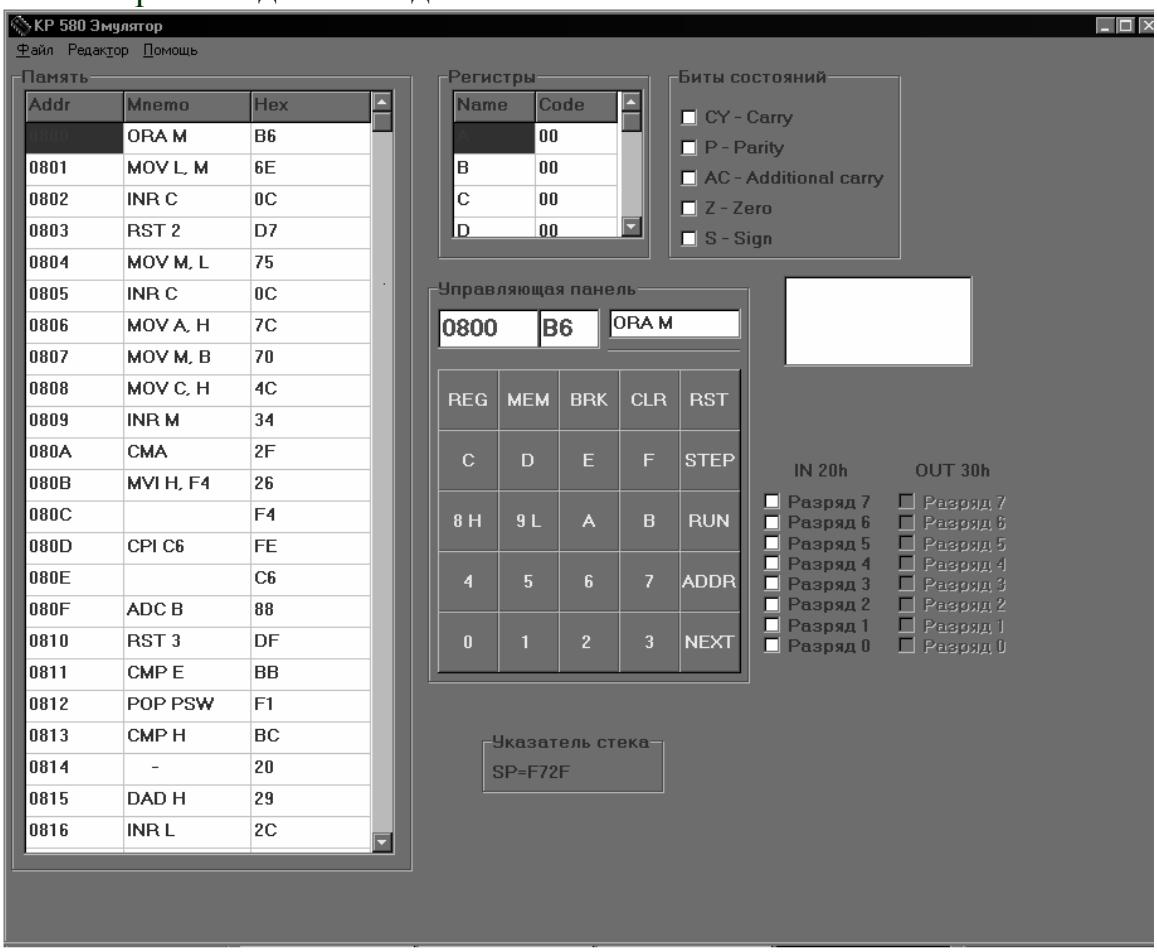


Рис. 1. Интерфейс симулятора микропроцессора КР580ВМ80 (I8080)

Программа может выполнять следующие функции:
- ввод информации и вызов директив с помощью клавиатуры;
- отображение вводимой информации в шестнадцатеричном коде на дисплее симулятора;
- запуск, выполнение и отладка программ пользователя.
- чтение и модификацию содержимого памяти, регистров
- выполнение и отладку пользовательских программ.

Для ввода программы пользователя необходимо :
- в меню “Файл” выбрать “Новый” (рис. 2).

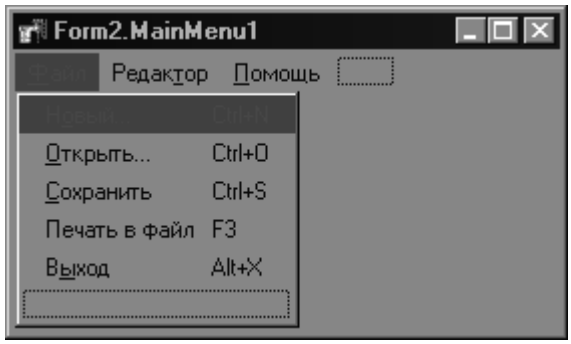


Рис. 2. Вид меню «Файл»

Теперь симулятор готов к записи программы пользователя. В блоке памяти по каждому адресу в исходном состоянии записываются пустые команды NOP (рис. 3).

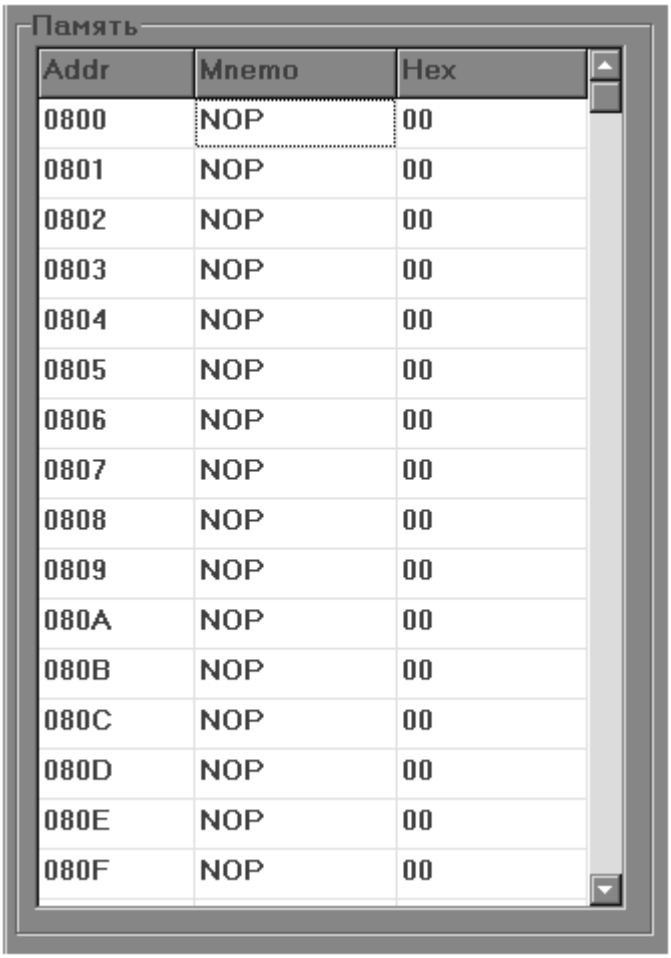


Рис.3. Окно отображения состояния памяти

Блок памяти содержит три поля:

- поле адресов ячеек памяти;
- поле мнемокодов команд;
- шестнадцатеричный (машинный) код команды.

Новую команду можно ввести несколькими способами:

1-й. способ:

а) указать в области памяти на нужный адрес и перейти к панели управления;

а) на управляющей панели нажать кнопку ADDR;

б) переместить указатель в ту часть дисплея, где отображается мнемокод команды, и записать новый, вводя его с клавиатуры компьютера (рис. 4).

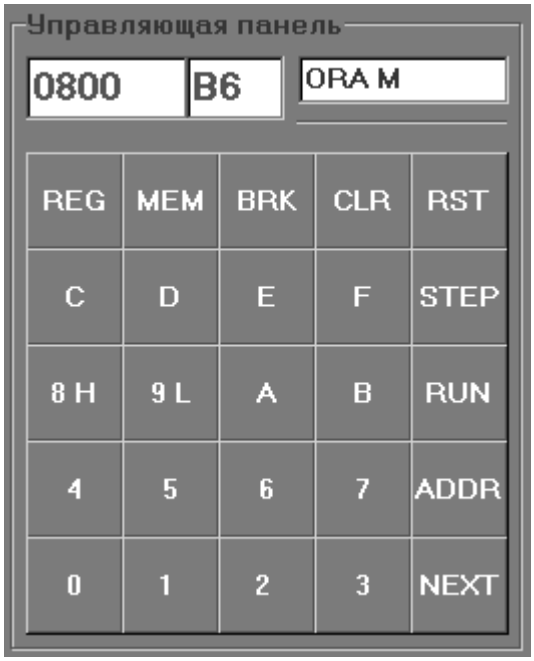


Рис. 4. Панель управления

2-ой способ:

а) указать в области памяти на нужный адрес, перейти к панели управления;

б) нажать клавиши ADDR, MEM и ввести код команды с клавиатуры управляющей панели.

Чтобы перейти к следующей ячейке памяти нажать ADDR, NEXT или нажать ADDR и ввести адрес, по которому запишется команда, пользуясь клавиатурой эмулятора. Адрес также можно выбрать на панели, отображающей состояние памяти, пользуясь полосой прокрутки.

После занесения программы в память, можно приступать к ее отладке и запуску. Сначала устанавливаем указатель памяти на начальный адрес. Эта процедура осуществляется нажатием кнопки RST.

После этого пользователь может вызвать пошаговый режим выполнения программы, нажав кнопку STEP (одно нажатие – выполнение одной команды), либо запустить программу клавишей RUN.

Проследить выполнение и увидеть результаты работы программы можно, воспользовавшись окнами отображения регистров, битов состояний, состоянием портов ввода-вывода (см. рис. 5, 6, 7).

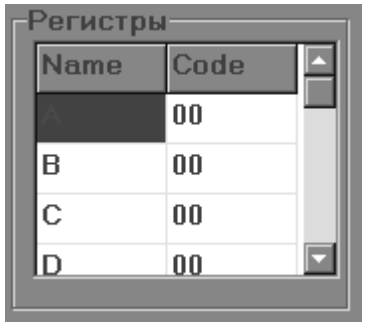


Рис. 5. Отображение состояния регистров

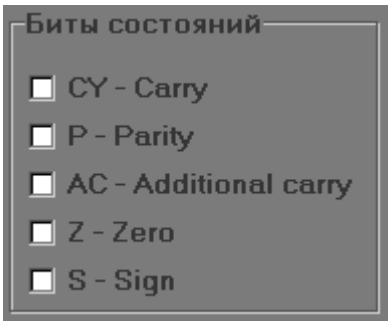


Рис. 6. Отображение битов состояний

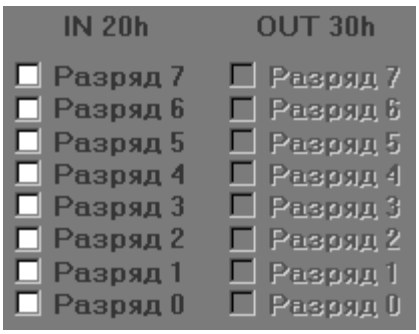


Рис. 7. Состояние портов ввода-вывода

В случае наличия ошибки в программе – симулятор выдает предупреждающее сообщение. Чтобы отредактировать введенную информацию достаточно остановить выполнение программы кнопкой RST и перейти к нужной ячейке памяти. Для этого предусмотрено и меню правки (рис. 8) со смещением на один адрес вверх или вниз с удалением текущей команды.

Рис. 8. Меню правки

В окне интерфейса также имеется специально выделенная область, в которую можно вносить различные записи о выполняемой программе на данном симуляторе: условие задания, комментарии.

Полученные результаты можно сохранять в удобном для пользователя виде, выбрав в меню «Файл» соответствующий формат файла: текстовый или файл памяти.

Выход из программы осуществляется нажатием комбинации клавиш Alt+X.

ISBN 985-485-085-4