МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**С.Н. Торгаев, Г.С. Воробьева,
И.С. Мусоров, Д.С. Чертихина**

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ: МИКРОПРОЦЕССОР INTEL 8080

*Рекомендовано в качестве учебного пособия
Редакционно-издательским советом
Томского политехнического университета*

Издательство
Томского политехнического университета
2014

УДК 681.322(075.8)
ББК 32.973.26-04я73
Т60

Торгаев С.Н.

Т60 Основы микропроцессорной техники: микропроцессор Intel 8080: учебное пособие / С.Н. Торгаев, Г.С. Воробьева, И.С. Мусоров, Д.С. Чертихина; Томский политехнический университет. − Томск: Изд-во Томского политехнического университета, 2014. – 208 с.

В пособии изложены основы функционирования микропроцессоров на примере микросхемы Intel 8080. Представлено подробное описание работы микропроцессора при выполнении кода программ и обработке внешних запросов (прерывание, ПДП и т. д.). Также приводится подробный курс лабораторных работ по изучению системы команд микропроцессора.

Предназначено для студентов, обучающихся по направлениям 11.03.04 «Электроника и наноэлектроника», 12.03.04 «Биотехнические системы и технологии».

УДК 681.322(075.8)
ББК 32.973.26-04я73

Рецензенты

Доктор технических наук, профессор ТГУ
Н.В. Евтушенко

Кандидат физико-математических наук
старший научный сотрудник Института оптики атмосферы СО РАН
Д.В. Шиянов

Содержание

ВВЕДЕНИЕ

На сегодняшний день микропроцессоры и микроконтроллеры нашли широкое применение практически во всех областях науки и техники, вытесняя традиционную цифровую технику на «жесткой логике». Достоинствами применения микропроцессорных устройств являются высокая гибкость, простота проектирования, а также возможность построения сложных алгоритмов обработки информации. Цифровые устройства на «жесткой логике» в настоящее время обычно используются при проектировании устройств, требующих максимального быстродействия, и в качестве устройств сопряжения микропроцессоров (микроконтроллеров) с другими устройствами.

Данное учебное пособие посвящено одному из базовых микропроцессоров, которым является микросхема Intel 8080. Изучение студентами ВУЗов принципов работы данного микропроцессора позволяет на простом уровне понять работу микропроцессоров и микроконтроллеров в целом. В пособии приводятся подробные описания архитектуры микропроцессора, действий, выполняемых микропроцессорам при выполнении кода программы, а также реакции микропроцессора на различные внешние сигналы.

ГЛАВА 1. КЛАССИФИКАЦИЯ И ХАРАКТЕРИСТИКИ МИКРОПРОЦЕССОРОВ. СТРУКТУРА МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

Микропроцессор – это программно-управляемое устройство, которое осуществляет обработку цифровой информации и управление этим процессом [1, 2, 3].

Микропроцессорные средства – это набор совместимых микросхем, таких как микропроцессор, микросхемы оперативной и постоянной памяти, устройства ввода-вывода и.т.д.

Микропроцессорной системой (МПС) называется вычислительная или управляющая система, которая построена на основе микропроцессорных средств [1, 2, 3].

Микрокомпьютер с небольшими вычислительными ресурсами и упрощенной системой команд, предназначенный не для только выполнения вычислений, но и для выполнения процедур логического управления различным оборудованием, называется ***микроконтроллером (МК)***. Также отличительной особенностью микроконтроллеров является наличие на кристалле большого многообразия периферийных устройств.

В микропроцессорных системах обработку всей информации и управление всем вычислительным процессом осуществляет микропроцессор. Именно микропроцессор выполняет все арифметические и логические операции (такие как сложение, умножение, сдвиг, сравнение и т.д.), временное хранение кодов программы и данных, пересылку данных между периферийными устройствами и т.д. Все дополнительные функции, такие как хранение программ и данных, связь между устройствами микропроцессорной системы, связь с персональным компьютером и т.д. выполняют периферийные устройства.

Микропроцессор выполняет все операции последовательно, то есть одну за другой. На данный момент существует большое количество микропроцессоров способных выполнять параллельно несколько операций. Недостатком последовательного выполнения операций является тот факт, что время выполнения алгоритма существенно зависит от его сложности, а следовательно скорость работы микропроцессорной системы будет невелика [4].

Все операции, которые будет выполнять микропроцессор, описываются управляющей программой. Программа состоит из команд, то

есть цифровых кодов (кодов операций), которые «говорят» микропроцессору, какую операцию необходимо выполнить. Набор команд, которые может выполнять микропроцессор, образуют его систему команд.

Микропроцессоры (микроконтроллеры) функционируют согласно предложенному в 1945 году Джоном фон Нейманом принципу программного управления, который заключается в следующем:

1. Информация кодируется в двоичной форме и разделена на единицы (элементы информации) – ***слова*** (количество бит, которое может обрабатывать микропроцессор одновременно – полубайт, байт, 2 байта и т. д., определяет разрядность шины данных, АЛУ).

2. Разнотипные слова информации – команды или данные – кодируются одинаковым способом и различаются только способом использования.

3. Информационные слова размещаются в ячейках памяти ЭВМ и идентифицируются номерами ячеек, называемыми ***адресами*** слов.

4. Алгоритм вычислений представляется в виде последовательности управляющих слов, называемых ***командами.*** Команды определяют:
а) наименование операции КОП;
б) информационные слова (числа), участвующие в операции.

5. Выполнение вычислений, согласно представленному алгоритму сводится к последовательному выполнению команд в порядке, ***однозначно определенном программой.*** Первым адресом является ***пусковой*** адрес.

1.1. Классификация микропроцессоров

1. По числу больших интегральных схем (БИС) различают [3]:

- **однокристальные;**
- **многокристальные;**
- **многокристальные секционные микропроцессоры.**

Однокристальные микропроцессоры реализованы в виде одной БИС или СБИС (сверхбольшой интегральной схемы). Возможности однокристальных микропроцессоров ограничены ресурсами кристалла и корпуса.

В многокристальном микропроцессоре организовано разделение структуры на функциональные блоки, реализованные в виде отдельных БИС или СБИС. Функциональные блоки микропроцессора выполняют определенные функции и могут работать независимо друг от друга.

Как правило, многокристальный процессор разбивается на следующие функциональные блоки: *операционный процессор, управляющий процессор* и *интерфейсный процессор.*

Операционный процессор выполняет обработку всех данных, а *управляющий процессор* служит для выборки, декодирования и вычисления адресов и также определяет последовательность команд. *Интерфейсный процессор* служит для подключения памяти, различных периферийных устройств и выполняет функцию канала прямого доступа к памяти (ПДП) [3].

2. По назначению различают:

- **универсальные микропроцессоры;**
- **специализированные микропроцессоры.**

Универсальные микропроцессоры предназначены для широкого круга задач. *Специализированные* микропроцессоры ориентированы на выполнение определенных функций, что позволяет значительно увеличить производительность. Область применения специализированных микропроцессоров достаточно широка. Как правило, они применяются для управления сложными техническими устройствами и технологическими процессами на производстве, транспорте, связи, военном деле, медицине и т.д. Особым классом специализированных микропроцессоров являются *цифровые сигнальные процессоры*, обладающие высокой производительностью при обработке аналоговых сигналов [3].

3. По виду обрабатываемых входных сигналов различают:

- **цифровые микропроцессоры;**
- **аналоговые микропроцессоры.**

В аналоговых микропроцессорах входные сигналы предварительно преобразуются в цифровую форму. После этого происходит обработка их в цифровом виде. Преобразованные в аналоговую форму сигналы поступают на выход.

4. По характеру временной организации работы разделяют:

- **синхронные микропроцессоры;**
- **асинхронные микропроцессоры.**

5. По количеству выполняемых программ разделяют:

- **однопрограммные микропроцессоры;**
- **многопрограммные микропроцессоры.**

Однопрограммные микропроцессоры могут выполнять только одну программу, а выполнение следующей программы возможно только по окончанию предыдущей. В многопрограммных микропроцессорах может одновременно выполняется несколько программ, что позволяет осуществлять управление большим числом источников или приемников информации [3].

1.2. Характеристики микропроцессоров

Основными характеристиками микропроцессоров являются:

1. ***Мощность микропроцессора***. Под мощностью МП принято понимать его способность обрабатывать данные. Её оценивают с помощью трех основных характеристик:

• ***Длина слова данных*** (количество бит (разрядов) обрабатываемых одновременно) может быть 4, 8, 16, 32 и 64 бита. Это наиболее часто используемая характеристика.

• ***Количество адресуемых слов памяти*** (зависит от количества разрядов шины адреса (ША). Для 16-разрядной ША это будет 2^{16}=65536=65К или 64К; для 20-разрядной ША 2^{20}=1048576=1М, т.е. одни миллион слов памяти, если слово соответсвует 1 байту, то 1 Мб, если слово = 2 байта, то 2 Мб). В памяти каждому слову присваивается ***адрес*** – номер его местоположения. Для того чтобы извлечь слово из памяти микропроцессору необходимо обратиться по соответствующему адресу. Все адреса памяти представляются в двоичной форме и начинаются с нулевого. Значения максимального адреса памяти в микропроцессорах разного типа различны, и чем больше это значение, тем больше вычислительная мощность микропроцессора.

• ***Скорость выполнения команд*** (время выполнения команд, тактовая частота).

2. ***Тип микроэлектронной технологии,*** используемый при изготовлении.

3. ***Количество кристаллов, образующих микропроцессор***, количество элементов (транзисторов) на кристалл, размеры кристалла;

4. ***Количество выводов корпуса кристалла***.

5. ***Тип управляющего устройства***, схемное или микропрограммное управление.

6. ***Эффективность системы команд***. Количество команд, выполняемые операции, возможные способы адресации, наличие команд работы со стековой памятью, команд операций с битами, десятичными числами, числами с плавающей точкой и т.д.

7. ***Число уровней прерываний***.

8. ***Возможность прямого доступа к памяти***.

9. ***Пропускная способность интерфейса ввода-вывода.***

10. ***Количество и уровни питающих напряжений.***

11. ***Номинальные параметры используемых сигналов, мощность, рассеиваемая БИС.***

12. ***Число входящих в МПК дополнительных БИС и СИС и выполняемые ими функции.***

13. ***Наличие и доступность*** для использования аппаратно-программных ***средств поддержки проектирования*** программ для микропроцессора и отладку микропроцессорного устройства.

1.3. Архитектуры микропроцессоров

Архитектура микропроцессора – это логическая организация микропроцессора, определяющая его возможности аппаратной, и программной реализации функций, необходимых для построения микропроцессорных систем.

Понятие архитектуры микропроцессора отражает:

- структуру, т. е. совокупность компонентов, составляющих микропроцессор, и связей между ними;
- способы представления и форматы данных;
- способы обращения ко всем элементам структуры, которые доступны для пользователя (программно-доступны);
- набор операций, выполняемых микропроцессором, т. е. его система команд;
- форматы управляющих слов и сигналов, которые вырабатываются микропроцессором и поступают в него;
- реакцию микропроцессора на поступающие внешние сигналы и другие характеристики.

Существует два вида архитектур микропроцессоров: **RISC (Reduced Instruction Set Computing)** и **CISC (Complex Instruction Set Computing)**.

1. Для RISC архитектуры характерно несколько шин данных и адресов и упрощённая система команд.

2. Для CISC архитектуры характерен полный набор команд, имеется одна шина данных и одна шина адреса. Данная архитектура более лояльна к пользователю.

Современные микропроцессоры имеют RISC ядро и CISC оболочку.

На данный момент разрабатываются процессоры с архитектурой типа **MISC** (**Minimum Instruction Set Computing**) с минимальным набором команд и весьма высоким быстродействием (в настоящее время эти модели находятся в стадии разработки).

Гарвардская архитектура

Гарвардская архитектура характеризуется тем, что память для данных и память программ располагаются в разных местах, таким образом, в данной архитектуре возможно совмещение операций вызова команды из памяти и ее выполнения. Стандартные микропроцессоры, характеризуются архитектурой фон-Неймана, в которой данные и команды передаются, как показано на рис. 1.1 [7].

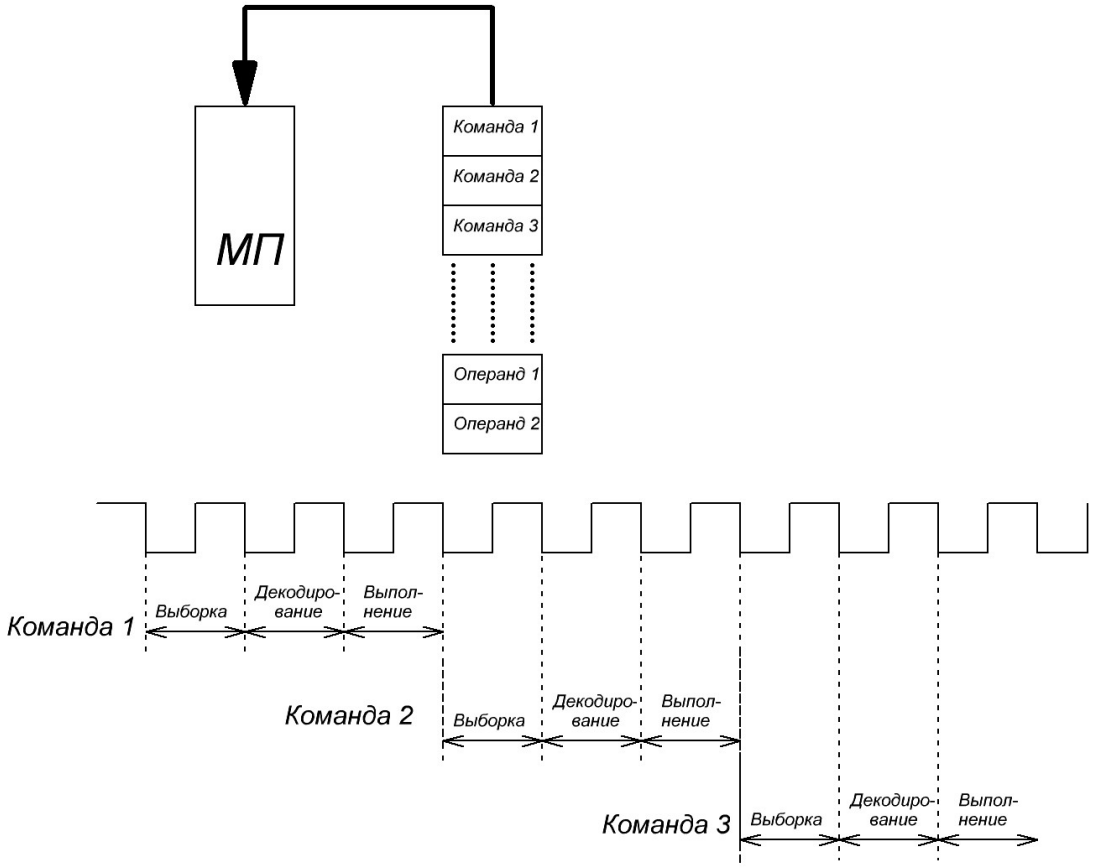

Рис. 1.1. Структура стандартного микропроцессора (архитектура фон-Неймана)

Предположим, что при использовании микропроцессора со стандартной архитектурой (фон-Неймана) требуется выполнить 3 команды. Обычно каждая из команд включает в себя три этапа:

1) этап выборки команды (кода операции) из памяти;

2) этап декодирования команды;

3) этап выполнения команды.

В стандартном микропроцессоре команды программы (т.е. программный код) и данные содержатся в одной области памяти (рис. 1.1).

Следовательно, вызов следующей команды невозможен при выполнении текущей, это связано с тем, что эти операции обе требуют обращения к памяти. В гарвардской архитектуре (рис. 1.2), в которой коды операций и данные хранятся в различных областях памяти, возможно совмещение вызова следующей команды во время выполнения текущей операции (рис. 1.2) [7].

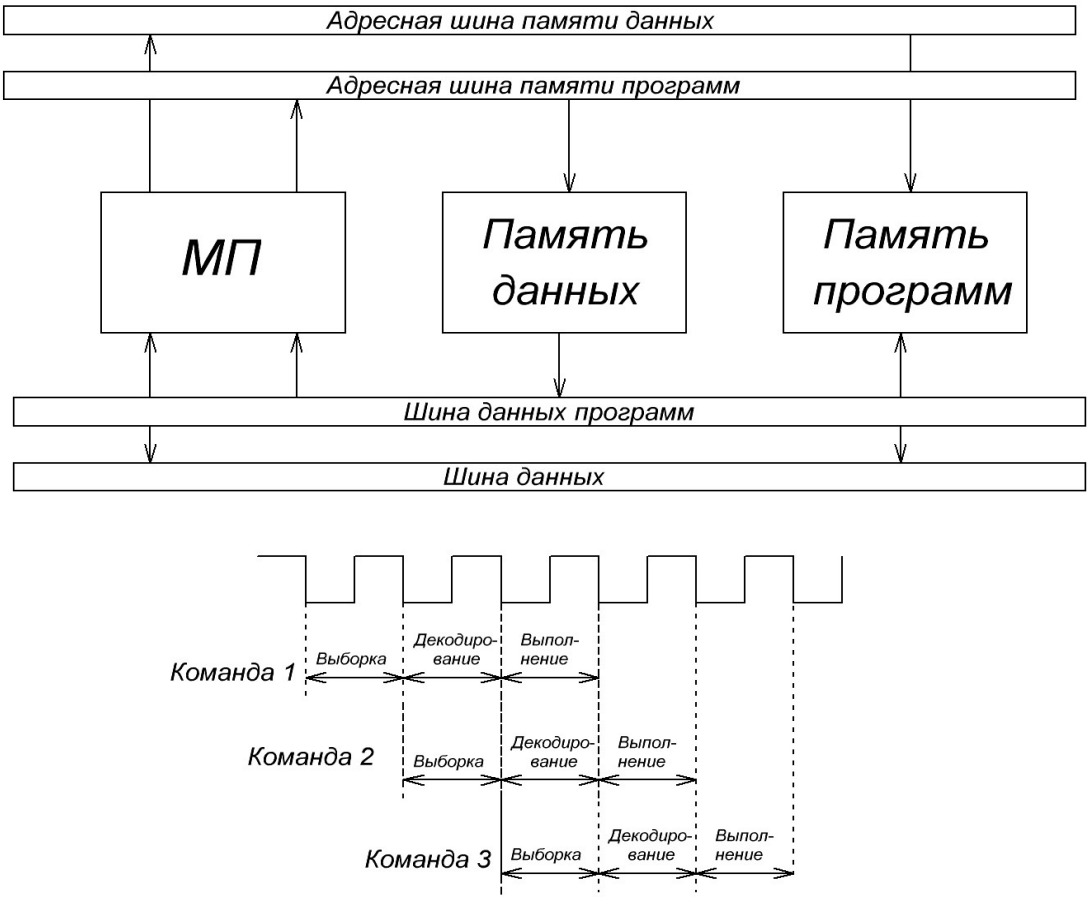

Рис. 1.2. Гарвардская структура

Как правило, память программы содержит программный код, а память для хранения данных содержит значения переменных. Классическая гарвардская архитектура используется лишь в небольшом количестве микропроцессоров, предназначенных для цифровой обработки сигналов (например, Motorola DSP56000 и т.д.), а чаще всего применяется модифицированная архитектура (например, в микропроцессорах TMS320 фирмы Texas Instruments). В модифицированной Гарвардской архитектуре память также разделена на две части, но в отличие от строгой гарвардской архитектуры в модифицированной архитектуре разрешена связь между двумя областями памяти [7].

Конвейерная обработка

Конвейерной обработкой называется метод, позволяющий совместить выполнение несколько операций. При конвейерной обработке задача разбивается на подзадачи, которые совмещаются в процессе выполнения. Данный метод позволяет повысить скорость работы системы и часто применяется при цифровой обработке сигналов. Подзадачи, на которые разбивается основная задача, называются каскадами конвейера.

Как отмечалось ранее, команда может быть разбита на три этапа. В конвейерной обработке каждый этап команды можно рассматривать как каскад конвейера. Таким образом, можно организовать одновременное выполнение команд (рис. 1.3).

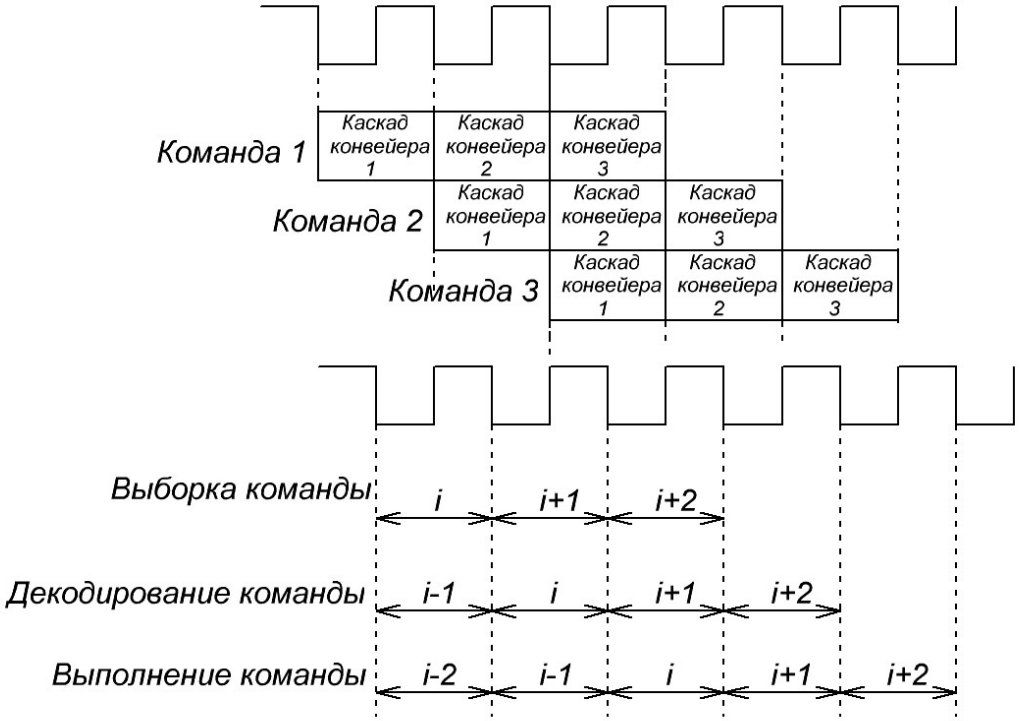

Рис. 1.3. Конвейерная обработка

На рис. 1.3 приведена временная диаграмма работы трехкаскадного конвейера. При конвейерной обработке каждый этап занимает один машинный цикл. Таким образом, в течение одного машинного цикла могут быть одновременно активны до трех команд, но каждая из них будет находиться на разной стадии обработки. При использовании конвейерной обработки команд все три части команды (т.е. выборка из памяти, декодирование и выполнение) независимы, тем самым обеспечивается выполнение нескольких команд одновременно. На рис. 1.3 показано, что в ходе 1-го цикла микропроцессор может одновременного извлекать

из памяти *i*-ую команду, декодировать (*i* - 1)-ую команду и выполнять (*i* - 2)-ую команду.

На данный момент практически во всех современных микроконтроллерах осуществляется конвейерная обработка. Например, трехкаскадный конвейер, рассмотренный выше, применяется в процессорах TMS320 (компании Texas Instruments). Благодаря использованию в данных микропроцессорах внутреннего распределения потока команд удается достаточно сильно уменьшить время выполнения каждой команды.

Пропускная способность микропроцессоров и микроконтроллеров с конвейерной обработкой определяется числом команд, которые обрабатываются в конвейере за единицу времени. Время перехода команды к следующему этапу (рис. 1.3) равно одному циклу и определяется наиболее медленным каскадом. Реальное увеличение скорости будет ниже из-за служебных задержек на организацию конвейера, задержек в регистрах и т.п.

Конвейерная обработка достаточно сильно влияет на работу с памятью в микропроцессорной системе. В микроконтроллере с конвейерной обработкой увеличивается число обращений к памяти, особенно при увеличении числа этапов обработки. При реализации цифровой обработки сигналов конвейерная обработка облегчается при использовании гарвардской архитектуры, в которой, как отмечалось ранее, данные и команды размещаются в различных областях памяти [7].

1.4. Структура микропроцессорной системы

На рис. 1.4 приведена упрощённая типичная структура микропроцессорной системы.

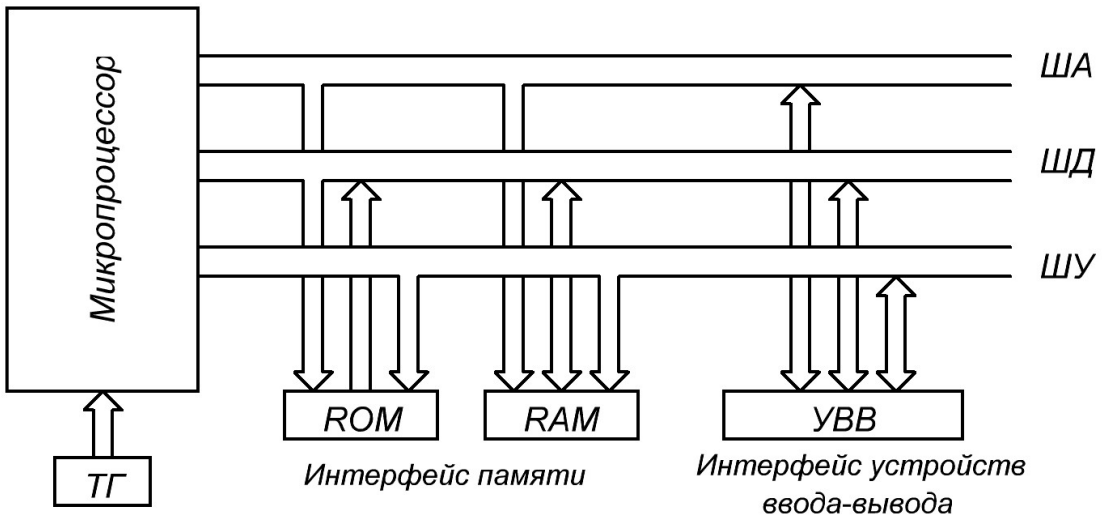

Рис. 1.4. Структура микропроцессорной системы

В микропроцессорной системе центральное место занимает микропроцессор. Микропроцессор выполняет следующие функции:
1) арифметические и логические операции;
2) управление процессом обработки информации;
3) взаимодействие устройств, входящих в систему.

Работа микропроцессора происходит под воздействием сигналов схемы синхронизации и начальной установки (таймера), часто выполненной в виде отдельного кристалла.

Шинная структура связей

Шинная структура связей позволяет достичь максимальной универсальности и упростить протоколы обмена информации в микропроцессорных системах [4, 6]. На рис. 1.5 представлена обобщенная схема шинной структуры связей.

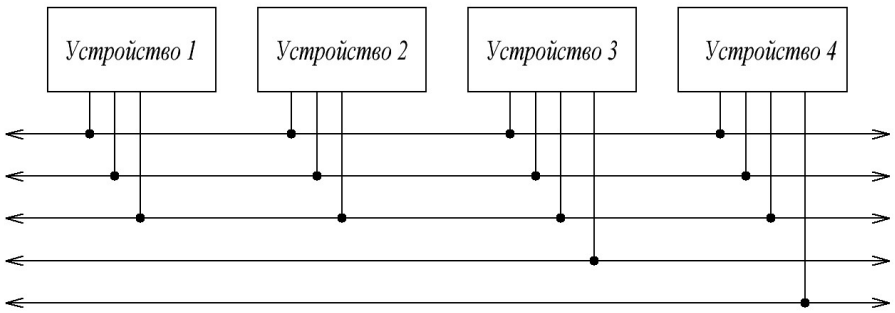

Рис. 1.5. Шинная структура связей

В данной схеме все сигналы между различными устройствами передаются по одним и тем же линиям связи, но с разделением во времени. Такой способ называется мультиплексированной передачей. Данная структура подразумевает так называемую двунаправленную передачу, т.е. передача по всем линиям связи осуществляется в обоих направлениях. Таким образом, при шинной структуре связи уменьшается количество линий связи, при этом упрощаются (унифицируются) протоколы обмена информацией [4].

Основным достоинством данной структуры связей является унификация протоколов обмена информацией, а соответственно, все узлы, отвечающие за обмен с шиной в этих устройствах, должны быть единообразны, унифицированы.

Основным недостатком данной структуры связей является то, что вся информация между различными устройствами передается последовательно (по очереди), что снижает быстродействие системы. Также к

недостаткам необходимо отнести то, что все устройства подключаются к линиям связи параллельно, следовательно, неисправность любого устройства может вывести из строя всю систему.

В системах с шинной структурой связей, как правило, применяются следующие выходные каскады микросхем:

- выход с открытым коллектором;
- выход с тремя состояниями с возможностью отключения (с Z-состоянием).

Данные выходные каскады могут объединять несколько выходов микросхем для получения мультиплексированных и двунаправленных линий связи. При этом в случае применения микросхем с Z-состоянием необходимо, чтобы на линии всегда был только один активный выход, а все остальные выходы находились в третьем состоянии. Каскады с открытым коллектором могут работать одновременно [4].

Представленная на рис. 1.4 структура микропроцессорной системы имеет ***магистрально-модульный принцип построения***. Различные блоки данной системы являются унифицированными модулями.

Связь между различными модулями и обмен информацией между ними осуществляется по коллективным шинам (магистралям), к которым подключены все модули микропроцессорной системы. ***В каждый момент времени возможен обмен информацией только между двумя модулями системы.*** Таким образом, обмен информацией производится путем ***разделения во времени*** модулями системы коллективных шин (магистралей).

Магистральный принцип сопряжения модулей (интерфейса) предполагает наличие:

1) информационно-логической совместимости различных модулей микропроцессорной системы, которая обеспечивается за счет использования единых способов представления информации;

2) алгоритма управления обменом информации между различными моделями;

3) единых форматов команд;

4) способа синхронизации всех компонентов системы.

Для большинства современных микропроцессоров характерна трехшинная структура, которая содержит шину адреса (***ША***), двунаправленную шину данных (***ШД***) и шину управления (***ШУ***). Также в микропроцессорных системах выделяют еще шину питания (***ШП***). Типовая структура микропроцессорной системы предполагает наличие общего сопряжения (общего или единого интерфейса) для модулей па-

мяти - ПЗУ и ОЗУ и периферийных устройств – внешних ЗУ, устройств ввода/вывода.

Шина адреса предназначена для определения адреса того устройства, с которым микропроцессор обменивается информацией. Каждому периферийному устройству в микропроцессорной системе и каждой ячейке памяти присваивается собственный адрес. Когда код какого-то адреса появляется на ША, устройство, которому этот адрес принадлежит, понимает, что ему предстоит обмен информацией. В микропроцессорной системе ША может быть как однонаправленной, так и двунаправленной.

Шина данных используется для передачи информационных кодов (кодов операций или данных) между различными устройствами и модулями микропроцессорной системы. Как правило, в пересылке информации участвует микропроцессор, передающий код данных в какое-то устройство или в ячейку памяти или же принимающий код данных из какого-то устройства или из ячейки памяти. Передача информации между устройствами возможна и без участия процессора. ШД в МПС является двунаправленной.

В состав ШУ входят отдельные управляющие сигналы. Каждый из этих сигналов несет свою уникальную функцию. Ряд сигналов служит для стробирования принимаемых или передаваемых данных, а другие управляющие сигналы используются для подтверждения приема-передачи данных, для установки всех устройств, которые входят в МПС, в исходное состояние, для тактирования устройств и т.д. Линии ШУ бывают как однонаправленные, так и двунаправленные.

Шина питания предназначена для питания системы. В ее состав входят линии питания (одна или несколько) и общий провод. В МПС может быть как один источник питания (чаще +5 В) так и несколько (обычно еще -5В, ±3.3В, ±12В). В современных микроконтроллерах и микропроцессорах, как правило, применяется низковольтное питание. Это связано с уменьшением размеров внутренних элементов кристаллов и необходимостью повышения частоты работы микросхем. Все устройства микропроцессорной системы подключены к линиям питания параллельно.

В качестве периферийных устройств в микропроцессорных системах используются дисплеи, АЦП, ЦАП, таймеры и др.

Периферийное устройство соединяется с шинами микропроцессора не непосредственно, а ***через программируемый связной адаптер (ПСА) или программируемый периферийный адаптер (ППА)***. Программируемый периферийный адаптер обслуживает периферийные устройства с

передачей информации в параллельном коде, программируемый связной адаптер обслуживает устройства с передачей информации в последовательном коде. Наличие программно настраиваемых адаптеров делает весьма гибкой и функционально богатой систему ввода-вывода информации в микропроцессорную систему.

В структуре микропроцессора интерфейс является узким местом из-за ограниченного числа выводов корпуса. Узкий интерфейс МП приводит к необходимости использования двунаправленных ***линий*** передачи информации, что сопровождается усложнением схем буферов и необходимостью использования временного мультиплексирования шин. Мультиплексирование шин позволяет при ограниченном числе линий передавать по ним различную информацию: адреса, данные или команды. Однако это требует наличия специальных линий для идентификации передаваемой информации и приводит к снижению скорости передачи информации через интерфейс.

Одним из классических микропроцессоров является Intel 8080. Изучение данного микропроцессора позволяет ознакомиться с внутренней структурой и принципами выполнения программного кода микропроцессоров в целом.

ГЛАВА 2. МИКРОПРОЦЕССОР INTEL 8080

2.1. Интерфейс микропроцессора Intel 8080

Интерфейс микропроцессора ***Intel 8080*** представлен на рис. 2.1.

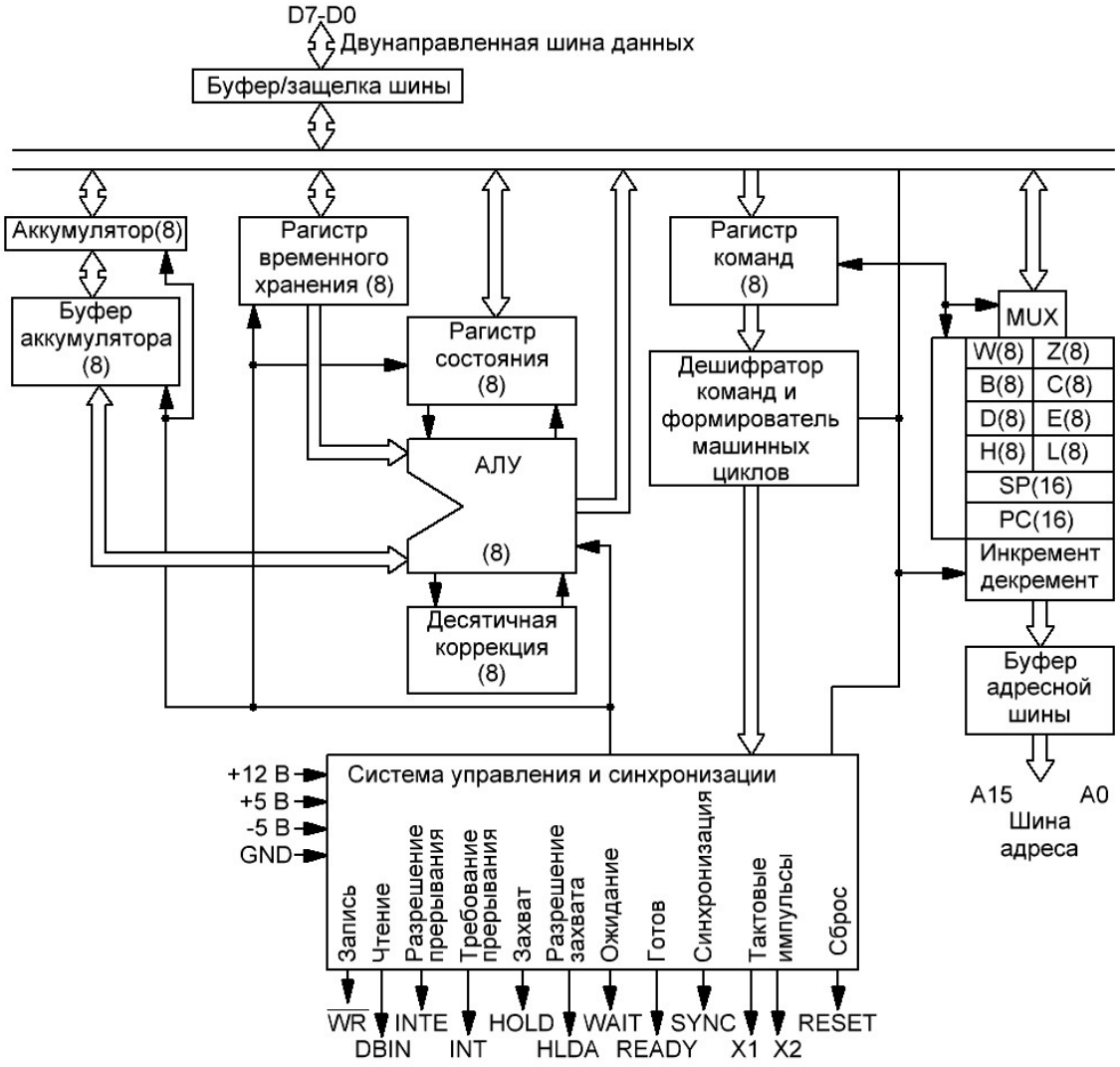

Рис. 2.1. Интерфейс микропроцессора ***Intel 8080***

В микропроцессоре имеются три внутренние шины: шина данных, шина адреса и шина управления, работающие по принципам разделения во времени [5].

Шина адреса (ША), которая состоит из 16 линий $A_{15\text{-}0}$, предназначена для передачи адресов от МП в память и порты ввода-вывода. Ад-

ресный канал обеспечивает адресацию внешней памяти объемом до 64 Кбайт, а также адресацию 256 устройств ввода и 256 устройств вывода.

Шина данных (информационная) D_7-D_0 есть двунаправленный трехстабильный вход – выход. Канал данных обеспечивает двухсторонний обмен информацией микропроцессора с памятью и устройствами ввода-вывода для обработки данных и команд.

Управление состоянием трехстабильных буферов осуществляется внешними сигналами выбора кристалла *CS* (разрешение кристалла) и *DS* (выбора устройства). Они формируются дешифрированием сигналов со старших линий шины адреса.

Шина управления (ШУ), которая состоит из 10 линий, служит для передачи управляющих сигналов, которые определяют порядок функционирования и характер компонентов микропроцессорной системы МПС.

A. Группа сигналов управления состоянием микропроцессора:

1. ***Входной сигнал сброса RESET (RST)***, устанавливающий внутренние узлы МП в исходное состояние и начинающий выполнение программы с нулевой ячейка памяти.

2. ***Входной сигнал готовности READY*** служит для подачи выходного сигнала, указывающего на готовность действительных данных к вводу в микропроцессор или на готовность памяти и внешних устройств к приему информации с шин данных. Сигнал позволяет синхронизовать работу микропроцессора с памятью и внешними устройствами любого быстродействия.

3. ***Выходной сигнал ожидания WAIT*** служит для выдачи выходного сигнала, подтверждающего, что микропроцессор находится в состоянии ожидания, что микропроцессор остановлен.

B. Группа сигналов управления шинами адреса и данных:

1. ***Выходной сигнал считывания*** (приема) ***DBIN (чтение, прием)*** предназначен для выдачи выходного сигнала, показывающего, что шины данных находятся в режиме приема, т.е. микропроцессор ожидает поступления данных от памяти, устройств ввода-вывода или других устройств системы. Слово из внешнего регистра, которое будет представлено на ШД, загружается в один из внутренних регистров микропроцессора.

2. ***Выходной сигнал записи WR (выдача, запись)*** служит для выдачи выходного сигнала, показывающего, что микропроцессор выдал на шины данных действительно данные, для записи в память или внешнее устройства ввода-вывода.

3. ***Входной сигнал запроса шины*** (захвата) ***HOLD*** предназначен для подачи сигнала от внешних устройств с целью запроса доступа к шинам адреса и данных. Входной сигнал запроса (захвата) шины «говорит» микропроцессору о том, что периферийное устройство должно использовать ША и ШД для прямого обмена данными с памятью МПС без участия МП. При поступлении этого сигнала микропроцессор временно останавливает выполнение текущей программы и переводит буферные регистры шин адреса и данных в высокоимпедансное состояние, т.о. реализует режим прямого доступа к памяти (ПДП).

4. ***Выходной сигнал подтверждения запроса шин*** (захвата) ***HLDA*** является ответным на сигнал «захват» и показывает, что шины адреса и данных находятся в высокоимпедансном состоянии и ими может распоряжаться ПУ, инициировавшее ПДП.

C. Группа сигналов, связанных с прерываниями, т.е. с запросами на обмен информацией от периферийных устройств:

1. ***Входной сигнал запроса на прерывание INT*** означает готовность к обмену данными периферийного устройства с микропроцессором. Микропроцессор в ответ на этот сигнал останавливает выполнение текущей программы, запоминает ее состояние и адрес возврата, обслуживает периферийные устройства, после окончания обработки прерывания микропроцессор восстанавливает исходное состояние и возобновляет выполнение прерванной программы.

2. ***Выходной сигнал разрешения прерывания INTE*** инициирует состояние внутреннего триггера разрешения прерывания и определяет возможность обслуживания микропроцессором запросов на прерывания от периферийных устройств. При высоком уровне ***INTE*** микропроцессор реагирует на сигнал запроса ***INT***, а при низком уровне сигналы запроса на прерывания игнорируются микропроцессором. Сигнал ***INTE*** является выходным сигналом внутреннего триггера разрешения прерываний. Его состоянием можно управлять программно. ***Команда EI – разрешает прерывания. Команда DI – запрещает прерывания.***

D. Линии синхронизации и питания. МП имеет 3 линии синхронизации

22 – Ф1

15 – Ф2

19 – SYNC

Двухфазные непрерывающиеся сигналы от главного генератора синхронизации подаются на входы ***Ф1*** и ***Ф2***. С их помощью определяются моменты выполнения большинства действий в МПС, т.е. создается

временная база функционирования системы. Производительность системы определяется периодом сигналов синхронизации.

Начало тактов синхронизации ***SYNC*** идентифицирует начало каждого машинного цикла (МЦ), в течение которого микропроцессор адресуется к внешнему регистру и обменивается с ним данными, а так же, при необходимости, производит внутреннее преобразование данных.

Одновременно с сигналом ***SYNC*** МП выдает на ШД байт состояния, который содержит информацию о действиях МП в текущем МЦ.

20 - +5В ***28 - +12В***
11 - -5В ***2 - GND***

2.2. Архитектура микропроцессора Intel 8080

Арифметико-логическое устройство (АЛУ)

Арифметико-логическое устройство выполняет обработку данных, арифметические и логические операции над 8-разрядными числами в процессе межрегистровой пересылки комбинационного типа. Арифметико-логическое устройство имеет два порта: один входной и один выходной. К каждому порту присоединен его собственный буферный регистр, который способен хранить для АЛУ ***одно слово данных***. Входные порты позволяют арифметико-логическому устройству принимать данные как с внутренней шины данных, так и из специального регистра, который называется ***аккумулятором (А)***. Единственный выходной порт АЛУ предоставляет ему возможность пересылать слово данных в аккумулятор.

Основной задачей ***аккумулятора*** является хранение слова данных, которое было послано в него из выходного порта АЛУ или извлечено из памяти. Когда арифметико-логическое устройство складывает два слова данных, одно из них находится в аккумуляторе. После выполнения сложения ***результат*** посылается в аккумулятор.

В зависимости от вида выполняемой операции арифметико-логическое устройство оперирует одним или двумя словами данных. Например, при сложении используются два слова данных, и следовательно, оба входа АЛУ. При выполнении операции инвертирования слова АЛУ использует только аккумулятор.

Функции АЛУ зависят от типа микропроцессора, но самыми распространенными являются сложение, вычитание, логическое умножение (И), логическое сложение (ИЛИ), исключающее ИЛИ, инверсия, сдвиги вправо и влево, приращение положительное (инкремент), приращение отрицательное (декремент).

Регистры микропроцессора

Регистры являются важной составной частью любого микропроцессора, т.к. они участвуют в реализации основных логических функций; регистры делятся на ***специальные регистры и регистры общего назначения (РОН)***.

Количество и назначение регистров зависит от архитектуры микропроцессора. ***i8080*** содержит программно доступные 8-разрядные регистры:

- аккумулятор;
- регистры общего назначения – ***B, C, D, E, H, L***;
- регистр признаков ***F*** (регистр состояния),

и 16-разрядные специализированные

- счетчик команд(программный счетчик ***PC***);
- регистр-указатель стека (***SP***);
- сдвоенный регистр косвенного адреса ***HL*** (регистр адреса/данных), где ***H*** – регистр старшего байта адреса, ***L*** – младшего.

Кроме того, имеются программно недоступные регистры:

- 8-разрядные регистры временного хранения ***T, W, Z***;
- 8-разрядный регистр команд ***IR***;
- 16-разрядный регистр адреса ***PA***.

Имеется возможность использования содержимого пар регистров ***B*** и ***C, D*** и ***E, H*** и ***L*** как составных слов двойной длины.

Аккумулятор

Аккумулятор – главный регистр микропроцессора. Все арифметические и логические операции выполняются за счет использования арифметико-логического устройства и аккумулятора. Каждая из таких операций над данными (операндами) предполагает размещение одного из них в аккумуляторе, а другого в памяти или другом регистре. Например, при сложении двух чисел (данных в виде слов) ***D*** и ***B***, расположенных в аккумуляторе и памяти соответственно, результирующая сумма ***C*** загружается в аккумулятор, замещая слово ***D***. Таким образом, результат операции арифметико-логического устройства всегда размещается в аккумуляторе, а исходные данные теряются.

Также аккумулятор используется при операциях программируемой передачи данных между различными частями микропроцессора. Выполнение данных операций производится в два этапа:

1) сначала осуществляется пересылка данных –из источника в аккумулятор;

2) а затем из аккумулятора данные пересылаются в пункт назначения.

Кроме того, микропроцессор выполняет некоторые действия над данными непосредственно в аккумуляторе.

В команде адрес аккумулятор в явном виде отсутствует. На использование аккумулятора в операции указывает КОП команды. Иными словами, в отношении аккумулятора применяется неявная адресация, что позволяет применять одноадресные команды, которые имеют сравнительно короткий формат.

Счетчик команд, программный счетчик (PC)

Счетчик команд – один из наиболее главных регистров микропроцессора. Как отмечалось выше, программа – это последовательность команд, которые хранятся в памяти микропроцессорной системы и которые предназначены для инструктирования микропроцессора, что необходимо делать для решения поставленной задачи. Счетчик команд (программный счетчик) следит за тем, какая команда выполняется в данный момент, а какая команда будет выполняться в следующей. Разрядность счетчика команд определяется разрядностью шины адреса. Чтобы обратиться к любому из этих адресов, счетчик команд должен располагать 16 двоичными разрядами. ***Где бы команды не располагались, они следуют друг за другом в определенном порядке.***

По внешнему сигналу ***RESET*** или при включении питания в счетчик команд загружаются данные из заданной проектировщиком области памяти (как правило, в счетчик команд при этом загружается нулевой адрес). В связи с этим, перед запуском начальный адрес для программы необходимо поместить в область памяти, которая указана проектировщиком. В отличие от аккумулятора и других регистров общего назначения счетчик команд не выполняет операции различного типа.

Перед выполнением программы в счетчик команд необходимо загрузить адрес, который содержит первую команду программы пользователя. Регистр адреса памяти (***RA***) и шина адреса расположены после счетчика команд. Адрес, содержащийся в программном счетчике, копируется из счетчика команд в регистр адреса памяти (***RA***), после чего содержимое данных регистров становится одинаковым.

Адрес начальной команды отправляется по шине адреса к схемам управления микросхемой памяти, в результате этого происходит считывание содержимого области памяти с соответствующего адреса. Далее происходит пересылка этой команды в специальный регистр микропроцессора, который называется регистром команд ***IR***. После считывания

команды из памяти микропроцессора делается приращение содержимого счетчика команд (приращение осуществляется на количество байт, которое занимает данная команда в области памяти). Счетчик команд получает это приращение в момент, когда микропроцессор начинает выполнять текущую команду. Следовательно, счетчик команд начинает указывать на адрес команды, которая будет выполняться следующей (адрес следующей команды). Об этом необходимо помнить, в связи с тем, что программируя работу микропроцессора, можно столкнуться с необходимостью использования текущего значения счетчика команд. Для этого нужно знать, ***что в каждый момент времени счетчик команд указывает не на команду, выполняемую в данный момент, а на команду, следующую за ней.***

На рис. 2.2 представлена схема считывания памяти.

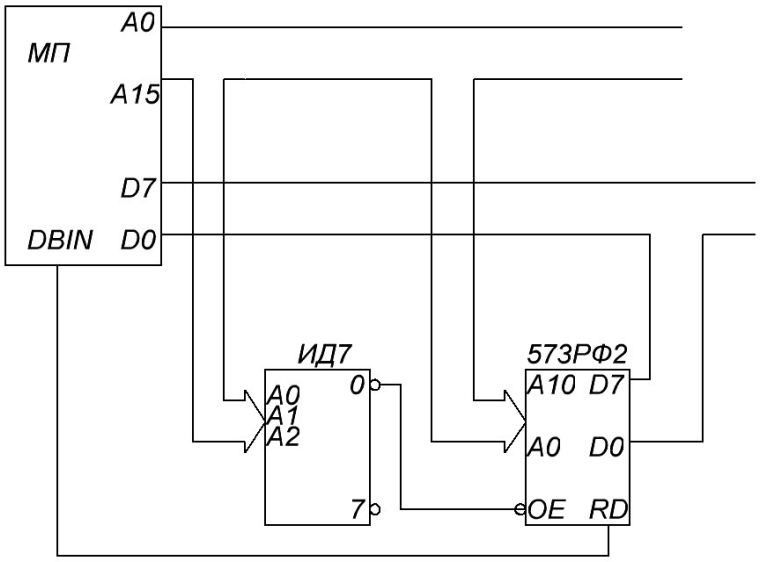

Рис. 2.2. Пример структуры считывания с памяти

Регистр адреса памяти

При обращении к памяти микропроцессорной системы регистр адреса памяти (*RA*) указывает на адрес ***ячейки памяти, которая подлежит использованию микропроцессором. Выходом данного регистра является шина адреса***. которая используется для выбора области памяти или порта ввода-вывода.

В течение цикла выборки кода операции регистр адреса памяти и счетчик команд имеют одинаковое содержимое, т.е. и регистр адреса, и счетчик команд указывают на местоположение извлекаемой из памяти команды. После декодирования счетчик команд получает приращение

равное размеру выполняемой команды. При этом р**егистр адреса памяти приращения не получает.**

При выполнении команды содержимое регистра адреса памяти зависит от выполняемой команды. Если при выполнении команды микропроцессору необходимо выполнить еще одно обращение к памяти, то регистр адреса памяти подлежит дополнительному использованию, а при этом содержимое программного счетчика остается неизменным.

На данный момент практически все микропроцессоры имеют команды, которые позволяют загружать этот регистр содержимым программного счетчика, регистрами общего назначения или области памяти.

Регистр команд (IR)

Регистр команд хранит код операции текущей выполняемой команды. Регистр команд соединен с внутренней шиной данных, однако он не может отправлять данные, а может только принимать данные. Не смотря на то, что функции регистра команд ограничены, его роль в работе микропроцессора велика, т.к. выход этого регистра является частью дешифратора команд. Таким образом, команда сначала извлекается из памяти, затем значение счетчика команд изменяется, и он начинает указывать на адрес следующей команды. При считывании команды из памяти на внутреннюю шину данных помещается копия кода операции, которая после этого пересылается в регистр команд. Далее начинается выполнения команды. В то время когда дешифратор команд считывает содержимое регистра команд (***IR***), он сообщает микропроцессору, действия, необходимые для реализации операций команды. Число разрядов регистра команд определяется типом микропроцессора и чаще всего совпадает с числом разряда слова данных.

Пример: Выполнение программы:

JMP 0800h
MOV B,C
ADI 0C0h

Память программ имеет вид:

0000h	*JMP (0C3h)*
0001h	*00 LB*
0002h	*08 HB*
0003h	*…*
0004h	*…*

0800h	*MOV B,C*
0801h	*ADI*
0802h	*C0*
0803h	00
0804h	…

• После запуска системы или сброса (***RESET***) в программный счетчик записывается нулевой адрес. (***PC***)←0000h.

• Данный адрес поступает в регистр адреса памятии в регистр команд считывается код первой команды: (***IR***)←00C3h.

• После этого дешифратор команд читает содержимое регистра команд (***IR***) и «говорит» микропроцессору, что необходимо сделать, а в программный счетчик записывается адрес следующей команды: (***PC***)←0003h. При этом осуществляется последовательное считывание кода операции и операндов данной команды посредствам инкрементирования значения регистра адреса памяти.

• Так как команда JMP обеспечивает переход в область памяти по адресу 0800h, следовательно, при ее выполнении в счетчик команд (программный счетчик) записывается адрес перехода: (***PC***)←0800h.

• Далее в регистр команд записывается код операции, находящийся по адресу 0800h. Так как по данному адресу лежит однобайтная команда, то в программный счетчик записывается значение (***PC***)←0801h.

• После выполнения команды MOV в регистр команд записывается код операции сложения аккумулятора со значением 0C0h (ADI 0C0h), при этом дешифратор команд, анализируя, что данная команда является трехбайтной, записывает в счетчик команд значение (***PC***)←0804h. При этом осуществляется последовательное считывание кода операции и операндов данной команды посредствам инкрементирования значения регистра адреса памяти.

Регистр состояния, регистр признаков, F-регистр, флаги

Регистр состояния – специализированный регистр (регистр кода состояния) – это набор ячеек памяти (триггеров), показывающих состояние ЦП. Каждая ячейка памяти физически представляет собой триггер, однако программисты предпочитают их называть ***флагами*** или ***флажками***. Каждый триггер регистра кода состояния содержит информацию, отражающую ***результат выполнения последней команды программы.*** Эта информация используется для организации условных переходов. В

рассматриваемом микропроцессоре ***i8080*** отдельные биты кода состояния имеют значения, представленные в табл. 2.1.

Таблица 2.1

Регистр состояния ***i8080***

Имя	**Разряд**	**Имя и описание**
S	7	***Признак знака.*** Принимает значение старшего разряда результата. 0101 1110 + 0101 1010 = 1011 1000, S=1 0101 1110 + 0001 1010 = 0111 1000, S=0
Z	6	***Признак нуля.*** Если результат равен 0, то Z=1, иначе Z=0. 0101 1110 + 0101 1010 = 1011 1000, Z=0 1111 1111 + 0000 0001 = 1 0000 0000, Z=1
-	5	Не используется, равен 0
AC	4	***Признак вспомогательного переноса.*** Если есть перенос между тетрадами байта, то AC=1, иначе AC=0. 0110 1110 + 0000 1000 = 0111 0110, AC=1 0110 1110 + 0000 0001 = 0110 1111, AC=0
-	3	Не используется, равен 0
P	2	***Признак четности.*** Если число единиц в байте результата четно, то P=1, иначе P=0. 0110 1110 + 0000 1000 = 0111 0110, P=0 0110 1110 + 0000 0001 = 0110 1111, P=1
-	1	Не используется, равен 1

Окончание табл. 2.1

C	0	***Признак переноса (заёма).*** Если при выполнении команды возник перенос из старшего разряда или заём в старший разряд, то C=1, иначе C=0. *1110 1110 + 1111 0000 = 1 1101 1110, C=1* *0110 1110 + 0001 0001 = 0111 1111, C=0*

Содержимое ***аккумулятора*** и регистра кода состояния ***F*** называют ***словом состояния программы и сокращенно обозначают PSW.*** Наличие флага переноса позволяет организовать на малоразрядном процессоре обработку данных любой длины путём последовательной обработки байтов операндов. Этот же регистр используется для создания замкнутого контура путём соединения самого старшего и самого младшего разрядов аккумулятора, что необходимо для циклического сдвига содержимого аккумулятора вправо или влево в соответствии с программой.

Буферные регистры АЛУ

Буферные регистры АЛУ используются для временного хранения одного слова данных. Необходимость в таком регистре возникает из-за отсутствия в АЛУ своего запоминающего устройства. Так как в состав АЛУ входят только комбинационные схемы, то при поступлении исходных данных на вход арифметико-логического устройства результирующие данные немедленно появляются на его выходе, как следствие выполнения операций данной программы. Получив данные с внутренней шины микропроцессора, АЛУ должно модифицировать их и поместить обработанные данные в аккумулятор, что неосуществимо без регистра временного хранения данных. Буферные регистры недоступны для программиста.

Если на вход буферного регистра временного хранения данных могут поступать данные только с внутренней шины данных, то в буфер аккумулятора данные могут поступать и с выхода аккумулятора. Буфер аккумулятора позволяет избежать одновременного подсоединения входа и выхода АЛУ к одной и той же точке.

РОН-СОЗУ, регистровые пары

РОН-СОЗУ, регистровые пары используются для хранения операндов, промежуточных и конечных результатов и т.д. Особое место занимает пара ***HL, называемая регистром данных/адреса,*** состоящая из двух 8-разрядных регистров, которые могут быть использованы как вместе, так и раздельно. Они обозначаются ***H*** и ***L*** соответственно старшему (***HIGH***) и младшему (***LOW***) байтам. При использовании этих двух регистров совместно мы обращаемся к паре ***HL***. Подобно аккумулятору, регистры ***H*** и ***L*** являются универсальными в том смысле, что их возможно инкрементировать, декрементировать, загружать данными и считывать данные. Регистровая пара ***HL*** может использоваться как адресный регистр и хранить адрес назначения в ходе размещения данных в памяти или являться адресом источника при загрузке аккумулятора. Таким образом, регистры ***H*** и ***L*** могут быть использованы для размещения данных в памяти и манипуляций с ними и как ***указатель адреса.***

Указатель стека

Указатель стека (верхушка стека) – 16-разрядный регистр-счетчик, содержимым которого ***всегда является адрес***. Назначение стека в том, чтобы ***сохранять текущее содержание*** всех регистров, если происходит прерывание основной программы. Образно стек можно представить в виде записи значений на отдельных листах бумаги и складывании их стопкой. Извлечение всегда происходит в обратном порядке, т.е. только с верхушки стека. Иными словами соблюдается принцип ***«последним вошел – первым вышел» (LIFO).*** В любой момент времени в стек можно включить дополнительную информацию, но при извлечении первой будет та, которая включена последней. С помощью стека можно организовать вложенные подпрограммы. При этом основная программа вызывает подпрограмму, которая может вызвать новую подпрограмму и т.д. При вызове первой подпрограммы адрес возврата к основной программе заносится в стек. При обращении ко второй подпрограмме адрес возврата к первой снова заносится в стек и т.д. По мере выполнения подпрограмм адреса возврата постепенно извлекаются из стека до тех пор, пока не произойдет возврат к основной программе (рис. 2.3).

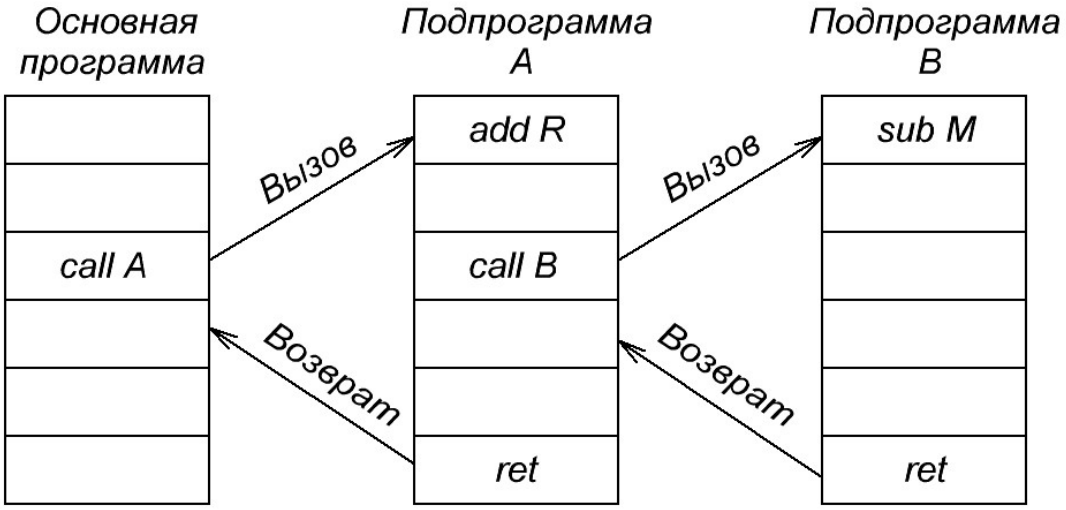

Рис. 2.3. Работа с подпрограммами

Однако основное назначение стека – это обслуживание прерываний. При прерывании содержимое ***PC*** и ***F***, а также все текущие данные из различных регистров последовательно вводятся в набор ячеек памяти (стек). Когда прерывание будет завершено, программа возвращается к верхушке стека и забирает данные с его поверхности, соблюдая принцип «последним вошел – первым вышел». Данные из стека выбираются до тех пор, пока не будет восстановлено положение до прерывания; в результате программа может вернуться к завершению прерванных действий.

Микропроцессор может иметь специальные регистры для сохранения данных, но чаще для этого используется специально отведенная последовательность ячеек памяти. В микропроцессоре ***i8080*** в качестве стека используется произвольная зона ОЗУ, а в микропроцессоре размещается только 16-разрядный регистр ***SP***. Содержимое этого регистра показывает адрес в памяти верхнего элемента списка данных. Область стека заполняется от старших адресов к младшим, т.е. можно сказать, что стек растет вверх. Каждое обращение к стеку для занесения в него слова данных сопровождается автоматическим уменьшением (декрементом) содержимого указателя стека на 1, а каждое обращение для извлечения информации из стека сопровождается инкрементом, т.е. добавлением к содержимому указателя стека (рис. 2.4).

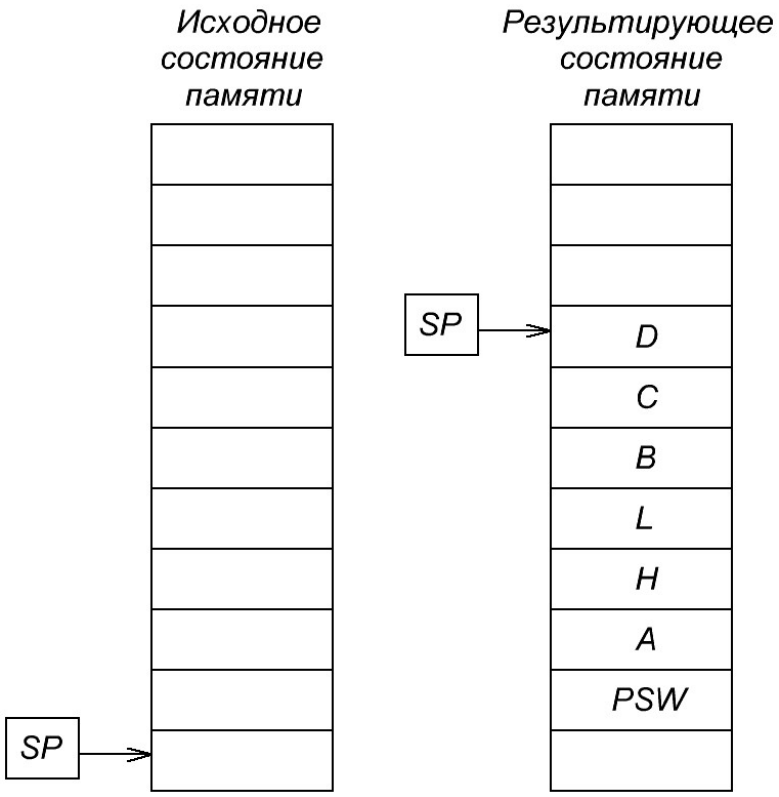

Рис. 2.4. Стек

В различных микропроцессорах и микроконтроллерах стек бывает двух типов: предекрементный (вначале декремент указателя стека, а потом запись в стек) и преинкрементный (вначале инкремент указателя стека, а потом запись в стек). Это важно помнить при инициализации стека. Инициализация стека осуществляется определением начального адреса, хранящегося в регистре указателя стека:

LXI SP, 220Ah; загрузка в ***(SP)←220Ah.***

Положение стека определяется программистом. Старшим адресом Указатель стека загружается старшим адресом, представляющим собой вершину стека (220Ah рис.2.5), что на единицу старше первой ячейки памяти стека 2209h (рис. 2.5). Запись данных осуществляется с помощью команды ***PUSH (поместить),*** или ***CALL (вызвать).*** Считывание данных из стека осуществляется при помощи команд ***POP (извлечь)*** или ***RETURN (возврат).***

Рис. 2.5. SP=220A

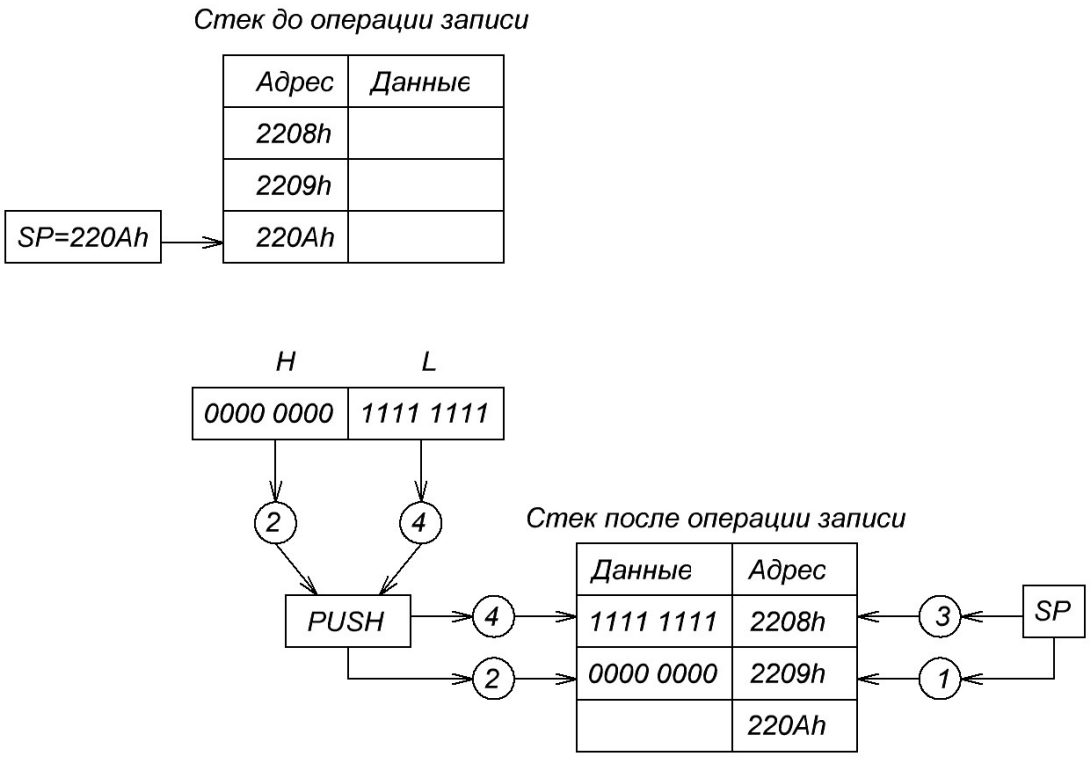

Рис. 2.6. Помещение в стек HL

Команда загрузки в стек (***PUSH***) идет следующим образом (на примере помещения в стек пары ***HL***, рис. 2.6):

1. Указатель стека декрементируется от 220Ah до 2209h.
2. ***SP*** указывает ячейку памяти 2209h по адресной шине и старший байт помещается в стек.
3. Указатель стека декрементируется от 2209h до 2208h.
4. ***SP*** указывает ячейку памяти 2208h по адресной шине и старший байт помещается в стек.

Схемы управления

Схемы управления в микропроцессорах играют важную роль, которая заключается в ***поддержании требуемой последовательности функционирования*** всех остальных его звеньев. По распоряжению схем управления очередная команда извлекается из регистра команд. В первую очередь выясняется, что необходимо сделать с данными, а далее генерируется последовательность действий для выполнения команды.

Работа схем управления обычно микропрограммируется. Одной из главных функций схем управления является ***декодирование команды***, которая находится в регистре команд, с помощью дешифратора команд, выдающего в результате ***сигналы, которые необходимы для выполнения команды.***

В рассматриваемом микропроцессоре изображены линии управления, соединяющие схемы управления со всеми узлами микропроцессора и внешними блоками: памяти и ввода-вывода.

Одной из важных входных линий управления является линия связи с генератором тактовых импульсов, синхронизирующим во времени работу микропроцессора. Она осуществляет соединение микропроцессора с внешними устройствами.

Схемы управления также выполняют другие специальные функции, такие как, управление процессами прерываний. Прерывание – запрос от других устройств ввода-вывода, который поступает на схемы управления. Прерывание связано с использованием внутренней ШД микропроцессора. Выбор, когда и в какой последовательности другие устройства могут использоваться внутренней ШД, осуществляется схемами управления.

Управляющее слово микропроцессора

Для нормального функционирования микропроцессорной системы недостаточно управляющих сигналов, генерируемых самим микропро-

цессором. МПС в каждом МЦ должна получать более полную информацию о состоянии МП. В условиях узкого интерфейса МП, когда внешних выводов для индикации внутреннего состояния (байта состояния) МП недостаточно, эта задача решается с использованием мультиплексирования ШД и представления внутреннего состояния МП во внешнем по отношению к МП регистре слове состояния (РСС).

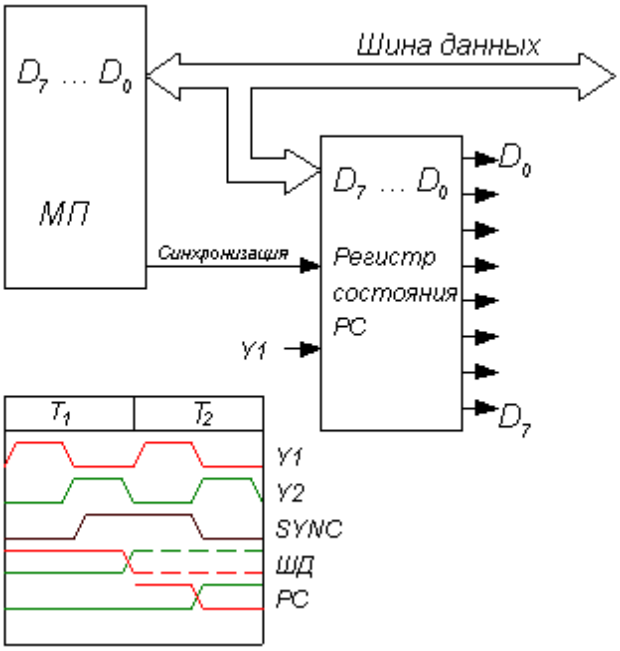

Рис. 2.7. Загрузка байта состояния в регистр состояния

Процессор в ***первом такте каждого МЦ*** генерирует на ШД байт состояния процессора (управляющее слово), которое содержит информацию о процессах, происходящих в МП. Сигнал синхронизации ***SYNC вырабатывается в начале каждого машинного цикла*** и используется в качестве сигнала, который идентифицирует информацию, представленную на ШД, как управляющее слово (байт состояния). Под воздействием сигнала синхронизации ***SYNC*** во внешний регистр управляющего слова загружается байт состояния. Сигналы, представленные на выходе 8-разрядного регистра состояния, ***используются как сигналы управления периферией МПС.***

Если при реализации любой команды первым машинным циклом в цикле команды является цикл выборки команды, то остальные машинные циклы в цикле команды следуют в произвольном порядке. Этот порядок определяется кодом команды. Всего ***i8080*** имеет 10 типов машинного цикла и соответственно 10 кодов управляющего слова, идентифицирующих эти циклы. Каждый разряд управляющего слова «заво-

дится» на соответствующие управляющие входы адаптеров или схем сопряжения с УВВ. Таким образом, в соответствии с текущим состоянием микропроцессора определяется их режим функционирования.

Следовательно, в условиях «узкого интерфейса» управление микропроцессорной системой осуществляется генерацией управляющих воздействий на двух уровнях:

1) на уровне управляющих сигналов (микроприказов по шине управления), ~***WR, DBIN*** и др. (появляются в каждом такте);
2) на уровне микроприказов путем генерации байта состояния в каждом машинном цикле.

Выходы регистра состояния и управляющие линии корпуса микропроцессора образуют шину управления микропроцессорной системы. Двенадцать линий шины системного управления позволяют МПС работать со сложным многофункциональным периферийным оборудованием. При этом использование временного мультиплексирования ШД для вывода во внешний регистр состояния сигналов управления микропроцессорной системы снижает производительность системы на 40%.

2.3. Тактирование и синхронизация микропроцессорной системы

Команды микропроцессора могут быть от 1 до 3-х байт и состоят из машинных циклов от M_1 до M_5. Машинный цикл состоит из машинных тактов от 3-х до 5 T_1-T_5, 1 такт равен 0.5 мкс, каждая команда выполняется от 2-х до 8 мкс. Первым и обязательным для всех команд является машинный цикл M_1, в котором производится выборка кода операции из памяти. В свою очередь, каждый машинный цикл для своего завершения требует от трех до 5 машинных тактов: T_1, T_2, T_3, T_4, T_5, два из которых T_1 и T_2 посылают адрес в память, такт T_3 приводит к считыванию команды или данных из памяти, такты T_4 и T_5 соответствуют выполнению команды (рис. 2.8). Временная диаграмма определяет основной цикл команды микропроцессора при наличии внешнего управляющего сигнала готовности (***READY***), информирующего о готовности периферийного оборудования к обмену с микропроцессором. В первом такте синхронизации T_1 микропроцессор выставляет на ША адрес следующей команды. После этого начинается цикл, в котором осуществляется выборка команды. В это же время на линии синхронизации ***SYNC*** появляется импульс, который:

1) идентифицирует информацию на шине данных как байт состояния и загружает его во внешний регистр состояния;

2) свидетельствует о начале машинного цикла ***выборка***.

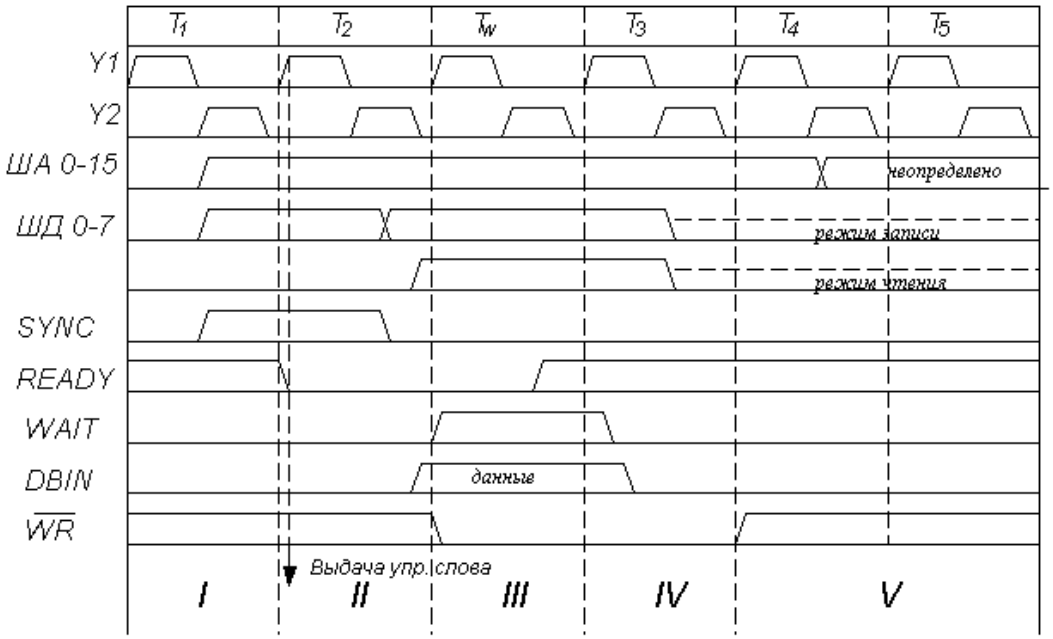

Рис. 2.8. Диаграммы работы микропроцессора

I – Формирование управляющего слова

II – Проверка управляющих сигналов, ***RAEDY, HLTA, HOLD, INT***

III – Остановка или ожидание сигнала готовности (не обязательно)

IV – Выборка данных, команды или запись данных

V – Выполнение команды, если необходимо (не обязательно)

По окончании сигнала синхронизации буферная схема ШД находится в режиме ввода, о чем свидетельствует единичный сигнал ***DBIN*** (чтение) на линии ШУ. В такте T_2, микропроцессором выполняется проверка готовности внешнего устройства к обмену, если адаптером внешнего устройства или памяти генерируется сигнал ***READY*** (готов). Микропроцессор переходит в состояние ожидания. В этом состоянии микропроцессор будет находиться до тех пор, пока на линии управления ***READY*** не появится сигнал высокого уровня, свидетельствующий о готовности памяти или периферийного оборудования к обмену. В состоянии T_w микропроцессор простаивает. В такте T_3 следуют чтение или запись слова в память, такты T_4 и T_5 отводятся для реализации операции, заданной кодом команды.

2.4. Функционирование микропроцессора

Рассмотрим действия, которые производит микропроцессор в ходе выполнения команд программы.

А. ***Цикл команды***

В микропроцессоре ***Intel 8080*** каждая команда выполняется в течение одного-пяти машинных циклов, ***называемыми процессорными циклами.***

Число машинных циклов команд определяется количеством ***обращений к внешним подсистемам*** (памяти, УВВ). Первым и обязательным машинным циклом всех команд является цикл выборки кода операции. Каждый машинный цикл состоит из 2 фаз. На этапе ***обязательной фазы***, которая называется ***фазой адресации***, микропроцессор, используя шину адреса, идентифицирует внешний регистр и обменивается с ним информацией. ***Необязательная фаза***, которая называется ***фазой выполнения***, необходима для дешифрирования кода операции команды и выполнения команды.

МЦ состоит из 3-5 машинных тактов, ***называемых функциональными состояниями или состояниями МП.*** Такты определяются как временной интервал между двумя нарастающими фронтами сигналов синхронизации ***Y1***, а их число в МЦ определяет содержание выполняемой команды. Циклы команд варьируются от 4 до 18 тактов. Если частота синхронизации равна 1,25 МГц, то циклы команд составляют от 3.2 до 14.4 мкс.

Первые три такта всех машинных циклов являются унифицированными и образуют фазу адресации. В T_1 осуществляется адресация внешнего регистра-источника или регистра-получателя. По переднему фронту сигнала ***Y2*** микропроцессор выдает сигналы адреса на ША. ***Они сохраняются стабильными до поступления переднего фронта Y2 такта, следующего за тактом T_3 текущего МЦ. Источником адреса могут быть следующие регистры: PC, SP, BC, DE, HL, WZ*** – в командах ВВ и передачи управления.

Такт T_2, всегда следующий за T_1, предназначен для проверки необходимости реакции на управляющие сигналы, которые влияют на работу МП. По переднему фронту сигнала ***Y2*** проверяются уровни внешних сигналов ***запроса прямого доступ к памяти HOLD***, ***готовности READY***, а также внутреннего сигнала подтверждения ***останова HLTA***. Кроме того, ***в*** такте ***T_2 последнего МЦ каждой команды*** проверяется уровень сигнала ***запроса прерывания INT***. В настоящем пособии будем считать, что указанные сигналы имеют уровни, не изменяющие нормального течения машинного цикла (***READY***=1, ***HOLD***=0, ***HLTA***=0, ***INT***=0).

Помимо проверки значений управляющих сигналов, ***в зависимости от функций*** текущего машинного цикла, в ***T_2*** производятся следующие действия:

1. Если текущий МЦ связан с обращением к памяти, то осуществляется инкремент ***PC***. ***PC***=***PC***+1;

2. Если в текущем МЦ осуществляется считывание (ввод) данных в микропроцессор, то по переднему фронту сигнала ***Y2*** будет сгенерирован сигнал считывания ***DBIN***;

3. Если в текущем МЦ осуществляется запись (вывод) данных из микропроцессора, то по переднему фронту сигнала ***Y2*** на ШД формируются сигналы выводимого слова.

Такт T_3 предназначен для обмена информацией между внутренним регистром микропроцессора и адресованным в T_1 внешним регистром. Если текущий машинный цикл связан с вводом данных в микропроцессор, то сигналы данных от внешнего регистра, которые коммутируются сигналом ***DBIN***, должны быть стабилизированы до заднего фронта сигнала ***Y1*** и оставаться стабильными более 130 нс после переднего фронта сигнала ***Y2*** такта T_3. Если в текущем МЦ микропроцессор выводит данные, то по фронту сигнала ***Y1*** генерируется L-активный сигнал ~***WR***. По фронту сигнала ***Y2*** снимается сигнал ***DBIN***, а сигнал ~***WR*** снимается по фронту следующего сигнала ***Y1***.

В первом МЦ всех команд, в тактах T_4 и T_5, производится дешифрирование кода операции, все необходимые внутренние передачи и преобразования данных.

Рассмотренный принцип формирования МЦ дает возможность определить число машинных циклов в каждой команде.

Однобайтные команды пересылки ***mov r2, r1*** выполняются за один МЦ: выборка кода операции (фаза адресации) и передача между внутренними регистрами (фаза выполнения). Однобайтные команды ***mov r, M*** или ***mov M r*** выполняется за два МЦ: выборка кода операции (M_1) и передача между внутренним регистром и ячейкой памяти адресуемой Н-парой (M_2). Двухбайтная команда ***ORI*** состоит из трех МЦ: выборка кода операции (М1), считывание второго байта операнда (M_2) и выполнения операции (M_3). Трехбайтная команда ***STA*** выполняется за 4 МЦ: три МЦ выборки команды и один МЦ, в котором происходит запись содержимого аккумулятора (***A)*** в ячейке памяти. Команда ***XTHL*** является однобайтной, но самой длительной, и выполняется за 5 МЦ: в первом машинном цикле производится выборка кода операции, два МЦ используются для передачи содержимого двух верхних ячеек стека в регистры ***W, Z*** и два МЦ предназначены для передачи в указанные ячейки стека содержимого регистров (***H, L***) с последующей передачей содержимого (***W, Z***) в (***H, L***).

Число МЦ восьми команд условных вызовов (***CC, CNC, CZ*** и др.) зависят от значения условия, которое проверяется в них, т.е. состояния соот-

ветствующего триггера флажка. Если условие, ***анализируемое в T_4 и T_5 МЦ1, не удовлетворяется***, то эти команды будут выполняться в течение циклов M_1, M_2, M_3 выборки 3-хбайтной команды. Если условие удовлетворяется, то условные вызовы будут выполняться в течение 5 МЦ. В двух дополнительных циклах (M_4 и M_5) содержимое программного счетчика загружается в стек.

В. ***Байт состояния***

Несколько упрощенно можно считать, что микропроцессорная система на базе МП ***Intel 8080*** состоит из пяти внешних подсистем:

1) подсистема памяти;
2) подсистема стека;
3) подсистема ввода;
4) подсистема вывода;
5) подсистема прерываний.

Все подсистемы подключаются к ША и ШД параллельно, что позволяет производить дешифровку адреса внешнего регистра, который формируется в T_1, одновременно всеми подсистемами.

Однако в любом МЦ может происходить обмен информацией микропроцессора только с одной подсистемой. Это означает, что для однозначной идентификации внешнего регистра адресной информации на ША недостаточно и ША необходимо дополнять «старшими байтами», которые будут адресовать подсистему. Для отладки микропроцессорной системы обычно требуется информация о текущем состоянии микропроцессора. Такую дополнительную информацию содержит байт состояния, который выдается на ШД в T_1 всех МЦ по переднему фронту сигнала ***Y2***. Выходной Н-активный сигнал синхронизации ***SYNC***, также генерируемый в T_1 по переднему фронту ***Y2***, служит импульсным сигналом, идентифицирующим на линии ШД байта состояния. Стабильные уровни сигналов состояния и высокий уровень сигнала ***SYNC*** сохраняются до переднего фронта сигнала ***Y2*** в такте T_2. Допустимые задержки сигналов состояния и сигнала SYNC выбраны так, чтобы спадающий фронт сигнала ***SYNC*** приходился на стабильные уровни сигналов состояния.

Сигнал ***SYNC*** можно использовать непосредственно для загрузки байта состояния с ШД во внешний регистр-защелку, где он сохраняется до следующего МЦ.

Отдельные биты байта состояния имеют следующее содержание:

D_0 (***INTA***) — бит подтверждение прерывания. Показывает ответную реакцию микропроцессора на запрос на прерывание, который был принят им от периферийного устройства. Формируется только в МЦ M_1 и используется для ввода команды рестарта ***RST***.

D$_1$ (***WR#***) — бит запись/вывод. Определяет, является ли микропроцессор получателем информации (***WR#***= 1) или источником информации (***WR#***=0) в текущем МЦ.

D$_2$ (***STACK***) — бит стек. Определяет указатель стека в качестве источника адреса (на ША в текущем МЦ находится содержимое указателя стека).

D$_3$ (***HLTA***) — бит подтверждения останова. Формируется в МЦ ***M***2 команды останова ***HLT*** и указывает на прекращение выполнения МП программы.

D$_4$ (***OUT***) — бит вывода. Формируется в МЦ ***M***$_3$ команды ***OUT*** и идентифицирует выбор подсистемы вывода. На ША находится адрес порта вывода.

D$_5$ (***M***1) — бит первого машинного цикла команды. Формируется в цикле ***M***$_1$ всех команд для определения выборки кода операции из памяти программ.

D$_6$ (***INP***) — бит ввода. Формируется в МЦ ***M***$_3$ команды ***IN*** и идентифицирует выбор подсистемы ввода. Адрес порта ввода находится на ША.

D$_7$ (***MEMR***) — бит считывания из памяти. Определяет выполнение считывания из памяти в текущем МЦ.

Основной функцией битов состояния является генерирование управляющих сигналов для внешних подсистем. Поэтому определение их диктуется простотой интерфейса, а не однозначной идентификацией машинного цикла. В табл. 2.2 приведены биты байта состояния для существующих типов машинных циклов.

Таблица 2.2

*Типы машинных циклов **i8080***

	D7	D6	D5	D4	D3	D2	D1	D0	
	MEMR	IN	M1	OUT	HLTA	STACK	~WO	INTA	
1	1	0	1	0	0	0	1	0	Выборка команды
2	1	0	0	0	0	0	1	0	Чтение из памяти
3	0	0	0	0	0	0	0	0	Запись в память
4	1	0	0	0	0	1	1	0	Чтение из стека
5	0	0	0	0	0	1	0	0	Запись в стек
6	0	1	0	0	0	0	1	0	Чтение из УВВ
7	0	0	0	1	0	0	0	0	Запись в УВВ
8	0	0	1	0	0	0	1	1	Разрешение прерывания
9	1	0	0	0	1	0	0	0	Останов
10	0	0	1	0	1	0	1	1	Разрешение останова

На рис. 2.9 приведена схема формирования основных управляющих сигналов, используемых в микропроцессорных системах на базе микропроцессора ***Intel 8080***. Из рисунка видно, что активным уровнем всех приведенных управляющих сигналов являются низкий уровень. Это значительно упрощает интерфейс с внешними подсистемами, имеющими обычно входы выбора устройства ***DS*** или выбора интегральной схемы ***CS*** с низким активным уровнем.

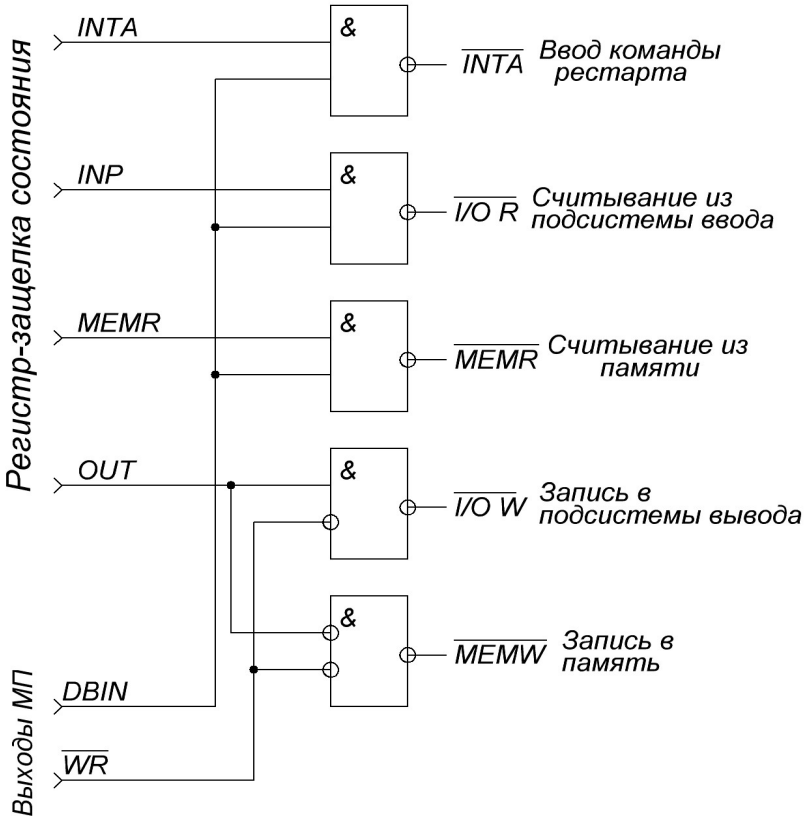

Рис. 2.9. Схема формирования управляющих сигналов

С. ***Особые случаи циклов команд***

Во всех машинных циклах в такте T_2 микропроцессором осуществляется проверка значения входных сигналов готовности ***READY***, запроса прямого доступа к памяти ***HOLD***, а также значение внутреннего сигнала подтверждения останова ***HLTA***. Уровень сигнала прерывания проверяется в последнем МЦ всех команд. Подробнее рассмотрим реакции МП на указанные сигналы.

Реакция микропроцессора на сигнал READY

Внешняя подсистема, например память, может приостановить действия микропроцессора установкой ***L***-уровня на линии ***READY,*** «растя-

нув» тем самым машинный цикл на целое число периодов синхронизации, в случае если быстродействия этой подсистемы недостаточно для синхронного обмена данными с микропроцессором. В случае поступления сигнала ***READY,*** микропроцессор не переходит к такту T_3 и вводит состояние ожидания T_w (Рис. 2.10). В состояниях ожидания адрес внешнего регистра, который был выдан в такте T_1, а также сигнал ***DBIN***, если текущий МЦ M_i связан передачей данных в МП, сохраняются на ША. Для окончания (подтверждения) перехода в состояние T_w микропроцессор по переднему фронту сигнала ***Y1*** формирует высокий уровень на линии ***WAIT***.

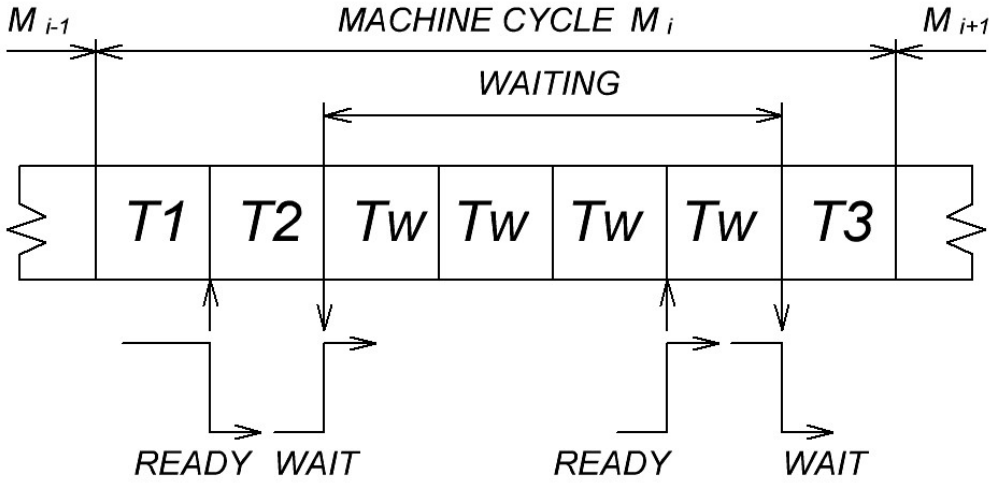

Рис. 2.10. Введение состояний ожидания

Действие ***L***-уровня на входе ***READY*** определяет продолжительность состояний ожидания. Внешней подсистеме, когда она готова к обмену данными, необходимо установить на линии ***READY*** высокий уровень. Высокий уровень должен быть стабилизирован до спада сигнала ***Y2*** для правильной реакции микропроцессора на готовность. После этого микропроцессором вводится такт T_3 и по переднему фронту ***Y1*** снимается сигнал ***WAIT***.

Реакция микропроцессора на сигнал HOLD

Обмен данными между периферийными устройствами с высокой скоростью передачи данных организуется в режиме прямого доступа к памяти (ПДП). В случае инициирования запроса на ПДП периферийным устройством (точнее, контроллером ПДП), микропроцессор прекращает выполнение программы и переводит внутренние буферы ША и

ШД в высокоимпедансное состояние. Контролер ПДП начинает управлять шинами, организуя обмен данными между периферийным устройством и памятью микропроцессорной системы.

Для инициирования ПДП необходимо установить высокий уровень на входной линии запроса прямого доступа к памяти ***HOLD***. В пакетном режиме, который является наиболее простым, высокий уровень **HOLD** сохраняется до окончания передачи блока данных, но прямой доступ к памяти может осуществляться и для передачи отдельных байтов. Микропроцессор будет реагировать на высокий уровень ***HOLD*** в текущем МЦ M_i, только если этот уровень стабилизируется до переднего фронта сигнала ***Y2*** (Рис. 2.11). Рекомендуется синхронизация между сигналами ***HOLD*** и ***Y1***.

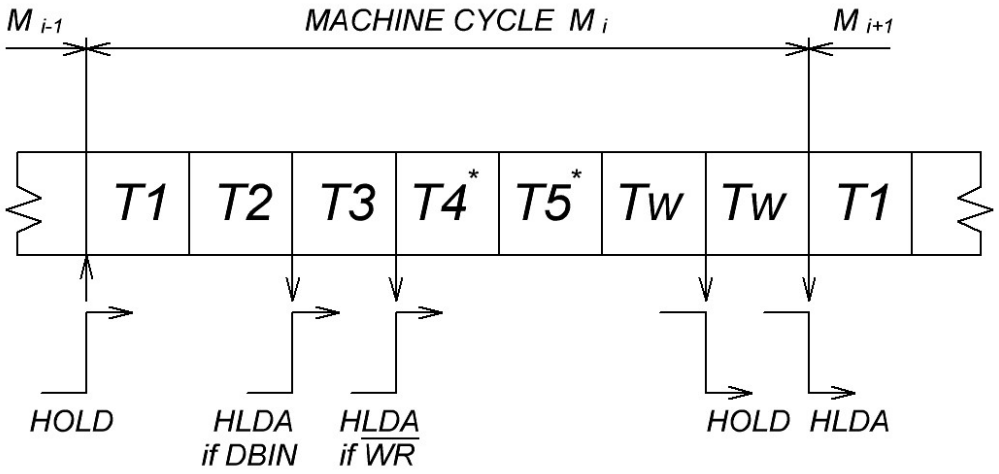

Рис. 2.11. Реакция микропроцессора на сигнал HOLD

В такте ***T***1 микропроцессором адресуется внешний регистр, обмен с которым будет производиться в такте ***T***3. В случае если режим прямого доступа к памяти будет разрешен до такта ***T***3, то в дальнейшем необходимо будет снова адресовать внешний регистр. При поступлении запроса микропроцессор обязательно заканчивает обмен информации с внешним регистром или памятью и далее переходит в режим прямого доступа к памяти и вводит специальные такты ожидания T_{WH}. При такой логике реакции микропроцессора сигнал готовности ***READY*** обладает более высоким приоритетом по сравнению с сигналом запроса прямого доступа к памяти ***HOLD***.

После такта T_3 текущего МЦ M_i следуют либо необязательные такты T_4 и T_5, которые отводятся на внутреннее преобразование данных (в частности, выполнение команд), либо первый такт T_1 следующего ма-

шинного цикла. Микропроцессор по нарастающему фронту ***Y1*** формирует ***H***-уровень ответного сигнала ***HLDA***, который является сигналом подтверждения запроса режима прямого доступа к памяти, а по фронту сигнала ***Y2*** микропроцессор переводит буферные регистры шины адрес и шины данных в 3-е состояние. Если текущий машинный цикл связан со считыванием данных в микропроцессор, то установка сигнала ***HLDA*** и перевод шины адреса и шины данных в высокоимпедансное состояние будут осуществляться в такте T_3, а если текущий машинный цикл связан с записью данных из микропроцессора, то установка сигнала ***HLDA*** и перевод шины адреса и шины данных в высокоимпедансное состояние будут осуществляться в такте, следующем за тактом T_3; при этом такты T_4 и T_5 будут совмещаться с режимом ПДП. После окончания действия режима ПДП микропроцессор введет такт T_1 следующего машинного цикла.

Когда закончится передача данных между периферийным устройством и памятью микропроцессорной системы, на линии ***HOLD*** устанавливается низкий уровень, и микропроцессор выходит из режима ПДП, т.е. заканчиваются такты ожидания T_{WH}.

Реакция микропроцессора на команду останова HLT

В первом машинном цикле M_1, который состоит из 4 тактов T_1 — T_4, осуществляется выборка и дешифрирование команды останова, а в следующем машинном цикле M_2 производится выполнение данной команды. В первом такте T_1 второго машинного цикла M_2 микропроцессор выставляет на адресную шину содержимое ***PC***, а на шине данных выставляется байт состояния, в котором установлен бит подтверждения команды останова ***HLTA***. Во втором такте T_2 по фронту тактового сигнала ***Y2*** буферные регистры шины адреса и шины данных переводятся в высокоимпедансное состояние, и по фронту тактового сигнала ***Y1*** выставляется высокий уровень на ножке ***WAIT***. Тем самым микропроцессором прекращается процесс выполнения программы.

Существует два способа вывода микропроцессора из состояния останова:

1. Посредствам подачи сигнала сброса высокого уровня ***RESET.*** Продолжительность данного сигнала должна составлять ***не менее 3 периодов синхронизации***. При подаче сигнала сброса микропроцессор по нарастающему фронту сигнала ***Y1*** генерирует внутренний сигнал сброса, по которому в программный счетчик загружаются нули (т.е. микропроцессор начинает свою работу с нулевого адреса) и устройство

управления формирует первый такт первого машинного цикла – цикла выборки кода операции;

2. Посредствам поступления запроса на прерывание ***INT***. Микропроцессор будет реагировать на данный сигнал в том случае, если произведена внутренняя установка разрешения прерываний (***INTE***=1). Таким образом, для выхода микропроцессора из состояния останова посредствам прерывания необходимо разрешить прерывания до команды останова. При поступлении запроса на прерывание микропроцессор вводит цикл M_1 выборки команды рестарта.

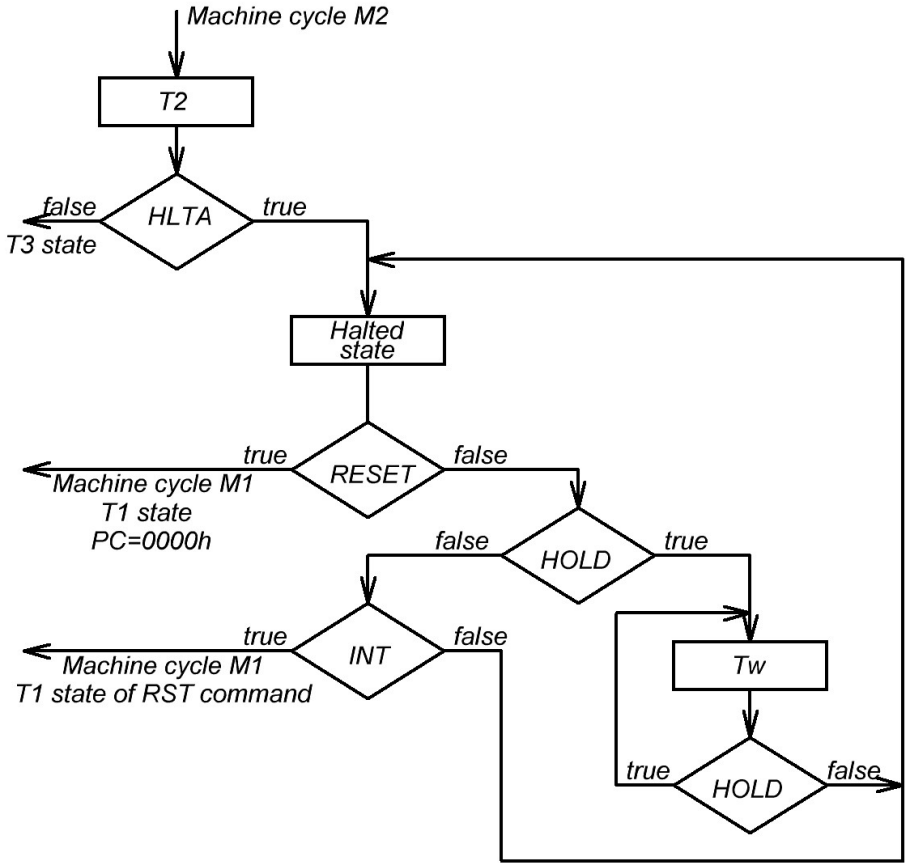

Рис. 2.12. Введение состояния останова

Находясь в состоянии останова микропроцессор, в обычном режиме реагирует на сигнал запроса режима ПДП ***HOLD***. Работа микропроцессора в состоянии останова показана на рис. 2.12.

Реакция микропроцессора на сигнал INT

При работе микропроцессора на входную ножку запроса на прерывание ***INT*** подключаются медленные периферийные устройства, кото-

рые будут участвовать в процессе обмена информации с микропроцессором. При этом выходной сигнал разрешения прерываний ***INTE*** будет «говорить» периферийным устройствам, будет ли микропроцессор реагировать на запросы (***INTE*** = 1) или нет (***INTE***=0).

При поступлении запроса на прерывание (***INT***=1, ***INTE***=1) микропроцессор выполняет следующие действия: прерывает выполнение текущей программы; сохраняет содержимое программного счетчика в стек; передает управление программе обработчика прерываний периферийного устройства, от которого произошел запрос; далее восстанавливает исходное состояние программы и продолжает ее выполнение.

При обработке прерываний в микропроцессорных системах используется приоритетная векторная система прерываний. Сигнал запроса на прерывание идентифицируется высоким уровнем на ножке ***INT***. Данный запрос может поступить в любой момент машинного цикла команды. В последнем такте последнего машинного цикла всех команд (кроме команды разрешения прерываний) при условии, что сигнал ***INT*** =1 и сигнал ***INTE***=1 по фронту тактового сигнала ***Y2***, внутренний триггер прерываний устанавливается в 1. После этого микропроцессор вводит машинный цикл «ПРЕРЫВАНИЕ».

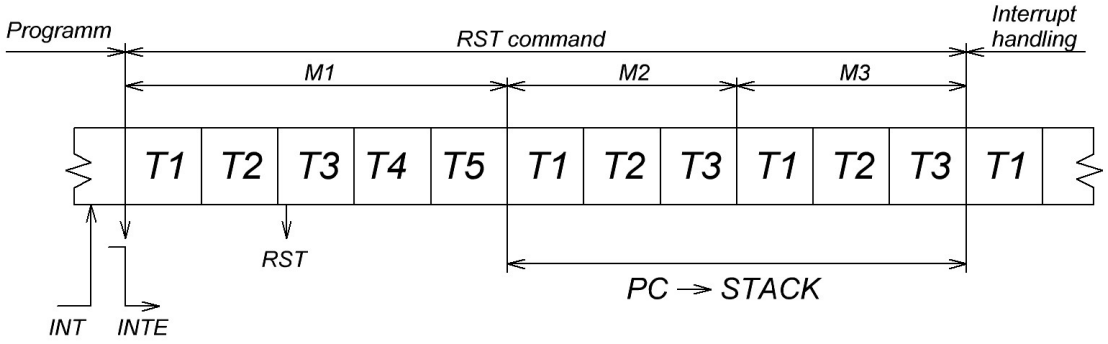

Рис. 2.13. Выполнение команды рестарта

В первом такте машинного цикла «ПРЕРЫВАНИЕ» микропроцессор выставляет на шине адреса содержимое программного счетчика, при этом на шину данных выставляется байт состояния (***M***$_1$ = l, ***INTA***=1, ***MEMR***= 0). В этом же такте по фронту тактового сигнала $Ф_2$ на выходе ***INTE*** формируется низкий уровень (рис. 2.13.). Таким образом, микропроцессор не будет отвечать на другие запросы на прерывание до тех пор, пока снова не будут разрешены прерывания.

Во втором такте ***T***$_2$ формируется сигнал ***DBIN***, по которому в цикле «ПРЕРЫВАНИЕ» происходит считывание кода операции из подсистемы прерываний. При этом не происходит инкремента программного

счетчика, и он продолжает хранить адрес команды, которая должна была выполняться.

Для того чтобы загрузить содержимое программного счетчика в стек подсистема прерываний формирует команду вызова. Для ускорения реакции микропроцессора на прерывание в систему команд микропроцессора включена 1-байтная команда, которая называется рестарт (***RESTART***, или ***RST***) и имеет код операции 11***AAA***111.

Значение ***AAA*** называется вектором прерывания и формируется периферийным устройством, которое сгенерировало запрос на прерывание. В такте T_3 данного машинного цикла команда рестарта загружается в регистр команд микропроцессора, а следующие такты T_4 и T_5 необходимы на дешифрирование команды рестарта.

По команде рестарта ***RST*** инициируются выполнения следующих действий:

1. Содержимое программного счетчика загружается в стек. Для этого микропроцессор вводит специальные машинные циклы M_2 и M_3, которые называются «ЗАПИСЬ В СТЕК».
2. В программный счетчик загружается код 00000000 00***AAA***000, который переадресует микропроцессор к обработчику прерывания.

После команды рестарта микропроцессор вводит цикл «ВЫБОРКА КОДА ОПЕРАЦИИ» первой команды подпрограммы обработчика прерываний. Последней командой в обработчике прерываний является команда возврата ***RET***, которая загружает из стека в ***PC*** адрес возврата.

2.5. Контрольные вопросы

1. В чем заключается основная функция АЛУ:
   а) выполнять операции сложения;
   б) служить источником сигналов для аккумулятора;
   в) изменять данные посредством арифметических или логических операций;
   г) выполнять все перечисленные функции.
2. Для большинства логических и арифметических операций, которые выполняются микропроцессором, необходимы два «участка» операции – два операнда. Один из них расположен в регистре или памяти. Укажите место, где находится другой операнд:
   а) в регистре команд;
   б) в аккумуляторе;
   в) в регистре адреса памяти;
   г) в счетчике команд.
3. Диапазон адресов 16-разрядного микропроцессора равен 2^{16}=65 536. Чему должно равняться число разрядов счетчика команд этого микропроцессора:
   а) 4;
   б) 4;
   в) 16;
   г)32.
4. Регистром какого типа является счетчик команд:
   а) специального назначения:
   б) особого назначения;
   в) памяти;
   г) всех перечисленных типов.
5. На какую команду программы указывает счетчик команд после извлечения из памяти очередной команды:
   а) последнюю выполненную;
   б) следующую команду, подлежащую выполнению;
   в) текущую выполняемую;
6. Произведите сложение приводимых ниже 8-разрядные двоичных чисел, указав состояние единичных разрядов регистра состояния: флаг переноса (C), флаг нулевого результата (Z) и флаг отрицательного результата (S).
   а) 00001111+11110000;
   б) 01010100+11001100;
   в) 00111011+11000101;

г) 00000001+01111111;
д) 11111111+11111111;
е) 00001111+00010000;
ж) 00000001+11111110;
з) 11000000+10000001;

7. Дайте описание программы установки разряда нулевого результата в единичное состояние после троекратного увеличения на 1 содержимого любого 8-разрядного регистра. Укажите чему равно начальное значение данного регистра.
8. Необходимо прибавить к младшему байту счетчика команд слово памяти. Полученный адрес со смещением необходимо поместить в регистр адреса памяти. Опишите последовательность действий микропроцессора для выполнения данных операций.
9. Какую роль могут играть регистры B, C и D:
   а) счетчика команд;
   б) регистров общего назначения;
   в) регистра адреса памяти;
   г) регистрированной пары DC.
10. Шина микропроцессора служит для двусторонней связи. Это означает:
    а) все данные перемещаются в двух противоположных направлениях;
    б) данные могут перемещаться в направлении, необходимом для завершения передачи;
    в) каждый функциональный узел микропроцессора имеет два входных порта;
    г) каждый узел должен иметь входной и выходной порты.
11. На что микропроцессору указывает код операции:
    а) где выполнять операцию;
    б) что делать;
    в) что делать и где выполнять.
12. Микропроцессор сопоставляет каждой команде единственную комбинацию двоичных цифр. Верно ли утверждение, что каждая команда имеет лишь одно мнемоническое обозначение? Ответ необходимо обосновать.
13. Время выполнения какой из ниже перечисленных способов адресации памяти является наименьшим:
    а) прямой;
    б) неявной;
    в) косвенно-регистровой;
    г) непосредственной.

14. Второй байт команды с непосредственной адресацией представляет собой:
    а) адрес области памяти, принадлежащей диапазону от 0_{10} до 255_{10};
    б) все перечисленные;
    в) 8-битовые данные;
    г) байт, легко доступный многим командам.
15. 16-разрядный микропроцессор использует 22-битовое адресное поле, которое обеспечивает доступ к 4 194 304 ячейкам памяти. К любой области памяти позволяет обращаться команда прямой адресации, первое слово которой занято кодом операции. Сколько байтов будет использоваться в случае формирования самой длинной подобной команды? Ответ необходимо обосновать.
16. Косвенную адресацию называют регистровой потому, что регистр, на который указывает команда:
    а) содержит адрес данных;
    б) содержит обрабатываемые данные;
    в) используется для областей памяти с адресами от 0_{10} до 255_{10};
    г) проверяет косвенным образом эти данные.
17. Перечислите все виды адресации памяти в порядке возрастания скорости выполнения.
18. Если микропроцессор имеет 16-разрядную шину адреса, то он может адресоваться:
    а) к 65 536 8-битовым словам памяти;
    б) к 65 536 словам памяти;
    в) к 32 768 1-байтовым словам памяти;
    г) к 16 8-битовым словам памяти.
19. Информация какого рода передается по линиям шины микроконтроллера:
    а) сигналы управления и питание;
    б) данные;
    в) адрес памяти;
    г) все перечисленные виды информации.
20. У микропроцессора имеются 4 порта ввода-вывода, содержащих восемь линий данных. Эти порты называются:
    а) порты последовательного вывода;
    б) порты параллельного ввода-вывода;
    в) порты вывода адресов;
    г) порты последовательного вывода.

ГЛАВА 3. ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1

Основы работы с лабораторным макетом микропроцессора

Теоретические сведения

Изучение процессора ***Intel 8080*** осуществляется с использованием отладочного комплекта Cyclone II FPGA компании Altera (рис. 3.1). Симулятор работы процессора реализован на базе микросхемы программируемой логической интегральной схемы (ПЛИС).

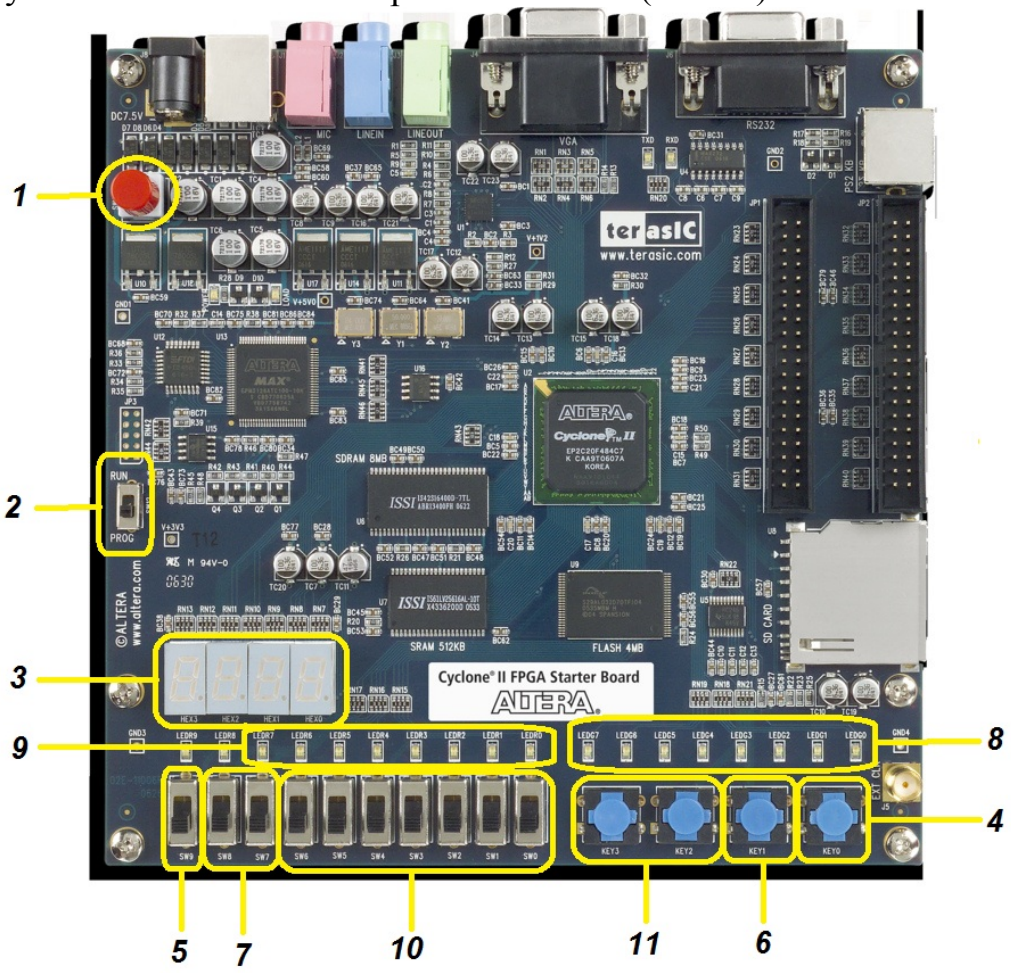

Рис. 3.1. Отладочный комплект Cyclone II FPGA

Включение устройства осуществляется посредствам кнопки ***1*** (рис. 3.1). При этом питание лабораторного модуля обеспечивает USB-порт персонального компьютера. Для работы модуля в режиме эмуля-

тора микропроцессора ***Intel 8080*** необходимо, чтобы переключатель ***2*** находился в положении ***RUN*** (верхнее положение).

Под цифрой ***3*** на рис. 1 расположены четыре семисегментных индикатора ***HEX0-HEX3***, которые отображают содержимое программного счетчика ***PC*** микропроцессора. При работе процессора после нажатия кнопки ***RESET*** (кнопка ***4 – KEY0***) в программный счетчик записывается значение ***0000h***.

Тумблер ***5 (SW9)*** переключает микропроцессор в режимы ***СТАРТ-СТОП***. В положении ***СТАРТ*** (нижнее положение тумблера) микропроцессор выполняет программу, заданную пользователем в автоматическом режиме. В верхнем положении тумблера (***СТОП***) микропроцессор приостанавливает выполнение текущей программы и переходит в пошаговый режим работы. В данном режиме микропроцессор выполняет команды только по нажатию кнопки ***6 (KEY1)***.

Таблица 3.1

Частоты работы микропроцессора

Положение тумблеров		Частота
SW8	***SW7***	
↓	↓	25 Гц
↓	↑	2.5 кГц
↑	↓	250 кГц
↑	↑	25 МГц

Тумблеры ***7*** (***SW7-SW8***) осуществляют переключение частоты работы микропроцессора. В табл. 3.1 представлены частоты работы микропроцессора в зависимости от положения тумблеров ***SW7-SW8.***

На светодиодах ***8*** (***LEDG0-LEDG7***) отображаются данные поступающие на внутреннюю шину данных микропроцессора.

Светодиоды ***9*** (***LEDR0-LEDR7***) подключены к порту ввода/вывода микропроцессора и отображают выводимые данные. Адрес порта вывода – 10h, вывод в порт (т.е. на светодиоды) осуществляется посредствам команды ***OUT***.

Переключатели ***10*** (***SW0 – SW6***) подключены к порту ввода микропроцессора (адрес порта 12h). При нижнем положении переключателя на порт поступает логический 0, а при верхнем положении – логическая 1. Считывание данных с порта осуществляется посредствам команды ***IN***. Кнопки ***11*** (***KEY2 – KEY3***) подключены к младшим разрядам порта ввода с адресом 11h. При нажатии кнопки на порт ввода поступает логическая 1.

Окно программы эмулятора микропроцессора ***Intel 8080*** представлено на рис. 3.2. При подключении лабораторного модуля к персональному компьютеру и открытии программы эмулятора необходимо подключить модель к программе посредствам кнопки «***Открыть устройство***». После подключения устройство будут активированы кнопки «***Очистить***», «***Открыть файл***» и «***Сохранить файл***».

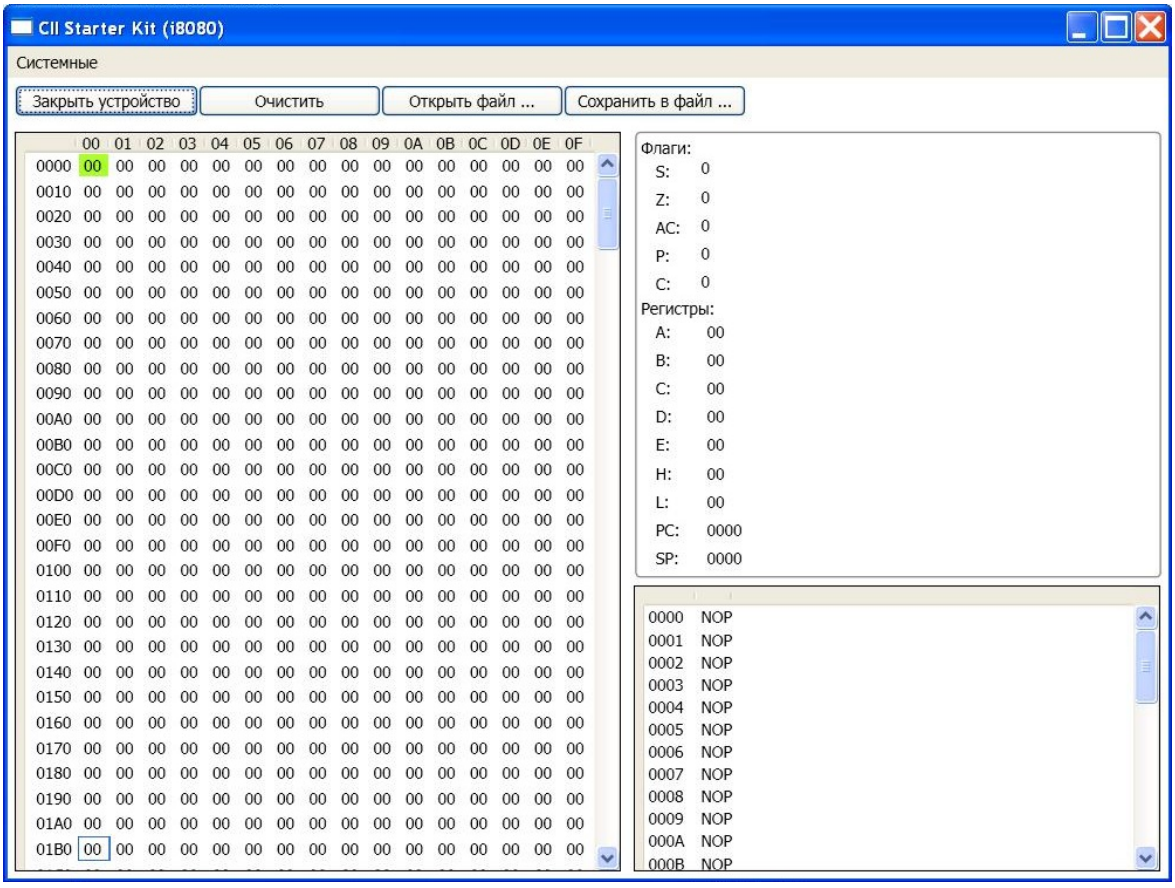

Рис. 3.2. Окно программы эмулятора микропроцессора ***Intel 8080***

В левом окне программы эмулятора показано адресное пространство памяти. Выполняемая программа записывается в виде шестнадцатеричных кодов операций и операндов в ячейки памяти память. В ходе занесения кодов программы в правом нижнем окне можно наблюдать мнемоническое обозначение программы.

В окне «***Флаги***» показаны биты флагов регистра состояния: ***C*** – перенос; ***AC*** – вспомогательный перенос; ***S*** – знак; ***Z*** – ноль; ***P*** – четность. Доступ к флагам возможен только программным путем.

В окне «***Регистры***» показано содержимое следующих регистров:

- 16-разрядный счетчик команд (***PC***);
- 16-разрядный регистр указатель стека (***SP***);

• 8-разрядный регистр-накопитель (***A***);

• шесть 8 –разрядных регистров общего назначения: ***B,C,D,E,H,L***.

Изменение содержимого данных регистров возможно, как посредствам команд микропроцессора, так и в ручном режиме, путем записи в них необходимых данных с клавиатуры в шестнадцатеричном коде.

Пример реализации программы записи данных в регистры ***A*** и ***B*** представлен на рис. 3.3.

MVI A, 0AAh
MVI B, 11h

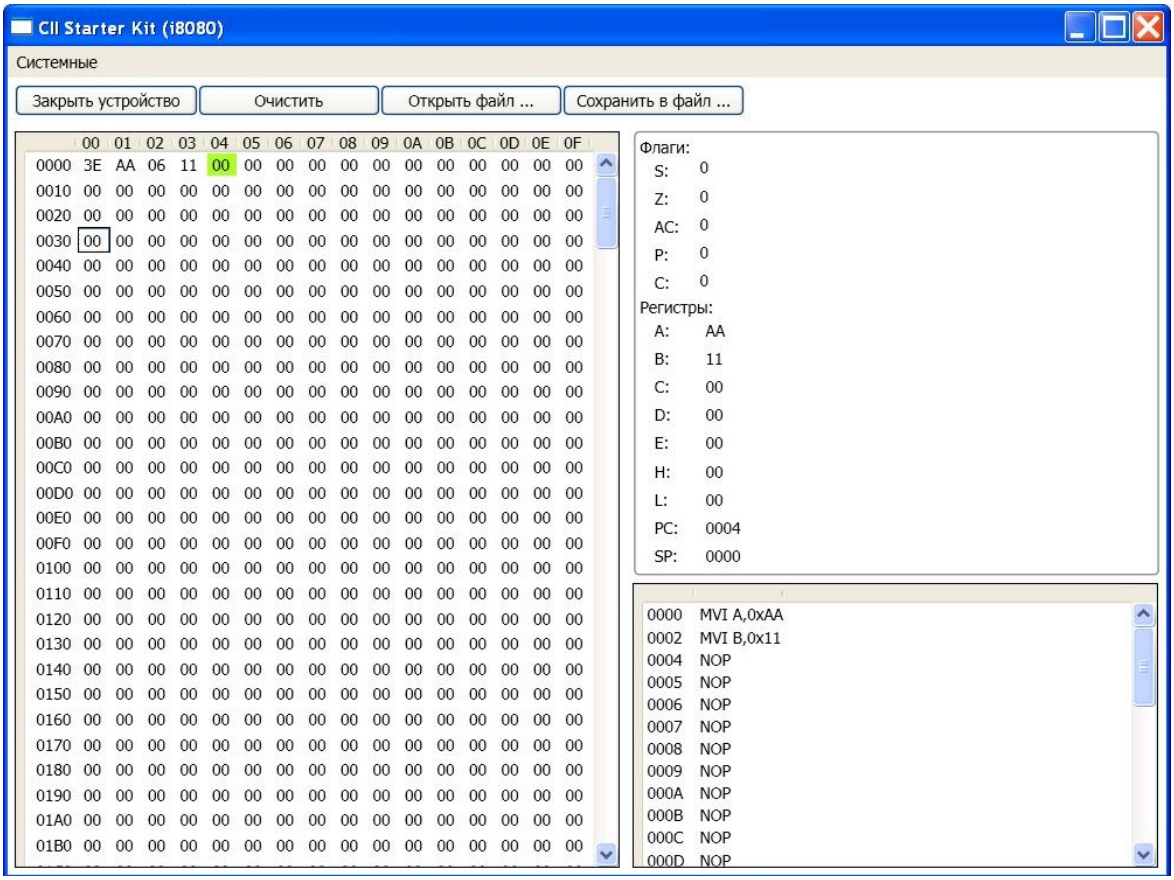

Рис. 3.3. Окно программы эмулятора микропроцессора ***Intel 8080***

Программа работы

1. Подключить отладочный комплект Cyclone II FPGA к компьютеру и включить его питание.

2. Запустить программу эмулятора микропроцессора ***Intel 8080***:
D:\CII_Project\CII_Project v1.1\CII Starter Kit.exe

3. Подключить отладочный макет к эмулятору посредствам кнопки «***Открыть устройство***».

4. Записать в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 3.2.

Таблица 3.2

Пример программы загрузки регистров

Адрес	Команда	Машинный код	Комментарий
0000	***MVI***	3E F0	загрузка регистра A←F0h
0002	***MVI***	06 0F	загрузка регистра B←0Fh

5. Ввести лабораторный макет в пошаговый режим работы и осуществить сброс микропроцессора (***RESET***). При этом значение программного счетчика микропроцессора должно установиться равным ***0000h***, а на внутреннюю шину данных будет выставлен код операции команды ***MVI A,F0h*** равный ***3E.*** Выставить минимальную частоту работы макета.

6. Наблюдая за внутренней шиной данных выполнить одну команду. В ходе выполнения первой команды, после считывания кода операции, на шину данных на короткое время будет выставлено число ***F0h***. После считывания операнда на шине данных будет выставлен код операции следующей команды - ***MVI B,0Fh***.

Лабораторная работа № 2

Команды загрузки регистров. Команды пересылки

Теоретические сведения

В микропроцессоре ***Intel 8080*** для программирования доступны следующие регистры:

- 16-разрядный счетчик команд (***PC***) содержит адрес команды, которая подлежит выполнению;
- 16-разрядный регистр указатель стека (***SP***), определяет адрес специализированной области ОЗУ-стека;
- 8-разрядный регистр-накопитель (***A***), используется для хранения и накопления результата в арифметических, логических операциях, а также в операциях ввода-вывода и сдвига. Кроме того, он может быть использован в качестве регистра общего назначения для хранения данных;
- шесть 8 –разрядных регистров общего назначения (РОН): ***B,C,D,E,H,L***;
- 8-разрядный регистр признаков ***PSW*** (***F***) содержит биты условий: ***C*** – перенос, ***AC*** – вспомогательный перенос, ***S*** – знак, ***Z*** – ноль, ***P*** – четность. Данные биты устанавливаются в зависимости от результата операции при выполнении арифметических, логических команд, команд сдвига и сравнения. Распределение условий в байте признаков показано в табл. 3.3.

Таблица 3.3

Распределение битов в байте состояния

7	6	5	4	3	2	1	0
S	***Z***	***0***	***AC***	***0***	***P***	***1***	***C***

Регистры общего назначения могут использоваться для манипуляции 16-разрядными данными. Для этого регистры объединяются в пары следующим образом: ***BC, DE, HL***, где соответственно, первый регистр используется для хранения старшего байта (например, ***B***), а второй – для хранения младшего байта (***C***), а также ***PSW*** (***A + C***) слово состояния программы.

Программа работы

1. Команды загрузки регистров общего назначения

Общий вид команды: ***MVI R, d8,***

где ***R*** – идентификатор регистра: ***A, B, C, D, E, H, L***, а ***d8*** – непосредственный операнд (байтовое число).

Пример: Запишите в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 3.4.

Таблица 3.4

Программа загрузки внутренних регистров микропроцессора

Адрес	Команда	Машинный код	Комментарий
0000	***MVI A,00***	3E 00	загрузка регистра ***A←00h***
0002	***MVI B,01***	06 01	загрузка регистра ***B←01h***
0004	***MVI C,02***	0E 02	загрузка регистра ***C←02h***
0006	***MVI D,03***	16 03	загрузка регистра ***D←03h***
0008	***MVI E,04***	1E 04	загрузка регистра ***E←04h***
000A	***MVI H,05***	26 05	загрузка регистра ***H←05h***
000C	***MVI L,06***	2E 06	загрузка регистра ***L←06h***

Выполнить эту последовательность команд в пошаговом режиме и пронаблюдать, как изменяется содержимое регистров общего назначения ***A, B, C, D, E, H, L.*** Значения регистров должны быть следующими:

A = 00h; B = 01h; C = 02h; D = 03h; E = 04h; H = 05h; L = 06h.

2. Команды загрузки регистров 16-разрядными данными

Общий вид команды: ***LXI R, <dh dl>***

где ***R*** – идентификатор пары регистров: ***B, D, H;***

dh – старший байт 16-разрядного операнда;

dl – младший байт 16-разрядного операнда.

Пример: Записать в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 3.5.

Таблица 3.5

Программа загрузки регистровых пар

Адрес	Команда	Машинный код	Комментарий
0000	***LXI B, 3132h***	01 32 31	Загрузка регистровой пары ***BC*** числом ***3132h***. Младший байт данных загружается в регистр ***C***, а старший байт – в регистр ***B***.

Окончание табл. 3.5

0003	***LXI D, 3334h***	11 34 33	Загрузка регистровой пары ***DE*** числом ***3334h***. Младший байт данных загружается в регистр ***E***, а старший байт – в регистр ***D***.
0006	***LXI H, 3536h***	21 36 35	Загрузка регистровой пары ***HL*** числом ***3536h***. Младший байт данных загружается в регистр ***L***, а старший байт – в регистр ***H***.

Примечание: в памяти располагается сначала младший байт операнда, затем – старший.

Выполнить выше описанную последовательность команд в пошаговом режиме и пронаблюдать, как изменяется содержимое регистров ***B, C, D, E, H, L***. Значения регистров должны быть следующими:

B = 31h; C = 32h; D = 33h; E = 34h; H = 35h; L = 36h.

3. Команды загрузки регистра указателя стека.

Команда непосредственной загрузки регистра указатель стека имеет вид: ***LXI SP, <dh dl>,***

где ***dh*** – старший байт 16-разрядного операнда;

dl – младший байт 16-разрядного операнда.

Пример: Записать в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 3.6.

Таблица 3.6

Программа загрузки указателя стека

Адрес	Команда	Машинный код	Комментарий
0000	***LXI SP, 0B01h***	31 01 0B	загрузка указателя стека: ***SP←0B01h***
0003	***LXI SP, 0100h***	31 00 01	загрузка указателя стека: ***SP←0100h***

Выполнить эту последовательность команд в пошаговом режиме и просмотрите содержание регистра указателя стека после каждого шага выполнения программы.

После первого шага, т.е. после выполнения микропроцессором первой команды, содержимое регистра указателя стека должно быть равным ***SP = 0B01h***.

После второго шага, т.е. после выполнения микропроцессором второй команды, содержимое регистра указателя стека должно быть равным ***SP = 0100h***.

Команда косвенной загрузки регистра указатель стека имеет вид:

SPHL

По этой команде в указатель стека загружается содержимое регистровой пары ***HL***. Поэтому, чтобы в указатель стека загрузить, например, число ***0210h***, его предварительно надо загрузить в регистровую пару ***HL***.

Пример: Записать в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 3.7.

Таблица 3.7

Программа косвенной загрузки указателя стека

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0210h***	21 10 02	загрузка ***HL←0210h***
0003	***SPHL***	F9	загрузка ***SP←HL***

Выполнить эту последовательность команд в пошаговом режиме и просмотрите содержание регистра указателя стека после выполнения программы. После выполнения содержимое регистра указателя стека должно быть равным ***SP = 0210h***.

4. Команды пересылки.

Общий вид команды: ***MOV R1, R2,***

где ***R1*** – идентификатор регистра получателя: ***A, B, C, D, E, H, L***;

R2 – идентификатор регистра источника: ***A, B, C, D, E, H, L***.

Пример: Записать в память, начиная с адреса ***0000h***, последовательность команд, представленных в табл. 3.8.

Таблица 3.8

Программа пересылки данных между регистрами

Адрес	Команда	Машинный	Комментарий
0000	***MVI A, FFh***	3E FF	загрузка регистра ***A←FFh***
0002	***MOV B, A***	47	пересылка B←***A***
0003	***MOV C, B***	48	пересылка ***C←B***
0004	***MOV D, C***	51	пересылка ***D←C***
0005	***MOV E, D***	5A	пересылка ***E←D***
0006	***MOV H, E***	63	пересылка ***H←E***
0007	***MOV L, H***	6C	пересылка ***L←H***

Выполнить эту последовательность команд в пошаговом режиме и пронаблюдать, как изменяется содержимое регистров ***A, B, C, D, E, H, L.*** Значения регистров должны быть следующими:

A = FFh; B = FFh; C = FFh; D = FFh; E = FFh; H = FFh; L = FFh.

5. Команда загрузки счетчика команд.

Общий вид команды: ***PCHL***

По этой команде в счетчик команд записывается содержимое пары регистров ***HL***. Т.о., для того, чтобы загрузить в счетчик команд адрес ***0100h***, необходимо сначала это число загрузить в регистровую пару ***HL***.

Пример: Записать в память по адресу ***0000h***, последовательность команд, представленных в табл. 3.9.

Таблица 3.9

Программа косвенной загрузки программного счетчика

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0100h***	21 00 01	***HL←0100h***
0003	***PCHL***	E9	***PC ← HL***

Выполнить эту последовательность команд в пошаговом режиме и пронаблюдайте, как изменяется содержимое регистров ***H, L*** и программного счетчика ***PC.***

Контрольные задания

1. Написать и выполнить программу загрузки регистров в соответствии с табл. 3.10.

Таблица 3.10

Варианты заданий загрузки регистров

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
	B:←F0h	*B:←01h*	*B:←AEh*	*B:←21h*	*B:←23h*	*B:←BBh*
	C:←33h	*C:←35h*	*C:←FBh*	*C:←16h*	*C:←45h*	*C:←CCh*
	D:←EEh	*D:←EAh*	*D:←35h*	*D:←E1h*	*D:←10h*	*D:←D1h*
	E:←AAh	*E:←A1h*	*E:←26h*	*E:←D5h*	*E:←62h*	*E:←EEh*
	H:←00h	*H:←A1h*	*H:←16h*	*H:←01h*	*H:←A5h*	*H:←AAh*
	L:←19h	*L:←18h*	*L:←AAh*	*L:←20h*	*L:←97h*	*L:←FFh*
	A:←FFh	*A:←F5h*	*A:←FEh*	*A:←25h*	*A:←F1h*	*A:←43h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
	B:←23h	*B:←23h*	*B:←0Eh*	*B:←11h*	*B:←23h*	*B:←8Bh*
	C:←45h	*C:←45h*	*C:←0Bh*	*C:←66h*	*C:←33h*	*C:←1Ch*

Окончание табл. 3.10

	D:←10h	*D:←1Ah*	*D:←05h*	*D:←11h*	*D:←15h*	*D:←11h*
	E:←62h	*E:←A1h*	*E:←36h*	*E:←55h*	*E:←1Dh*	*E:←E3h*
	H:←A5h	*H:←1Bh*	*H:←06h*	*H:←00h*	*H:←D1h*	*H:←A5h*
	L:←97h	*L:←19h*	*L:←0Ah*	*L:←22h*	*L:←2Dh*	*L:←F5h*
	A:←95h	*A:←53h*	*A:←0Eh*	*A:←66h*	*A:←F3h*	*A:←44h*

Проверить правильность выполнения программы.

2. Написать и выполнить программу загрузки регистровых пар в соответствии с табл. 3.11.

Таблица 3.11

Варианты заданий загрузки регистровых пар

Вариант	**1**	**2**	**3**
	BC←FFFFh	*BC←00FFh*	*BC←0000h*
	DE←0123h	*DE←0124h*	*DE←0F0Fh*
	HL←55AAh	*HL←5555h*	*HL←1579h*
Вариант	**4**	**5**	**6**
	BC←0E0Eh	*BC←1100h*	*BC←DDDDh*
	DE←E0E0h	*DE←4545h*	*DE←ABCDh*
	HL←000Eh	*HL←536Ah*	*HL←DCBAh*
Вариант	**7**	**8**	**9**
	BC←13EFh	*BC←1234h*	*BC←FEDCh*
	DE←A734h	*DE←5678h*	*DE←BA98h*
	HL←1FA9h	*HL←9ABCh*	*HL←7654h*
Вариант	**10**	**11**	**12**
	BC←3210h	*BC←2468h*	*BC←0000h*
	DE←35DFh	*DE←ACE2h*	*DE←1111h*
	HL←5555h	*HL←468Ah*	*HL←2222h*

Проверить правильность работы программы.

3. Написать и выполнить программы загрузки регистра указателя стека, в соответствии с табл. 3.12, с использованием команд непосредственной и косвенной загрузки.

Таблица 3.12

Варианты заданий загрузки регистра указателя стека

Вариант	**1**	**2**	**3**
	SP←0100h	*SP←0101h*	*SP←0802h*
	SP←0200h	*SP←0202h*	*SP←0812h*
	SP←0300h	*SP←0203h*	*SP←0822h*
Вариант	**4**	**5**	**6**
	SP←0200h	*SP←0800h*	*SP←0702h*

Окончание табл. 3.12

	SP←0201h	*SP←0900h*	*SP←0802h*
	SP←0202h	*SP←0000h*	*SP←0902h*
Вариант	**7**	**8**	**9**
	SP←0200h	*SP←000Ah*	*SP←0110h*
	SP←0222h	*SP←000Bh*	*SP←0120h*
	SP←0333h	*SP←0A0Bh*	*SP←0130h*
Вариант	**10**	**11**	**12**
	SP←0800h	*SP←0110h*	*SP←0115h*
	SP←0825h	*SP←0111h*	*SP←0178h*
	SP←0850h	*SP←0800h*	*SP←0564h*

Проверить правильность работы программы.

4. Написать и выполнить программу пересылки, в соответствии с табл. 3.13, предварительно загрузив регистры.

Таблица 3.13

Варианты заданий пересылки данных

Вариант	**1**	**2**	**3**
	B←A (значением 00h)	*D←H (значением 01h)*	*B←C (значением 10h)*
	C←L (значением 0Eh)	*E←C (значением 02h)*	*C←A (значением 20h)*
	H←B (значением 0Fh)	*B←L (значением 03h)*	*A←B (значением 30h)*
Вариант	**4**	**5**	**6**
	L←H (значением 0Fh)	*H←A (значением FFh)*	*C←B (значением 15h)*
	C←A (значением 0Eh)	*E←A (значением EEh)*	*C←H (значением 25h)*
	D←E (значением 0Bh)	*B←C (значением D1h)*	*A←L (значением 35h)*
Вариант	**7**	**8**	**9**
	A←H (значением 22h)	*B←C (значением 00h)*	*B←C (значением 15h)*
	C←B (значением 11h)	*D←E (значением 0Eh)*	*C←A (значением 20h)*
	D←L (значением 10h)	*H←L (значением 0Fh)*	*A←B (значением 2Ah)*
Вариант	**10**	**11**	**12**
	H←A (значением 00h)	*B←A (значением AAh)*	*A←H (значением 21h)*
	E←A (значением 01h)	*C←L (значением 0Ah)*	*C←B (значением 22h)*
	B←C (значением 05h)	*H←B (значением 00h)*	*D←L (значением 23h)*

Проверить правильность выполнения программы.

5. Написать и выполнить программу перехода с адреса ***0000h*** на адреса в соответствии с табл. 3.14.

Таблица 3.14

Варианты заданий перехода на адрес

Вариант	**1**	**2**	**3**
	0100h	*0050h*	*0080h*
	0200h	*0060h*	*0090h*
	0300h	*0070h*	*0100h*

Окончание табл. 3.12

Вариант	**4**	**5**	**6**
	0100h	*0800h*	*0900h*
	0200h	*0200h*	*0200h*
	0800h	*0100h*	*0100h*
Вариант	**7**	**8**	**9**
	0100h	*0200h*	*0100h*
	0200h	*0250h*	*0200h*
	0800h	*0700h*	*0150h*
Вариант	**10**	**11**	**12**
	0110h	*0100h*	*0050h*
	0210h	*0150h*	*0100h*
	0250h	*0700h*	*0150h*

Проверить правильность выполнения программы.

Лабораторная работа № 3

Методы адресации памяти. Команды работы с памятью

Теоретические сведения

Память представляется как последовательность ячеек размером в один байт. Каждая ячейка имеет свой адрес в диапазоне от 0 до 65535. Для удобства обычно используется шестнадцатеричное значение адреса, тогда диапазон адресации составляет ***0000h – FFFFh***.

В микропроцессорной системе адресации адрес ячейки памяти указывается в самой команде во втором и третьем байтах команды (прямая адресация). В общем виде это выглядит следующим образом

КОП ad16

где КОП – код операции (чтение или запись); ***ad16*** – адрес ячейки памяти.

В памяти такая команда будет размещена следующим образом

КОП ad16 (младший байт) ad16 (старший байт),

т.е. после байта кода операции располагается сначала младший байт адреса, а затем – старший.

Косвенная адресация предполагает, что адрес ячейки памяти будет располагаться в регистровых парах ***HL, DE, BC***. Для каждой конкретной команды работы с памятью закреплена своя регистровая пара. Таким образом, прежде чем выполнить такую команду необходимо сначала задать адрес в соответствующей регистровой паре.

Например,

LXI H, 0800h

MOV M, A; запись в память содержимое регистра ***A*** по адресу, находящемуся в регистровой паре ***HL*** или

LXI D, 0900H

STAX D; запись в память содержимое регистра ***A*** по адресу, находящемуся в регистровой паре ***DE***

Программа работы

1. Команды записи в память с прямой адресацией.

Существуют две команды прямой адресации записи в память:

STA ad16 запись в память по прямому адресу ***ad16*** содержимого регистра ***A***;

SHLD ad16 запись в память содержимого регистровой пары ***HL***. Причем по адресу ***ad16*** будет записано содержимое регистра ***L***, а по адресу ***ad16+1*** будет записано содержимое регистра ***H***.

Пример: Запишите в память, начиная с адреса ***0000h***, коды следующих команд, используя прямую адресацию (табл. 3.15)

Таблица 3.15

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MVI A,FFh***	3E FF	Запись в аккумулятор значения ***FFh***
0002	***STA 0110h***	32 10 01	Запись в память содержимого регистра ***A*** по адресу ***0110h***
0005	***LXI H,3536h***	21 36 35	Загрузка регистровой пары ***HL*** числом ***3536h***. Младший байт данных загружается в регистр ***L***, а старший байт – в регистр ***H***.
0008	***SHLD 0150h***	22 50 01	Запись в память содержимого регистра ***L*** по адресу ***0150h***, содержимого регистра ***H*** по адресу ***0151h***

Выполните эту последовательность команд в пошаговом режиме и пронаблюдайте, как изменяется содержимое регистров ***A, H, L*** и содержимое ячеек памяти ***0110h, 0150h, 0151h***. Значения регистров и ячеек памяти должны быть следующими

A = FFh; H = 35h; L = 36h; (0110h) = FFh; (0150h) = 36h; (0151h) = 35h

2. Команды чтение памяти с прямой адресацией.

Аналогично командам записи с прямой адресацией существуют две команды чтения памяти с конкретным адресом

LDA ad16 загрузка регистра ***A*** из ячейки памяти с адресом ***ad16***;

LHLD ad16 чтение памяти по прямому адресу ***ad16*** в регистровую пару ***HL***. При этом в регистр ***H*** будет записано содержимое ячейки с адресом ***ad16+1***, а в регистр ***L*** содержимое ячейки памяти с адресом ***ad16.***

Пример: Запишите в память по адресу ***0000h*** коды следующих команд (табл. 3.16).

Таблица 3.16

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LDA 0190h***	3A 90 01	Чтение в регистр ***A*** содержимого ячейки с адресом ***0190h***
0003	***LHLD 0190h***	2A 90 01	Чтение в регистр ***L*** содержимого ячейки с адресом ***0190h***, а в регистр ***H*** содержимого ячейки с адресом ***0191h***

Вручную внесите в ячейки памяти следующие значения
(0190h) = ABh; (0191h) = CDh.

Выполните эту последовательность команд в пошаговом режиме и пронаблюдайте, как изменяется содержимое регистров ***A, H, L***. Значения регистров должны быть следующими:

A = ABh; H = CDh; L = ABh.

3. Команды чтения/записи в память с косвенной адресацией.

Общий вид команды

MOV M, R запись в память содержимого регистра;

MOV R, M загрузка регистра из ячейки памяти, адрес который находится в регистровой паре ***HL***. ***R*** – регистр общего назначения ***A, B, C, D, E, H, L***.

Пример: Запишите в память, начиная с адреса ***0000h***, коды следующей программы (табл. 3.17).

Таблица 3.17

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MVI A, 0AAh***	3E AA	загрузка регистров
0002	***MVI B, 0BBh***	06 BB	
0004	***MVI C, 0CCh***	0E CC	
0006	***MVI D, 0DDh***	16 DD	
0008	***MVI E, 0EEh***	1E EE	
000A	***LXI H, 0100h***	21 00 01	загрузка ***HL=0100h***, адрес ***M***
000D	***MOV M, A***	77	запись в ***M=A***, по адресу ***HL***
000E	***LXI H, 0101h***	21 01 01	
0011	***MOV M, C***	71	
0012	***LXI H, 0102h***	21 02 01	
0015	***MOV M, B***	70	
0016	***LXI H, 0103h***	21 03 01	
0019	***MOV M, E***	73	
001A	***LXI H, 0104h***	21 04 01	
001D	***MOV M, D***	72	
001E	***LXI H, 0105h***	21 05 01	
0021	***MOV M, H***	74	
0022	***LXI H, 0106h***	21 06 01	
0025	***MOV M, L***	75	

Выполните эту последовательность команд. Значения ячеек памяти должны быть следующими

0100h = AAh; 0101h = CCh; 0102h = BBh; 0103h = EEh; 0104h =DDh; 0105h =01h;

0106h = 06h

Пример: Запишите в память, начиная с адреса ***0000h***, коды следующей программы (табл. 3.18).

Таблица 3.18

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0100h***	21 00 01	загрузка ***HL=0100h***, адрес ***M***
0003	***MOV E, M***	5E	чтение ***E=M***, по адресу ***HL***
0004	***LXI H, 0101h***	21 01 01	и т.д.
0007	***MOV D, M***	56	
0008	***LXI H, 0102h***	21 02 01	
000B	***MOV C, M***	4E	
000C	***LXI H, 0103h***	21 03 01	
000F	***MOV B, M***	46	
0010	***LXI H, 0104h***	21 04 01	
0013	***MOV A, M***	7E	
0014	***LXI H, 0105h***	21 05 01	
0017	***MOV H, M***	66	
0018	***LXI H, 0106h***	21 06 01	
001B	***MOV L, M***	6E	

Заполните вручную соответствующие ячейки памяти (***0100h = AAh, 0101h = CCh, 0102h = BBh, 0103h = EEh, 0104h = DDh, 0105h = 01h, 0106h = 06h***).Выполните эту последовательность команд. Значения регистров должны быть следующими

A = DDh; B = EEh; C = BBh; D = CCh; E = Aah; H = 01h; L = 06h

4. Команды чтения/записи при адресации через регистровые пары *BC, DE*.

STAX B запись содержимого регистра ***A*** в память, адрес в регистровой паре ***BC***;

STAX D запись содержимого регистра ***A*** в память, адрес в регистровой паре ***DE***;

LDAX B чтение содержимого памяти в регистр ***A***, адрес в регистровой паре ***BC***;

LDAX D чтение содержимого памяти в регистр ***A***, адрес в регистровой паре ***DE***.

Пример: Запишите в память, начиная с адреса 0000h, коды программы (табл. 3.19).

Таблица 3.19

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LXI B, 0100h***	01 00 01	Загрузка ***BC←0100h***
0003	***MVI A, 0Fh***	3E 0F	Загрузка ***A←0Fh***
0005	***STAX B***	02	Запись в ***M←A*** по адресу ***BC***
0006	***LXI D, 0110h***	11 10 01	Загрузка в ***DE←0110h***
0009	***MVI A, 0F0h***	3E F0	Загрузка в ***A←F0h***
000B	***STAX D***	12	Запись в ***M←A*** по адресу ***DE***

Выполните эту последовательность команд. Значения ячеек памяти должны быть следующими

0100h = 0Fh, 0110h = F0h.

***Пример*:** Запишите в память, начиная с адреса 0000h, коды программы (табл. 3.20).

Таблица 3.20

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LXI D, 0100h***	11 00 09	загрузка в ***DE←0100h***
0003	***LDAX D***	1A	чтение в ***A←M*** по адресу ***DE***
0004	***MOV L, A***	6F	пересылка ***L←A***
0005	***LXI B, 0110h***	01 10 09	загрузка в ***BC←0110h***
0008	***LDAX B***	0A	чтение в ***A←M*** по адресу ***BC***
0009	***MOV H, A***	67	;пересылка ***H←A***

Заполните вручную соответствующие ячейки памяти (***0100h←0Fh, 0110h←F0h***). Выполните эту последовательность команд. Значения регистров должны быть следующими

H←F0h, L←0Fh.

Контрольные задания

1. Напишите и выполните программу записи данных в память из регистра ***A***, в соответствии с табл. 3.21. Для этого используйте команду загрузки регистра ***A*** и команду записи в память регистра ***A*** по прямому адресу.

Таблица 3.21

Варианты заданий записи данных в память

Вариант 1	Адрес	*0100*	*0105*	*0107*	*010C*	*0120*	*0126*
	Данные	*00h*	*01h*	*05h*	*0Ah*	*BBh*	*12h*
Вариант 2	Адрес	*0200*	*0202*	*0205*	*020C*	*0215*	*0220*
	Данные	*25h*	*12h*	*50h*	*A0h*	*BCh*	*1Dh*
Вариант 3	Адрес	*0101*	*0110*	*0120*	*0130*	*0135*	*0140*
	Данные	*15h*	*A1h*	*5Ah*	*A6h*	*00h*	*21h*
Вариант 4	Адрес	*0100*	*0105*	*0107*	*010B*	*0123*	*0126*
	Данные	*22h*	*33h*	*44h*	*55h*	*AAh*	*CCh*
Вариант 5	Адрес	*0200*	*0210*	*0220*	*0226*	*0228*	*0240*
	Данные	*FFh*	*10h*	*01h*	*03h*	*0Bh*	*12h*
Вариант 6	Адрес	*0100*	*0105*	*010C*	*0115*	*0120*	*0130*
	Данные	*66h*	*12h*	*05h*	*0Ah*	*D0h*	*D3h*

Проверьте правильность работы программы.

2. Напишите и выполните программу записи данных из регистровой пары ***HL*** в соответствии с табл. 3.22. Для этого используйте команду загрузки регистровой пары ***HL*** и команду записи в память регистровой пары ***HL*** по прямому адресу.

Таблица 3.22

Варианты заданий записи данных из регистровой пары

Вариант 1	Адрес	*0100*	*0101*	*0111*	*0112*	*0120*	*0121*
	Данные	*00h*	*07h*	*09h*	*0Bh*	*B0h*	*12h*
Вариант 2	Адрес	*0200*	*0201*	*0205*	*0206*	*0210*	*0211*
	Данные	*25h*	*12h*	*50h*	*0Ah*	*0Ch*	*D1h*
Вариант 3	Адрес	*0101*	*0102*	*0120*	*0121*	*0135*	*0136*
	Данные	*15h*	*1Ah*	*A5h*	*6Ah*	*00h*	*21h*
Вариант 4	Адрес	*0100*	*0101*	*0107*	*0108*	*0123*	*0124*
	Данные	*22h*	*33h*	*44h*	*55h*	*AAh*	*CCh*
Вариант 5	Адрес	*0210*	*0211*	*0220*	*0221*	*0228*	*0229*
	Данные	*FFh*	*1Ah*	*A1h*	*03h*	*B0h*	*12h*
Вариант 6	Адрес	*0150*	*0151*	*0161*	*0162*	*0200*	*0201*
	Данные	*66h*	*B2h*	*05h*	*A0h*	*0Dh*	*3Dh*

Проверьте правильность работы программы.

3. Напишите и выполните программу записи данных в память в соответствии с табл. 3.23. Для этого используйте команды загрузки регистровой пары ***HL*** и регистра ***A*** и команды записи в память регистровой пары ***HL*** и регистра ***A*** по прямому адресу.

Таблица 3.23

Варианты заданий записи данных в память

Вариант 1	Адрес	*0100*	*0101*	*0111*	*0115*	*0120*	*0121*
	Данные	*22h*	*17h*	*09h*	*0Ah*	*80h*	*90h*
Вариант 2	Адрес	*0150*	*0201*	*0202*	*0210*	*0212*	*0213*
	Данные	*35h*	*42h*	*1Ah*	*0Ah*	*C0h*	*D3h*
Вариант 3	Адрес	*0101*	*0102*	*0120*	*0125*	*0135*	*0136*
	Данные	*16h*	*1Ah*	*50h*	*60h*	*0Ah*	*2Dh*
Вариант 4	Адрес	*0100*	*0105*	*0107*	*0108*	*0110*	*0124*
	Данные	*25h*	*13h*	*43h*	*56h*	*0Ah*	*C0h*
Вариант 5	Адрес	*0210*	*0211*	*0220*	*0221*	*0228*	*0240*
	Данные	*0Fh*	*10h*	*12h*	*D3h*	*B1h*	*1Ch*
Вариант 6	Адрес	*0150*	*0151*	*0160*	*0162*	*0200*	*0201*
	Данные	*65h*	*B4h*	*05h*	*A1h*	*0Ch*	*35h*

Проверьте правильность работы программы.

4. Напишите и выполните программу загрузки регистров ***B, C, D, E, H, L*** из памяти в соответствии с табл. 3.24. Используйте команды чтения памяти в регистр ***A*** по непосредственному адресу и команды пересылки. Значение соответствующих ячеек памяти необходимо заполнить вручную.

Таблица 3.24

Варианты заданий загрузки регистров

Вариант 1	Адрес	*0100h*	*0105h*	*0107h*	*010Ch*	*0120h*	*0126h*
	Данные	***00h***	***01h***	***15h***	***1Ah***	***1Bh***	***22h***
	Регистр	***B***	***C***	***D***	***E***	***H***	***L***
Вариант 2	Адрес	*0200h*	*0205h*	*0215h*	*020Ch*	*0216h*	*0220h*
	Данные	***35h***	***26h***	***51h***	***A1h***	***B0h***	***11h***
	Регистр	***C***	***B***	***D***	***H***	***E***	***L***
Вариант 3	Адрес	*0101h*	*0110h*	*0125h*	*0133h*	*0135h*	*0145h*
	Данные	***16h***	***19h***	***50h***	***A6h***	***0Bh***	***21h***
	Регистр	***B***	***D***	***C***	***E***	***H***	***L***

Окончание табл. 3. 24

Вариант 4	Адрес	*0100h*	*0105h*	*0110h*	*0111h*	*0112h*	*0126h*
	Данные	***21h***	***35h***	***44h***	***56h***	***0Ah***	***C0h***
	Регистр	***B***	***C***	***D***	***E***	***H***	***L***
Вариант 5	Адрес	*0200h*	*0211h*	*0221h*	*0226h*	*0230h*	*0240h*
	Данные	***F0h***	***10h***	***19h***	***13h***	***B0h***	***23h***
	Регистр	***C***	***B***	***D***	***H***	***E***	***L***
Вариант 6	Адрес	*0100h*	*0105h*	*010Ch*	*0115h*	*0120h*	*0130h*
	Данные	***69h***	***37h***	***56h***	***0Ah***	***D6h***	***03h***
	Регистр	***L***	***D***	***C***	***E***	***H***	***B***

Проверьте правильность работы программы.

5. Напишите и выполните программу загрузки регистров ***B, C, D, E, H, L*** из памяти в соответствии с табл. 3.25. Используйте команды чтения памяти в регистровую пару ***HL*** и команды пересылки. Значение соответствующих ячеек памяти необходимо заполнить вручную.

Таблица 3.25

Варианты заданий загрузки регистров

Вариант 1	Адрес	*0100h*	*0101h*	*0108h*	*0109h*	*0121h*	*0122h*
	Данные	*01h*	*05h*	*25h*	*10h*	*1Bh*	*2Ah*
	Регистр	*B*	*C*	*D*	*E*	*L*	*H*
Вариант 2	Адрес	*0200h*	*0201h*	*0215h*	*0216h*	*0220h*	*0221h*
	Данные	*35h*	*26h*	*51h*	*0A1h*	*0B0h*	*11h*
	Регистр	*B*	*C*	*D*	*E*	*L*	*H*
Вариант 3	Адрес	*0101h*	*0102h*	*0125h*	*0126h*	*0135h*	*0136h*
	Данные	*16h*	*29h*	*60h*	*0A7h*	*3Bh*	*A1h*
	Регистр	*B*	*C*	*D*	*E*	*L*	*H*
Вариант 4	Адрес	*0110h*	*0111h*	*0150h*	*0151h*	*0212h*	*0213h*
	Данные	*99h*	*35h*	*88h*	*56h*	*5Ah*	*0Ch*
	Регистр	*B*	*C*	*D*	*E*	*L*	*H*
Вариант 5	Адрес	*0200h*	*0201*	*0221*	*0222h*	*0230h*	*0231*
	Данные	*0F5h*	*0EEh*	*1Eh*	*1Ah*	*0BBh*	*33h*
	Регистр	*B*	*C*	*D*	*E*	*L*	*H*
Вариант 6	Адрес	*0105h*	*0106h*	*010Ch*	*010Dh*	*0183h*	*0184h*
	Данные	*19h*	*38h*	*96h*	*0A1h*	*16h*	*13h*
	Регистр	*B*	*C*	*D*	*E*	*L*	*H*

Проверьте правильность работы программы.

6. Напишите и выполните программу перезаписи данных из одних ячеек памяти (адрес 1) в другие (адрес 2) в соответствии с табл. 3.26. Значение соответствующих ячеек памяти необходимо заполнить вручную.

Таблица 3.26

Варианты заданий перезаписи данных

Вариант 1	Адрес 1	*0100h*	*0101h*	*0108h*	*0109h*	*0121h*	*0125h*
	Адрес 2	*0200h*	*0201h*	*0208h*	*0209h*	*0235h*	*0240h*
	Данные	*11h*	*51h*	*25h*	*1Ah*	*10h*	*2Bh*
Вариант 2	Адрес 1	*0200h*	*0201h*	*0215h*	*0216h*	*0220h*	*0229h*
	Адрес 2	*0250h*	*0251h*	*0208h*	*0209h*	*0135h*	*0140h*
	Данные	*35h*	*26h*	*51h*	*A1h*	*B0h*	*11h*
Вариант 3	Адрес 1	*0101h*	*0102h*	*0120h*	*0126h*	*0135h*	*0136h*
	Адрес 2	*0300h*	*0301h*	*0308h*	*0310h*	*0335h*	*0336h*
	Данные	*26h*	*29h*	*61h*	*A8h*	*4Bh*	*11h*
Вариант 4	Адрес 1	*0110h*	*0111h*	*0150h*	*0180h*	*0212h*	*0213h*
	Адрес 2	*0200h*	*0201h*	*0205h*	*0207h*	*0250h*	*0251h*
	Данные	*9Ah*	*35h*	*90h*	*58h*	*6Ah*	*1Ch*
Вариант 5	Адрес 1	*0200*	*0201*	*0221*	*0222h*	*0230h*	*0240h*
	Адрес 2	*0100*	*0101*	*0121*	*0122h*	*0140h*	*0150h*
	Данные	*F7h*	*E0h*	*10h*	*A6h*	*0Bh*	*33h*
Вариант 6	Адрес 1	*0105h*	*0106h*	*010Ch*	*010Dh*	*0180h*	*0190h*
	Адрес 2	*0210h*	*0211h*	*0220h*	*0221h*	*0290h*	*0300h*
	Данные	*1Ah*	*3Dh*	*96h*	*A1h*	*1Eh*	*1Dh*

Проверьте правильность работы программы.

7. Напишите и выполните программу записи в память содержимого регистров в соответствии с табл. 3.27, используя команды записи в память с косвенной адресацией (через регистровую пару ***HL***).

Таблица 3.27

Варианты заданий записи в память содержимого регистров

	Регистры	**A**	**B**	**C**	**D**	**E**	**H**	**L**
Вариант 1	Адрес	*0100h*	*0101h*	*0102h*	*0103h*	*0104h*	*0105h*	*0106h*
	Данные	*FFh*	*EEh*	*BBh*	*00h*	*AAh*	*01h*	*06h*
Вариант 2	Адрес	*0200h*	*0201h*	*0202h*	*0203h*	*0204h*	*0205h*	*0206h*
	Данные	*00h*	*E1h*	*B1h*	*01h*	*A1h*	*02h*	*06h*

Окончание табл. 3. 27

Вариант 3	Адрес	*0110h*	*0111h*	*0112h*	*0113h*	*0114h*	*0115h*	*0116h*
	Данные	*34h*	*12h*	*B0h*	*50h*	*A5h*	*01h*	*16h*
Вариант 4	Адрес	*0125h*	*0126h*	*0127h*	*0128h*	*0129h*	*0130h*	*0131h*
	Данные	*99h*	*E1h*	*0Bh*	*05h*	*00h*	*01h*	*31h*
Вариант 5	Адрес	*0130h*	*0131h*	*0132h*	*0133h*	*0134h*	*0135h*	*0136h*
	Данные	*22h*	*E7h*	*05h*	*06h*	*05h*	*01h*	*36h*
Вариант 6	Адрес	*0100h*	*0101h*	*0102h*	*0103h*	*0104h*	*0105h*	*0106h*
	Данные	*98h*	*10h*	*20h*	*30h*	*22h*	*01h*	*06h*

Проверьте правильность работы программы.

8. Напишите и выполните программу чтения содержимого памяти в соответствующие регистры (табл. 3.28), используя команды чтения из памяти с косвенной адресацией (через регистровую пару ***HL***).

Таблица 3.28

Варианты заданий чтения содержимого памяти

	Реги-стры	**D**	**B**	**C**	**A**	**E**	**H**	**L**
Вариант 1	Адрес	*0100h*	*0101h*	*0102h*	*0103h*	*0104h*	*0105h*	*0106h*
	Данные	*0FFh*	*0EEh*	*0BBh*	*00h*	*0AAh*	*01h*	*06h*
Вариант 2	Адрес	*0200h*	*0201h*	*0202h*	*0203h*	*0204h*	*0205h*	*0206h*
	Данные	*00h*	*0E1h*	*0B1h*	*01h*	*0A1h*	*02h*	*06h*
Вариант 3	Адрес	*0110h*	*0111h*	*0112h*	*0113h*	*0114h*	*0115h*	*0116h*
	Данные	*34h*	*12h*	*0B0h*	*50h*	*0A5h*	*01h*	*16h*
Вариант 4	Адрес	*0125h*	*0126h*	*0127h*	*0128h*	*0129h*	*0130h*	*0131h*
	Данные	*99h*	*0E1h*	*0Bh*	*05h*	*00h*	*01h*	*31h*
Вариант 5	Адрес	*0130h*	*0131h*	*0132h*	*0133h*	*0134h*	*0135h*	*0136h*
	Данные	*22h*	*0E7h*	*05h*	*06h*	*05h*	*01h*	*36h*
Вариант 6	Адрес	*0100h*	*0101h*	*0102h*	*0103h*	*0104h*	*0105h*	*0106h*
	Данные	*98h*	*10h*	*20h*	*30h*	*22h*	*01h*	*06h*

Проверьте правильность работы программы.

9. Напишите и выполните программу записи данных в две области памяти, используя для адресации регистровую пару ***BC*** и регистровую пару ***DE*** в соответствии с табл. 3.29.

Таблица 3.29

Варианты заданий записи данных в две области памяти

Вариант 1	Адрес 1	*0100h*	*0101h*	*0108h*	*0109h*	*0121h*	*0125h*
	Адрес 2	*0200h*	*0201h*	*0208h*	*0209h*	*0235h*	*0240h*
	Данные	*11h*	*51h*	*25h*	*1Ah*	*10h*	*2Bh*
Вариант 2	Адрес 1	*0200h*	*0201h*	*0215h*	*0216h*	*0220h*	*0229h*
	Адрес 2	*0250h*	*0251h*	*0208h*	*0209h*	*0135h*	*0140h*
	Данные	*35h*	*26h*	*51h*	*A1h*	*B0h*	*11h*
Вариант 3	Адрес 1	*0101h*	*0102h*	*0120h*	*0126h*	*0135h*	*0136h*
	Адрес 2	*0300h*	*0301h*	*0308h*	*0310h*	*0335h*	*0336h*
	Данные	*26h*	*29h*	*61h*	*A8h*	*4Bh*	*11h*
Вариант 4	Адрес 1	*0110h*	*0111h*	*0150h*	*0180h*	*0212h*	*0213h*
	Адрес 2	*0200h*	*0201h*	*0205h*	*0207h*	*0250h*	*0251h*
	Данные	*9Ah*	*35h*	*90h*	*58h*	*6Ah*	*1Ch*
Вариант 5	Адрес 1	*0200h*	*0201h*	*0221h*	*0222h*	*0230h*	*0240h*
	Адрес 2	*0100h*	*0101h*	*0121h*	*0122h*	*0140h*	*0150h*
	Данные	*F7h*	*E0h*	*10h*	*A6h*	*0Bh*	*33h*
Вариант 6	Адрес 1	*0105h*	*0106h*	*010Ch*	*010Dh*	*0180h*	*0190h*
	Адрес 2	*0210h*	*0211h*	*0220h*	*0221h*	*0290h*	*0300h*
	Данные	*1Ah*	*3Dh*	*96h*	*A1h*	*1Eh*	*1Dh*

Проверьте правильность работы программы.

10. Напишите и выполните программу перезаписи данных из одной области памяти (адресуйтесь через ***BC***) в другую область памяти (адресуйтесь через ***DE***) в соответствии с табл. 3.30. Значение соответствующих ячеек памяти необходимо заполнить вручную.

Таблица 3.30

Варианты заданий загрузки регистров

Вариант 1	Адрес 1	*0100h*	*0105h*	*0108h*	*0109h*	*0121h*	*0125h*
	Адрес 2	*0200h*	*0205h*	*0208h*	*0209h*	*0221h*	*0225h*
	Данные	***11h***	***51h***	***25h***	***1Ah***	***10h***	***2Bh***
Вариант 2	Адрес 1	*0200h*	*0201h*	*0215h*	*0216h*	*0220h*	*0229h*
	Адрес 2	*0100h*	*0101h*	*0115h*	*0116h*	*0120h*	*0129h*
	Данные	***35h***	***26h***	***51h***	***A1h***	***B0h***	***11h***
Вариант 3	Адрес 1	*0101h*	*0102h*	*0120h*	*0126h*	*0135h*	*0136h*
	Адрес 2	*0301h*	*0302h*	*0320h*	*0326h*	*0335h*	*0336h*
	Данные	***26h***	***29h***	***61h***	***A8h***	***4Bh***	***11h***

Окончание табл. 3.30

Вариант 4	Адрес 1	*0110h*	*0120h*	*0150h*	*0180h*	*0212h*	*0213h*
	Адрес 2	*0210h*	*0220h*	*0050h*	*0080h*	*0112h*	*0113h*
	Данные	***9Ah***	***35h***	***90h***	***58h***	***6Ah***	***1Ch***
Вариант 5	Адрес 1	*0200h*	*0201h*	*0221h*	*0222h*	*0230h*	*0240h*
	Адрес 2	*0100h*	*0101h*	*0121h*	*0122h*	*0130h*	*0140h*
	Данные	***F7h***	***E0h***	***10h***	***A6h***	***0Bh***	***33h***
Вариант 6	Адрес 1	*0105h*	*0106h*	*010Ch*	*010Dh*	*0180h*	*0190h*
	Адрес 2	*0205h*	*0206h*	*020Ch*	*020Dh*	*0280h*	*0290h*
	Данные	***1Ah***	***3Dh***	***96h***	***A1h***	***1Eh***	***1Dh***

Проверьте правильность работы программы.

Лабораторная работа № 4

Арифметические команды

Теоретические сведения

В микропроцессоре ***Intel 8080*** предусмотрены следующие команды двоичной арифметики: сложение 8-разрядных чисел; сложение 16-разрядных чисел; вычитание 4-разрядных чисел; инкремент; декремент.

Все арифметические операции с 8-разрядными операндами предполагают, что один из операторов размещается в регистре аккумуляторе, а другой либо в регистре, либо в памяти (при этом адрес ячейки задается в регистровой паре ***HL***),либо является непосредственным числом, заданным в самой команде. Вычитания производятся всегда из регистра аккумулятора. Результат арифметической операции записывается в аккумуляторе. Кроме того, по результату арифметических операций сложения и вычитания устанавливаются биты признаков ***C*** – переноса, ***Z*** – нуля, ***S*** – знака, ***P*** – четности, ***AC*** – вспомогательного переноса.

Команды сложения 16-разрядных чисел, так называемой удвоенной точности, предусматривают, что один из операндов находится в регистровой паре ***HL***, а второй – либо в ***DE***, либо в ***BC***. Результат записывается в ***HL***. Кроме того, по результату операции устанавливается либо сбрасывается бит переноса – ***C***.

Команды инкремента увеличивают содержимое регистров, ячейки памяти по адресу в ***HL*** и регистровых пар на 1. Команда инкремент регистра и памяти изменяет биты признаков ***Z, S, P, AC***. Инкремент регистровых пар не затрагивает биты признаков.

Команды декремента уменьшают содержимое регистров, ячейки памяти по адреса в ***HL*** и регистровых пар на 1. Затрагиваемые биты признаков аналогичны команде инкремент.

Программа работы

1. Команды сложения 8-разрядных чисел.

ADD R сложение аккумулятора с содержимым одного из регистров ***B, C, D, E, H, L***.

ADD M сложение аккумулятора с содержимым ячейки памяти (адрес в ***HL***).

ADI d8 сложение аккумулятора с непосредственным числом ***d8***.

ADC R сложение аккумулятора с содержимым одного из регистров и бита переноса ***C***.

ADC M сложение аккумулятора с содержимым ячейки памяти (адрес ***HL***) и бита переноса ***C***.

ACI d8 сложение аккумулятора с непосредственным числом ***d8*** и бита переноса ***C***.

Пример: Запишите в памяти, начиная с адреса ***0000h***, коды программы (табл. 3.31), реализующей

A=A+B+(M)+1

Таблица 3.31

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***ADD B***	80	***A←A+B***
0001	***LXI H, 0100h***	21 00 01	Загрузка ***HL←0100h***
0004	***ADD M***	86	***A←A+(M)***
0005	***ADI 1***	C6 01	***A←A+1***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.32 и проверьте полученные результаты.

Таблица 3.32

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
A	*00h*	*00h*	*00h*	*F0h*	*FFh*	*55h*
B	*00h*	*02h*	*10h*	*0Eh*	*00h*	*AAh*
M	*00h*	*03h*	*45h*	*00h*	*00h*	*FFh*
Вариант	7	**8**	**9**	**10**	**11**	**12**
A	*00h*	*0Ah*	*AAh*	*0Fh*	*F0h*	*66h*
B	*05h*	*02h*	*11h*	*0Eh*	*01h*	*0Ah*
M	*10h*	*13h*	*45h*	*15h*	*09h*	*11h*

***Пример*:** Запишите в памяти, начиная с адреса 0000h, коды программы сложения 16-разрядных чисел, используя команды 8-разрядного сложения (табл. 3.33)

(HL)=(DE)+(BC)

Таблица 3.33

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, C***	79	
0001	***ADD E***	83	Сложение младших байтов, установка бита переноса, если переполнение
0002	***MOV L, A***	6F	Младший байт результата в регистр ***L***
0003	***MOV A, B***	78	
0004	***ADC D***	8A	Сложение старших байтов с учетом переноса
0005	***MOV H, A***	67	Старший байт результата в регистр ***H***

Выполните программ, предварительно задавая исходные значения в соответствии с табл. 3.34. Проверьте результат.

Таблица 3.34

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
BC	*0001h*	*02C5h*	*F000h*	*8137h*	*809Fh*	*FFFFh*
DE	*00FEh*	*03F1h*	*0FFFh*	*72D9h*	*8121h*	*0000h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
BC	*0005h*	*01C0h*	*F100h*	*1234h*	*819Fh*	*00FFh*
DE	*00F0h*	*02C1h*	*0FF0h*	*70D0h*	*8122h*	*0001h*

2. Команды вычитания 8-разрядных чисел.

SUB R вычитание из аккумулятора содержимого одного из регистров ***B, C, D, E, H, L***;

SUB M вычитание из аккумулятора содержимого ячейки памяти (адрес в ***HL***);

SUI d8 вычитание из аккумулятора непосредственного числа ***d8***;

SBB R вычитание из аккумулятора содержимого одного из регистров ***B, C, D, E, H, L*** минус бит переноса ***C***;

SBB M вычитание из аккумулятора содержимого ячейки памяти (адрес в ***HL***) минус бит переноса ***C***;

SBI d8 вычитание из аккумулятора непосредственного числа ***d8*** минус бит переноса ***C***.

Пример: Запишите в памяти, начиная с адреса ***0000h***, коды программы (табл. 3.35), реализующей функцию

(A)=(A)-(B)-(M)-1

Таблица 3.35

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***SUB B***	90	***(A)=(A)-(B)***
0001	***LXI H, 0100h***	21 00 01	Загрузка ***HL=0100h*** адрес ***M***
0004	***SUB M***	96	***(A)=(A)-(M)***
0005	***SBI 1***	DE 01	***(A)=(A)-1***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.36. Проверьте полученные результаты.

Таблица 3.36

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
A	*FFh*	*00h*	*01h*	*25h*	*00h*	*05h*
B	*01h*	*FFh*	*01h*	*20h*	*00h*	*06h*
M	*01h*	*00h*	*00h*	*04h*	*10h*	*FFh*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
A	*F0h*	*03h*	*11h*	*15h*	*03h*	*15h*
B	*01h*	*F0h*	*13h*	*10h*	*01h*	*06h*
M	*02h*	*11h*	*02h*	*04h*	*02h*	*F0h*

Пример*:** Запишите в памяти, начиная с адреса ***0000h, коды программы вычитания 16-разрядных чисел (табл. 3.37)

(HL)=(DE)-(BC)

Таблица 3.37

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, E***	7B	
0001	***SUB C***	91	Вычитание младший байтов ***(A)=(E)-(C)***
0002	***MOV L, A***	6F	Если ***E<C,*** то перенос=1
0003	***MOV A, D***	7A	
0004	***SBB B***	98	Вычитание старшего байта с учетом переноса ***(D)-(B)-C***
0005	***MOV H, A***	67	

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.38. Проверьте полученный результат.

Таблица 3.38

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
BC	*0001h*	*02C5h*	*F000h*	*8137h*	*809Fh*	*FFFFh*
DE	*00FEh*	*03F1h*	*0FFFh*	*72D9h*	*8121h*	*0000h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
BC	*0005h*	*01C0h*	*F100h*	*1234h*	*819Fh*	*00FFh*
DE	*00F0h*	*02C1h*	*0FF0h*	*70D0h*	*8122h*	*0001h*

3. Команды сложения с удвоенной точностью.

DAD H сложение ***(HL)=(HL)+(HL)***

DAD B сложение ***(HL)=(HL)+(BC)***

DAD D сложение ***(HL)=(HL)+(DE)***

Пример: Запишите в памяти, начиная с адреса **0000h**, коды программы (табл. 3.39), реализующей

(HL)=(BC)+(DE)

Таблица 3.39

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV H, B***	60	Пересылка ***(H)←(B)***
0001	***MOV L, C***	69	Пересылка ***(H)←(B)***
0002	***DAD D***	19	***(HL)=(HL)+(DE)***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.40. Проверьте результаты.

Таблица 3.40

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
BC	*0000h*	*7FFFh*	*8000h*	*55AAh*	*ECB9h*	*FFFFh*
DE	*7FFFh*	*8000h*	*8000h*	*AA55h*	*1347h*	*8000h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
BC	*0100h*	*7000h*	*1000h*	*11AAh*	*ECBDh*	*F00Fh*
DE	*7FF0h*	*8010h*	*AA00h*	*0055h*	*0001h*	*8011h*

4. Команды инкремента.

INR R увеличение на 1 содержимого регистра ***A, B, C, D, E, H, L***;

INR M увеличение на 1 содержимого ячейки памяти, адрес в ***HL***;

INX R увеличение на 1 содержимого регистровой пары ***BC, DE, HL, SP*** (указателя стека). В команде указывается идентификатор старшего регистра, например, ***INX B.***

Пример: Запишите в памяти, начиная с адреса ***0000h***, код команды

Таблица 3.41

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***INR R***	--	***(R)=(R)+1***

Выполните данную команду для следующих регистров и исходных значений и проверьте полученные результаты (табл. 3.42).

Таблица 3.42

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
R	***A***	***B***	***C***	***D***	***E***	***H***
Исх.зн.	*01h*	*05h*	*07h*	*10h*	*12h*	*13h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
R	***L***	***A***	***B***	***D***	***E***	***H***
Исх.зн.	*1Ah*	*1Dh*	*50h*	*CAh*	*0Bh*	*ABh*

Пример*:** Запишите в памяти, начиная с адреса ***0000h, коды команд (табл. 3.43).

Таблица 3.43

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0100h***	21 00 01	Загрузить ***HL=0100h***, адрес ***M***
0003	***INR M***	34	***(M)=(M)+1***

Выполните данную последовательность команд, для следующих исходных значений содержимого ячейки памяти (табл. 3.44) и проверьте полученные результаты.

Таблица 3.44

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
Исх.зн.	*11h*	*05h*	*07h*	*1Ch*	*12h*	*19h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
Исх.зн.	*1Ah*	*10h*	*51h*	*C0h*	*0Bh*	*A0h*

Пример: Запишите в памяти, начиная с адреса ***0000h***, код команды.

Таблица 3.45

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***INX R***	--	***(R)=(R)+1***

Выполните данную команду, для следующих пар регистров и исходных значений и проверьте полученные результаты (табл. 3.46).

Таблица 3.46

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
R	***BC***	***DE***	***HL***	***BC***	***DE***	***HL***
Исх.зн.	*010F*	*05AA*	*070B*	*0FFF*	*FA12*	*AF13*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
R	***BC***	***DE***	***HL***	***BC***	***DE***	***HL***
Исх.зн.	*1AFD*	*1D35*	*5123*	*CA00*	*0B0F*	*AB01*

5. Команда декремента.

DCR R уменьшение на 1 содержимого регистра ***A, B, C, D, E, H, L***;

DCR M уменьшение на 1 содержимого ячейки памяти, адрес в ***HL***;

DCX R уменьшение на 1 содержимого регистровой пары ***BC, DE, HL, SP*** (указателя стека). В команде указывается идентификатор старшего регистра, например, ***DCX B***.

Пример: Запишите в память начиная с адреса ***0000h***, код команды

Таблица 3.47

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***DCR R***	--	***(R)=(R)-1***

Выполните данную команду, для следующих регистров и исходных значений и проверьте полученные результаты (табл. 3.48).

Таблица 3.48

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
R	***A***	***B***	***C***	***D***	***E***	***H***
Исх.зн.	*11h*	*05h*	*0Ah*	*10h*	*12h*	*10h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
R	***L***	***A***	***B***	***D***	***E***	***H***
Исх.зн.	*01h*	*10h*	*5Fh*	*0Ah*	*1Bh*	*00h*

Пример*:** Запишите в памяти, начиная с адреса ***0000h, код команды

Таблица 3.49

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0100h***	21 00 01	Загрузить ***HL=0100h***, адрес ***M***
0003	***DCR M***	35	***(M)=(M)-1***

Выполните данную последовательность команд, для следующих исходных значений содержимого ячейки памяти (табл. 3.50) и проверьте полученные результаты.

Таблица 3.50

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
Исх.зн.	*F1h*	*05h*	*01h*	*21h*	*11h*	*19h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
Исх.зн.	*F0h*	*FFh*	*25h*	*C0h*	*00h*	*0Fh*

Пример*:** Запишите в памяти, начиная с адреса ***0000h, код команды

Таблица 3.51.

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***DCX R***	2B	***(R)=(R)-1***

Выполните данную команду, для следующих пар регистров и исходных значений и проверьте полученные результаты (табл. 3.52).

Таблица 3.52

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
R	***BC***	***DE***	***HL***	***BC***	***DE***	***HL***
Исх.зн.	*010Fh*	*05AAh*	*0700h*	*0F00h*	*FA12h*	*AF1Fh*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
R	***BC***	***DE***	***HL***	***BC***	***DE***	***HL***
Исх.зн.	*1AF0h*	*1D35h*	*5100h*	*CA01h*	*0B0Ah*	*AB0Ah*

Контрольные задания

1. Напишите и выполните команду реализующую ***(C)=(D)+(E),*** в соответствии с табл. 3.53.

Таблица 3.53

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
D	*10h*	*FFh*	*C7h*	*19h*	*AAh*	*E5h*
E	*80h*	*01h*	*08h*	*49h*	*55h*	*F0h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
D	*70h*	*0Fh*	*C5h*	*12h*	*0Ah*	*EEh*
E	*81h*	*11h*	*88h*	*46h*	*50h*	*F0h*

2. Напишите и выполните программу сложения содержимого двух ячеек памяти (содержимое ячеек необходимо заполнить вручную)

(M1)=(M2)+(M3)
Адрес M1=0100h
Адрес M2=0101h
Адрес M3=0102h

Таблица 3.54

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
M2	*00h*	*FEh*	*D5h*	*22h*	*61h*	*19h*
M3	*F0h*	*02h*	*C2h*	*BBh*	*95h*	*33h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
M2	*11h*	*F0h*	*DDh*	*23h*	*66h*	*18h*
M3	*F1h*	*12h*	*02h*	*B1h*	*09h*	*3Ah*

3. Напишите и выполните программу сложения ***(HL)=(BC)+(E)+4E5Fh*** и заполните таблицу

Таблица 3.55

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
E	*00h*	*FEh*	*D5h*	*22h*	*61h*	*19h*
BC	*0AFFh*	*00FEh*	*0123h*	*0A15h*	*2122h*	*2223h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
E	*11h*	*F0h*	*DDh*	*23h*	*66h*	*18h*
BC	*0AF0h*	*A100h*	*9123h*	*0615h*	*2100h*	*0E23h*

4. Напишите и выполните программу реализующую
(C)=(D)-(E)-10h

Таблица 3.56

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
D	*01h*	*DAh*	*F0h*	*85h*	*9Fh*	*FFh*
E	*FEh*	*10h*	*0Fh*	*72h*	*81h*	*0Ah*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
D	*16h*	*0Ch*	*39h*	*75h*	*F0h*	*EEh*
E	*98h*	*10h*	*01h*	*02h*	*16h*	*DDh*

5. Напишите и выполните программу вычитания содержимого двух ячеек памяти (содержимое ячеек необходимо заполнить вручную)
(M1)=(M2)-(M3); Адрес M1=0100h; Адрес M2=0101h; Адрес M3=0102h

Таблица 3.57

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
M2	*AAh*	*0Eh*	*D5h*	*22h*	*61h*	*19h*
M3	*F0h*	*02h*	*C2h*	*B0h*	*91h*	*3Ah*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
M2	*11h*	*F0h*	*D0h*	*21h*	*66h*	*18h*
M3	*FFh*	*12h*	*02h*	*B1h*	*19h*	*36h*

6. Напишите и выполните программу вычитания ***(HL)=(BC)-(E)-0FFFh***.

Таблица 3.58

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
E	*00h*	*F0h*	*15h*	*23h*	*62h*	*18h*
BC	*0AF0h*	*01F1h*	*5123h*	*1A14h*	*2112h*	*2222h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
E	*11h*	*F1h*	*15h*	*26h*	*67h*	*00h*
BC	*0A12h*	*0101h*	*5123h*	*1A15h*	*2113h*	*2332h*

7. Напишите и выполните программу заполнения массива по заданному индексу элемента массива в соответствии с таблицей.

Таблица 3.59

Варианты заданий

Вариант	**1**					
Базовый элемент массива	0100h					
Номер элемента	*00h*	*03h*	*06h*	*11h*	*18h*	*1Fh*
Содержимое элемента массива	*00h*	*01h*	*02h*	*03h*	*04h*	*05h*
Вариант	**2**					
Базовый элемент массива	0200h					
Номер элемента	*00h*	*03h*	*07h*	*10h*	*12h*	*15h*
Содержимое элемента массива	*01h*	*11h*	*02h*	*AAh*	*B4h*	*15h*
Вариант	**3**					
Базовый элемент массива	0300h					
Номер элемента	*01h*	*05h*	*06h*	*12h*	*1Ah*	*1Bh*
Содержимое элемента массива	*00h*	*22h*	*33h*	*44h*	*04h*	*55h*
Вариант	**4**					
Базовый элемент массива	0150h					
Номер элемента	*02h*	*07h*	*0Ah*	*0Bh*	*12h*	*13h*
Содержимое элемента массива	*05h*	*07h*	*12h*	*13h*	*42h*	*55h*
Вариант	**5**					
Базовый элемент массива	0050h					
Номер элемента	*00h*	*01h*	*03h*	*05h*	*07h*	*09h*
Содержимое элемента массива	*05h*	*17h*	*F1h*	*BAh*	*ABh*	*33h*
Вариант	**6**					
Базовый элемент массива	0100h					
Номер элемента	*00h*	*02h*	*04h*	*06h*	*08h*	*0Ah*
Содержимое элемента массива	*A0h*	*A1h*	*A2h*	*A2h*	*A3h*	*AFh*

8. Напишите и выполните программу перезаписи содержимого массива 1, заданного в задании **4.3.7**, в массив 2 в соответствии таблицей.

Таблица 3.60

Варианты заданий

Вариант	**1**					
Базовый элемент массива 2	0200h					
Номер элемента 2	*02h*	*04h*	*06h*	*08h*	*0Ah*	*0Ch*
Вариант	**2**					
Базовый элемент массива 2	0100h					
Номер элемента 2	*02h*	*04h*	*07h*	*08h*	*0Ah*	*0Bh*

Окончание табл. 3.60

Вариант	**3**					
Базовый элемент массива 2	0200h					
Номер элемента 2	*01h*	*02h*	*03h*	*04h*	*05h*	*06h*
Вариант	**4**					
Базовый элемент массива 2	0300h					
Номер элемента 2	*00h*	*01h*	*03h*	*07h*	*09h*	*0Bh*
Вариант	5					
Базовый элемент массива 2	0150h					
Номер элемента 2	*01h*	*02h*	*06h*	*07h*	*09h*	*0Ah*
Вариант	**6**					
Базовый элемент массива 2	0200h					
Номер элемента 2	*01h*	*02h*	*03h*	*04h*	*05h*	*06h*

9. Напишите и выполните программу заполнения массива памяти данными, используя команды инкремент пары регистров и регистра.

Таблица 3.61

Варианты заданий

Вариант	**1**	**2**	**3**
Массив	*0050h – 0054h*	*0060h – 0064h*	*0070h – 0074h*
Данные	*00h – 04h*	*10h – 14h*	*11h – 15h*
Вариант	**4**	**5**	**6**
Массив	*0080h – 0084h*	*0090h – 0094h*	*0100h – 0104h*
Данные	*00h – 04h*	*0Ah – 0Eh*	*20h – 24h*
Вариант	**7**	**8**	**9**
Массив	*0200h – 0204h*	*0210h – 0214h*	*00220h – 0224h*
Данные	*50h – 54h*	*1Ah – 1Eh*	*00h – 04h*
Вариант	**10**	**11**	**12**
Массив	*0230h – 0234h*	*0240h – 0244h*	*0250h – 0254h*
Данные	*0Bh – 0Fh*	*1Bh – 1Fh*	*05h – 09h*

10. Напишите и выполните программу заполнения массива памяти данными, используя команды декремент пары регистров и регистра.

Таблица 3.62

Варианты заданий

Вариант	**1**	**2**	**3**
Массив	*0054h – 0050h*	*0064h – 0060h*	*0074h – 0070h*
Данные	*05h – 01h*	*16h – 12h*	*19h – 15h*

Окончание табл. 3.60

Вариант	**4**	**5**	**6**
Массив	*0084h – 0080h*	*0094h – 0090h*	*0104h – 0100h*
Данные	*03h – FFh*	*0Eh – 0Ah*	*25h – 21h*
Вариант	**7**	**8**	**9**
Массив	*0204h – 0200h*	*0214h – 0210h*	*00224h – 0220h*
Данные	*54h – 50h*	*1Eh – 1Ah*	*09h – 05h*
Вариант	**10**	**11**	**12**
Массив	*0234h – 0230h*	*0244h – 0240h*	*0254h – 0250h*
Данные	*0Fh – 0Bh*	*1Fh – 1Bh*	*0Ah – 06h*

Лабораторная работа № 5

Логические команды

Теоретические сведения

Для реализации логических операций в системе команд микропроцессора К580ВМ80А предусмотрены следующие логические команды: логическое сложение; логическое умножение; исключающее ИЛИ; инверсия.

Все логические команды выполняются побитно с 8-разрядными операндами. При этом один из операндов размещается в регистре – накопителе, аккумуляторе, а второй – либо в одном из регистров общего назначения, либо в ячейке памяти или задается во втором байте команды. Результат выполнения команды записывается в аккумулятор. При этом бит переноса - сбрасывается в 0, а остальные биты устанавливаются в соответствии с результатом выполнения команды.

Команды логического сложения реализуют логическую операцию «***ИЛИ***». Результат равен 1, если хотя бы один из соответствующих битов равен 1, и равен 0, если оба равны 0. Например

$10101001 \vee 00110010 = 10111011$,

где $\vee$ - обозначение операции «***ИЛИ***».

Команды логического умножения реализуют логическую операцию «***И***». Результат равен 1, если оба бита равны 1, и равен 0, если хотя бы один из них равен 0. Например

$10101001 \wedge 00110010 = 00100000$

где $\wedge$ - обозначение операции «***И***».

Команды «***ИСКЛЮЧАЮЩЕГО ИЛИ***» реализуют логическую операцию сложения по модулю два. Результат равен 1,если соответствующие биты противоположны <1 и 0>, и равен 0, если они одинаковы. Например

$10101001 \oplus 00110010 = 10011011$

где $\oplus$ – обозначение операции «***ИСКЛЮЧАЮЩЕЕ ИЛИ***».

Команда инверсии реализует операцию «***ОТРИЦАНИЕ***» содержимого аккумулятора. Например

$\overline{10101001} = 01010110$.

Программа работы

1. Команды логического сложения

ORA R с регистром ***A, B, C, D, E, H, L***;

ORA M с ячейкой памяти, адрес ячейки памяти ***HL***;

ORI d8 с непосредственным операндом.

Пример: Запишите в память, начиная с адреса ***0000h*** коды программы (табл. 3.63), реализующей выражение

$A \leftarrow A \vee C \vee M \vee 80h$

Таблица 3.63

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***ORA C***	B1	$A \leftarrow A \vee C$
0001	***LXI H, 0100h***	21 00 01	$HL \leftarrow 0100h$
0004	***ORA M***	B6	$A \leftarrow A \vee M$
0005	***ORI 80h***	F9 80	$A \leftarrow A \vee 80h$

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.65

Таблица 3.64

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
A	*01h*	*0Bh*	*0Ch*	*0Dh*	*0Eh*	*0Fh*
C	*12h*	*01h*	*0Ah*	*10h*	*10h*	*11h*
M = 0100h	*25h*	*F0h*	*10h*	*00h*	*02h*	*00h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
A	*02h*	*10h*	*10h*	*22h*	*AAh*	*99h*
C	*11h*	*05h*	*01h*	*10h*	*00h*	*10h*
M = 0100h	*22h*	*10h*	*10h*	*55h*	*02h*	*01h*

Запишите в память, начиная с адреса ***0000h***, коды программы, реализующей выражения (табл. 3.65)

$HL \leftarrow BC \vee DE$

Таблица 3.65

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, C***	79	Пересылка $A \leftarrow C$
0001	***ORA E***	B3	$A \leftarrow A \vee E$
0002	***MOV L, A***	6F	Пересылка $L \leftarrow A$, младшего байта результата.
0003	***MOV A, B***	78	Пересылка $A \leftarrow B$

Окончание табл. 3.65

0004	***ORA D***	B2	***A←A∨D***
0005	***MOV H, A***	67	Пересылка ***H←A***, старшего байта результата.

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.66

Таблица 3.66

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
BC	*0111h*	*0B22h*	*0C00h*	*0F0Dh*	*0E00h*	*0F10h*
DE	*1200h*	*0100h*	*0A00h*	*1011h*	*1001h*	*1100h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
BC	*02AAh*	*1001h*	*1011h*	*2200h*	*AA01h*	*9925h*
DE	*1100h*	*05EAh*	*0122h*	*1033h*	*0010h*	*1011h*

2. Команды логического умножения.

ANA R с регистром ***A, B, C, D, E, H, L***;

ANA M с ячейкой памяти, адрес ячейки памяти ***HL***;

ANI d8 с непосредственным операндом.

Пример: Запишите в память, начиная с адреса ***0000h*** коды программы, реализующей выражение (табл. 3.67)

A←A∧C∧M∧7Fh

Таблица 3.67

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***ANA C***	A2	***A←A∧D***
0001	***LXI H,0100h***	21 00 01	***HL←0100h***
0004	***ANA M***	A6	***A←A∧M***
0005	***ANI 7Fh***	E6 7F	***A←A∧7Fh***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.68

Таблица 3.68

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
A	*01h*	*0Bh*	*0Ch*	*0Dh*	*0Eh*	*0Fh*
C	*12h*	*01h*	*0Ah*	*10h*	*10h*	*11h*
M = 0100h	*25h*	*F0h*	*10h*	*00h*	*02h*	*00h*

Окончание табл. 3.68

Вариант	**7**	**8**	**9**	**10**	**11**	**12**
A	*02h*	*10h*	*10h*	*22h*	*AAh*	*99h*
C	*11h*	*05h*	*01h*	*10h*	*00h*	*10h*
M = 0100h	*22h*	*10h*	*10h*	*55h*	*02h*	*01h*

Пример*:** Запишите в память, начиная с адреса ***0000h, коды программы, реализующей выражения (табл. 3.69)

HL←BC^DE

Таблица 3.69

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, C***	79	Пересылка ***A←C***
0001	***ANA E***	A3	***A←A^E***
0002	***MOV L, A***	6F	Пересылка ***L←A*** младшего байта результата
0003	***MOV A, D***	7A	Пересылка ***A←D***
0004	***ANA B***	A0	***A←A^B***
0005	***MOV H, A***	67	Пересылка ***H←A***, старшего байта результата.

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.70

Таблица 3.70

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
BC	*0111h*	*0B22h*	*0C00h*	*0F0Dh*	*0E00h*	*0F10h*
DE	*1200h*	*0100h*	*0A00h*	*1011h*	*1001h*	*1100h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
BC	*02AAh*	*1001h*	*1011h*	*2200h*	*AA01h*	*9925h*
DE	*1100h*	*05EAh*	*0122h*	*1033h*	*0010h*	*1011h*

3. Команды «ИСКЛЮЧАЮЩЕЕ ИЛИ».

XRA R с регистром ***A, B, C, D, E, H, L***;

XRA M с ячейкой памяти, адрес ячейки памяти ***HL***;

XRI d8 с непосредственным операндом.

Пример*:** Запишите в память, начиная с адреса ***0000h коды программы, реализующей выражение (табл. 3.71)

$A \leftarrow A \oplus C \oplus M \oplus AAh$

Таблица 3.71

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***XRA A***	AF	$A \leftarrow A \oplus A$, $A=0$
0001	***XRA E***	AB	$A \leftarrow A \oplus E$
0002	***LXI H, 900h***	21 00 09	$HL \leftarrow 0100h$
0005	***XRA M***	AE	$A \leftarrow A \oplus M$
0006	***XRI 7Fh***	EE AA	$A \leftarrow A \oplus AAh$

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.72

Таблица 3.72

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
A	*01h*	*0Bh*	*0Ch*	*0Dh*	*0Eh*	*0Fh*
C	*12h*	*01h*	*0Ah*	*10h*	*10h*	*11h*
M = 0100h	*25h*	*F0h*	*10h*	*00h*	*02h*	*00h*
Вариант	**7**	**8**	**9**	**10**	**11**	**12**
A	*02h*	*10h*	*10h*	*22h*	*AAh*	*99h*
C	*11h*	*05h*	*01h*	*10h*	*00h*	*10h*
M = 0100h	*22h*	*10h*	*10h*	*55h*	*02h*	*01h*

Пример*:** Запишите в память, начиная с адреса ***0000h, коды программы, реализующей выражения (табл. 3.73)

$HL \leftarrow (A \vee B) \wedge DE \oplus (HL \oplus C)$

Таблица 3.73

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***ORA B***	B0	$A \leftarrow A \vee B$
0001	***ANA E***	A3	$A \leftarrow A \wedge E$
0002	***MOV A, C***	79	Пересылка $A \leftarrow C$
0003	***XRA L***	A	$A \leftarrow A \oplus L$
0004	***XRA E***	AB	$A \leftarrow A \oplus E$

Окончание табл. 3.73

0005	***MOV L, A***	6F	Пересылка ***L←A***, младшего байта результата
0006	***MOV A,***	7C	Пересылка ***A←H***, старшего байта результата
0007	***XRA D***	AA	***A←A*** $\oplus$ ***D***
0008	***MOV H,***	67	Пересылка ***H←A***, старшего байта результата

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.74

Таблица 3.74

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
A	*05h*	*73h*	*21h*	*DEh*	*15h*	*D5h*
B	*66h*	*2Fh*	*EBh*	*32h*	*31h*	*05h*
DE	*6712h*	*36CDh*	*BAFFh*	*F235h*	*F235h*	*F235h*
HL исх	*3355h*	*7921h*	*68ACh*	*DBF1h*	*DB1Eh*	*D2FEh*
C	*A2h*	*36h*	*78h*	*C5h*	*1Dh*	*0Dh*

4. Команда ИНВЕРСИЯ.

CMA инверсия аккумулятора.

Пример*:** Запишите в память, начиная с адреса ***0000h коды программы, реализующей выражение (табл. 3.75)

A← NOT B AND NOT C

Таблица 3.75

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, B***	78	Пересылка ***A←B***
0001	***CMA***	2F	***A←NOT A***
0002	***MOV B, A***	47	Пересылка ***B←A***
0003	***MOV A, C***	79	Пересылка ***A←C***
0004	***CMA***	2F	***A←NOT A***
0005	***ANA B***	A0	***A←A*** $\wedge$ ***B***
0006	***CMA***	2F	***A←NOT A***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.76

Таблица 3.76

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
B	*00h*	*FFh*	*25h*	*39h*	*ABh*	*EDh*
C	*FFh*	*02h*	*56h*	*21h*	*13h*	*C3h*
A	*FFh*	*FFh*	*77h*	*39h*	*BBh*	*EFh*

Пример: Запишите в память, начиная с адреса ***0000h***, коды программы, реализующей выражения (табл. 3.77)

M3←NOT (NOT M1 ∨ NOT M2)

Адреса ячеей памяти ***M1-900h, M2-901h, M3-900h.***

Таблица 3.77

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
000	***LXI H, 900h***	21 00 09	***HL←900h*** адрес ***M***
003	***MOV A, M***	7E	***A←M1***
004	***CMA***	2F	***A← NOT M1***
005	***MOV C, A***	4F	Пересылка ***C←A***, промежуточного результата
006	***INX H***	23	***HL←HL+1***, адрес ***M2***
007	***MOV A, M***	7E	***A←M2***
008	***CMA***	2F	***A← NOT M2***
009	***ORA C***	B1	***A←A ∨ C***
00A	***CMA***	2F	***A← NOT A***
00B	***INX H***	23	***HL←HL+1***,адрес ***M3***
00C	***MOV M, A***	77	***M←A***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.78.

Таблица 3.78

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
M1	*37h*	*43h*	*09h*	*78h*	*78h*	*78h*
M2	*29h*	*5Eh*	*F6h*	*95h*	*95h*	*95h*
M3	*21h*	*42h*	*94h*	*10h*	*10h*	*10h*

Контрольные задания

1. Напишите и выполните программы реализации выражения (табл. 3.79).

HL←B∨C∨DE∨8800h

Таблица 3.79

Варианты заданий

Вариант 1	Регистры	***C***	***B***	***D***	***E***
	Данные	*00h*	*10h*	*11h*	*F0h*
Вариант 2	Регистры	***C***	***B***	***D***	***E***
	Данные	*01h*	*11h*	*A2h*	*00h*
Вариант 3	Регистры	***C***	***B***	***D***	***E***
	Данные	*00h*	*11h*	*0Bh*	*0Ah*
Вариант 4	Регистры	***C***	***B***	***D***	***E***
	Данные	*05h*	*06h*	*11h*	*A0h*
Вариант 5	Регистры	***C***	***B***	***D***	***E***
	Данные	*A0h*	*0Ah*	*00h*	*11h*
Вариант 6	Регистры	***C***	***B***	***D***	***E***
	Данные	*05h*	*19h*	*00h*	*12h*

2. Напишите и выполните программу реализации выражения (табл. 3.80)

(M3)=(M1)∨(M2)

Адреса ячеек памяти соответственно равны

M1=0100h

M2=0101h

M3=0102h

Таблица 3.80

Варианты заданий

Вариант 1	Ячейки	***M1***	***M2***
	Данные	*0Fh*	*E0h*
Вариант 2	Ячейки	***M1***	***M2***
	Данные	*11h*	*02h*
Вариант 3	Ячейки	***M1***	***M2***
	Данные	*AAh*	*55h*
Вариант 4	Ячейки	***M1***	***M2***
	Данные	*01h*	*10h*
Вариант 5	Ячейки	***M1***	***M2***
	Данные	*02h*	*F0h*

Окончание табл. 3.80

Вариант 6	Ячейки	***M1***	***M2***
	Данные	*0Bh*	*B1h*

3. Напишите и выполните программы реализации выражения (табл. 3.81)

HL←(HL∧BC∨DE)∧03FFh

Таблица 3.81

Варианты заданий

Вариант 1	Регистры	***BC***	***HL***	***DE***
	Данные	*00FFh*	*0010h*	*1000h*
Вариант 2	Регистры	***BC***	***HL***	***DE***
	Данные	*0101h*	*2125h*	*2211h*
Вариант 3	Регистры	***BC***	***HL***	***DE***
	Данные	*2233h*	*4400h*	*1000h*
Вариант 4	Регистры	***BC***	***HL***	***DE***
	Данные	*0010h*	*5000h*	*300Fh*
Вариант 5	Регистры	***BC***	***HL***	***DE***
	Данные	*0F0Fh*	*0FA0h*	*0001h*
Вариант 6	Регистры	***BC***	***HL***	***DE***
	Данные	*0F0Bh*	*1000h*	*22FFh*

4. Напишите и выполните программу реализации выражения (табл. 3.82)

(M2)←A∧(M1)∨C∧D

Адреса ячеей памяти ***M1=0100h, M2=0101h***.

Таблица 3.82

Варианты заданий

Вариант 1	Регистры	***A***	***M1***	***C***	***D***
	Данные	*00h*	*10h*	*FFh*	*0Ah*
Вариант 2	Регистры	***A***	***M1***	***C***	***D***
	Данные	*FFh*	*05h*	*15h*	*20h*
Вариант 3	Регистры	***A***	***M1***	***C***	***D***
	Данные	*0Eh*	*12h*	*22h*	*55h*
Вариант 4	Регистры	***A***	***M1***	***C***	***D***
	Данные	*A1h*	*1Ah*	*11h*	*BBh*
Вариант 5	Регистры	***A***	***M1***	***C***	***D***
	Данные	*EEh*	*11h*	*BBh*	*22h*
Вариант 6	Регистры	***A***	***M1***	***C***	***D***
	Данные	*0Fh*	*77h*	*EEh*	*B1h*

5. Напишите и выполните программу реализации выражения (табл. 3.83)

H←L ⊕ D ∨ E ∧ H ⊕ B ∨ C ⊕ 0Fh

Таблица 3.83

Варианты заданий

Вариант 1	Регистры	***HL***	***DE***	***BC***
	Данные	*0000h*	*5610h*	*FF48h*
Вариант 2	Регистры	***HL***	***DE***	***BC***
	Данные	*FF00h*	*0235h*	*1B35h*
Вариант 3	Регистры	***HL***	***DE***	***BC***
	Данные	*1001h*	*F00Fh*	*0CBAh*
Вариант 4	Регистры	***HL***	***DE***	***BC***
	Данные	*E3E0h*	*15B1h*	*B32Bh*
Вариант 5	Регистры	***HL***	***DE***	***BC***
	Данные	*2377h*	*EDAEh*	*B341h*
Вариант 6	Регистры	***HL***	***DE***	***BC***
	Данные	*0BC5h*	*1125h*	*2020h*

6. Напишите и выполните программу реализации выражения (табл. 3.84).

M←M ⊕ AAh ∧ H ⊕ B ∨ E

Таблица 3.84

Варианты заданий

Вариант 1	Регистры	***M***	***H***	***B***	***E***
	Данные	*EEh*	*B1h*	*EEh*	*B1h*
Вариант 2	Регистры	***M***	***H***	***B***	***E***
	Данные	*FFh*	*05h*	*FFh*	*0Ah*
Вариант 3	Регистры	***M***	***H***	***B***	***E***
	Данные	*10h*	*FFh*	*77h*	*EEh*
Вариант 4	Регистры	***M***	***H***	***B***	***E***
	Данные	*EEh*	*EEh*	*11h*	*B1h*
Вариант 5	Регистры	***M***	***H***	***B***	***E***
	Данные	*05h*	*B1h*	*FFh*	*0Ah*
Вариант 6	Регистры	***M***	***H***	***B***	***E***
	Данные	*FFh*	*05h*	*15h*	*20h*

7. Напишите и выполните реализации выражения (таблица 3.85).

HL←NOT DE ⊕ NOT BC ∧ NOT C ∨ NOT B

Таблица 3.85

Варианты заданий

Вариант 1	Регистры	***D***	***B***	***C***	***E***
	Данные	*FFh*	*0Ch*	*33h*	*10h*
Вариант 2	Регистры	***D***	***B***	***C***	***E***
	Данные	*EEh*	*23h*	*32h*	*F3h*
Вариант 3	Регистры	***D***	***B***	***C***	***E***
	Данные	*ACh*	*26h*	*B2h*	*41h*
Вариант 4	Регистры	***D***	***B***	***C***	***E***
	Данные	*12h*	*73h*	*00h*	*16h*
Вариант 5	Регистры	***D***	***B***	***C***	***E***
	Данные	*1Ah*	*A4h*	*53h*	*AEh*
Вариант 6	Регистры	***D***	***B***	***C***	***E***
	Данные	*F4h*	*C1h*	*22h*	*EEh*

8. Напишите и выполните реализации выражения (табл. 3.86).

M6← NOT M1 $\oplus$ ***M2*** $\vee$ ***NOT M3*** $\wedge$ ***NOT M4*** $\vee$ ***M5***

Адреса ячеек памяти

M1=900h ***M4=903h***

M2=901h ***M5=904h***

M3=902h ***M6=905h***

Таблица 3.86

Варианты заданий

Вариант 1	Регистр	***M1***	***M2***	***M3***	***M4***	***M5***
	Данные	*FFh*	*AAh*	*A4h*	*00h*	*74h*
Вариант 2	Регистр	***M1***	***M2***	***M3***	***M4***	***M5***
	Данные	*00h*	*BEh*	*4Bh*	*56h*	*00h*
Вариант 3	Регистр	***M1***	***M2***	***M3***	***M4***	***M5***
	Данные	*05h*	*26h*	*33h*	*ACh*	*FFh*
Вариант 4	Регистр	***M1***	***M2***	***M3***	***M4***	***M5***
	Данные	*34h*	*BBh*	*51h*	*C4h*	*A7h*
Вариант 5	Регистр	***M1***	***M2***	***M3***	***M4***	***M5***
	Данные	*56h*	*FAh*	*43h*	*DCh*	*00h*
Вариант 6	Регистр	***M1***	***M2***	***M3***	***M4***	***M5***
	Данные	*C3h*	*FFh*	*15h*	*ADh*	*CCh*

Лабораторная работа № 6

Команды сравнения

Теоретическое обоснование

Система команд микропроцессора ***Intel 8080*** содержит три типа команд сравнения:

1. Сравнение содержимого аккумулятора с содержимым регистра;
2. Сравнение содержимого аккумулятора с содержимым ячейки памяти;
3. Сравнение содержимого аккумулятора с непосредственным операндом.

Команды сравнения выполняются посредством внутреннего вычитания из содержимого аккумулятора, соответственно, содержимого регистра, ячейки памяти и непосредственного операнда. Содержимое аккумулятора при этом не изменяется. В результате сравнения устанавливаются биты признаков следующим образом (табл. 3.87).

Таблица 3.87

Значения флагов при выполнении команды сравнения

Результат сравнения	Признак	
	(Ноль, бит Z)	(Перенос C)
Равно	1	0
Больше	0	0
Меньше	0	1

Главным образом эти команды используются перед командами условных переходов (переход по признаку), которые будут рассматриваться в последующей лабораторной работе.

Программа работы

1. Команды сравнения содержащего регистра с содержимым аккумулятора.

CMP A команда сравнения регистра A с содержимым регистра ***A***
CMP B команда сравнения регистра A с содержимым регистра ***B***
CMP C команда сравнения регистра A с содержимым регистра ***C***
CMP D команда сравнения регистра A с содержимым регистра ***D***
CMP E команда сравнения регистра A с содержимым регистра ***E***
CMP H команда сравнения регистра A с содержимым регистра ***H***
CMP L команда сравнения регистра A с содержимым регистра ***L***

Пример: Запишите в память, начиная с адреса ***0000h***, коды программ сравнения содержимого регистров ***C*** и ***B***.

Таблица 3.88

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, C***	79	Пересылка ***(A)←(C)***
0001	***CMP B***	B8	Сравнение с ***(B), (A)-(B)***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.89. Проверьте полученные результаты.

Таблица 3.89

Варианты заданий

C	*01h*	*01h*	*FFh*	*43h*	*55h*	*03h*	*20h*
B	*01h*	*09h*	*00h*	*FFh*	*55h*	*55h*	*15h*
Ноль ***Z***	*1*	*0*	*0*	*0*	*1*	*0*	*0*
Перенос ***C***	*0*	*1*	*0*	*1*	*0*	*0*	*0*
Знак ***S***	*0*	*1*	*1*	*1*	*0*	*0*	*0*
Четность ***P***	*1*	*0*	*1*	*1*	*1*	*1*	*0*

2. Команды сравнения с памятью

CMP M сравнение содержимого регистра ***A*** с содержимым ячейки памяти, адрес которой задан в регистровой паре ***HL***.

Пример: Запишите в память, начиная с адреса ***0000h***, коды программы сравнения с содержимым ячейки памяти (табл. 3.90).

Таблица 3.90

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0100h***	21 00 01	Загрузка ***HL=0100h***
0003	***CMP M***	BE	Сравнение содержимого ***A*** и ***M***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.91. Проверьте полученные результаты.

Таблица 3.91

Пример программы

A	***21h***			***BAh***			***E9h***		
M=0100h	*00h*	*21h*	*39h*	*19h*	*FFh*	*BAh*	*E9h*	*10h*	*F5h*
Ноль ***Z***	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*0*	*0*
Перенос ***C***	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*1*
Знак ***S***	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*1*
Четность ***P***	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*0*	*0*

3. Команды сравнения с непосредственным операндом.

CPI d8 сравнение содержимого регистра ***A*** с числом, заданным во втором байте команды, ***d8*** – байт.

Пример: Запишите в память, начиная с адреса ***0000h***, колы команды сравнения с непосредственным операндом.

Таблица 3.92

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***CPI 7FH***	FE 7F	Сравнение ***A*** и 7Fh

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.93.

Таблица 3.93

Вариант задания

Вариант	**1**	**2**	**3**	**4**	**5**	**6**
A	*00h*	*80h*	*7Fh*	*B3h*	*25h*	*F7h*
Ноль ***Z***	*0*	*0*	*1*	*0*	*0*	*0*
Перенос ***C***	*1*	*0*	*0*	*0*	*1*	*0*
Знак ***S***	*1*	*0*	*0*	*0*	*1*	*0*
Четность ***P***	*1*	*0*	*1*	*0*	*1*	*1*

Контрольные задание

1. Напишите и выполните программу сравнения содержимого регистров ***H*** и ***L***. Заполните табл. 3.94.

Таблица 3.94

Вариант задания

Вариант 1	***H***	***L***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	FFh	***0Fh***				

Окончание табл. 3.94

Вариант 2	***H***	***L***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	00h	***0Ah***				
Вариант 3	***H***	***L***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	01h	***00h***				
Вариант 4	***H***	***L***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	ABh	***BAh***				
Вариант 5	***H***	***L***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	10h	***10h***				
Вариант 6	***H***	***L***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	12h	***11h***				

2. Напишите и выполните программу сравнения содержимого регистра ***A*** с содержимым ячейки памяти по адресу ***0110h***. Заполните табл. 3.95.

Таблица 3.95

Вариант задания

Вариант 1	***A***	***M***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	00h	***FFh***				
Вариант 2	***A***	***M***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	05h	***05h***				
Вариант 3	***A***	***M***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	FFh	***01h***				
Вариант 4	***A***	***M***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	12h	***15h***				
Вариант 5	***A***	***M***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	0Ah	***0Bh***				
Вариант 6	***A***	***M***	Ноль ***Z***	Перенос ***C***	Знак ***S***	Четность ***P***
	22h	***20h***				

3. Напишите и выполните программу сравнения содержимого регистра ***H*** с непосредственным операндом. Заполните табл. 3.96.

Таблица 3.96

Вариант задания

Вариант 1	Операнд	*5Ah*				
	A	*01h*	*10h*	*5Ah*	*FFh*	*5Bh*
	Ноль ***Z***					
	Перенос ***C***					

Окончание табл. 3.96

	Знак ***S***					
	Четность ***P***					
Вариант 2	Операнд	*05h*				
	A	*10h*	*1Ah*	*05h*	*01h*	*51h*
	Ноль ***Z***					
	Перенос ***C***					
	Знак ***S***					
	Четность ***P***					
Вариант 3	Операнд	*10h*				
	A	*11h*	*10h*	*00h*	*01h*	*50h*
	Ноль ***Z***					
	Перенос ***C***					
	Знак ***S***					
	Четность ***P***					
Вариант 4	Операнд	*FEh*				
	A	*01h*	*FEh*	*FFh*	*F0h*	*5Ah*
	Ноль ***Z***					
	Перенос ***C***					
	Знак ***S***					
	Четность ***P***					
Вариант 5	Операнд	*0Ah*				
	A	*00h*	*11h*	*0Ah*	*1Fh*	*0Bh*
	Ноль ***Z***					
	Перенос ***C***					
	Знак ***S***					
	Четность ***P***					
Вариант 6	Операнд	*22h*				
	A	*22h*	*11h*	*33h*	*FFh*	*00h*
	Ноль ***Z***					
	Перенос ***C***					
	Знак ***S***					
	Четность ***P***					

Лабораторная работа № 7

Команды сдвига

Теоретическое обоснование

В системе команд микропроцессора ***Intel 8080*** предусмотрены следующие команды сдвига: циклический сдвиг влево ***RLC***; циклический сдвиг вправо ***RRC***; сдвиг влево через перенос ***RAL***; сдвиг вправо через перенос ***RAR***.

Команды сдвига выполняются в регистре – накопителе, аккумуляторе над 8 – разрядными операндами. Результат заносится в аккумулятор.

Команда циклического сдвига влево ***RLC*** перемещает каждый бит байта на один разряд влево. При этом содержимое старшего разряда записывается в младший разряд и в бит переноса.

Исходное содержимое флага переноса аккумулятора

Таблица 3.97

Содержимой байта до выполнения команды циклического сдвига влево

Флаг переноса	Данные							
C	***b7***	***b6***	***b5***	***b4***	***b3***	***b2***	***b1***	***b0***

После ***RLC***

Таблица 3.98

Содержимой байта после выполнения команды циклического сдвига влево

Флаг переноса	Данные							
C	***b6***	***b5***	***b4***	***b3***	***b2***	***b1***	***b0***	***b7***

Команда циклического сдвига вправо ***RRC*** перемещает каждый бит байта на 1 разряд вправо. При этом содержимое младшего разряда записывается в старший разряд и в бит переноса.

Исходное содержимое флага переноса и аккумулятора

Таблица 3.99

Содержимой байта до выполнения команды циклического сдвига вправо

Флаг переноса	Данные							
C	***b7***	***b6***	***b5***	***b4***	***b3***	***b2***	***b1***	***b0***

После ***RRC***

Таблица 3.100

Содержимой байта после выполнения команды циклического сдвига вправо

Флаг переноса	Данные							
b0	***b0***	***b7***	***b6***	***b5***	***b4***	***b3***	***b2***	***b1***

Команда сдвига влево через перенос ***RAL*** перемещает содержимое каждого бита байта влево на 1 разряд. При этом содержимое бита переноса записывается в младший разряд, а содержимое старшего разряда заносится в бит переноса. Используя эту команду, можно реализовать операцию умножения на число кратное 2.

Исходное содержимое флага переноса и аккумулятора

Таблица 3.101

Содержимой байта до выполнения команды сдвига влево

Флаг переноса	Данные							
C	***b7***	***b6***	***b5***	***b4***	***b3***	***b2***	***b1***	***b0***

После ***RAL***

Таблица 3.102

Содержимой байта после выполнения команды сдвига влево

Флаг переноса	Данные							
C	***b6***	***b5***	***b4***	***b3***	***b2***	***b1***	***b0***	***C***

Команда сдвига вправо через перенос перемещает содержимое каждого разряда байта на один разряд. При этом в старший разряд байта записывается значение бита переноса, а в него заноситься содержимое младшего разряда байта. Используя эту команду, можно реализовать операцию деления на число кратное 2.

Исходное содержимое флага переноса и аккумулятора

Таблица 3.103

Содержимой байта до выполнения команды сдвига вправо

Флаг переноса	Данные							
C	***b7***	***b6***	***b5***	***b4***	***b3***	***b2***	***b1***	***b0***

После ***RAR***

Таблица 3.104

Содержимой байта после выполнения команды сдвига вправо

Флаг переноса	Данные							
b0	***C***	***b7***	***b6***	***b5***	***b4***	***b3***	***b2***	***b1***

Программа работы

1. Команды циклического сдвига.

RLC циклический сдвиг вправо;

RRC циклический сдвиг вправо.

Пример: Запишите в память, начиная с адреса 0000h, коды программы реализующей операцию циклического сдвига байта на 4 разряда, используя команды ***RLC*** (табл. 3.105).

Таблица 3.105

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***RLC***	07	;циклический сдвиг влево на 1 разряд
0001	***RLC***	07	;циклический сдвиг влево на 1 разряд
0002	***RLC***	07	;циклический сдвиг влево на 1 разряд
0003	***RLC***	07	;циклический сдвиг влево на 1 разряд

Выполните эту программу, предварительно задавая исходные значения в соответствии с табл. 3.106.

Таблица 3.106

Вариант задания

A исх.	*00*	*0F*	*F0*	*81*	*A5*	*67*
A рез.	*00*	*F0*	*0F*	*18*	*5A*	*76*

Пример: Запишите в память, начиная с адреса ***0000h***, коды программы, реализующей операцию объединения старших тетрад байтов, в регистрах ***B*** и ***C***, в один (табл. 3.107).

Таблица 3.107

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, C***	79	
0001	***RRC***	0F	перемещение старшей тетрады 1-го байта
0002	***RRC***	0F	
0003	***RRC***	0F	на место младшей тетрады.
0004	***RRC***	0F	
0005	***ANI OFh***	E6 0F	выделение старшей тетрады 1-го байта.
0006	***MOV C, A***	4F	

Окончание табл. 3.107

0007	***MOV A, B***	78	
0008	***ANI F0h***	E6 F0	выделение старшей тетрады 2-го байта.
000B	***ORA C***	B1	объединение двух байт в один.

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.108. Сравните результаты.

Таблица 3.108

Вариант задания

C	*72h*	*F0h*	*51h*	*19h*
B	*9Fh*	*0Fh*	*A3h*	*86h*
A рез.	*97h*	*0Fh*	*A5h*	*81h*

2. Команды сдвига через перенос.

RAL сдвиг влево через перенос;

RAR сдвиг вправо через перенос.

Пример: Запишите в память, начиная с адреса ***0000h***, коды программы, реализующей циклический сдвиг влево на 1 разряд содержимого пары регистров ***HL*** (табл. 3.109).

Таблица 3.109

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***ORA A***	B7	сброс переноса в 0
0001	***MOV A, L***	7D	сдвиг влево L на 1 разряд через перенос
0002	***RAL***	17	0 – в младший разряд L
0003	***MOV L, A***	6F	
0004	***MOV A, H***	7C	сдвиг влево H на 1 разряд через перенос
0005	***RAL***	17	с учетом переноса из L
0006	***MOV H, A***	67	старший разряд P в перенос
0007	***MOV A, L***	1D	перенос – в младший разряд L
0008	***ACI 0***	CE 00	
000A	***MOV L, A***	6F	

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.110.

Таблица 3.110

Вариант задания

HL исх.	*FFFCh*	*8002h*	*3578h*	*FFFFh*	*0000h*	*1111h*
HL рез.	*FFF9h*	*0005h*	*6AF0h*	*FFFFh*	*0000h*	*2222h*

Пример*:** Запишите в память, начиная с адреса ***0000h, коды программы, реализующей операцию умножения на 4, содержимого регистра ***C***.

Таблица 3.111

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, C***	79	
0001	***ORA A***	B7	сброс бита переноса
0002	***RAL***	17	умножение на 2
0003	***RAL***	17	умножение на 2
0004	***MOV B, A***	47	результат в B

ПРИМЕЧАНИЕ исходное значение не должно превышать 63.

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.112.

Таблица 3.112

Вариант задания

C	*00h*	*02h*	*10h*	*2Fh*	*33h*	*3Ah*
B	*00h*	*08h*	*40h*	*BCh*	*CCh*	*E8h*

Пример: Запишите в память, начиная с адреса ***0000h***, коды программы, деления на 8 содержимого регистра ***H***. Целая часть результата помещена в регистр ***D***, остаток в ***E***.

Таблица 3.113

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A, H***	7C	
0001	***ORA A***	B7	сброс переноса в 0
0002	***RAR***	IF	***A=A/2***

Окончание табл. 3.113

0003	***ORA A***	B7	сброс переноса в 0
0004	***RAR***	IF	***A=A/2***
0005	***ORA A***	B7	сброс переноса в 0
0006	***RAR***	IF	***A=A/2***
0007	***MOV D, A***	57	***D=H/8*** – целая часть
0008	***MOV A, H***	7C	
0009	***ANI 07h***	E6 07	выделение остатка результата
000B	***MOV E, A***	5F	остаток в ***E***

Выполните программу, предварительно задавая исходные значения в соответствии с табл. 3.113. Проверьте результаты.

Таблица 3.113

Вариант задания

H	*05h*	*08h*	*10h*	*35h*	*79h*	*FFh*
D	*00h*	*01h*	*02h*	*06h*	*0Fh*	*1Fh*
E	*05h*	*00h*	*00h*	*05h*	*01h*	*07h*

Контрольные задания

1. Напишите и выполните программу (табл. 3.114) объединения младших тетрад двух байтов в один. Младшую тетраду 2-го байта поместить в старшую тетраду результирующего байта.

Таблица 3.114

Вариант задания

Вариант 1	*E (1 байт)*	*D (2 байт)*
	F0h	*11h*
Вариант 2	*E (1 байт)*	*D (2 байт)*
	0Fh	*AAh*
Вариант 3	*E (1 байт)*	*D (2 байт)*
	22h	*11h*
Вариант 4	*E (1 байт)*	*D (2 байт)*
	55h	*ABh*
Вариант 5	*E (1 байт)*	*D (2 байт)*
	35h	*53h*
Вариант 6	*E (1 байт)*	*D (2 байт)*
	77h	*01h*

2. Напишите и выполните программу, реализующую операцию логического умножения 3-х битов 5, 6, 7 одного байта на 3 бита 2, 3, 4 второго байта, предварительно переместив умножаемые биты этих байт на младшие позиции, и обнулив остальные разряды (табл. 3.115).

Таблица 3.115

Вариант задания

Вариант 1	*L (1 байт)*	*H (2 байт)*
	ABh	*17h*
Вариант 2	*L (1 байт)*	*H (2 байт)*
	F0h	*1Ah*
Вариант 3	*L (1 байт)*	*H (2 байт)*
	22h	*11h*
Вариант 4	*L (1 байт)*	*H (2 байт)*
	AAh	*BBh*
Вариант 5	*L (1 байт)*	*H (2 байт)*
	77h	*01h*
Вариант 6	*L (1 байт)*	*H (2 байт)*
	35h	*53h*

3. Напишите и выполните программу циклического сдвига на 3 разряда содержимого пары регистров (табл. 3.116)**.**

Таблица 3.116

Вариант задания

Вариант 1	*DE*
	FF00h
Вариант 2	*HL*
	22F0h
Вариант 3	*BC*
	0220h
Вариант 4	*DE*
	3563h
Вариант 5	*HL*
	ABCDh
Вариант 6	*BC*
	D32Ah

4. Напишите и выполните программу деления содержимого пары регистров ***BC*** на 8. ***BC=BC/8*** (табл. 3.117)**.**

Таблица 3.117

Вариант задания

Вариант 1	*DE*
	1F40h
Вариант 2	*HL*
	1DC8h
Вариант 3	*BC*
	42B0h
Вариант 4	*DE*
	5988h
Вариант 5	*HL*
	A7C8h
Вариант 6	*BC*
	4570h

Лабораторная работа № 8

Команды безусловного и условных переходов. ввод-вывод данных

Теоретическое обоснование

В системе команд микропроцессора КР580ВМ80А предусмотрены команды изменения последовательности выполнения команд для организации циклов, обработки условий передачи управления и т.д. Существуют два типа команд перехода безусловный и условный.

При выполнении команд безусловного перехода осуществляется передача управления по адресу, заданному во втором и третьем байтах команды, либо по адресу, заданному в регистровой паре.

Команды условного перехода выполняются в том случае, если установлен или сброшен соответствующий бит признака, в противном случае команда игнорируется и выполняется следующая за ней команда.

Существуют команды условного перехода для следующих битов признаков:

- бита нуля Z;
- бита переноса C;
- бита знака S;
- бита четности P.

Для каждого бита признака предусмотрены две команды перехода переход по установленному признаку (=1) и переход по сброшенному биту признака (=0).

Соответствие выполнения команды и признаков приведены в табл. 3.118.

Таблица 3.118

Значения флагов при выполнении команд переходов

Признак	Ноль ***Z***		Перенос ***C***		Четность ***P***		Знак ***S***	
Команда	***1***	***0***	***1***	***0***	***1***	***0***	***1***	***0***
JZ	Да	---	---	---	---	---	---	---
JNZ	---	Да	---	---	---	---	---	---
JC	---	---	Да	---	---	---	---	---
JNC	---	---	---	Да	---	---	---	---
JPE	---	---	---	---	Да	---	---	---
JPO	---	---	---	---	---	Да	---	---
JM	---	---	---	---	---	---	Да	---
JP	---	---	---	---	---	---	---	Да

Программа работы

1. Команды безусловного перехода

JMP ADR16 безусловный переход по адресу указанному по 2 и 3 байтах команды.

PCHL безусловный переход по адресу, заданному в ***HL***.

Пример: Запишите в память, начиная с адреса ***0000h***, коды программы, осуществляющей инкремент аккумулятора, переход по адресу ***0100h***, записанному в регистре ***HL*** и возврат в начало программы (табл. 3.119).

Таблица 3.119

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***LXI H, 0100h***	21 00 01	Загрузка в ***HL*** адреса перехода
0003	***MVI A, 00h***	3E 00	Загрузка в ***A=0***
0005	***INR A***		Инкремент ***A***
0006	***PCHL***	E9	Переход ***(PC)←(HL)***
0100	***JMP 0000h***		Переход по адресу ***0000h***

2. Команды перехода по признаку – ноль.

JZ ADR переход, если ***Z = 1***;

JNZ ADR переход, если ***Z = 0***.

Пример: Запишите в память, начиная с адреса ***0000h***, программу заполнения ***10h*** ячеек памяти нулями (табл. 3.120).

Таблица 3.120

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MVI C, 10h***	0E 10	Загрузка ***C←10h***, длина массива
0002	***LXI H, 100h***	21 00 01	***HL←0100h***, начальный адрес массива
0005	***MVI M, 0***	36	Загрузка ***(HL) = 0***
0007	***INX H***	23	***HL←HL+1***, следующий адрес
0008	***DCR C***	0D	***C = C-1***, длина массива
0009	***JNZ 005h***	C2 05 00	Продолжать, если длина массива не равна 0

Проверьте результаты выполнения программы.

Пример*:** Запишите в память, программу заполнения ***10h ячеек памяти нулями.

Таблица 3.121

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MVI C, 10h***	0E 10	Загрузка в ***C←10h***, длина массива
0002	***LXI H, 100h***	21 00 09	Загрузка ***HL←100h***, начальный адрес массива
0005	***MOV A,M***	7E	Чтение ***A←(HL)***
0006	***CPI 0***	FE 00	***A←0***
0008	***JZ 00Dh***	CA 0D 00	Переход, если да
000B	***MVI M,0***	36 00	Нет, загрузка ***(HL)←0***
000D	***INX H***	23	***HL←HL+1***, следующий адрес
000E	***DCR C***	0D	***C = C-1***, длина массива
000F	***JNZ 005h***	C2 05 00	Продолжать, если длина массива не равна 0

Проверьте результаты выполнения программы.

3. Команды перехода по признаку С – перенос.

JC ADR переход, если ***C=1***;

JNC ADR переход, если ***C=0***.

Пример*:** Запишите в память, начиная с адреса ***0000h, программу, выполняющую сложение аккумулятора с регистром В. Если присутствует перенос, в аккумулятор записывается ***FFh***, в противном случае, в аккумулятор кладется ***00h*** (табл. 3.122).

1) ***A=FFh B=10h;***

2) ***A=00h B=10h.***

Таблица 3.122

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MVI A,FFh***	3E FF	Загрузка A ←FFh
0001	***MVI B,10h***	06 10	Загрузка B ←10h
0003	***ORA B***	B0	***A←A∨B***
0005	***JC 100h***	DA 00 01	Если перенос = 1, перейти в ячейку с адресом 0100h
0006	***JNC 200h***	D2 00 02	Если перенос = 0, перейти в ячейку с адресом 0200h
1000	***MVI A,FFh***	3E FF	Загрузка A ←FFh
2000	***MVI A,00h***	3E 00	Загрузка A ←00h

4. Команды перехода по принципу Р – четность.

JPE ADR переход если Р = 1;

JPO ADR переход если Р = 0.

Пример: Запишите в память, начиная с адреса ***0000h***, программу дополнения байта по четности в старшем разряде. Исходное число в регистре ***C*** (табл. 3.123).

Таблица 3.123

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MOV A,C***	79	Пересылка ***A←C*** исходного байта
0001	***ANI 7Fh***	E6 7F	Обнуление старшего разряда
0003	***ORA A***	B7	***A*** ∨ ***A*** – установка бита четности
0004	***JPO***	E2 09 00	Переход, если исходный байт четный
0007	***ORI 80h***	F6 80	Дополнить до четности
0009	***MOV C,A***	4F	Результат

Выполните программу, задавая исходные значения в соответствии с табл. 3.124. Проверьте результат.

Таблица 3.124

Варианты заданий

C исходное	***00***	***FF***	***B6***	***80***	***CD***	***75***
C результат	***00***	***FF***	***36***	***00***	***4D***	***F5***

5. Команды перехода по признаку – знак.

JM ADR переход, если ***S= 1***.

JP ADR переход, если ***S= 0***.

8.2.6. Команды ввода/вывода.

IN adr8 считывание данных с порта;

OUT adr8 вывод данных в порт.

Пример: Запиши в память, начиная с адреса ***0000h***, код программы, которая, при нажатой кнопке, инкрементирует переменную и выводит ее значение в порт со светодиодами. Адрес порта с переключателями ***12h***, адрес порта со светодиодами ***10h*** (табл. 3.125).

Таблица 3.125

Пример кода программы

Адрес	Команда	Машинный код	Комментарий
0000	***MVI A, 00H***	3E 00	Загрузка ***A←00h***
0002	***MVI C, FFH***	0E FF	Загрузка ***C←FFh***
0004	***MVI D, FFH***	16 FF	Загрузка ***D←FFh***
0006	***MOV B,A***	47	Загрузка ***B←A***
0007	***IN 12H***	DB 12	Считывание порта ***12H***
0008	***DCR A***	3D	***A←A-1***
0009	***JC 07H***	DA 07 00	Переход, если ***C=1***
000C	***MOV A,B***	78	Загрузка ***A←B***
000D	***INR A***	3C	***A←A+1***
000E	***MOV B,A***	47	Загрузка ***B←A***
000F	***OUT 10H***	D3 10	Вывод в порт ***10h***
0010	***CALL 200H***	CD 00 02	Вызов подпрограммы по адресу ***0200h***
0013	***JMP 07H***	C3 07 00	Переход на адрес ***07h***
2000	***DCR C***	0D	***C←C-1***
2001	***JNC 200H***	D2 00 02	Переход, если ***C=0***
2004	***DCR D***	15	***D←D-1***
2005	***JNC 204H***	D2 04 02	Переход, если ***C=0***
2008	***RET***	C9	Возврат из подпрограммы

Контрольные задания

1. Организовать на выводах порта эффект бегущей «1».
2. Организовать на выводах порта эффект бегущего «0».
3. Организовать мерцание светодиодов.
4. Организовать смену частоты мерцания светодиодов нажатием кнопки. При отпущенной кнопке одна частота мерцания, при нажатой кнопке другая частота.
5. Организовать смену направления бегущей «1» нажатием кнопки. При отпущенной кнопке «1» бегает в одну сторону, а при нажатой кнопке, в другую.
6. Организовать смену направления бегущего «0» нажатием кнопки. При отпущенной кнопке «0» бегает в одну сторону, а при нажатой кнопке, в другую.
7. Организовать эффект светофора. Старшие 3 бита порта – красный цвет, средние 2 бита – желтый и младшие 3 бита – зеленый.

8. Организовать эффект эквалайзера. Бегущая «1» туда и обратно.
9. Организовать эффект эквалайзера. Бегущий «0» туда и обратно.
10. Организовать инкремент и декремент по нажатию кнопки. По нажатию одной кнопки производится инкремент переменной и вывод ее в порт, по нажатию другой декремент и вывод в порт.
11. Организовать на выводах порта эффект бегущей «1». Длина пути бегущей «1» определяется нажатием кнопки. Какая кнопка нажата, до туда и бежит «1».
12. Организовать на выводах порта эффект бегущего «0». Длина пути бегущего «0» определяется нажатием кнопки. Какая кнопка нажата, до туда и бежит «0».

Все временные задержки организовать с помощью подпрограммы.

Приложение 1

Форматы чисел. Позиционная система счисления

В позиционных системах счисления, числовое значение цифры зависит от ее позиции в последовательности цифр, которыми представлено число.

В общем случае последовательность цифр имеет следующий вид:

$x = a_{n-1}a_{n-2}.....a_1a_0$,

где $0 \le a_k \le b-1$, при этом ***b*** – основание системы счисления.

На сегодняшний день наиболее распространёнными являются системы счисления с основанием ***b*** = 2, 10, 16.

В случае если ***b*** >10, то для записи числа в соответствующей системе счисления используются специальные обозначения, соответствующие цифрам 10, 11 и т.д. Например в шестнадцатеричной системе счисления такими символами являются буквы:

$A \to 10$

$B \to 11$

$C \to 12$

$D \to 13$

$E \to 14$

$F \to 15.$

Например:

$535_{10} - десятичная\,система$

$0001\,0111_2 \to 0001\,0111b \to bx0001\,0111 - двоичная\,система$

$537_8 - восьмеричная\,система$

$8CA_{16} \to 8CAh \to 0x8CA - шестнадцатеричная\,система$

Положительное целое число ***x*** в ***b***-ричной системе счисления представляется в виде конечной линейной комбинации степеней числа ***b***.

$$x = \sum_{k=0}^{n-1} a_k b^k$$

где $0 \le a_k \le b-1$.

Например:

$539_{10} = 5 \cdot 10^2 + 3 \cdot 10^1 + 9 \cdot 10^0 = 500 + 30 + 9 = 539$

$0001\,0111_2 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 4 + 2 + 1 = 23$

$537_8 = 5 \cdot 8^2 + 3 \cdot 8^1 + 7 \cdot 8^0 = 320 + 24 + 7 = 351$

$8CA_{16} = 8 \cdot 16^2 + 12 \cdot 16^1 + 10 \cdot 16^0 = 2048 + 192 + 10 = 2250$

Микропроцессорная техника оперирует двоичными цифрами. Независимо от изображения чисел и цифр в программе пользователя, микропроцессор всегда преобразует их в последовательность двоичных цифр: 0 и 1.

Используют следующие сокращения для обозначения форматов двоичных чисел.

Бит – двоичная цифра, имеющая два значения (0, 1).

С помощью двух бит можно представить четыре числа (00, 01, 10, 11).

С помощью ***n*** бит можно представить 2^n чисел.

Тетрада – это комбинация из четырех бит, она описывает 16 комбинаций чисел, т.е. совпадает с числом цифр в 16-теричной системе счисления. Т.о. любую комбинацию тетрады мы можем записать либо 4 битами, либо одной цифрой или буквой 16-ной системы счисления (таблица П 1.1)

Таблица П 1.1

Запись чисел от 0 до 15 в различных системах счисления

10-теричная	двоичная	16-теричная
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D

Окончание табл. П 1.1

14	1110	E
15	1111	F

Байт – это группа из 8 бит.

Байт позволяет описать 256 различных чисел. Биты в байте номеруются справа, налево начиная с нуля.

Самое младшее число в 16-теричной системе счисления, формата байт, записывается в виде 00_{16}, а самое старшее число записывается в виде FF_{16}.

Например:

$0001\,0111_2 = 17_{16}$

$0001\,1100_2 = 1Ch$

$1010\,1111_2 = 0AFh$

Слово – это комбинация из 16 бит или 2 байт. Слово содержит:

$2^{16} = 65536$ комбинаций.

Для краткой записи больших степеней числа 2 (что используется при характеристике объема памяти) величину 2^{10} обозначают буквой К, которая читается «килобайт» (Кбайт), число 2^{20} обозначают М, которая читается «мегабайт» (Мбайт), число 2^{30} обозначают Г, которая читается «гигабайт» (Гбайт).

Перевод десятичного числа в другую систему счисления

Для того чтобы перевести целое десятичное число ***x*** в систему счисления с основанием ***b*** необходимо исходное число ***x*** последовательно делить на основание ***b*** до получения частного равного нулю. Искомым представлением числа является последовательность остатков от операций деления, причем первый остаток дает младшую цифру искомого числа.

Например:

Перевод числа 23 в двоичную систему:

$23/2 = 11(остаток 1)$

$11/2 = 5(остаток 1)$

$5/2 = 2(остаток 1)$

$2/2 = 1(остаток 0)$

$1/2 = 0(остаток 1)$

$23_{10} = 0001\,0111_2$

Перевод числа 2250 в шестнадцатеричную систему:

$$2250/16 = 140(остаток 10)$$
$$140/16 = 8(остаток 12)$$
$$8/16 = 0(остаток 8)$$
$$2250_{10} = 8CA_{16}$$

Знаковые двоичные числа

Для изображения знака числа при кодировке знаковых чисел используется старший двоичный разряд. Для отображения положительного числа старший двоичный разряд полагают равным нулю, для отрицательных чисел этот разряд равен 1.

При записи знаковых чисел всегда задают его формат, т.е. число бит, выделяемых для записи знакового числа. Обычно этот формат состоит из восьми бит. В прямой кодировке старший разряд используется как знаковый разряд, а оставшиеся младшие разряды используются для обозначения модуля числа.

Например:

$$0001\,0111_2 = 23_{10}$$
$$1001\,0111_2 = 175_{10} - без\ знаковое$$
$$1001\,0111_2 = -23_{10} - знаковое$$

Максимальное положительное число в прямой кодировке в двоичном представлении записывается как $01111111 = 127_{10}$.

Минимальное положительное число в прямой кодировке в двоичном представлении записывается как $11111111 = -127_{10}$

Прямой код в МП технике использовался только на ранних этапах развития. В настоящее время для записи знаковых чисел используют дополнительный код. В дополнительном коде кодировка всех положительных чисел совпадает с прямой кодировкой, а для кодировки отрицательных чисел ввели специальное правило. Чтобы получить дополнительный код отрицательного числа, состоящего из ***n*** бит, необходимо записать ***n***-битный модуль этого числа, а затем проинвертировать все биты и к полученному числу арифметически прибавить единицу.

Чтобы получить модуль отрицательного числа, записанного в дополнительном коде, необходимо выполнить следующие операции:

1. Записать n-битный код отрицательного числа;
2. Проинвертировать все биты;
3. Арифметически прибавить единицу.

Например:

$0001\,0111_2 = 23_{10}$

Инверсия $-1110\,1000_2$

Прибавление $1-1110\,1001_2 = -23_{10}$

$1110\,1001_2 = 233_{10}$ – *без знаковое*

$1110\,1001_2 = -23_{10}$ – *знаковое*

При этом $11111111_2 = -1_{10}$.

Числа с фиксированной точкой

Запись дробной части произвольного числа ***x*** в ***b***-ричной позиционной системе с фиксированной точкой основывается на представлении этого числа в виде многочлена

$$x = a_{-1}b^{-1} + a_{-2}b^{-2} + a_{-3}b^{-3} + ... + a_{-m}b^{-m}.$$

При переводе дробной части чисел из десятичной системы счисления в систему с основанием ***b*** обычно используют следующий алгоритм:

1. Дробная часть умножается на ***b***, после чего целая часть отбрасывается;
2. Вновь полученная дробная часть умножается на ***b*** и т.д.
3. Процедура продолжается до тех пор, пока дробная часть не станет равной нулю.
4. Целые части выписываются после запятой в порядке их получения (с лева на право). Т.е. если целая часть равно 0, то выписываем «0». Если целая часть равна 1, то выписываем «1».
5. Результатом может быть либо конечная, либо периодическая дробь в системе счисления с основанием ***b***. Поэтому, когда дробь является периодической, приходится обрывать умножение на каком-либо шаге и довольствоваться приближенной записью исходного числа в системе с основанием ***b***.

Пример 1: Запишем в двоичном коде число ***521.4375.***

Перевод целой части числа:

$521 / 2 = 260(остаток 1)$

$260 / 2 = 130(остаток 0)$

$130 / 2 = 65(остаток 0)$

$65 / 2 = 32(остаток 1)$

$32 / 2 = 16(остаток 0)$

$16 / 2 = 8(остаток 0)$

$8 / 2 = 4(остаток 0)$

$4 / 2 = 2(остаток 0)$

$2 / 2 = 1(остаток 0)$

$1 / 2 = 0(остаток 1)$

$521_{10} = 10\ 0000\ 1001_2$

Перевод дробной части числа:

$0.4375 \cdot 2 = 0.875$ *записываем* 0

$0.875 \cdot 2 = 1.75$ *записываем* 1

$0.75 \cdot 2 = 1.5$ *записываем* 1

$0.5 \cdot 2 = 1$ *записываем* 1

$.4375_{10} = .0111_2$

Таким образом, полная запись числа ***521.4375*** в двоичном коде будет иметь следующий вид: $0010\ 0000\ 1001.0111_2 = 209.7_{16}$. Произведем проверку

$0010\ 0000\ 1001.0111_2$

$x = 1 \cdot 2^9 + 1 \cdot 2^3 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} =$

$= 512 + 8 + 1 + 0.25 + 0.125 + 0.0625 = 521.4375$

Например 2: Запишем в двоичном коде число ***1.94.***

Перевод целой части числа:

$1 / 2 = 0(остаток 1)$

$1_{10} = 0001_2$

Перевод дробной части числа:

$0.94 \cdot 2 = 1.88$ *записываем* 1

$0.88 \cdot 2 = 1.76$ *записываем* 1

$0.76 \cdot 2 = 1.52$ *записываем* 1

$0.52 \cdot 2 = 1.04$ *записываем* 1

$0.04 \cdot 2 = 0.08$ *записываем* 0

$0.08 \cdot 2 = 0.16$ *записываем* 0

и т.д.

$.94_{10} = .111100_2$

Таким образом, полная запись числа ***1.94*** в двоичном коде будет иметь следующий вид: $0001.1111_2 = 1.F_{16}$. Произведем проверку

0001.1111_2

$x = 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} =$

$= 1 + 0.5 + 0.25 + 0.125 + 0.0625 = 1.9375$.

Числа с плавающей точкой

Не редко возникает необходимость обрабатывать очень большие числа (например, расстояние между звёздами) или наоборот очень маленькие числа (например, размеры атомов или электронов). При таких вычислениях приходится использовать числа с большой разрядностью. В то же время знать расстояние между звёздами с точностью до миллиметра нет необходимости. Числа с фиксированной точкой неэффективны в вычислениях с такими величинами.

Для записи таких чисел в десятичной арифметике используется алгебраическая форма. Число записывают в виде ***мантиссы***, умноженной на 10 в степени, которая отображает ***порядок числа***.

Например:

$0.2 \cdot 10^{25}$ *или* $0.16 \cdot 10^{-30}$

Такая форму записи используют и для двоичных чисел. Она называется **запись числа с плавающей точкой**. Значение мантиссы не может превышать единицы и после запятой в мантиссе не может записываться ноль.

Для представления чисел с одинарной точностью (float) и с двойной точностью (double) существует стандарт IEEE 754. Для того, чтобы записать число в формате с плавающей точкой одинарной точности тре-

буется 32-битовое слово. Для записи чисел с двойной точностью требуется 64-битовое слово. Обычны числа хранятся в нескольких соседних ячейках памяти МП. На рис. 3 приведены форматы числа с плавающей точкой одинарной точности и числа с плавающей точкой удвоенной точности.

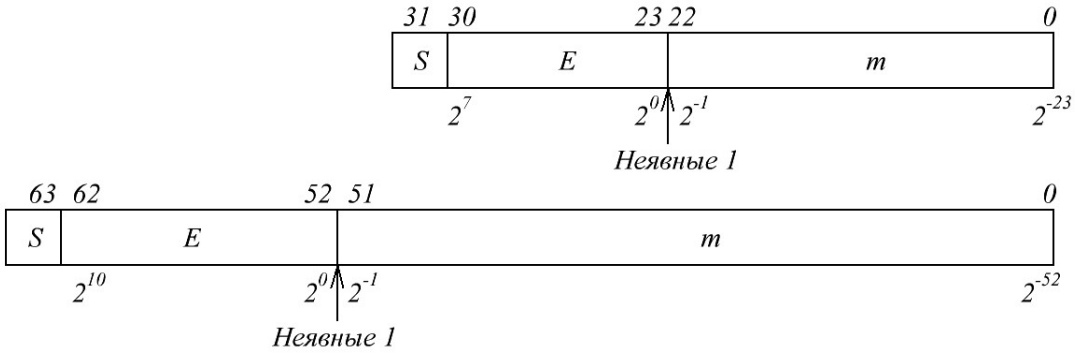

Рис. П1. Формат числа с плавающей точкой

На рисунке буквой ***S*** обозначается знак числа. Если S равен 0 - это положительное число, 1 - число отрицательное.

Буквой ***E*** обозначен смещённый порядок числа. Смещение необходимо для того, чтобы не вводить в число еще один знак. Смещённый порядок всегда положителен. В числах с плавающей точкой одинарной точности для порядка выделено восемь бит. В числах с плавающей точкой двойной точности для смещённого порядка отводится 11 бит. Значение смещения для чисел с плавающей точкой одинарной точности равно 127, а для двойной точности смещение равно 1023. В десятичной системе исчисления в мантиссе после запятой могут содержаться цифры от 1 до 9, а в двоичной системе исчисления только 1. В связи с этим, в числе с плавающей точкой, после двоичной запятой, для хранения единицы отдельный бит не выделяется. Единица используется в качестве двоичной запятой. Кроме того, принято, что мантисса всегда больше 1. То есть значение мантиссы лежит в диапазоне от 1 до 2.

Расчет значения числа по его двоичному коду с плавающей точкой можно определить из соотношения:

$$x = (-1)^S \cdot (1.m) \cdot 2^E$$

Свойства чисел с плавающей точкой:

1. В нормализованном виде любое отличное от нуля число представимо в единственном виде. Недостатком такой записи является тот факт, что невозможно представить число 0.

2. Так как старший бит двоичного числа, записанного в нормализованной форме, всегда равен 1, его можно опустить. Это используется в стандарте IEEE 754.

3. В отличие от целочисленных стандартов (например, integer), имеющих равномерное распределение на всем множестве значений, числа с плавающей точкой (double, например) имеют квазиравномерное распределение.

4. Вследствие свойства 3, числа с плавающей точкой имеют постоянную относительную погрешность (в отличие от целочисленных, которые имеют постоянную абсолютную погрешность).

5. Очевидно, не все действительные числа возможно представить в виде числа с плавающей точкой.

6. Точно в таком формате представимы только числа, являющиеся суммой некоторых обратных степеней двойки (не ниже -53). Остальные числа попадают в некоторый диапазон и округляются до ближайшей его границы. Таким образом, абсолютная погрешность составляет половину величины младшего бита.

7. В формате double представимы числа от 2.3×10^{-308} до 1.7×10^{-307}.

Приведем примеры расчета чисел по двоичному коду.

Например: Вычислить четырехбайтное число с плавающей точкой:
11000001 01001000 00000000 00000000

- Так как знаковый бит равен 1 то число отрицательное.
- Экспонента, равная 10000010, соответствует числу 130 в десятичном виде. После вычитания из 130 числа 127, получим число 3.
- Далее необходимо записать мантиссу:

1.100 1000 0000 0000 0000 0000

- Таким образом, десятичное число: $1100.1_2 = 12.5_{10}$.

Также можно рассчитать значение числа по формуле указанной выше:

$$x = (-1)^1 \cdot (1.100\,1000) \cdot 2^{130-127} = -1.5625 \cdot 2^3 = -12.5.$$

Задания для самостоятельного контроля

1. Перевести числа в десятичную систему счисления (с пояснениями):

0001 0111*b*

1011 0001*b*

0*BAh*

8*CAh*

2. Перевести числа в двоичную и шестнадцатеричную системы счисления:

565_{10}

53_{10}

3. Записать отрицательные числа в дополнительном коде:

-50_{10}

-15_{10}

-1_{10}

4. Записать числа в двоичной системе счисления с фиксированной точкой:

127.4375_{10}

12.431_{10}

125.125_{10}

5. Перевести двоичные числа с плавающей точкой в десятичную систему счисления:

11000000 00110000 00000000 00000000

01000000 10111100 00000000 00000000

Приложение 2

Система команд микропроцессора Intel 8080

Команда	Код	Описание
ADD A	*87h*	Логическое И ***A*** и ***A***
ADD B	*80h*	Логическое И ***B*** и ***A***
ADD C	*81h*	Логическое И ***C*** и ***A***
ADD D	*82h*	Логическое И ***D*** и ***A***
ADD E	*83h*	Логическое И ***E*** и ***A***
ADD H	*84h*	Логическое И ***H*** и ***A***
ADD L	*85h*	Логическое И ***L*** и ***A***
ADD M	*86h*	Логическое ***M**(**HL**) + **A***
ADI d8	*C6h*	Логическое *d8* + ***A***
ADC A	*8Fh*	Логическое И ***A*** и ***A*** и ***C***
ADC B	*88h*	Логическое И ***B*** и ***A*** и ***C***
ADC C	*89h*	Логическое И ***C*** и ***A*** и ***C***
ADC D	*8Ah*	Логическое И ***D*** и ***A*** и ***C***
ADC E	*8Bh*	Логическое И ***E*** и ***A*** и ***C***
ADC H	*8Ch*	Логическое И ***H*** и ***A*** и ***C***
ADC L	*8Dh*	Логическое И ***L*** и ***A*** и ***C***
ADC M	*8Eh*	Логическое И ***M**(**HL)*** и ***A*** и ***C***
ACI d8	*CEh*	Логическое И ***d8*** и ***A*** и ***C***
ANA A	*A7h*	Проверка ***A***
ANA B	*A0h*	Логическое И ***B*** с ***A***
ANA C	*A1h*	Логическое И ***C*** с ***A***
ANA D	*A2h*	Логическое И ***D*** с ***A***
ANA E	*A3h*	Логическое И ***E*** с ***A***
ANA H	*A4h*	Логическое И ***H*** с ***A***
ANA L	*A5h*	Логическое И ***L*** с ***A***
ANA M	*A6h*	Логическое И Loc(***HL***) с ***A***
ANI d8	*E6h*	Логическое И непосредственные данные с ***A***
CALL a16	*CDh*	Передать управление подпрограмме по адресу a16
CZ a16	*CCh*	Вызвать подпрограмму по адресу a16, если нуль
CNZ a16	*C4h*	Вызвать подпрограмму по адресу a16, если не нуль
CP a16	*F4h*	Вызвать подпрограмму по адресу a16, если плюс
CM a16	*FCh*	Вызвать подпрограмму по адресу a16, если минус
CC a16	*DCh*	Вызвать подпрограмму по адресу a16, если перенос
CNC a16	*D4h*	Вызвать подпрограмму по адресу a16, если нет переноса

CPE a16	*ECh*	Вызвать подпрограмму по адресу a16, если чётно
CPO a16	*E4h*	Вызвать подпрограмму по адресу a16, если нечётно
CMA	*2Fh*	Инвертировать ***A***
CMC	*3Fh*	Инвертировать перенос
CMP A	*BFh*	Установить флаг ***FZ***
CMP B	*B8h*	Сравнить ***A*** с ***B***
CMP C	*B9h*	Сравнить ***A*** с ***C***
CMP D	*Bah*	Сравнить ***A*** с ***D***
CMP E	*BBh*	Сравнить ***A*** с ***E***
CMP H	*BCh*	Сравнить ***A*** с ***H***
CMP L	*BDh*	Сравнить ***A*** с ***L***
CMP M	*BEh*	Сравнить ***A*** с M*(***HL***)
CPI d8	*FEh*	Сравнить ***A*** с непосредственными данными, заданными в команде
DAA	*27h*	Десятичная коррекция аккумулятора (совершенно бесполезная команда. Я так и ни разу ей и не воспользовался:)
DAD B	*09h*	Сложить ***BC*** с ***HL***
DAD D	*19h*	Сложить ***DE*** с ***HL***
DAD H	*29h*	Сложить ***HL*** с ***HL*** (удвоение ***HL***)
DAD SP	*39h*	Сложить ***SP*** с ***HL***
DCR A	*3Dh*	Декремент ***A***
DCR B	*05h*	Декремент ***B***
DCR C	*0Dh*	Декремент ***C***
DCR D	*15h*	Декремент ***D***
DCR E	*1Dh*	Декремент ***E***
DCR H	*25h*	Декремент ***H***
DCR L	*2Dh*	Декремент ***L***
DCR M	*35h*	Декремент M*(***HL***)
DCX B	*0Bh*	Декремент ***BC***
DCX D	*1Bh*	Декремент ***DE***
DCX H	*2Bh*	Декремент ***HL***
DCX SP	*3Bh*	Декремент ***SP***
DI	*F3h*	Запретить прерывания
EI	*FBh*	Разрешить прерывания
HLT	*76h*	Останов процессора
IN pp	*DBh*	Ввести данные из порта pp
INR A	*3Ch*	Инкрементировать ***A***
INR B	*04h*	Инкрементировать ***B***
INR C	*0Ch*	Инкрементировать ***C***
INR D	*14h*	Инкрементировать ***D***
INR E	*1Ch*	Инкрементировать ***E***

INR H	*24h*	Инкрементировать ***H***
INR L	*2Ch*	Инкрементировать ***L***
INR M	*34h*	Инкрементировать содержимое ***M**** (***HL***)
INX B	*03h*	Инкрементировать ***BC***
INX D	*13h*	Инкрементировать ***DE***
INX H	*23h*	Инкрементировать ***HL***
INX SP	*33h*	Инкрементировать ***SP***
JMP a16	*C3h*	Перейти по адресу a16
JZ a16	*CAh*	Перейти по адресу a16, если нуль
JNZ a16	*C2h*	Перейти по адресу a16, если не нуль
JP a16	*F2h*	Перейти по адресу a16, если плюс
JM a16	*FAh*	Перейти по адресу a16, если минус
JC a16	*Dah*	Перейти по адресу a16, если перенос
JNC a16	*D2h*	Перейти по адресу a16, если нет переноса
JPE a16	*EAh*	Перейти по адресу a16, если паритет чётный
JPO a16	*E2h*	Перейти по адресу a16, если паритет нечётный
LDA a16	*3Ah*	Загрузить ***A*** из ячейки с адресом a16
LDAX B	*0Ah*	Загрузить ***A*** из ячейки с адресом ***M**** (***BC***)
LDAX D	*1Ah*	Загрузить ***A*** из ячейки с адресом ***M**** (***DE***)
LHLD a16	*2Ah*	Загрузить в ***HL*** содержимое ячейки с адресом a16
LXI B,d16	*01h*	Загрузить в ***BC*** непосредственные данные d16
LXI H,d16	*21h*	Загрузить в ***HL*** непосредственные данные d16
LXI SP,d16	*31h*	Загрузить в ***SP*** непосредственные данные d16
MOV A,A	*7Fh*	Переслать из ***A*** в ***A***
MOV A,B	*78h*	Переслать из ***B*** в ***A***
MOV A,C	*79h*	Переслать из ***C*** в ***A***
MOV A,D	*7Ah*	Переслать из ***D*** в ***A***
MOV A,E	*7Bh*	Переслать из ***E*** в ***A***
MOV A,H	*7Ch*	Переслать из ***H*** в ***A***
MOV A,L	*7Dh*	Переслать из ***L*** в ***A***
MOV A,M	*7Eh*	Переслать из ***M**** (***HL***) в ***A***
MOV B,A	*47h*	Переслать из ***A*** в ***B***
MOV B,B	*40h*	Переслать из ***B*** в ***B***
MOV B,C	*41h*	Переслать из ***C*** в ***B***
MOV B,D	*42h*	Переслать из ***D*** в ***B***
MOV B,E	*43h*	Переслать из ***E*** в ***B***
MOV B,H	*44h*	Переслать из ***H*** в ***B***
MOV B,L	*45h*	Переслать из ***L*** в ***B***
MOV B,M	*46h*	Переслать из ***M**** (***HL***) в ***B***
MOV C,A	*4Fh*	Переслать из ***A*** в ***C***
MOV C,B	*48h*	Переслать из ***B*** в ***C***
MOV C,C	*49h*	Переслать из ***C*** в ***C***

MOV C,D	*4Ah*	Переслать из ***D*** в ***C***
MOV C,E	*4Bh*	Переслать из ***E*** в ***C***
MOV C,H	*4Ch*	Переслать из ***H*** в ***C***
MOV C,L	*4Dh*	Переслать из ***L*** в ***C***
MOV C,M	*4Eh*	Переслать из ***M**** (***HL***) в ***C***
MOV D,A	*57h*	Переслать из ***A*** в ***D***
MOV D,B	*50h*	Переслать из ***B*** в ***D***
MOV D,C	*51h*	Переслать из ***C*** в ***D***
MOV D,D	*52h*	Переслать из ***D*** в ***D***
MOV D,E	*53h*	Переслать из ***E*** в ***D***
MOV D,H	*54h*	Переслать из ***H*** в ***D***
MOV D,L	*55h*	Переслать из ***L*** в ***D***
MOV D,M	*56h*	Переслать из ***M**** (***HL***) в ***D***
MOV E,A	*5Fh*	Переслать из ***A*** в ***E***
MOV E,B	*58h*	Переслать из ***B*** в ***E***
MOV E,C	*59h*	Переслать из ***C*** в ***E***
MOV E,D	*5Ah*	Переслать из ***D*** в ***E***
MOV E,E	*5Bh*	Переслать из ***E*** в ***E***
MOV E,H	*5Ch*	Переслать из ***H*** в ***E***
MOV E,L	*5Dh*	Переслать из ***L*** в ***E***
MOV E,M	*5Eh*	Переслать из ***M**** (***HL***) в ***E***
MOV H,A	*67h*	Переслать из ***A*** в ***H***
MOV H,B	*60h*	Переслать из ***B*** в ***H***
MOV H,C	*61h*	Переслать из ***C*** в ***H***
MOV H,D	*62h*	Переслать из ***D*** в ***H***
MOV H,E	*63h*	Переслать из ***E*** в ***H***
MOV H,H	*64h*	Переслать из ***H*** в ***H***
MOV H,L	*65h*	Переслать из ***L*** в ***H***
MOV H,M	*66h*	Переслать из ***M**** (***HL***) в ***H***
MOV L,A	*6Fh*	Переслать из ***A*** в ***L***
MOV L,B	*68h*	Переслать из ***B*** в ***L***
MOV L,C	*69h*	Переслать из ***C*** в ***L***
MOV L,D	*6Ah*	Переслать из ***D*** в ***L***
MOV L,E	*6Bh*	Переслать из ***E*** в ***L***
MOV L,H	*6Ch*	Переслать из ***H*** в ***L***
MOV L,L	*6Dh*	Переслать из ***L*** в ***L***
MOV L,M	*6Eh*	Переслать из ***M**** (***HL***) в ***L***
MOV M,A	*77h*	Переслать из ***A*** в ***M***
MOV M,B	*70h*	Переслать из ***B*** в ***M***
MOV M,C	*71h*	Переслать из ***C*** в ***M***
MOV M,D	*72h*	Переслать из ***D*** в ***M***
MOV M,E	*73h*	Переслать из ***E*** в ***M***

MOV M,H	*74h*	Переслать из ***H*** в ***M***
MOV M,L	*75h*	Переслать из L в ***M***
MVI A,d8	*3Eh*	Переслать ***d8*** в ***A***
MVI B,d8	*06h*	Переслать ***d8*** в ***B***
MVI C,d8	*0Eh*	Переслать ***d8*** в ***C***
MVI D,d8	*16h*	Переслать ***d8*** в ***D***
MVI E,d8	*1Eh*	Переслать ***d8*** в ***E***
MVI H,d8	*26h*	Переслать ***d8*** в ***H***
MVI L,d8	*2Eh*	Переслать ***d8*** в ***L***
MVI M,d8	*36h*	Переслать ***d8*** в ***M**** (***HL***)
NOP	*00h*	Нет операции
ORA A	*B7h*	Проверить ***A*** и сбросить перенос
ORA B	*B0h*	Логическая операция ***A*** ИЛИ ***B***
ORA C	*B1h*	Логическая операция ***A*** ИЛИ ***C***
ORA D	*B2h*	Логическая операция ***A*** ИЛИ ***D***
ORA E	*B3h*	Логическая операция ***A*** ИЛИ ***E***
ORA H	*B4h*	Логическая операция ***A*** ИЛИ ***H***
ORA L	*B5h*	Логическая операция ***A*** ИЛИ ***L***
ORA M	*B6h*	Логическая операция ***A*** ИЛИ ***M***
ORI d8	*F6h*	Логическая операция ***A*** ИЛИ ***d8***
OUT pp	*D3h*	Записать ***A*** в порт ***pp***
PCHL	*E9h*	Передать управление по адресу в ***HL***
POP B	*C1h*	Извлечь слово из стека в ***BC***
POP D	*D1h*	Извлечь слово из стека в ***DE***
POP H	*E1h*	Извлечь слово из стека в ***HL***
POP PSW	*F1h*	Извлечь слово из стека в ***PSW***
PUSH B	*C5h*	Поместить в стек содержимое ***BC***
PUSH D	*D5h*	Поместить в стек содержимое ***DE***
PUSH H	*E5h*	Поместить в стек содержимое ***HL***
PUSH PSW	*F5h*	Поместить в стек содержимое ***PSW***
RAL	*17h*	Циклический сдвиг ***C*** + ***A*** влево
RAR	*1Fh*	Циклический сдвиг ***C*** + ***A*** вправо
RLC	*07h*	Сдвинуть ***A*** влево на один разряд с переносом
RRC	*0Fh*	Сдвинуть ***A*** вправо на один разряд с переносом
RIM	*20h*	Считать маску прерывания (только в 8085)
RET	*C9h*	Возврат из подпрограммы
RZ	*C8h*	Возврат из подпрограммы, если ***Z***=0
RNZ	*C0h*	Возврат из подпрограммы, если ***Z***=1
RP	*F0h*	Возврат из подпрограммы, если ***P***=1
RM	*F8h*	Возврат из подпрограммы, если ***P***=0
RC	*D8h*	Возврат из подпрограммы, если ***C***=1
RNC	*D0h*	Возврат из подпрограммы, если ***C***=0

RPE	*E8h*	Возврат из подпрограммы, если паритет чётный
RPO	*E0h*	Возврат из подпрограммы, если паритет нечётный
RST 0	*C7h*	Запуск программы с адреса *0*
RST 1	*CFh*	Запуск программы с адреса *8h*
RST 2	*D7h*	Запуск программы с адреса *10h*
RST 3	*DFh*	Запуск программы с адреса *18h*
RST 4	*E7h*	Запуск программы с адреса *20h*
RST 5	*EFh*	Запуск программы с адреса *28h*
RST 6	*F7h*	Запуск программы с адреса *30h*
RST 7	*FFh*	Запуск программы с адреса *38h*
SIM	*30h*	Установить маску прерывания (только в 8085)
SPHL	*F9h*	Загрузить ***SP*** из ***HL***
SHLD a16	*22h*	Записать ***HL*** по адресу a16
STA a16	*32h*	Записать ***A*** по адресу a16
STAX B	*02h*	Записать ***A*** по адресу ***M**** (***BC***)
STAX D	*12h*	Записать ***A*** по адресу ***M**** (***DE***)
STC	*37h*	Установить флаг переноса (***C***=1)
SUB A	*97h*	Вычесть ***A*** из ***A*** (очистить ***A***)
SUB B	*90h*	Вычесть ***B*** из ***A***
SUB C	*91h*	Вычесть ***C*** из ***A***
SUB D	*92h*	Вычесть ***D*** из ***A***
SUB E	*93h*	Вычесть ***E*** из ***A***
SUB H	*94h*	Вычесть ***H*** из ***A***
SUB L	*95h*	Вычесть ***L*** из ***A***
SUB M	*96h*	Вычесть ***M*** из ***A***
SUI d8	*D6h*	Вычесть *d8* из ***A***
SBB A	*9Fh*	Вычесть ***A*** из ***A*** (очистить ***A***)
SBB B	*98h*	Вычесть с заёмом ***B*** из ***A***
SBB C	*99h*	Вычесть с заёмом ***C*** из ***A***
SBB D	*9Ah*	Вычесть с заёмом ***D*** из ***A***
SBB E	*9Bh*	Вычесть с заёмом ***E*** из ***A***
SBB H	*9Ch*	Вычесть с заёмом ***H*** из ***A***
SBB L	*9Dh*	Вычесть с заёмом ***L*** из ***A***
SBB M	*9Eh*	Вычесть с заёмом ***M*** из ***A***
SBI d8	*DEh*	Вычесть с заемом *d8* из ***A***
XCHG	*EBh*	Обмен содержимым ***DE*** и ***HL***
XTHL	*E3h*	Обмен содержимого вершины стека с содержимым ***HL***
XRA A	*AFh*	Исключающее ИЛИ ***A*** с ***A*** (очистка ***A***)
XRA B	*A8h*	Исключающее ИЛИ ***B*** с ***A***
XRA C	*A9h*	Исключающее ИЛИ ***C*** с ***A***
XRA D	*AAh*	Исключающее ИЛИ ***D*** с ***A***

XRA E	*ABh*	Исключающее ИЛИ ***E*** с ***A***
XRA H	*Ach*	Исключающее ИЛИ ***H*** с ***A***
XRA L	*ADh*	Исключающее ИЛИ ***L*** с ***A***
XRA M	*AEh*	Исключающее ИЛИ ***M**** (***HL***) с ***A***
XRI d8	*EEh*	Исключающее ИЛИ *d8* с ***A***

* М(регистровая пара) – ячейка памяти, адрес которой лежит в соответствующей регистровой паре.

Список литературы

1. Ершова Н.Ю., Иващенков О.Н., Курсков С.Ю. Микропроцессоры. – Санкт-Петербург, 2002.
2. Микропроцессоры: в 3-х кн. / под ред. С.В. Преснухина. – М.: Высшая школа, 1986. – Кн.1. – 495 с. – Кн. 2. – 383 с. – Кн. 3. – 351 с.
3. Ливенцов С.Н., Вильнин А.Д., Горюнов А.Г. Основы микропроцессорной техники: учебное пособие. – Томск: ТПУ, 2007. – 118 с.
4. Молодецкий В.Б., Кривенков М.В., Пахомов А.Н., Кудашев С.В. Микропроцессорная техника: учебное пособие. – Красноярск: ИПК СФУ, 2008. - 126с.
5. Sunil Mathur. Microprocessor 8085 and its interfacing. – New Delhi: PHI Learning Private Limitted, 2011. – 868 p.
6. Основы микропроцессорной техники: учебное пособие / Ю.В. Новиков, П.К. Скоробогатов. – 4-е изд., испр. – Москва: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2009. – 357 с.: ил.
7. Цифровая обработка сигналов: практический подход, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2008. – 992 с.: ил.

Учебное издание

ТОРГАЕВ Станислав Николаевич
ВОРОБЬЕВА Галина Степановна
МУСОРОВ Илья Сергеевич
ЧЕРТИХИНА Дарья Сергеевна

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ: МИКРОПРОЦЕССОР INTEL 8080

Учебное пособие

Компьютерная верстка *В.В. Михалев*

Зарегистрировано в Издательстве ТПУ
Размещено на корпоративном портале ТПУ

Национальный исследовательский Томский политехнический университет
Система менеджмента качества
Издательства Томского политехнического университета
сертифицирована в соответствии с требованиями ISO 9001:2008

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30
Тел./факс: 8(3822)56-35-35, www.tpu.ru