Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Севастопольский государственный университет»

# УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

к лабораторному практикуму по дисциплине
«**Архитектура вычислительных устройств информационных систем**»

для студентов, обучающихся по направлению
**09.03.02 “Информационные системы и технологии”**
очной и заочной форм обучения

**Севастополь**
**2021**

УДК 004.732
Учебно-методическое пособие к лабораторному практикуму по дисциплине «Архитектура вычислительных устройств информационных систем» / Сост. Чернега В.С., Дрозин А.Ю. — Севастополь: Изд-во СевГУ, 2021— 40 с.

Учебно-методическое пособие предназначено для оказания методической поддержки при проведения лабораторных работ по дисциплине «Архитектура вычислительных устройств информационных систем». Целью учебно-методического пособия является методическая помощь студентам в выполнении и защите лабораторных работ. Излагаются теоретические и практические сведения, а также методические рекомендации, необходимые для выполнения лабораторных работ. В пособии содержатся программы исследований и требования к содержанию отчета.

Учебно-методическое пособие рассмотрено и утверждено на методическом семинаре и заседании кафедры информационных систем
(протокол № 7 от 30 апреля 2021 г.)

Допущено учебно-методическим центром СевГУ в качестве учебно-методического пособия.

Рецензент: Кротов К.В., канд. техн. наук, доцент кафедры ИС

## Содержание

Лабораторная работа 1

# "Исследование архитектуры универсального 8-разрядного микропроцессора"

## 1. Цель работы

Исследовать архитектуру и основные блоки 8-разрядного процессора. Исследовать взаимодействие основных блоков процессора при выполнении команд разных типов. Приобрести навыки написания и отладки ассемблерных программ в эмуляторе KP580 Emulator.

## 2. Краткие теоретические сведения

Структурная схема 8-разрядного микропроцессора типа 8080, назначение функциональных блоков и его функционирование подробно описано в [6.1 — 6.4], а также изображена на рисунке 2.1. С точки зрения программиста процессор представляет собой ряд программно-доступных регистров общего назначения, арифметико-логическое устройство, выполняющее операции сложения и вычитания двоичных 8-разрядных чисел, логические операции, операции сдвига и некоторые другие действия. Для выполнения умножения и деления операндов требуется составлять отдельные программы.

К регистрам общего назначения относятся аккумулятор A и регистры B,C,D,E,H и L. Имеется также регистр признаков – регистр флагов F. 8-разрядный аккумулятор A используется в большинстве команд арифметической и логической обработки. Обычно он адресуется неявно и является как источником, так и приёмником операндов и результата;

Признаки результата операции фиксируются во флаговом регистре F. Пять флагов C, P, AC, Z и M упакованы в байт, три разряда которого не используются. Флаги имеют следующее функциональное назначение:

C (carry) – признак переноса из старшего разряда АЛУ;

P (parity) – признак четного числа единиц в результате операции;

AC (auxiliary carry) – признак дополнительного переноса из младших четырех разрядов (младшей тетрады) АЛУ. Используется наиболее часто при сложении чисел в двоично-десятичной форме;

Z (zero) – признак нулевого результата;

S (sign) – знак результата.

Значение флага указывает на результат выполнения какой-либо операции. Флаги всегда устанавливаются или сбрасываются автоматически после выполнения очередной команды, влияющей на флаги, в зависимости от результата операции. При этом флаг считается установленным, если флаговый разряд принимает значение 1, и сброшенным, если значение разряда равно 0.

Регистры общего назначения (РОН), кроме аккумулятора могут объединяться в пары (B-C,D-E и H-L) и использоваться как 16-битовые регистры.

Особенностью регистровой пары H-L является то, что она может неявно применяться для косвенной адресации памяти.

## 3. Описание лабораторной установки

Исследование архитектуры микропроцессора выполняется с помощью эмулятора KP580. Программа KP580 Emulator позволяет:

- написание программ на языке ассемблера, используя систему команд микропроцессора КР580ВМ80А;
- их отладку и просмотр выполнение в тактовом, командном и сквозном режимах;
- изучить особенности и порядок выполнения команд;
- приобрести навыки работы с внешними устройствами МП-системы;
- получить представления об организации внешней и внутренней (регистровой) памяти и стековой области;

В возможности эмулятора входит работа с 5-ю внешними устройствами, такими, как монитор, НГМД, НЖМД, сетевой адаптер и принтер; отладка и выполнение программ в тактовом, командном и сквозном режимах; работа со всем спектром системы команд данного МП; сохранение, загрузка и печать данных и результатов; ручной ввод данных в ОЗУ и РОН.

### 3.1 Главное окно программы

Главное окно программы изображено на рисунке 2.1. Содержимое главного окна программы:

1. Главное меню программы;
2. Структурная схема МП-системы;
3. Таблица содержимого ОЗУ МП-системы;
4. Внешние периферийные устройства, подключенные к портам МП-системы;
5. Панель редактирования значения выбранной (текущей) ячейки ОЗУ МП-системы;
6. Панель редактирования значения содержимого выбранного регистра общего назначения МП-системы;
7. Группа кнопок «Сброс» для обнуления всех ячеек ОЗУ и регистров общего назначения МП-системы;
8. Панель системы команд МП КР580ВМ80А (скрытый вид);
9. Группа кнопок «Выполнение» для выполнения программы МП-системой в сквозном, командном и тактовом режимах.

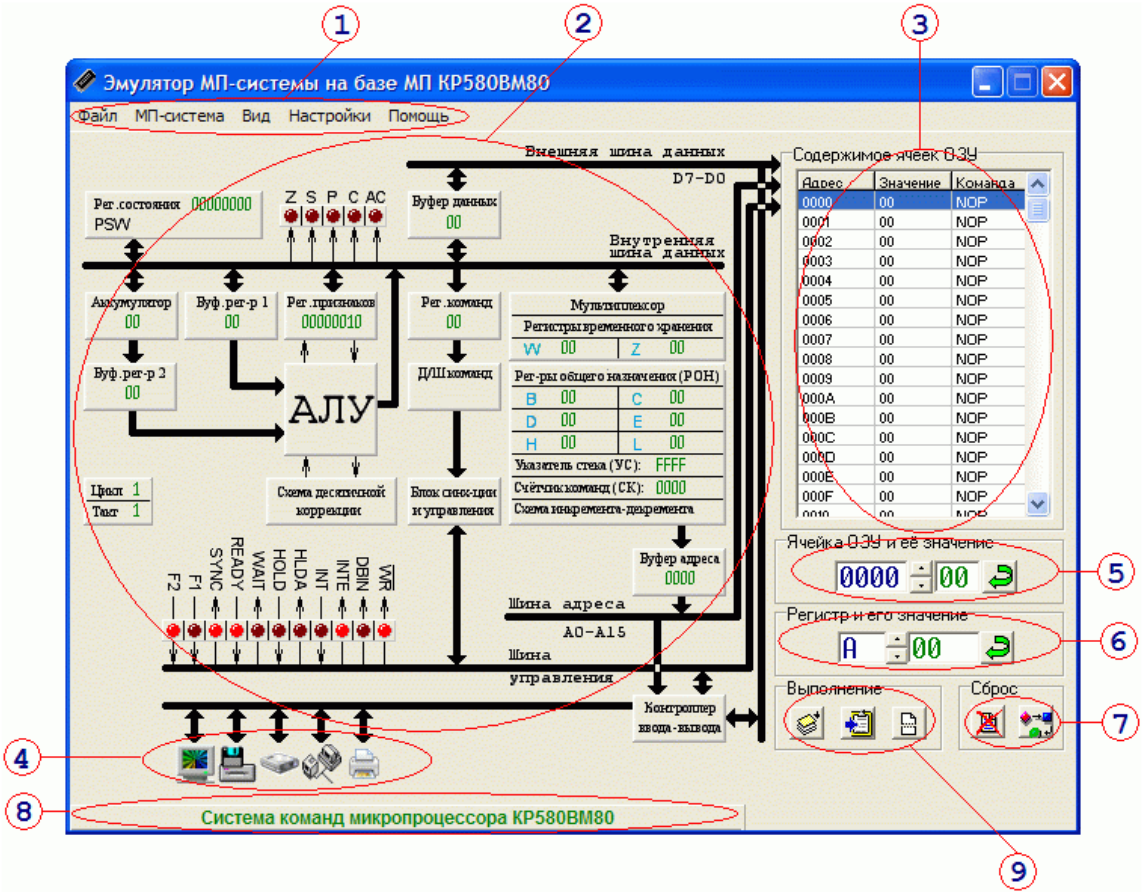



Рисунок 2.1 — Главное окно программы

## 2.2 Главное меню программы

Главное меню программы расположено в верхней части главного окна программы под его заголовком, как показано на рисунке 2.1 п.1, и содержит следующие пункты:

- "Файл";
- "МП-система";
- "Вид";
- "Настройки";
- "Помощь".

### 2.2.1 Меню «Файл»

Вызвать меню «Файл» можно нажав на соответствующий пункт меню («Файл»), или произвести вызов этого меню при помощи сочетания клавиш Alt+Ф. Содержимое меню «Файл» показано на рисунке 2.2 и включает в себя следующие пункты:

1. «Новый (очистить память и регистры)» — Служит для перевода МП-системы в исходное состояние, очищая (обнуляя) все ячейки ОЗУ и регистры общего назначения;

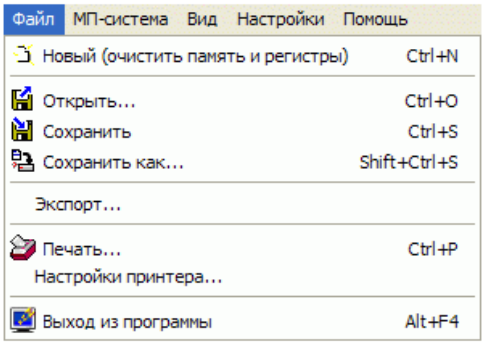



Рисунок 2.2 — Пункт меню «Файл» главного меню программы

2. «Открыть…» — Для открытия файла-образа содержимого ячеек ОЗУ и регистров общего назначения;
3. «Сохранить» - Для сохранения текущего файла-образа содержимого ячеек ОЗУ и регистров общего назначения. Если файл ещё не сохранён, то действие этого пункта меню аналогично пункту «Сохранить как…»;
4. «Сохранить как…» — Для сохранения файла-образа содержимого ячеек ОЗУ и регистров общего назначения с заданием имени файла, а также, выбором расположения этого файла в иерархии файловой системы носителей;
5. «Экспорт…» — Для экспорта в MS Word, MS Excel или текстовый файл выбранной части содержимого ячеек ОЗУ и значений выбранных регистров общего назначения;
6. «Печать…» — Для распечатки выбранной части содержимого ячеек ОЗУ и значений выбранных регистров общего назначения;
7. «Настройки принтера…» — Используется для задания параметров печати и выбора принтера при использовании пункта меню «Печать…»;
8. «Выход из программы» — Служит для завершения работы программы.

### 2.2.2 Меню «МП-система»

Вызвать меню «МП-система» можно нажав на соответствующий пункт меню («МП-система»), или произвести вызов этого меню при помощи сочетания клавиш Alt+ M.

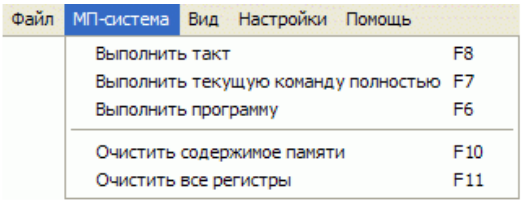



Рисунок 2.3 — Пункт меню «МП-система» главного меню программы

Содержимое меню «МП-система» показано на рисунке 2.3, и включает в себя следующие пункты:

1. «Выполнить такт» — Позволяет выполнить МП-системе один такт текущей команды. Действие аналогично действию кнопке «Выполнить такт» группы кнопок «Выполнение» главного окна программы;
2. «Выполнить текущую команду полностью» — Выполняет текущую команду МП-системы целиком, используя всю последовательность тактов, присущую данной команде. Действие аналогично действию кнопке «Выполнить текущую команду» группы кнопок «Выполнение» главного окна программы;
3. «Выполнить программу» — Запускает программу МП-системы на исполнение, начиная с текущей ячейки ОЗУ (команды). Действие аналогично действию кнопке «Выполнить программу» группы кнопок «Выполнение» главного окна программы;
4. «Очистить содержимое памяти» — Позволяет очистить (обнулить) все ячейки ОЗУ МП-системы. Действие аналогично действию кнопке «Очистить ОЗУ» группы кнопок «Сброс» главного окна программы;
5. «Очистить все регистры» — Позволяет очистить (обнулить) все регистры общего назначения МП-системы. Действие аналогично действию кнопке «Очистить РОН» группы кнопок «Сброс» главного окна программы.

### 2.2.3 Меню «Вид»

Вызвать меню «Вид» можно нажав на соответствующий пункт меню («Вид»), или произвести вызов этого меню при помощи сочетания клавиш Alt+ В.

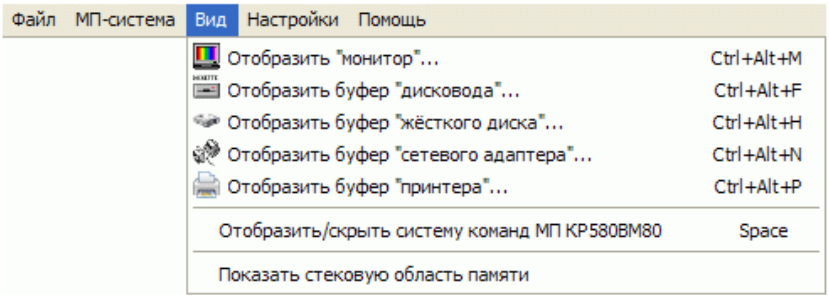



Рисунок 2.4 — Пункт меню «Вид» главного меню программы

Содержимое меню «Вид» показано на рисунке 2.4, и включает в себя следующие пункты:

1. «Отобразить «монитор»…» — Позволяет отобразить окно «Монитор КР580». Действие аналогично одиночному нажатию на пиктограмме «Отобразить монитор…» группы периферийных устройств, подключенных к МП-системе;
2. «Отобразить буфер «дисковода»…» — Позволяет отобразить окно «Дисковод КР580». Действие аналогично одиночному нажатию на пиктограмме «Отобразить буфер дисковода…» группы периферийных устройств, подключенных к МП-системе;

3. «Отобразить буфер «жёсткого диска»…» — Позволяет отобразить окно «Жёсткий диск КР580». Действие аналогично одиночному нажатию на пиктограмме «Отобразить буфер жёсткого диска…» группы периферийных устройств, подключенных к МП-системе;
4. «Отобразить буфер «сетевого адаптера»…» — Позволяет отобразить окно «Сетевой адаптер КР580». Действие аналогично одиночному нажатию на пиктограмме «Отобразить буфер сетевого адаптера…» группы периферийных устройств, подключенных к МП-системе;
5. «Отобразить буфер «принтера»…» — Позволяет отобразить окно «Принтер КР580». Действие аналогично одиночному нажатию на пиктограмме «Отобразить буфер принтера…» группы периферийных устройств, подключенных к МП-системе;
6. «Отобразить/скрыть систему команд МП КР580ВМ80» — Отображает (скрывает) панель системы команд МП КР580ВМ80, располагая её поверх структурной схемы МП-системы главного окна программы;
7. «Показать стековую область памяти» — Опускает (поднимает) прокрутку таблицы содержимого ячеек ОЗУ МП-системы до уровня ячейки, на которую указывает регистр-указатель стека МП-системы.

### 2.2.4 Меню «Настройки»

Вызвать меню «Настройки» можно нажав на соответствующий пункт меню («Настройки»), или произвести вызов этого меню при помощи сочетания клавиш Alt+Н.

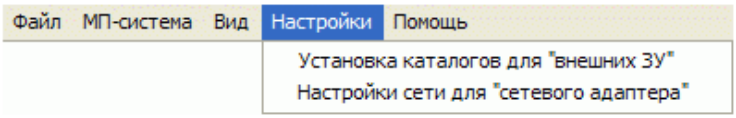



Рисунок 2.5 — Пункт меню «Настройки» главного меню программы

Содержимое меню «Настройки» показано на рисунке 2.5, и включает в себя следующие пункты:

1. «Установка каталогов для «внешних ЗУ»» — Позволяет установить пользовательский каталог иерархии файловой системы реальной машины для хранения фалов с данными, выведенными МП-системой в порт дисковода и жёсткого диска;
2. «Настройки сети для «сетевого адаптера»» — Позволяет настроить IP-адрес и TCP-порт реальной машины-приёмника данных, передаваемых в сеть.

### 2.2.5 Меню «Помощь»

Вызвать меню «Помощь» можно нажав на соответствующий пункт меню («Помощь»), или произвести вызов этого меню при помощи сочетания клавиш Alt+П.

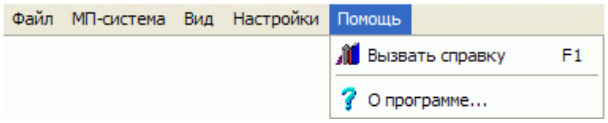



Рисунок 2.6 — Пункт меню «Помощь» главного меню программы

Содержимое меню «Помощь» показано на рисунке 2.6, и включает в себя следующие пункты:

1. «Вызвать справку» — Позволяет вызвать настоящее руководство;
2. «О программе…» — Вызывает окно, содержащее сведенья об авторах, версии программы, а также, адреса в Интернете для обновления версии программы.

2.3 Структурная схема МП-системы главного окна программы

Структурная схема МП-системы расположена в центральной части главного окна программы, как показано на рисунке 2.1 п.2, и содержит следующие элементы:

- Регистр слова состояния микропроцессора (PSW) МП-системы и его значение, представленное в двоичной системе счисления, а также, расшифровку этого значения, представленного в словесной форме;
- Буфер данных МП-системы и его значение, представленное в шестнадцатеричной системе счисления;
- Регистр-аккумулятор (A) МП-системы и его значение, представленное в шестнадцатеричной системе счисления;
- Буферные регистры МП-системы 1 и 2 и их значения, представленные в шестнадцатеричной системе счисления;
- Регистр признаков (флагов) МП-системы и его значение, представленное в двоичной системе счисления, а также, индикаторы расшифровки флагов: Z, S, P, C, AC;
- Регистр команд МП-системы и его значение, представленное в шестнадцатеричной системе счисления;
- Дешифратор команд МП-системы, индицирующий мнемонику текущей выполняемой команды, закреплённой на регистре команд;
- Счётчики машинных микроциклов и микротактов МП-системы, индицирующие свои текущие значения в десятичной системе счисления;
- Блок АЛУ МП-системы;
- Блок десятичной коррекции значения регистра-аккумулятора МП-системы;
- Блок синхронизации и управления МП-системой;
- Буфер адреса МП-системы и его значение, представленное в шестнадцатеричной системе счисления;
- Блок регистров общего назначения МП-системы и их значения, представленные в шестнадцатеричной системе счисления. Регистры B, C, D, E, H, L;

- Блок регистров временного хранения МП-системы и их значения, представленные в шестнадцатеричной системе счисления. Регистры W, Z;
- Схема инкремента/декремента МП-системы, индицирующее своё соответствующее действие условными обозначениями «+1» и «-1» соответственно;
- Регистр-указатель стека МП-системы и его значение, представленное в шестнадцатеричной системе счисления;
- Регистр-счётчик команд МП-системы и его значение, представленное в шестнадцатеричной системе счисления;
- Контролер ввода/вывода МП-системы;
- Индикаторы состояния и тактирования микропроцессора МП-системы: F1, F2, SYNC, READY, WAIT, HOLD, HLDA, INT, INTE, DBIN, WR;
- Порты МП-системы от 00h до 04h для монитора, дисковода, жёсткого диска, сетевого адаптера и принтера соответственно;
- Все элементы связаны между собой шинами: данных, адреса, управления, внутренней шиной данных и шиной внешних устройств (портов) в соответствии со структурной схемой (см. рис. 1).

Следующие элементы структурной схемы носят активных характер, позволяющий, при помощи щелчка мыши на их значении, отобразить и редактировать последнее в панели редактирования значений регистров (см. рисунок 2.1, п.6):

- Аккумулятор;
- Регистры блока РОН: B, C, D, E, H, L;
- Регистры временного хранения: W, Z;
- Указатель стека;
- Счётчик команд.

### 2.4 Таблица содержимого ОЗУ МП-системы

ОЗУ МП-системы представлено в виде блока (рисунок 2.1, п.3) с таблицей к которому схематично подведены шины управления, адреса и данных. Таблица условно разделена на 3 столбца:

- Столбец адреса ОЗУ — каждый адрес ячейки ОЗУ представлен в шестнадцатеричном виде и лежит в диапазоне от 0000h до FFFFh (0d…65535d), соответствуя тем самым максимально доступной адресации памяти для МП КР580ВМ80А (64КБ);
- Столбец значения ОЗУ — текущее значение, соответствующее данному адресу ОЗУ. Представлено в шестнадцатеричном виде и лежит в диапазоне от 00h до FFh (0d…255d);
- Столбец команды — расшифровка соответствующего значения ячейки ОЗУ МП-системы, лежащего по соответствующему адресу. Представлено в виде мнемокода на языке ассемблера. Однако стоит подразумевать, что не всегда мнемокод напрямую связан со значением

соответствующей ячейки, ввиду того, что предыдущая команда может быть, к примеру, двухбайтной, а стало быть, в данной ячейке подразумеваются данные от предыдущей команды, не имеющие никакого отношения к представленному мнемокоду.

При выборе строки этой таблицы (текущей ячейки) при помощи мыши или клавиатурных стрелок «↑» и «↓», изменяется значение номера выбранной ячейки на единицу соответственно, которое отражается в поле редактирования значения ячейки ОЗУ (см. рисунок 2.1, п.5), а также это выделение визуально отражает значение счётчика команд (PC) МП-системы;

В нижней (обычно) области таблицы содержимого ОЗУ установлено выделение коричневого цвета на ту ячейку ОЗУ, на которую указывает указатель стека (SP) МП-системы. Все нижестоящие ячейки (у которых адрес старше) окрашены жёлтым цветом, символизируя тем самым, стековую область ОЗУ. Фрагмент таблицы содержимого ОЗУ, случая, когда значение регистра-указателя стека равно FFFAh показан на рисунке 2.7.

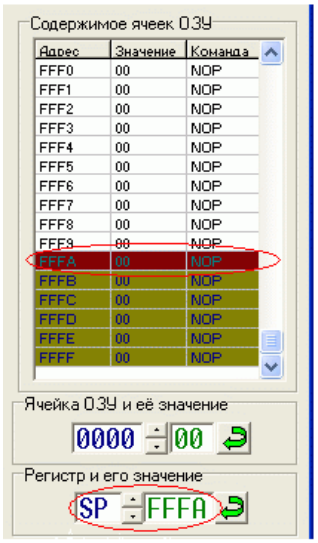



Рисунок 2.7 — Пункт меню «Помощь» главного меню программы

## 2.5 Внешние периферийные устройства


Рисунок 2.8 — Внешние периферийные устройства КР580

Внешние периферийные устройства МП-системы подключены к общей шине периферийных устройств, идущей от контроллера ввода/вывода (см. рисунок 2.8, рисунок 2.1, п.4). Всего к МП-системе подключено 5 устройств, соответственно портам ввода/вывода (00h…04h):

- Порт 00h. «Монитор КР580» — Представляет собой виртуальный монитор, обеспечивающий вывод графической или текстовой информации. Графический режим соответствует разрешению 256x256 пиксе-

лей и глубине цвета — 128 бит на пиксель, а текстовый — 39x20 символов и глубине цвета 128 бит на символ. Одновременно монитор поддерживает два этих режима, т.е. может содержать и текст и графику;

- Порт 01h. «Накопитель на гибких магнитных дисках КР580» — Представляет собой виртуальный буфер дисковода, обеспечивающий вывод данных в файл на накопитель на гибких дисках реальной машины в реальном времени при наличии дискеты в дисководе A;
- Порт 02h. «Накопитель на жёстких магнитных дисках КР580» — Представляет собой виртуальный буфер жёсткого диска, обеспечивающий вывод данных в файл в реальном времени на накопитель на жёстких дисках реальной машины;
- Порт 03h. «Сетевой адаптер КР580» — Представляет собой виртуальный полудуплексный буфер данных, обеспечивающий передачу данных в реальном времени по сети реальных вычислительных машин по протоколу TCP/IP. Адрес и порт указывается в «настройках «сетевого адаптера»»;
- Порт 04h. «Принтер КР580» — Представляет собой виртуальный буфер данных, обеспечивающий вывод данных на принтер реальной машины по согласию пользователя.

### 2.6 Панель редактирования значения выбранной (текущей) ячейки ОЗУ МП-системы

В правой части главного окна программы под таблицей содержимого ОЗУ МП-системы находится панель редактирования значения ячейки ОЗУ МП-системы (см. рисунок 2.1, п.5). Эта панель состоит из четырёх основных элементов:

- Поле ввода (отображения) текущего номера ячейки ОЗУ МП-системы — представляет собой четырёхзначное шестнадцатеричное число и служит для выбора редактируемой ячейки ОЗУ. Также может являться значением счётчика команд (PC) МП-системы. При выборе любой строки таблицы содержимого ОЗУ (номера ячейки), здесь также отражается номер выбранной ячейки, а также, это значение фиксируется на счётчике команд. При установке курсора в это поле, клавиатурные клавиши «↑» и «↓» также позволяют изменять значение номера ячейки на единицу соответственно;
- Прокрутка номера текущей ячейки ОЗУ — служит для удобства выбора номера текущей ячейки;
- Поле ввода значения выбранной ячейки ОЗУ — представляет собой двузначное шестнадцатеричное число и служит для редактирования значения выбранной ячейки ОЗУ МП-системы. При установке курсора в это поле, клавиатурные клавиши «↑» и «↓» позволяют изменять значение номера выбранной ячейки на единицу соответственно;

- Кнопка ввода нового значения в ОЗУ МП-системы — позволяет внести новое значение текущей (выбранной) ячейки в ОЗУ МП-системы. Клавиша «Enter» на клавиатуре может также осуществить подобное действие, но лишь в том случае, если курсор редактирования находится в поле ввода номера текущей ячейки ОЗУ, либо в поле ввода текущего значения выбранной ячейки ОЗУ.

2.7 Панель редактирования значения содержимого выбранного регистра общего назначения МП-системы

В правой части главного окна программы под панелью редактирования содержимого выбранной ячейки ОЗУ МП-системы находится панель редактирования значения содержимого выбранного регистра общего назначения МП-системы (см. рисунок 2.1, п.6). Эта панель состоит из четырёх основных элементов:

- Поле ввода (отображения) выбранного регистра МП-системы — представляет собой наименование регистра (A, B, C, D, E, H, L, W, Z, PC, SP) и служит для выбора редактируемого регистра. При установке курсора в это поле, клавиатурные клавиши «↑» и «↓» также позволяют изменять наименование выбранного регистра по порядку;
- Прокрутка наименования регистра - служит для удобства выбора регистра;
- Поле ввода значения выбранного регистра — представляет собой двузначное шестнадцатеричное число и служит для редактирования значения выбранного регистра МП-системы. При установке курсора в это поле, клавиатурные клавиши «↑» и «↓» позволяют изменять наименование выбранного регистра по порядку;
- Кнопка ввода нового значения в выбранный регистр МП-системы — позволяет внести новое значение выбранного регистра МП-системы. Клавиша «Enter» на клавиатуре может также осуществить подобное действие, но лишь в том случае, если курсор редактирования находится в поле ввода наименования регистра, либо в поле ввода текущего значения выбранного регистра.

2.8 Группа кнопок «Сброс»

В правой нижней части главного окна программы находится группа кнопок «Сброс» (см. рисунок 2.1, п. 7), и состоит из двух кнопок:

- Сброс ОЗУ — позволяет обнулить все ячейки ОЗУ МП-системы;
- Сброс регистров — позволяет обнулить все регистры МП-системы.

2.9 Панель системы команд МП КР580ВМ80А

В нижней части главного окна программы находится заголовок панели системы команд МП КР580ВМ80А, как показано на рисунке 2.1, п. 8. Это

скрытый (исходный) вид панели. При наведении курсора мыши на этот заголовок, панель «всплывает» поверх структурной схемы МП-системы. Её полный вид показан на рисунке 33.

Панель системы команд представлена в виде таблицы 16x16, строки и столбцы которой пронумерованы шестнадцатеричными цифрами, комбинация которых (строка-столбец) означает номер (код) команды. Для примера: строка 5h, столбец Bh, будут соответствовать команде «MOV E, E» с кодом 5Bh.

Система команд микропроцессора КР580ВМ80

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NOP | LXI B,d16 | STAX B | INX B | INR B | DCR B | MVI B,d8 | RLC | - | DAD B | LDAX B | DCX B | INR C | DCR C | MVI C,d8 | RRC |
| 1 | - | LXI D,d16 | STAX D | INX D | INR D | DCR D | MVI D,d8 | RAL | - | DAD D | LDAX D | DCX D | INR E | DCR E | MVI E,d8 | RAR |
| 2 | - | LXI H,d16 | SHLD adr | INX H | INR H | DCR H | MVI H,d8 | DAA | - | DAD H | LHLD adr | DCX H | INR L | DCR L | MVI L,d8 | CMA |
| 3 | - | LXI SP,d16 | STA adr | INX SP | INR M | DCR M | MVI M,d8 | STC | - | DAD SP | LDA adr | DCX SP | INR A | DCR A | MVI A,d8 | CMC |
| 4 | MOV B,B | MOV B,C | MOV B,D | MOV B,E | MOV B,H | MOV B,L | MOV B,M | MOV B,A | MOV C,B | MOV C,C | MOV C,D | MOV C,E | MOV C,H | MOV C,L | MOV C,M | MOV C,A |
| 5 | MOV D,B | MOV D,C | MOV D,D | MOV D,E | MOV D,H | MOV D,L | MOV D,M | MOV D,A | MOV E,B | MOV E,C | MOV E,D | MOV E,E | MOV E,H | MOV E,L | MOV E,M | MOV E,A |
| 6 | MOV H,B | MOV H,C | MOV H,D | MOV H,E | MOV H,H | MOV H,L | MOV H,M | MOV H,A | MOV L,B | MOV L,C | MOV L,D | MOV L,E | MOV L,H | MOV L,L | MOV L,M | MOV L,A |
| 7 | MOV M,B | MOV M,C | MOV M,D | MOV M,E | MOV M,H | MOV M,L | HLT | MOV M,A | MOV A,B | MOV A,C | MOV A,D | MOV A,E | MOV A,H | MOV A,L | MOV A,M | MOV A,A |
| 8 | ADD B | ADD C | ADD D | ADD E | ADD H | ADD L | ADD M | ADD A | ADC B | ADC C | ADC D | ADC E | ADC H | ADC L | ADC M | ADC A |
| 9 | SUB B | SUB C | SUB D | SUB E | SUB H | SUB L | SUB M | SUB A | SBB B | SBB C | SBB D | SBB E | SBB H | SBB L | SBB M | SBB A |
| A | ANA B | ANA C | ANA D | ANA E | ANA H | ANA L | ANA M | ANA A | XRA B | XRA C | XRA D | XRA E | XRA H | XRA L | XRA M | XRA A |
| B | ORA B | ORA C | ORA D | ORA E | ORA H | ORA L | ORA M | ORA A | CMP B | CMP C | CMP D | CMP E | CMP H | CMP L | CMP M | CMP A |
| C | RNZ | POP B | JNZ adr | JMP adr | CNZ adr | PUSH B | ADI d8 | RST 0 | RZ | RET | JZ adr | - | CZ adr | CALL adr | ACI d8 | RST 1 |
| D | RNC | POP D | JNC adr | OUT N | CNC adr | PUSH D | SUI d8 | RST 2 | RC | - | JC adr | IN N | CC adr | - | SBI d8 | RST 3 |
| E | RPO | POP H | JPO adr | XTHL | CPO adr | PUSH H | ANI d8 | RST 4 | RPE | PCHL | JPE adr | XCHG | CPE adr | - | XRI d8 | RST 5 |
| F | RP | POP PSW | JP adr | DI | CP adr | PUSH PSW | ORI d8 | RST 6 | RM | SPHL | JM adr | EI | CMP H | - | CPI d8 | RST 7 |

Рисунок 2.9 — Полный вид панели системы команд МП КР580ВМ80А

В ячейках самой таблицы указан мнемокод команд, а цвет ячейки визуально отражает принадлежность команд определённой группе. Всего команды условно разделены на 12 групп:

1. Однобайтовых пересылок (оранжевый цвет);
2. Двухбайтовых пересылок (жёлтый цвет);
3. Арифметических операций с одним операндом (коричнево-зелёный цвет);
4. Арифметических операций с двумя операндами (салатовый цвет);
5. Логических операций с одним операндом (зелёный цвет);
6. Логических операций с двумя операндами (ярко-зелёный цвет);
7. Установки признаков (сиреневый цвет);
8. Шестнадцатибитовых операций (красный цвет);
9. Сдвига содержимого аккумулятора (розовый цвет);
10. Передачи управления (светло-коричневый цвет);
11. Вызова и возврата из подпрограмм (светло-бирюзовый цвет);

12. Специальные (бледно-голубой цвет).

Панель команд облегчает программирование эмулятора, позволяя «переносить» при помощи мыши требуемые команды (значения) неограниченное число раз на строки таблицы содержимого ячеек ОЗУ МП-системы, тем самым заполняя ячейки ОЗУ требуемыми значениями (командами). Для этого следует:

1. Выбрать требуемое значение (команду), наведя курсор мыши на соответствующую ячейку таблицы системы команд;
2. Зажать левую кнопку мыши;
3. Не отпуская левой кнопки, перевести указатель мыши на нужную строку таблицы содержимого ячеек ОЗУ;
4. Отпустить кнопку мыши.

После чего, в соответствующей строке таблицы содержимого ячеек ОЗУ будет отражён номер и мнемокод «перенесённой» команды. За ненадобностью отображения панели системы команд, следует отвести курсор мыши на свободную область главного окна программы. В этом случае панель примет исходный (скрытый) вид.

### 2.10 Группа кнопок «Выполнение»

В правой нижней части главного окна программы находится группа кнопок «Сброс» (см. рисунок 2.1, п. 9), и состоит из трёх кнопок (справа-налево):

- Выполнить такт — позволяет выполнить один такт текущей команды, на которую указывает счётчик команд (PC) МП-системы. При этом, если команда выполнена не целиком, становятся недоступными некоторые элементы управления главного окна, а вступившие изменения значений в выполненном такте отмечаются красным цветом;
- Выполнить команду целиком — позволяет выполнить (довыполнить) все такты текущей команды, на которую указывает счётчик команд (PC) МП-системы;
- Выполнить программу — запускает программу на выполнение, начиная с адреса, на который указывает счётчик команд (PC) МП-системы. При этом данная кнопка принимает утопленный вид с пиктограммой «stop», что меняет её функцию на останов выполнения программы. Выполнение заканчивается по достижению команды HLT (76h), либо по принудительному останову нажатием на этой кнопке с пиктограммой «stop».

### 2.11 Возможности экспорта и печати данных эмулятора

Для удобства работы с рассматриваемым эмулятором предусмотрены следующие возможности работы с данными:

- Загрузка и сохранение образов содержимого ОЗУ и РОН;
- Частичная загрузка и сохранение программ эмулятора (подпрограмм);

- Экспорт содержимого ОЗУ и РОН эмулятора в MS Excel;
- Экспорт содержимого ОЗУ и РОН эмулятора в MS Word;
- Экспорт содержимого ОЗУ и РОН эмулятора в текстовый файл;
- Печать содержимого ОЗУ и РОН эмулятора.

## 2.12 Настройки программы

- Настройка каталогов для эмуляции внешних накопителей;
- Настройка адресов сети для эмуляции работы "сетевого адаптера".

### 2.12.1 Настройка каталогов для эмуляции внешних накопителей

Для организации работы с устройствами ввода-вывода эмулятора, а именно внешних накопителей, используются соответствующие каталоги на дисках реальной машины, для размещения выходных данных в файлах. По умолчанию это:

- Дисковод (порт в/в эмулятора 01h) - "A:\"
- Жёсткий диск (порт в/в эмулятора 02h) - "C:\"

Соответственно, по желанию пользователя, можно указать и другие каталоги. Для этого следует выбрать из главного меню программы пункт "Настройки→Установки каталогов для "внешних ЗУ"". В появившемся окне настроек каталогов укажите путь для хранения файлов с выходными данными эмулятора на накопителе на гибких магнитных дисках и на жёстком диске "C" реальной машины в соответствующих элементах-деревьях каталогов. После чего нажмите кнопку "ОК" чтобы изменения вступили в силу.

Если в дисководе отсутствует дискета, эмулятор выдаст соответствующее сообщение. При желании, можно вставить дискету и нажать появившуюся кнопку "Готово" в рассматриваемом окне, и продолжить выбор каталогов.

Окно настройки каталогов изображено на рисунке 41.

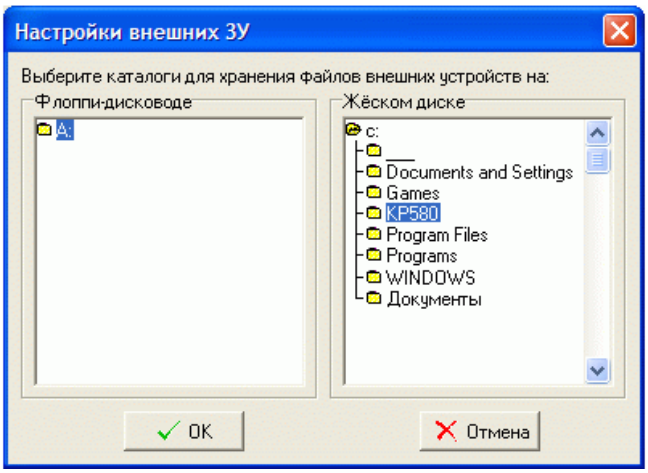



Рисунок 2.10 — Окно настройки каталогов для внешних накопителей

### 2.12.2 Настройка адресов сети для эмуляции работы сетевого адаптера

Для передачи и приёма данных эмулятора по сети при помощи команд ввода вывода, используя порт 03h, следует изначально настроить IP-адрес и TCP-порт реальной машины-получателя, на которой также установлен и запущен данный эмулятор.

Для этого следует выбрать из главного меню программы пункт "Настройки→Установки сети для "сетевого адаптера"". В появившемся окне настроек сети укажите в поле "Адрес" IP-адрес машины-приёмника, а в поле "Порт", соответственно, TCP-порт. После чего нажмите кнопку "OK" чтобы изменения вступили в силу.

Окно настройки сети изображено на рисунке 2.11.

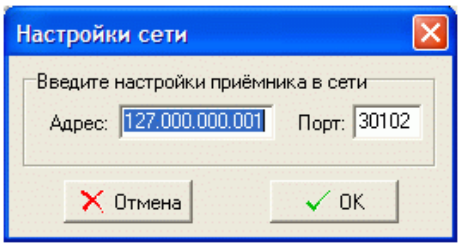



Рисунок 2.11 — Окно "настройки сети" с параметрами машины-приёмника сети для эмуляции работы "сетевого адаптера"

Таким образом, при работе с сетью на эмуляторах, можно настроить последние соответствующим образом так, чтобы данные передавались "по цепочке", "по кольцу", или, в простейшем случае, для двух машин.

## 2.13 Работа с программой

- Общие принципы работы с программой;
- Работа с внешними устройствами эмулятора.

### 2.13.1 Особенности работы с программой

Начало работы с программой в основном заключается в написании или загрузке программы на Ассемблере в эмулятор. Для этого можно воспользоваться либо панелью системы команд программы, либо панелью редактирования значений ячеек ОЗУ эмулятора, либо загрузить образ ОЗУ с носителя.

При необходимости, можно заполнить соответствующими значениями регистры общего назначения эмулятора. После чего, для подробного изучения каждого такта конкретной команды, можно воспользоваться кнопкой тактированного выполнения команды.

Для выполнения программы в командном режиме, (к примеру, для отладки программы) используется кнопка командного режима. Соответственно, для выполнения программы целиком, следует пользоваться кнопкой программного режима.

Если выполнение программы завершается командой останова 76h HLT, устанавливается флаг останова микропроцессора HLDA и выдаётся соответствующее сообщение. Работу с эмулятором можно продолжить, сняв флаг при

помощи пункта меню "МП-система→Снять флаг HLDA", либо воспользовавшись клавишей F12 на клавиатуре, либо произвести сброс РОН (см. ниже).

Для очистки (обнуления) РОН и/или ОЗУ эмулятора, воспользуйтесь группой кнопок "Сброс", либо пунктом меню программы "Файл→Новый (очистить память и регистры)"

После написания программы на языке Ассемблера, её можно сохранить в виде образа ОЗУ и РОН эмулятора на какой-либо носитель, и при следующей надобности, также загрузить в эмулятор. Загружать и сохранять можно не только весь образ, но и часть ОЗУ эмулятора.

Для удобства работы с написанными программами в виде таблиц или текстовых документов, предусмотрены возможности экспорта части содержимого ОЗУ и/или РОН эмулятора в MS Excel, MS Word и текстовый файл.

Также предусмотрена возможность печати части содержимого ОЗУ и/или РОН эмулятора на принтере.

### 2.13.2 Работа с внешними устройствами эмулятора

Для начала работы с внешними устройствами эмулятора, следует осуществить некоторые настройки каталогов жёсткого диска и дисковода, а также настройки сети реальной машины.

Работа со всеми внешними устройствами эмулятора заключается в отправке или приёме на (с) соответствующий(его) устройству порт(а) МП-системы значения из (в) регистра-аккумулятора. Это осуществляется путём выполнения на эмуляторе команд ввода-вывода, таких как IN (принять из порта) и OUT (вывести в порт).

#### Работа с монитором

"Монитор КР580" поддерживает отдельную систему команд, обеспечивающую вывод графической или текстовой информации. Графический режим соответствует разрешению 256x256 пикселей и глубине цвета — 128 бит на пиксель, а текстовый - 39x20 символов и глубине цвета 128 бит на символ. Одновременно монитор поддерживает два этих режима, т.е. может содержать и текст и графику.

Команды засылаются в порт 00h побайтно. Различаются 3-х байтные и 2-х байтные команды:

2-х байтная.

1-ый байт: 1-ый бит - 0-текст, 1-графика; остальные 7 бит на цвет, согласно формуле: FFFFFFh(RGB) / 127 * эти_7_бит.

2-ой байт: номер символа в кодовой таблице OEM/DOS.

3-х байтная.

1-ый байт: 1-ый бит - 0-текст, 1-графика; остальные 7 бит на цвет, согласно формуле: FFFFFFh(RGB) / 127 * эти_7_бит.

2 байт: координата по X.

3 байт: координата по Y.

Работа с дисководом. В порт дисковода КР580 (01h) засылаются или читаются значения, что приводит к их одновременному сохранению (чтению) в (из) файл(а) реальной машины, располагающейся на дискете.

Работа с жёстким диском (порт 02h)

Аналогично работе с дисководом, только связано с жёстким диском реальной машины.

Работа с сетевым адаптером

Отправленные значения в порт 03h пересылаются по сети реальных машин по протоколу TCP/IP на IP-адрес, указанный в настройках.

Работа с принтером

Отправленные значения в порт 04h временно хранятся в буфере до тех пор, пока пользователь не отправит их на печать самостоятельно. Печатаемые символы на принтере реальной машины соответствуют кодировке OEM/DOS.

## 3. Программа лабораторной работы

3.1 Изучить архитектуру МП КР580ВМ80 (выполняется в процессе домашней подготовки к лабораторной работе).

3.2 Изучить основные команды МП КР580ВМ80 (выполняется в процессе домашней подготовки к лабораторной работе).

3.3 Задавая различные команды (запись данных в регистр и в пару регистров, пересылки данных, суммирования при наличия переноса, чтения и записи в память, записи в стек, обращения к памяти путем косвенной адресации и др.) исследовать наличие и вид сигналов и данных на шинах процессора, содержимое регистров, значение флагов и взаимодействие блоков МП КР580ВМ80 в ходе выполнения команд.

## 4. Методические указания и рекомендации по выполнению работы

4.1. Для записи, например, числа 25 в регистр D, нужно использовать ассемблерную команду MVI D,25. Команда заносится в ячейку ОЗУ с адресом 0000 (окно 3 главного окна, изображенного на рис.2.1) путем перетягивания с помощью мышки кода этой команды из таблицы команд (рис.2.9). Затем следует увеличить адрес на 1 (окно 5) и в поле данных ввести с клавиатуры число 25. Обратите внимание, что число 25 будет интерпретироваться в поле «Команда» как команда DCR H. На это не нужно обращать внимание, так как это связано с недостатком эмулятора, который любые числа декодирует, как код команды.

4.2. Для записи двухбайтового числа, например 413F, в регистровую пару H,L нужно воспользоваться командой LXI H,413F. Занесение этой команды в память ОЗУ осуществляется аналогично п.4.1.

4.3. С целью исследования косвенной адресации составьте программу занесение в память ОЗУ массива N однобайтных чисел и затем сложения их и занесения суммы в одну из ячеек памяти. Здесь N – две последние цифры зачетной книжки студента.

4.4. Для исследования функционирования процессора нужно запустить потактовое выполнение составленной программы. Для этого следует перейти на нулевой адрес ОЗУ и начать последовательно нажимать на кнопку в окошке 9 «Выполнить такт». При этом нужно исследовать сигналы, выдаваемые на шины процессора.

## 5. Содержание отчета

4.1 Цель и программа работы.
4.2 Структурная схема МП КР580ВМ80.
4.3 Описание взаимодействия блоков микропроцессора при выполнении команд различной длины и различных типов.
4.4 Результаты проведенных исследований и расчетов времени выполнения команд.
4.5 Выводы по работе с анализом результатов выполненных исследований и расчетов.

## 6. Контрольные вопросы

1.1. Расскажите об основных блоках процессора 8-разрядного микропроцессора и их назначении.

2.1. Объясните понятие машинного цикла. Перечислите виды машинных циклов МП КР580ВМ80.

3.1. Проанализируйте подробно с привлечением временных диаграмм работу процессора при различных режимах работы: программно-управляемом, обслуживания прерываний, прямого доступа в память.

4.1. Перечислите основные внешние выходы МП КР580ВМ80, расскажите об их назначении.

5.1. С какой целью процесор вначале каждого машинного цикла выдает слово состояния цикла?

6.1. Для чего служат регистры общего назначения и каковы особенности их применения?

7.1. Объясните назначение регистра признаков и как используется значение флагов.

8.1. Каково назначение регистров W и Z?

9.1. Расскажите о роли счетчика команд в организации выполнения программы.

10.1.Расскажите о роли указателя стека в организации выполнения программы.

11.1.Расскажите об основных возможностях экранного отладчика KP580 Emulator.

12.1.Расскажите о режимах исполнения отдельных команд и целых программ в экранном отладчике KP580 Emulator.

13.1.Опишите возможности взаимодействия микропроцессора с внешними устройствами реализованные в экранном отладчике KP580 Emulator. Что такое десятичная коррекция и в каких случаях она применяется?

Лабораторная работа 2

# “Исследование методов реализации алгоритмов обработки данных на ассемблере 8-разрядного микропроцессора ”

## 1. Цель работы

Исследовать методы реализации типовых алгоритмов обработки данных на ассемблере процессора КР580ВМ80. Изучение основных команд пересылки данных, передачи управления и арифметических команд ассемблера микропроцессора. Исследование возможностей эмулятора и экранного отладчика KP580 Emulator. Приобретение практических навыков составления и отладки программ на языке Ассемблера.

## 2. Основные теоретические положения

2.1 Система команд микропроцессора

2.1.1 Классификация команд

Под командой понимают совокупность сведений, необходимых процессору для выполнения определенного действия при реализации программы. Множество команд, реализуемых в ЭВМ, образует систему команд, выбор которой является важнейшей задачей проектирования ЭВМ. Система команд определяет область применения и эффективность микропроцессорной системы управления. Несмотря на то, что подавляющее большинство алгоритмов может быть реализовано посредством ограниченного набора команд, большинство ЭВМ имеет 60–120 базовых команд. Под базовой понимают команду, которая определяет выполняемую операцию без учета модификаций данной команды за счет использования различных режимов адресации. Например, МП КР580ВМ80А имеет 78 базовых команд, однако с учетом модификаций число команд равняется 224. Это позволяет в ряде случаев существенно сократить длину программ, а следовательно, уменьшить время решения задачи и размер программы в памяти. Таким образом, система команд определяет возможности машины.

Теоретически ограничения на число команд ЭВМ нет; например, при введении команд из нескольких слов можно выделить больше бит под код операции. Каждый дополнительный бит в коде операции удваивает число команд. С другой стороны, чем сложнее команда, тем быстрее выполняется программа из-за сокращений числа обращений к памяти. Классификация команд по основным признакам представлена на рисунке 2.1.

Систему команд рассматриваемого микропроцессора КР580ВМ80А можно классифицировать по трем основным признакам:

1. Длине команды (одно-, двух- и трехбайтные);

2. Функциональному назначению (передачи, обработки данных, команды управления);

3. Архитектурным признакам (операции с регистрами, памятью и портами).


Рисунок 2.1 — Классификация команд

Современные тенденции развития ЭВМ показывают, что фирмы-разработчики микропроцессоров стараются создавать дополнительные наборы команд на основе уже существующих, сохраняя программную преемственность с предыдущими поколениями процессоров. Такие ресурсоемкие задачи, как расчет трехмерной графики, компрессия/декомпрессия аудио-видеоданных и другие, используют дополнительные наборы команд (3DNow, MMX, SSE, и др.), оптимизированные под соответствующие приложения.

### 2.1.2 Методы адресации

Для взаимодействия различных модулей в микроЭВМ должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода/вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т.е. однозначная комбинация бит. Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда – отдельные адресные пространства регистров, устройств ввода/вывода и внутренней памяти. Кроме того, память хранит как данные, так и команды. С другой стороны, при разработке микропроцессоров стараются использовать коды операций минимальной длины, что приводит к возникновению проблемы идентификации данных из-за короткого машинного слова. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти – это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес. В микропроцессоре КР580ВМ80А используется пять методов адресации:

1. Прямая – в команде задается адрес ячейки памяти, где расположен операнд; он указывается во втором (младшая часть адреса) и в третьем (старшая часть) байтах команды. К этой группе также относятся команды, в которых задается адрес порта ввода/вывода:

STA 8020H – требует четырех обращений к памяти;

IN 05H – требует двух обращений к памяти.

2. Прямая регистровая – в команде задается адрес регистра или пары регистров, где находится 8- или 16-битный операнд:

MOV A, B – требует одного обращения к памяти;

CMP C.

3. Непосредственная – операнд содержится в самой команде:

MVI A, 08H – требует двух обращений к памяти;

LXI M, 8020H – требует трех обращений к памяти.

4. Косвенная – адрес M ячейки памяти, где расположен операнд, определяется содержимым парного регистра, явно или неявно указанного в команде:

MOV A, M – пересылка в A из ячейки памяти, на которую указывает HL;

LDAX B – загрузка A из ячейки памяти, на которую указывает пара BC.

5. Неявная – адрес операнда не указывается в явном виде, а определяется кодом операции:

ADD B; A ← A+B, аккумулятор не задается в явном виде.

Следует отметить, что в одной команде могут использоваться два различных метода адресации, например, в команде MVI A, 08H используется прямая регистровая адресация для приемника и непосредственная для источника. В системах реального времени для повышения скорости вычислений программ необходимо максимально использовать регистровую адресацию.

### 2.1.3 Формат команд

Формат команды определяет ее структурные элементы, каждый из которых интерпретируется определенным образом при выполнении команды. Среди таких элементов (полей) выделяют:

- код операции, определяющий выполняемое действие;
- адрес ячейки памяти, регистра процессора, внешнего устройства;
- режим адресации;
- операнд при использовании непосредственной адресации;
- код анализируемых признаков для команд условного перехода.

Почти во всех форматах команд первые биты отводятся для кода операции, но далее форматы команд разных ЭВМ сильно отличаются друг от друга. Остальные биты должны определять операнды или их адреса, и поэтому

они используются для комбинации режимов, адресов регистров, адресов памяти, относительных адресов и непосредственных операндов. Обычно длина команды варьируется от 1 до 3 и даже 6 байт.

Число бит, отводимое под КОП, является функцией полного набора реализуемых команд. При использовании фиксированного числа бит под КОП для кодирования всех $m$ команд необходимо в поле КОП выделить $\log_2 m$ двоичных разрядов. Так как информация берется только из одной ячейки, эта ячейка называется источником; ячейка, содержимое которой изменяется, называется приемником.

Средняя длина команды в типичной программе для 8-разрядного микропроцессора равна двум байтам, а для программ более поздних 16-разрядных процессоров типа i8086 она примерно равна 4. Поэтому на программах с большим количеством логических операций 8-разрядные процессоры незначительно уступают 16-разрядным.

Положение полей в микропроцессоре КР580ВМ80А переменное, и в зависимости от команды, назначение поля может иметь следующее значение:

Byte 1 – содержит код операции, длину команды, адреса регистров;

Byte 2 – содержит адрес порта ввода/ вывода, 8-разрядный операнд или младшую часть 16-разрядного операнда;

Byte 3 – содержит старшую часть 16-разрядного операнда.


Рисунок 2.2 — Формат первого байта команды КР580ВМ80А

Многобайтная команда должна размещаться в последовательно расположенных ячейках памяти.

Адрес приемника (DDD) и адрес источника (SSS) задают регистр или ссылку на ячейку памяти (признак косвенной адресации), при этом адреса регистров кодируются следующим образом (таблица 2.1):

Таблица 2.1 — Кодировка адресов регистров и регистровых пар

| Регистр R | Код SSS или DDD | Регистр R | Код SSS или DDD | Регистровая пара RP | Код RP |
|---|---|---|---|---|---|
| B | 000 | H | 100 | BC | 00 |
| C | 001 | L | 101 | DE | 01 |
| D | 010 | M | 110 | HL | 10 |
| E | 011 | A | 111 | SP | 11 |

2.2 Программная реализация базовых алгоритмов управления

2.2.1 Программная реализация ветвлений

Ветвление - это такое место в программе, после которого в зависимости от какого-либо условия может начать выполняться тот или иной код. То есть, счетчик команд в результате выполнения команды ветвления может быть установлен по двум различным адресам, в зависимости от исхода проверки какого-либо условия. На рисунке 2.3 показаны фрагменты алгоритмов программ с ветвлением и без него.

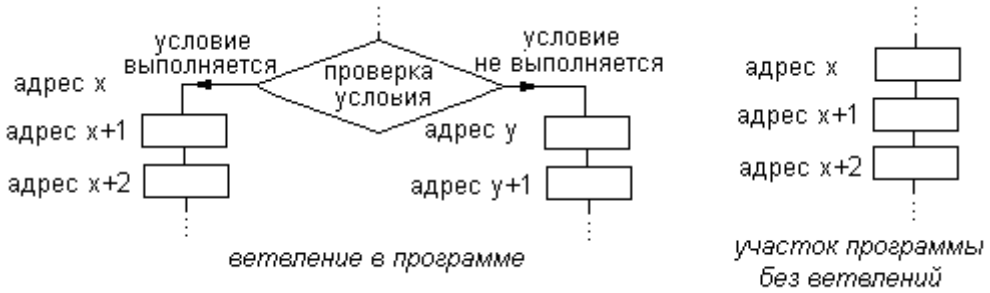



Рисунок 2.3 — Участки программы с ветвлением и без ветвления

Для организации ветвлений в микропроцессоре КР580ВМ80 используются следующие команды: JZ, JNZ, JC, JNC, JP, JM, JPE, JPO. Работу этих команды рассмотрим на примере первых четырех как наиболее важных для программиста.

JZ Адрес ; Перейти на Адрес, если Z бит = 1 *Jump if zero*.

JNZ Адрес ; Перейти на Адрес, если Z бит = 0 *Jump if not zero*.

JC Адрес ; Перейти на Адрес, если C бит = 1 *Jump if carry*.

JNC Адрес ; Перейти на Адрес, если C бит = 0 *Jump if no carry*.

Если провести аналогию с языками высокого уровня, то эти конструкции можно сравнить с конструкциями вида if then else и им подобными.

2.2.2 Программная реализация циклов и временных задержек

При разработке программного обеспечения часто возникает необходимость выполнения определенных действий заданное количество раз. Собственно, циклы это и есть повторение одного и того же участка кода заданное количество раз. В ассемблере для организации циклов нет специальных команд, подобных операторам for to , while do и прочим операторам языков высокого уровня.

Есть какое-то условие, при выполнении которого цикл заканчивается. Один из аргументов этого условия (или даже оба аргумента) в процессе выполнения цикла может изменяться. Если условие не выполнено, то цикл повторяется. Как только условие выполнится, счётчик команд устанавливается на адрес первой команды, следующей за командами цикла. Циклы бывают двух типов: с предпроверкой (или ещё говорят с предусловием), когда проверка условия происходит в начале цикла, и с постпроверкой (или по-другому — с постусловием), когда проверка условия происходит в конце цикла. На рисунке 2.4 показаны фрагменты алгоритмов с различными видами циклов.

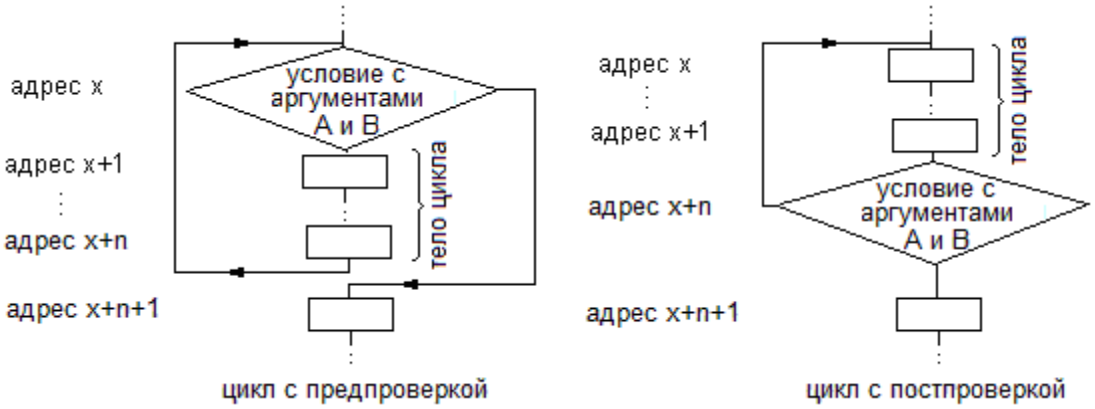



Рисунок 2.4 — Виды циклов

Цикл представляет собой по сути обычное ветвление, только один из переходов осуществляется назад, к тому коду, который мы уже выполняли. Такой алгоритм легко можно организовать с помощью описанных выше команд. Для этого используются программные счетчики, которые реализуются нижеизложенными способами.

Цикл с предпроверкой:

1. Путем инкремента содержимого регистра или ячейки памяти:

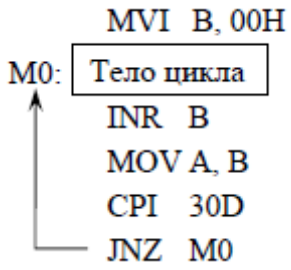
```
    MVI  B, 00H
M2: INR  B
    MOV A, B
    CPI  30D
    JNZ  M0
    Тело цикла
    JMP  M2
M0:
```

2. Путем декремента содержимого регистра или ячейки памяти:

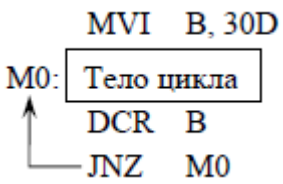
```
    MVI  B, 30D
M2: DCR  B
    JNZ  M0
    Тело цикла
    JMP  M2
M0:
```

Цикл с постпроверкой:

1. Путем инкремента содержимого регистра или ячейки памяти:




2. Путем декремента содержимого регистра или ячейки памяти:




В этих примерах тело цикла выполняется 30 раз. При необходимости в числе повторов более 255 можно использовать вложенные счетчики. Программный счетчик с количеством повторений 1200 может выглядеть следующим образом:

```
        MVI B, 30D      ; счетчик 30 циклов
M0:     MVI C, 40D      ; счетчик 40 циклов
M1:     Тело цикла
        DCR C
        JNZ M1
        DCR B
        JNZ M0
```

Временные задержки реализуются путем повторения циклов с известным временем выполнения. Так как микропроцессор КР580ВМ80А работает на частоте 2 МГц, то время выполнения одного такта составляет T = 0,5 мкс. Зная количество тактов, необходимых для выполнения определенных команд, можно рассчитать время выполнения любого участка программного кода:

```
MVI B, 100D ; 7 тактов, 100 повторений цикла
M0: NOP ; пустая операция 4 такта
NOP ; 4 такта
DCR B ; 5 тактов
JNZ M0 ; 10 тактов
```

В этом примере число тактов $N = 7+(4+4+5+10)\cdot 100 = 2307$. Тогда время выполнения составляет $\tau = 2307\cdot 0{,}5 = 1153{,}5$ мкс.

Для реализации задержек большей длительности программист может использовать двукратные циклы. Таким образом, нетрудно сформировать программную задержку определенной длительности:

```
MVI B, ... ; 7 тактов, … повторений цикла
M0: MVI C, 86D ; 7 тактов
```

```
M1: NOP ; 4 такта
NOP ; 4 такта
DCR C ; 5 тактов
JNZ M1 ; 10 тактов
DCR B ; 5 тактов
JNZ M0 ; 10 тактов
```

В этом примере число тактов N = B·(7+86·(4+4+5+10)+4+5+10) = B·2000. Для получения задержек на несколько секунд используют тройные циклы.

## 3. Программа выполнения работы

3.1. Изучить основные команды пересылки данных, логических и арифметических операций, организации ветвлений и циклов (выполняется в процессе домашней подготовки к лабораторной работе).

3.2. Изучить возможности эмулятора и экранного отладчика КР580 Emulator. Исследовать изменение в основных блоках процессора в ходе выполнения команд различных типов (выполняется в процессе домашней подготовки к лабораторной работе).

3.3. Составить блок-схему алгоритма функционирования программы в соответствии с заданным вариантом.

3.4. Реализовать ассемблерную программу в соответствии с заданным вариантом. Модифицировать программу, применяя различные виды команд, выполняющих одинаковые функции.

3.5. Исследовать длительности выполнения полученных программ в зависимости от используемых команд.

3.6. Сделать выводы по результатам проведенных исследований и расчетов.

Варианты заданий приведены в приложении А.

## 4. Содержание отчета

4.1. Цель и программа работы.

4.2. Блок-схема алгоритма программы в соответствии с заданным вариантом.

4.3. Листинги ассемблерных программы в соответствии с заданным вариантом.

4.4. Результаты проведенных исследований и расчетов.

4.5. Выводы по работе с анализом результатов выполненных исследований и расчетов.

## 5. Контрольные вопросы

5.1. Проведите классификацию команд ассемблера микропроцессора КР580ВМ80.
5.2. Расскажите о регистре флагов процессора КР580ВМ80. Какие команда влияют на состояние данного регистра, какие нет?
5.3. Приведите примеры команд логических операций ассемблера процессора КР580ВМ80 и назовите случаи их применения.
5.4. Расскажите о командах арифметических операций ассемблера процессора КР580ВМ80. Приведите примеры таких команд.
5.5. Какова роль счетчика команд в организации выполнения программы? Можно ли нарушить порядок изменения состояний программного счетчика?
5.6. Расскажите о командах ветвления ассемблера процессора КР580ВМ80. Приведите примеры таких команд с различными условиями.
5.7. Расскажите о принципах организации ветвлений в ассемблере процессора КР580ВМ80. Приведите примеры организации циклов с различными условиями останова.
5.8. Расскажите о командах логического и арифметического сдвигов, объясните разницу между ними. Приведите примеры выполнения сдвигов.
5.9. В чем заключается схожесть, а в чем отличие программного счетчика и указателя стека?
5.10. Каким образом можно обратиться к памяти с использованием косвенной адресации?
5.11. Какие команды используются для выполнения сложения двухбайтных чисел?

Лабораторная работа 3

# Исследования способов организации процесса ввода и вывода информации в 8-разрядном микропроцессоре

## 1. Цель работы

Исследовать способы подключения внешних устройств к 8-разрядному процессору, принципы организации обмена информацией между процессором и внешним устройством, работы с портами. Изучение основных команд работы с портами ассемблера процессора КР580ВМ80 и исследование воздействие их на порты и флаги.

## 2. Основные теоретические положения

К командам ввода-вывода МП КР580ВМ80 относятся команды IN<$A_1$> и OUT<$A_1$>. При выполнении команды IN<$A_1$> МП считывает число из входного устройства с адресом ($A_1$)($A_1$) и записывает его в аккумулятор. При выполнении команды OUT<$A_1$> МП записывает число из аккумулятора в выходное устройство с адресом ($A_1$)($A_1$). Так как адрес устройства указывается в одном байте команды, то с помощью этих команд МП может обмениваться информацией не более чем с 256 внешними устройствами.

В качестве простейших устройств ввода-вывода могут использоваться 8-разрядные регистры (например, многорежимный буферный регистр К589ИР12). Также в качестве устройств ввода-вывода могут применяться и более сложные схемы, например, программируемое устройство ввода-вывода в параллельном коде (КР580ИК55).

### 2.1 Простейшие устройства ввода-вывода

Схема подключения устройства ввода, выполненного на базе буферного регистра (БР) с переключателями (S0 – S7), приведена на рисунке 2.1. При замкнутом переключателе на вход регистра подается «0», а при разомкнутом «1». Переключатели используются для имитации передачи данных от внешнего устройства. На рисунке 2.1 приведена схема подключения выходного устройства, построенного на базе буферного регистра (БР). К выходу Q0 буферного регистра подключен динамик. Светодиоды, подключенные к выходам БР (Q4 – Q7), указывают число в двоичном виде, записанное в выходное устройство.

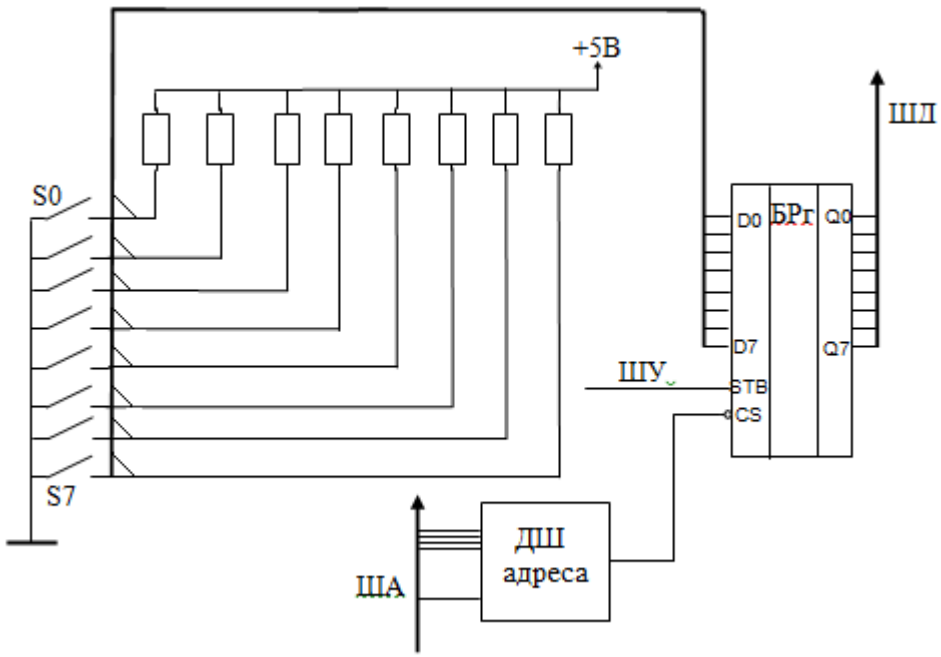



Рисунок 2.1 — Устройства ввода, выполненное на базе буферного регистра (БР) с переключателями (S)

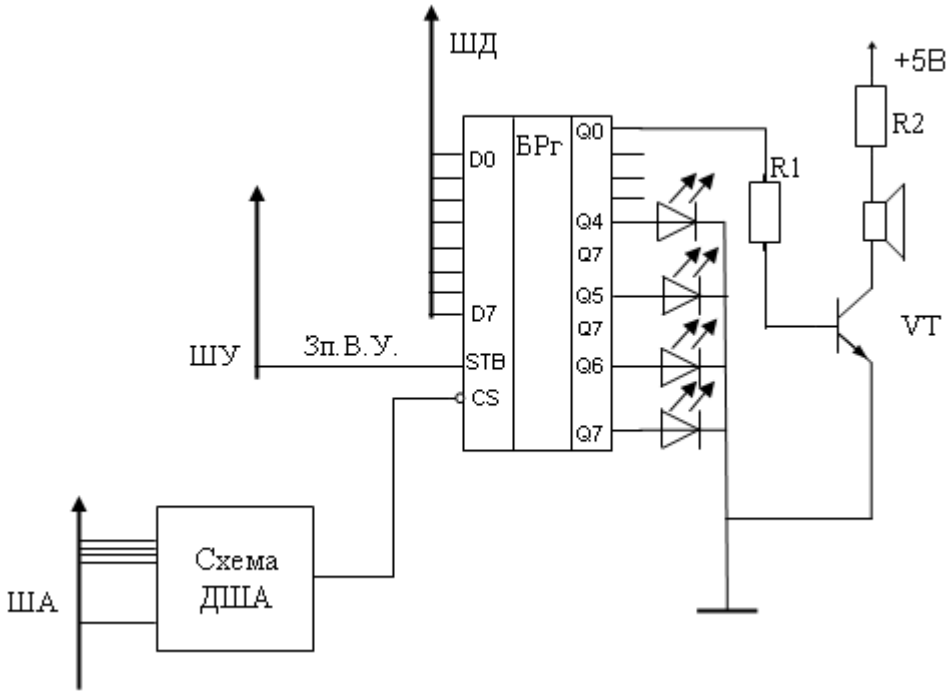



Рисунок 2.2 — Схема выходного устройства, построенного на базе буферного регистра (БРг) со светодиодами и динамиком

## 2.2 Вывод данных на семисегментный дисплей

В качестве устройства вывода информации, удобного для восприятия, часто используются дисплеи на базе индикаторов. Индикатор представляет собой восемь светодиодов с общим анодом в одном корпусе. Каждый индикатор имеет семь светодиодов для отображения сегментов цифр, а восьмой светодиод отображает десятичную точку (Рис.2.3). Индикатор может отображать цифры от 0 до 9, а также некоторые буквы.

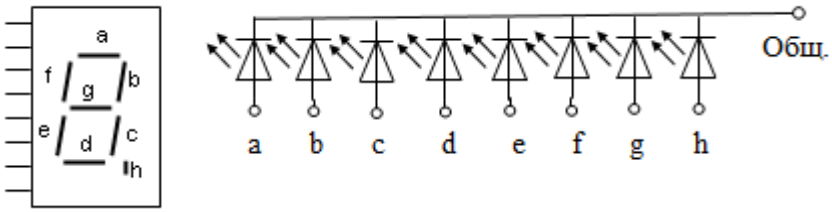



Рисунок 2.3 — Внешний вид и схема семисегментной светодиодной матрицы

Дисплей состоит из ячеек, каждая из которых представлена семисегментным индикатором. Вывод на дисплеи в микропроцессорных системах может осуществляться статическим или динамическим способом.

### 2.2.1 Организация статического режима работы дисплея

При статическом способе выводы сегментов каждого из индикаторов подключается к своему регистру. Для управления разрешением высвечивания символа на индикаторе используется отдельный регистр, причем анод каждого из индикаторов подсоединяется к соответствующему выходу этого регистра (рисунке 2.4).

Программа управления выводом информации на дисплей состоит из операции выдачи кода символа на соответствующий индикатор (регистр DSP) и вывода разрешающего сигнала на этот индикатор (регистр SKAN).

Одинаковые сегменты каждой ячейки индикатора связаны общей шиной и соединены регистром сегментов РгСг. Выходы анодов каждого из индикаторов подключены к регистру сканирования РгСк. Наличие уровня логической единицы в соответствующем разряде регистра сканирования РгСк приводит к высвечиванию символа в соответствующем индикаторе дисплея при наличии информации на шине данных.

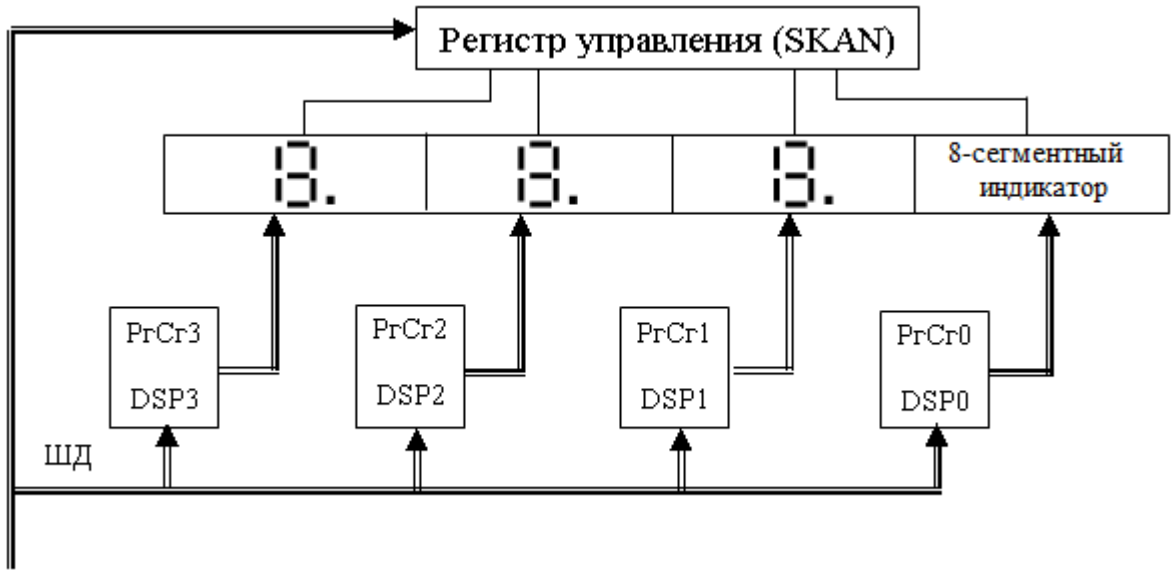



Рисунок 2.4 – Схема статической индикации данных

Вариант программы включения сегментов второй ячейки с помощью кода, задаваемого со входного регистра (порта ввода) имеет вид:

| Адрес | Метка | Мнемокод | Комментарий |
|---|---|---|---|
| 0800 | | MVI A,04 | Поместить в Акк число 00000100 |
| 2 | | OUT SKAN | Вывести число на РгСк и включить цифру 2 |
| 4 | M1 | IN 20 | Ввести данные в Акк из входного регистра |
| 6 | | OUT DSP2 | Записать их в регистр сегментов РгСг дисплея |
| 8 | | JMP M1 | Продолжить с метки M1 |

### 2.2.2 Организация мультиплексного режима работы дисплея

Схема подключения дисплея в мультиплексном режиме показана на рисунке 2.4. В этом режиме вывод информации на каждый индикатор дисплея выводится микроЭВМ последовательно. Цифра или символ на индикаторе высвечивается некоторый промежуток времени, задаваемый подпрограммой задержки.

Ниже приведен вариант программы обеспечивающей мультиплексный режим работы дисплея. Код цифр для вывода на каждую ячейку хранится в последовательных ячейках памяти с адресами 0900 – 0905. Код цифры для нулевой ячейки индикатора хранится в ячейке с адресом 0900. Начальный адрес подпрограммы временной задержки 0430.

| Адрес | Метка | Мнемокод | Комментарий |
|---|---|---|---|
| 0800 | | LXI B, 0100 | Загрузить в регистр B длительность задерж- |

| | | | |
|---|---|---|---|
| | | | ки |
| 03 | | XRA A | Очистить аккумулятор |
| 04 | M1 | LXI H, 0905 | Указать на адрес кода цифры 5 |
| 07 | | MVI D, 20 | Загрузить указатель цифры в регистр D |
| 09 | M2 | MOV A,M | Получить из ОЗУ код очередной цифры |
| 0A | | OUT DSP | Записать его в РгСг |
| 0C | | MOV A,D | Загрузить в аккумулятор указатель цифры |
| 0D | | OUT SKAN | Включить нужную цифру |
| 0F | | RAR | Указать на следующую цифру |
| 10 | | MOV D, A | Сохранить указатель цифры в регистре D |
| 11 | | CALL DEL | Вызвать подпрограмму временной задержки |
| 14 | | XRA A | Очистить аккумулятор |
| 15 | | OUT SKAN | Выключить цифры |
| 17 | | DCR L | Уменьшить на 1 содержимое регистра L |
| 18 | | ORA D | Все ли сообщения выведены? |
| 19 | | JNZ M2 | Если нет, то продолжить |

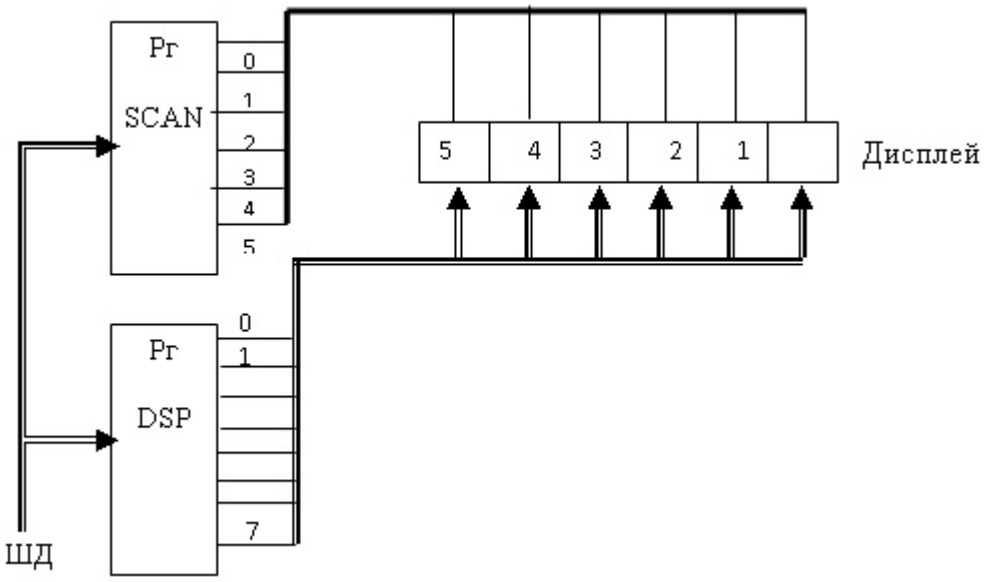



Рисунок 2.4 — Схема подключения индикаторов в динамическом режиме

## 3. Программа выполнения работы

3.1 Изучить команды работы с портами (выполняется в процессе домашней подготовки к лабораторной работе).

3.2 Изучить возможности эмулятора и экранного отладчика КР580 Emulator для работы с портами. Исследовать изменения в основных блоках процессора в ходе выполнения команд пересылки данных из

процессора во внешнее устройство и из внешнего устройства в процессор (выполняется в процессе домашней подготовки к лабораторной работе).

3.3 Составить структурную схему подключения внешнего устройства к процессору в соответствии с вариантом задания.

3.4 Составить блок-схему алгоритма функционирования программы в соответствии с заданным вариантом.

3.5 Реализовать программу в соответствии с вариантом.

3.6 Рассчитать длительность выполнения ассемблерной программы с учетом возможных ветвлений и циклов.

Варианты заданий приведены в приложении Б.

## 4. Содержание отчета

4.1 Цель и программа работы.

4.2 Структурная схема подключения внешнего устройства.

4.3 Блок-схема алгоритма программы в соответствии с вариантом.

4.4 Текст и листинг ассемблерной программы в соответствии с заданным вариантом.

4.5 Результаты проведенных исследований и расчетов.

4.6 Выводы по работе с анализом результатов выполненных исследований и расчетов.

## 5. Контрольные вопросы

5.1 Расскажите о классификации команд ассемблера процессора КР580ВМ80.

5.2 Расскажите о командах для работы с портами, обмена данными с внешними устройствами. Приведите примеры таких команд.

5.3 Начертите схемы подключения портов ввода и вывода к микропроцессору с цепями адресации.

5.4 Расскажите о назначении слова состояния машинного цикла, какую роль оно играет в процессе выполнениям команд работы с портами.

5.5 Объясните, с какой целью слово состояния машинного цикла записывается во внешний регистр. Расскажите когда и как это происходит.

5.6 Расскажите о методах организации вывода информации на 7-ми сегментные индикаторы.

5.7 Какие есть достоинства и недостатки динамического и статического методов вывода информации на 7-ми сегментные индикаторы.

5.8 Объясните устройство и принципы работы динамика.

5.9 Расскажите о принципах формирования звука с помощью динамика.

5.10 Напишите программу на языке ассемблера процессора КР580ВМ80 для формирования звуковых колебаний с помощью динамика с частотой 100 Гц.

5.11 Составьте схему и напишите программу включения и выключения бытового вентилятора, работающего от сети 220 В.

Лабораторная работа 4

# Исследование архитектуры 16-разрядных микропроцессоров и способов отладки ассемблерных программ в эмуляторе

## 1. Цель работы

Исследовать систему команд, архитектуру и основные блоки процессора Intel 8086 и взаимодействие основных блоков процессора при выполнении команд разных типов. Приобрести практические навыки написания ассемблерных программ и отладки их в эмуляторе микропроцессора — экранным отладчиком типа emu8086.

## 2. Основные теоретические положения

Основной архитектурной особенностью 16-разрядного процессора Intel 8086 (отечественный аналог К1810ВМ86) является 16-разрядная шина данных (ШД), 20-разрядная шина адреса, мультиплексированная с ШД, 16-разрядное АЛУ и 16-разрядные внутренние регистры. Для повышения быстродействия процессор 8086 логически разделен на два независимых блока, работающих параллельно: блок сопряжения с системной шиной BIU (*Bus Interface Unit*) и исполнительный EU (*Execution Unit*). Блок сопряжения считывает коды команд и операндов и сохраняет их в 6-байтовом конвейере команд, а исполнительный блок выбирает команды из конвейера, не дожидаясь, пока BIU доставит очередную команду.

Память в микро-ЭВМ на базе МП Intel 8086 логически организована как одномерный массив *байтов*, каждый из которых имеет адрес в диапазоне 0000 – FFFFFh. Любые два смежных байта могут рассматриваться как 16-битовое слово. Младший байт имеет меньший адрес, старший – больший. *Адресом слова считается адрес его младшего байта.*

В процессорах семейства Intel x86 применяется сегментная адресация памяти, при *которой* все пространство адресов делится на множество сегментов, т.е. пространство является сегментированным. Пространство памяти емкостью 1 Мбайт представляется как набор сегментов, определяемых программным путем. Сегмент состоит из смежных ячеек памяти и является независимой и отдельно адресуемой единицей памяти емкостью 64 Кбайт. Каждому сегменту системной программой назначается начальный (базовый) адрес, являющийся адресом первого байта сегмента в пространстве памяти. В

зависимости от вида хранимой информации различают три типа сегментов: для хранения кодов команд используется сегмент кода CS, в качестве стековой памяти используется сегмент стека SS, а для хранения данных служат сегменты DS и ES. Начальные адреса четырех сегментов, выбранных в качестве текущих, записываются в сегментные регистры CS, DS, SS, ES. Для обращения к команде и данным, находящимся в других сегментах, необходимо изменить содержимое сегментных регистров, что позволяет использовать все пространство памяти емкостью 1 Мбайт. Для хранения смещения в сегменте используются специальные регистры, выбор которых зависит от типа обращения к памяти (таблица 2.1).

Таблица 2.1 – Источники логического адреса

| Тип обращения к памяти | Сегмент (по умолчанию) | Вариант | Смещение |
|---|---|---|---|
| Выборка команд | CS | нет | IP |
| Стековые операции | SS | нет | SP |
| Переменная | DS | CS, SS, ES | EA |
| Цепочка-источник | DS | CS, SS, ES | SI |
| Цепочка-приемник | ES | нет | DI |

Команды всегда выбираются из текущего сегмента кода в соответствии с логическим адресом CS:IP. Стековые команды всегда обращаются к текущему сегменту стека по адресу SS:SP. Если при вычислении эффективного адреса EA используется регистр BP, то обращение производится также к стековому сегменту, а ячейки стекового сегмента рассматриваются как ОЗУ с произвольной выборкой. Операнды, как правило, размещаются в текущем сегменте данных, и обращение к ним организуется по адресу DS:EA. Однако программист может заставить МП обратиться к переменной, находящейся в другом текущем сегменте.

Значения *базы сегмента* и *смещения в сегменте представляют собой л*огический адрес. Базовый адрес и смещение в сегменте отображаются 16-разрядными числами. При обращении к памяти BIU на основании логического адреса формирует физический адрес следующим образом: значение базы сегмента смещается на четыре разряда влево, и полученное 20-разрядное число (с четырьмя нулями в младших четырех разрядах) складываются со

значением смещения в сегменте. Таким образом, база сегмента (с четырьмя нулями, добавленными в качестве младших разрядов) задает для памяти сегменты длинной 64 Кбайт, а значение сегмента в смещении – расстояние от начала сегмента до искомого адреса памяти. Максимально возможное смещение в сегменте равно 64 Кбайт.

МП содержит три группы регистров. К *первой группе* относятся РОН, используемые для хранения промежуточных результатов. Ко *второй* группе относятся *указатели и индексные регистры*, предназначенные для размещения или извлечения данных из выбранного сегмента памяти. Содержание этих регистров определяет значение смещения в сегменте при задании логического адреса. К *третьей* группе относятся *регистры сегментов*, задающие начальные адреса (базы) самих сегментов памяти. МП имеет регистр флаговых разрядов, используемых для указания состояния АЛУ при выполнении различных команд и управления его работой.

Таким образом в МП располагается тринадцать 16-разрядных регистров и девять флаговых разрядов АЛУ. Последний, тринадцатый регистр, называется указателем команд *IP (Instruction Pointer*). Он выполняет функции, аналогичные функциям программного счетчика в МП 8080 и не является программно доступным регистром. Доступ к его содержимому может быть осуществлен с помощью команд передачи управления.

Группа РОН включает в себя семь 8-разрядных регистров МП 8080 и один добавленный 8-разрядный регистр для того, чтобы объединить все регистры в четыре 16-разрядные пары. К ним может быть организован программный доступ как к 8- или 16-разрядным регистрам. 16-разрядные регистры обозначаются символами AX, BX, CX и т.д. При обращении к ним, как к 8-разрядным регистрам, их обозначают AL, AH, BL, BH и т.д. Все эти регистры могут быть использованы при выполнении арифметических и логических команд. Однако есть команды, использующие определенные регистры для специфических целей, тогда применяют мнемонические обозначения: аккумулятор (*accumulator*), база (*base*), счет (*count*), данные (*data*).

Группа указателей и индексных регистров состоит из четырех 16-разрядных регистров: регистры указатели SP – Stack Pointer и BP – Base Pointer; индексные регистры SI – Source index и DI – Destination index. Обычно эти регистры содержат информацию о смещении по адресам в выбранном сегменте и позволяют компактно писать программы каждый раз непосредственно, не приводя используемого адреса. С их помощью производится вычисление адресов программ. Чаще всего в регистрах *указателях* записано ад-

ресное смещение по отношению к *стековому* сегменту, а в *индексных* регистрах – адресное смещение по отношению к *сегменту данных*.

Регистр состояния (Флаговый регистр) содержит шестнадцать триггеров, из которых используется только 9. Эти триггеры отображают состояние процессора при выполнении последней арифметической или логической команды. Пять триггеров аналогичны МП 580-й серии. Дополнительные триггеры сигнализируют: OVERFLOW – переполнение; DIRECTION – направление – указывает направление работы с массивами (автоматическое увеличение или уменьшение на единицу адреса массива); INTERRUPT – прерывание – определяет для МП реагирования на прерывание; TRAP – (западня, ловушка) устанавливает для МП пошаговый режим.

Система команд ВМ86 содержит 91 мнемокоманду и позволяет совершать операции над байтами, двухбайтовыми символами, отдельными битами, а также цепочками байтов и слов. По функциональному признаку система команд МП 8086 разбивается на 6 групп: пересылка данных; арифметические операции; логические операции и сдвиги; передача управления; обработка цепочек; управление процессором.

## 3. Описание лабораторной установки

Лабораторный стенд для исследования архитектуры и способов программирования на языке ассемблера 16-разрядных микропроцессоров состоит из персонального компьютера, на котором установлен программный эмулятор 16-разрядного микропроцессора типа Intel 8086 (отечественный аналог МП КР1810). Эмулятор отображает на экране персонального компьютера программную модель исследуемого процессора, а также позволяет создавать и редактировать тексты программ на языке ассемблера МП 8086, выполнять их ассемблирование и исследование процессов модификации регистров процессора, дампов памяти и портов в пошаговом и реальном режимах отладки программ.

### 3.1 Особенности работы с экранным отладчиком emu8086

При запуске программы появляется окно приглашения (рисунок 3.1), в котором пользователю предлагается выбрать один из вариантов работы: создание нового файла; просмотр примеров программирования; запуск уже использовавшихся файлов или переход в режим помощи.

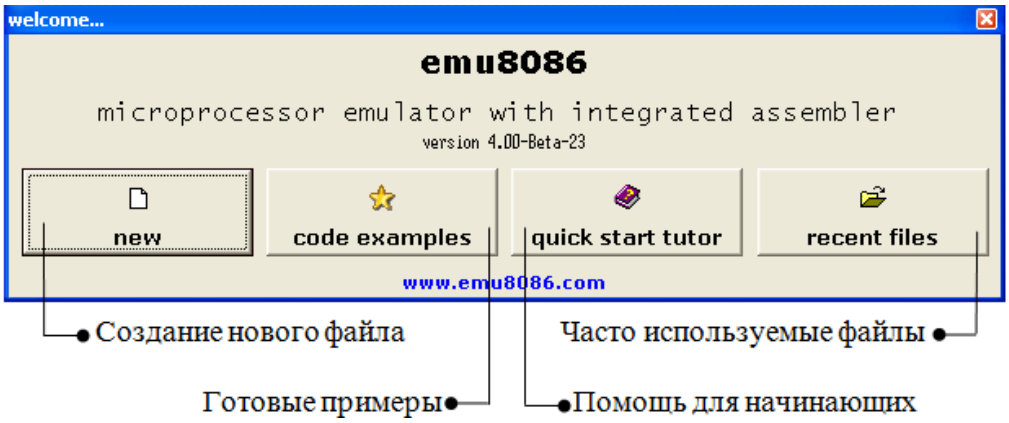



Рисунок 3.1 — Окно приглашения

При создании нового файла (рисунок 3.2) существует возможность выбрать шаблон файла.

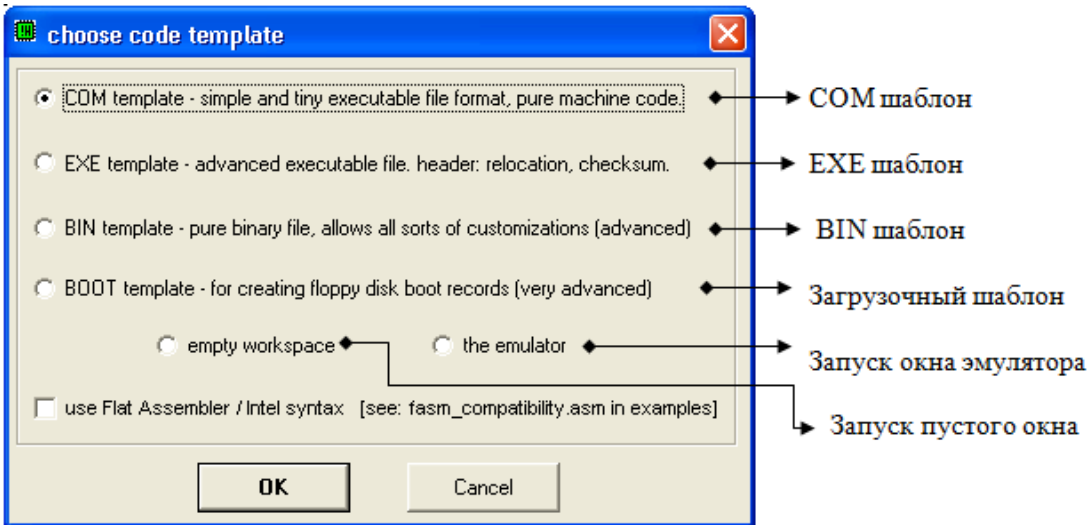



Рисунок 3.2 — Окно выбора шаблона

Например, при выборе варианта шаблона COM-файла, появится окно редактора с соответствующим стандартным кодом (рисунок 3.3). Окно редактора можно разделить на область панели инструментов и рабочую область, используемую для написания и редактирования ассемблерных программ.

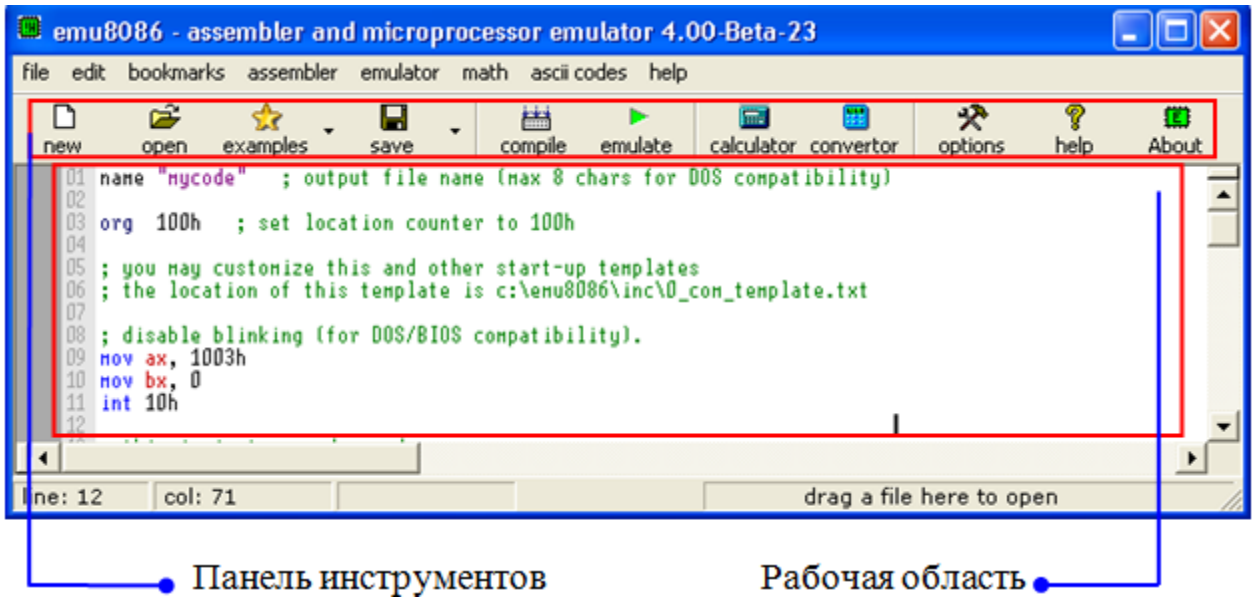



Рисунок 3.3 — Окно создания и редактирования кода

Панель инструментов позволяет осуществить быстрый доступ к самым часто используемым функциям. Пункты меню панели инструментов имеет следующее назначение:

new — Создание нового файла. Создание и ввод текста программы на ассемблере для последующего выполнения. Доступно использование комментариев – перед текстом комментария необходимо ставить “;”.

open — Открытие уже созданного и сохраненного файла для его редактирования или запуска.

examples — Открытие примера. Открывает один из доступных файлов, созданных разработчиками для показа возможностей функций ассемблера.

save — Сохранение на носитель информации файла, который вы создали или отредактировали.

compile —Создание исполняемого файла. Создание файла, который не будет зависеть от эмулятора и будет запускаться из операционной системы без дополнительных программ (*.exe, *.com).

emulate — Запуск программы в режиме эмуляции, в котором возможно отследить за каждым шагом выполнения программы, за содержимым ре-

гистров, флагов, ячеек памяти. Используется при проверке на работоспособность программы и при необходимости отыскать ошибки в коде.

calculator — Запуск встроенного калькулятора, который позволяет производить расчеты над числами в двоичной, восьмеричной, десятеричной, шестнадцатеричной формах. Результат можно выводить так же в любой удобной форме. Вид калькулятора представлен на рисунке 3.4.

convertor — Запуск конвертера основ (двоичная, восьмеричная, десятичная, …). Конвертер основ счисления позволяет легко и удобно переводить числа из одной основы в другую, т.е. осуществлять преобразования типа: bin->hex, dec->bin и т.п. Вид конвертера основ счисления показан на рисунке 3.5.

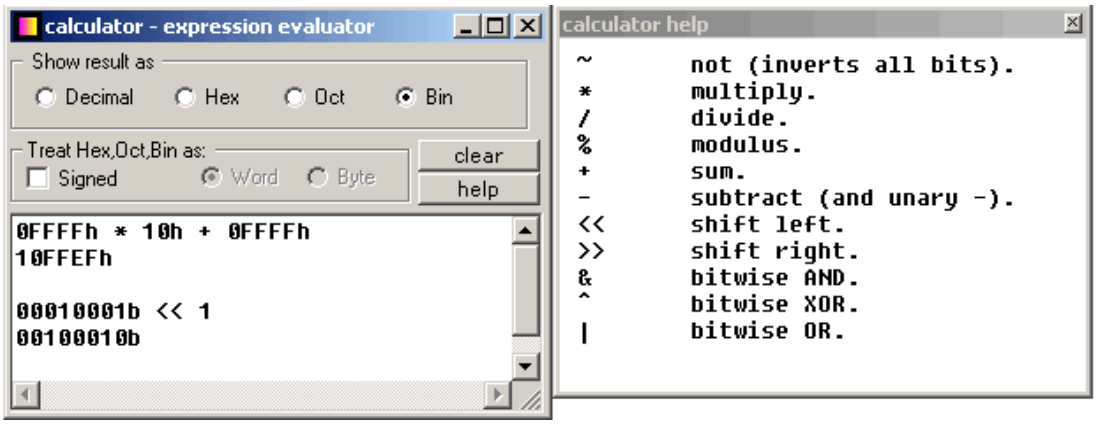



Рисунок 3.4 — Окно калькулятора и окно помощи для работы с ним

options — Настройки эмулятора. Позволяют настроить цвет и шрифт кода, комментариев, выделения, фона, панелей и других компонентов эмулятора.


Рисунок 3.5 — Окно конвертера

help — Вызов помощи (на англ. языке). Полное описание на английском языке функций работы с ассемблером, прерываниями, циклами и т.д. Сайт создателей программы и часто задаваемые вопросы(FAQ) в режиме online.

About — Информация о создателях. Регистрация продукта, информация о «координатах» создателей, версии и дате создания программы.

3.2 Работа отладчика в режиме эмуляции.

Общий вид главного окна экранного отладчика показан на рисунке 3.6. В левой части окна изображены программно доступные регистры микропроцессора 8086 и значение их содержимого в 16-ричной системе счисления.

В правой части окна выведен текст исследуемой программы в мнемонических кодах, а в центре отображается этот же фрагмент в машинных кодах (в 16-ричной системе счисления). Как видно из рисунка, программа размещена в сегменте с базовым адресом 0700h и начальном смещении 0100h.

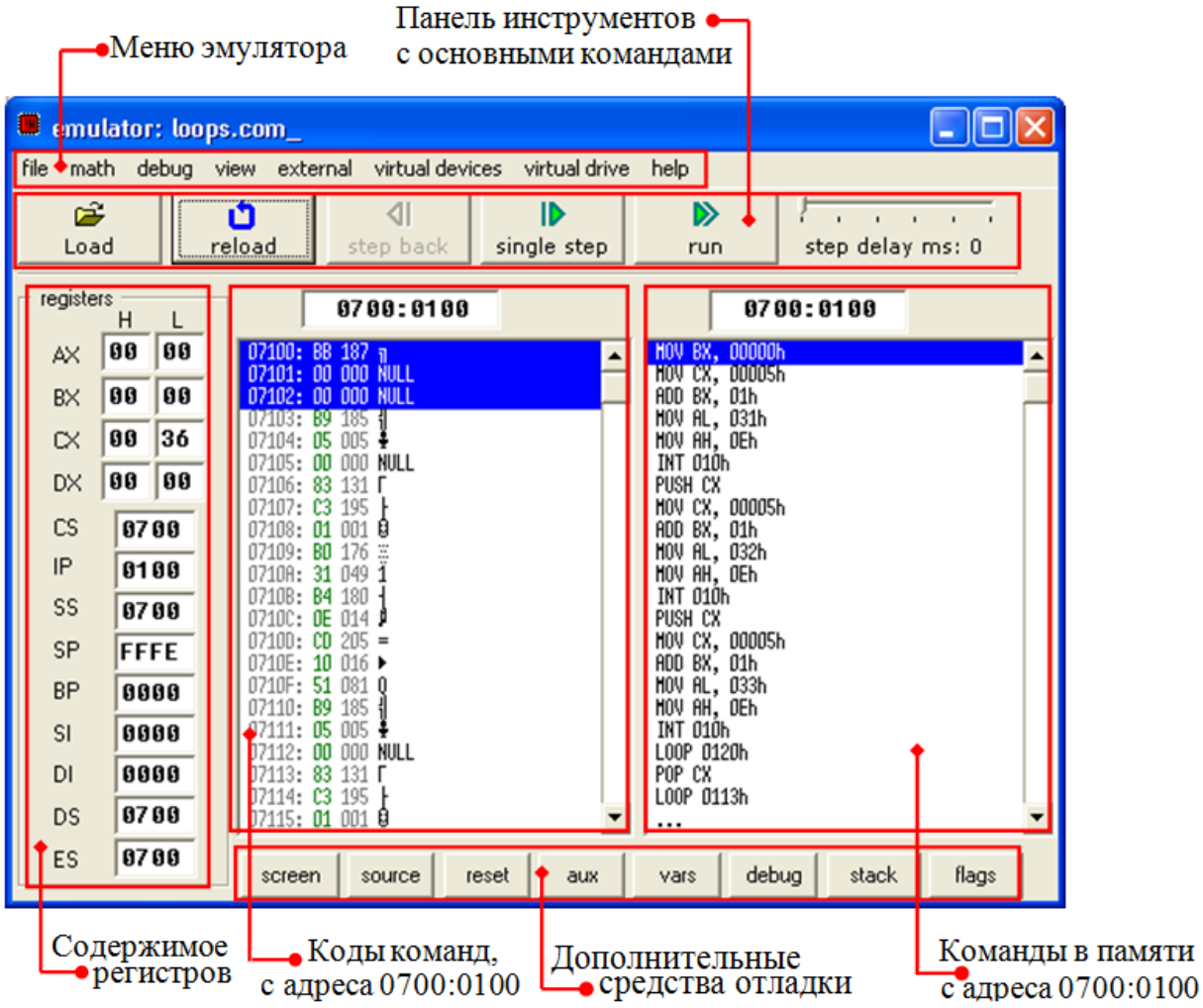



Рисунок 3.6 — Вид главного окна эмулятора

Двойным щелчком по любому адресу или содержимому регистров вызывается расширенный просмотр значений (рисунок 3.7). На панели инструментов в верхней части данного окна имеются следующие клавиши:

- Load —загрузка существующего файла для его эмуляции;
- Reload — выполнение программы с самого её начала;

- Step back — шаг назад на одну операцию при пошаговом режиме;
- Single step — шаг вперёд на одну операцию при пошаговом режиме;

Run — эмуляция без остановок между операциями.

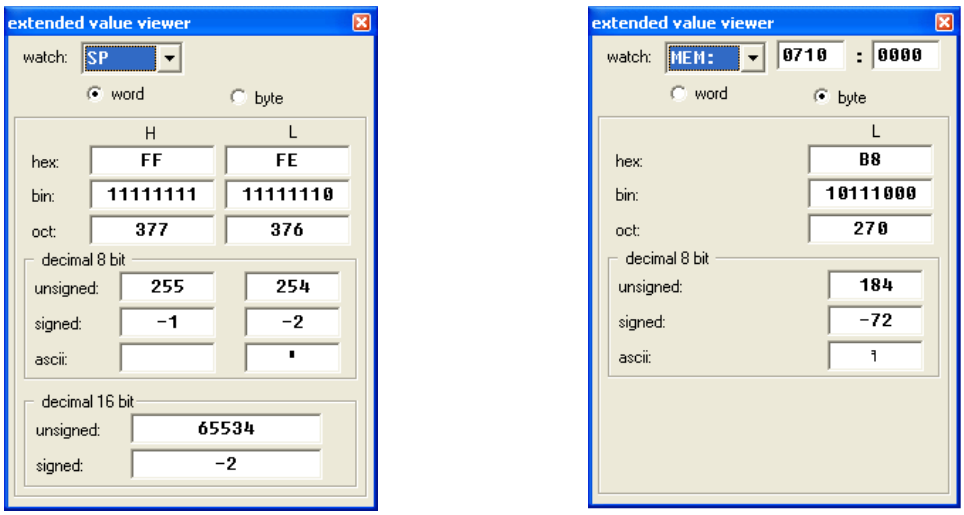
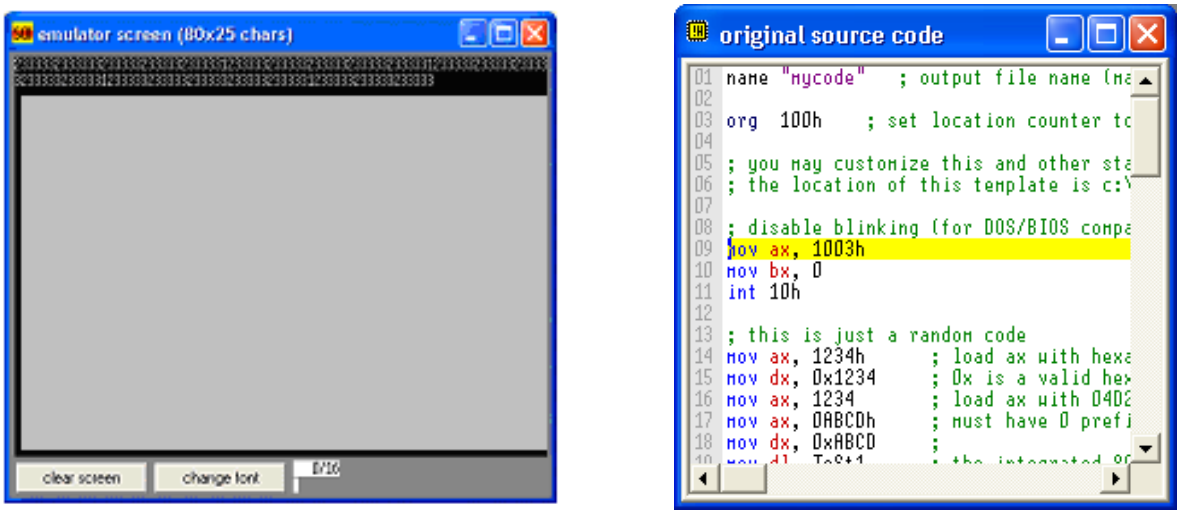



Рисунок 3.7 — Пример окон просмотра значений регистров

Главное окно эмулятора также предоставляет дополнительные возможности для отладки программ, которые инициируются путем нажатия одной из клавиш в нижней части главного окна, в частности:

**screen** — Вызов эмулируемого экрана. Если в программе используется работа с экраном (монитором), то выполнение этих операций отображается на эмулируемом экране, показанном на рисунке 3.8а.




а) б)

Рисунок 3.8 — Окно эмулирующее вывод данных на экран монитора (а) и окно просмотра исходного кода программы

**source** — Вызов окна просмотра оригинального кода. Для ориентации в своей программе и в кодах можно использовать окно, содержащее её оригинальный текст. Пример такого окна показан на рисунке 3.8,б.

**reset** — Сброс всех значений регистров. Тем самым осуществляется выполнение программы с начала. При этом отчищается эмулируемый экран.

**aux** — Пункт меню предоставляет дополнительные возможности для просмотра памяти, АЛУ, сопроцессора и др.

Окно памяти программ и данных **memory**. Используется для удобного просмотра содержимого памяти с произвольным доступом (рисунок 3.9)

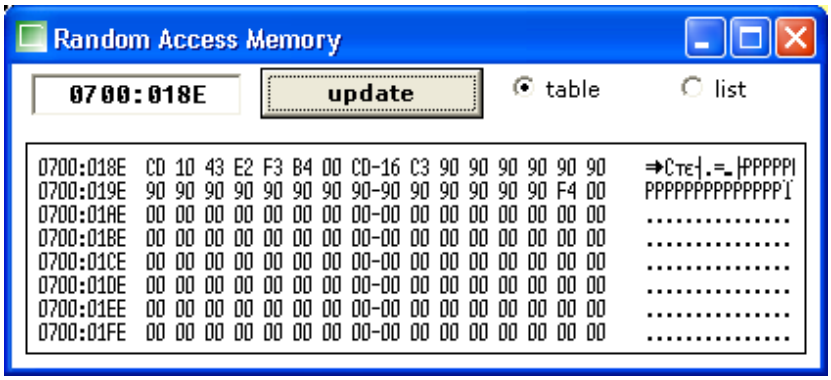



Рисунок 3.9 — Окно просмотра содержимого памяти

В данном окне можно просматривать последовательность располагаемых в памяти данных и перемещаться по памяти, вводя значение адреса вместо показанного на рисунке 0700:018E. Можно выбрать способ просмотра в виде таблицы или списка путём выбора table или list соответственно.

Окно **ALU** — отображает содержимое арифметико-логического устройства. На рисунке 3.10 показан пример работы АЛУ при сложении: первая строка указывает номер разряда, две следующие строки – это операнды, а последняя – результат.

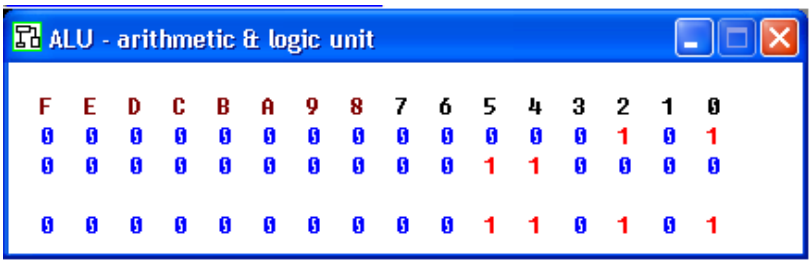



Рисунок 3.10— Окно просмотра содержимого ALU

В окне **FPU** — отображается содержимое регистров математического сопроцессора.

Окно **stop on condition** —необходимо использовать, если при отладке требуется остановить эмуляцию при каком-то условии значения регистров общего назначения, флагов или ячейки памяти. Вид этого окна показан на рисунке 3.11.

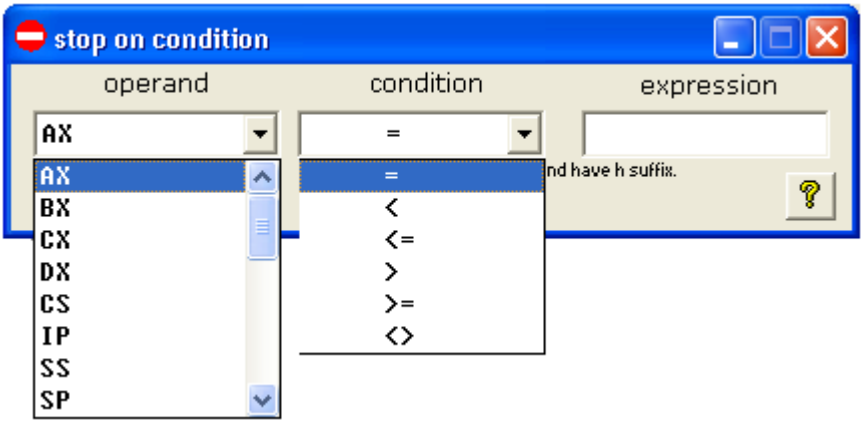
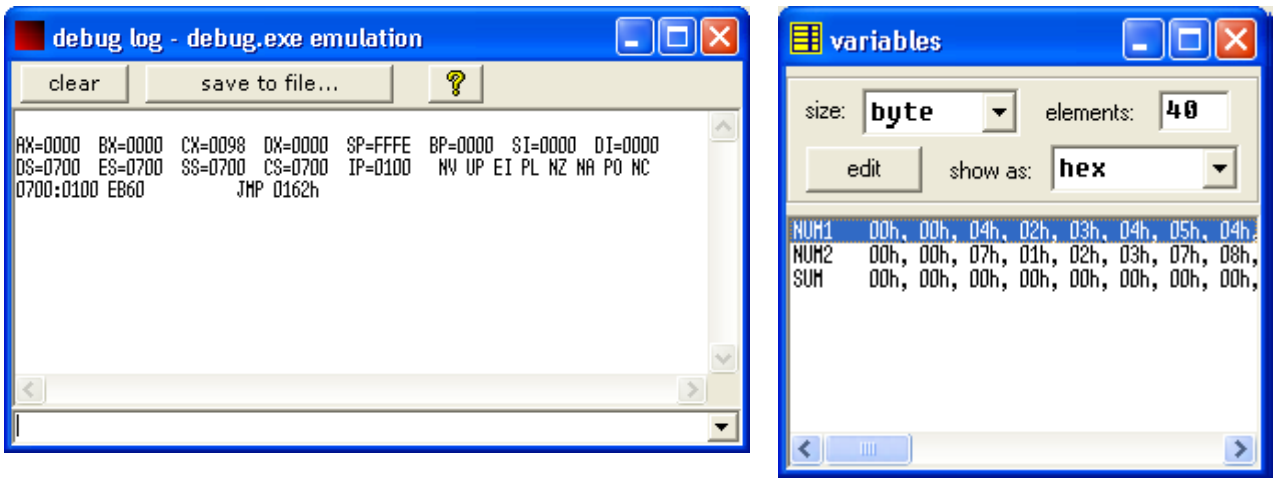



Рисунок 3.11 - Вид окна «остановка по условию»

Таблица символов **symbol table**. В этой таблице отображаются названия и параметры символов. В окне **listing** — отображается программа в машинных кодах. Вся эмулируемая программа представляется в виде кодов команд и адресов, по которым они располагаются.

Журнал отладки **debag** — сохраняет каждое изменение в ходе выполнения программы (рисунок 3.12а).




а) б)

Рисунок 3.12 — Окно журнала отладки а) и окно просмотра переменных б)

**vars** — Окно просмотра переменных (рисунок 3.12,б). Можно осуществить быстрый и прямой доступ ко всем переменным. В случае необходимости можно изменить их значение двойным щелчком мыши по необходимой переменной

Окно **stack** — оказывает возможность просмотреть содержимое стека или его изменения.

Окно состояние флагов **flags**. Позволяет просмотреть состояние всех флагов и при необходимости изменить их. При нажатии на кнопку “analyse”, появляется окно “lexical flag analyser” (рисунок 3.13), в котором расписывается значение каждого флага процессора.

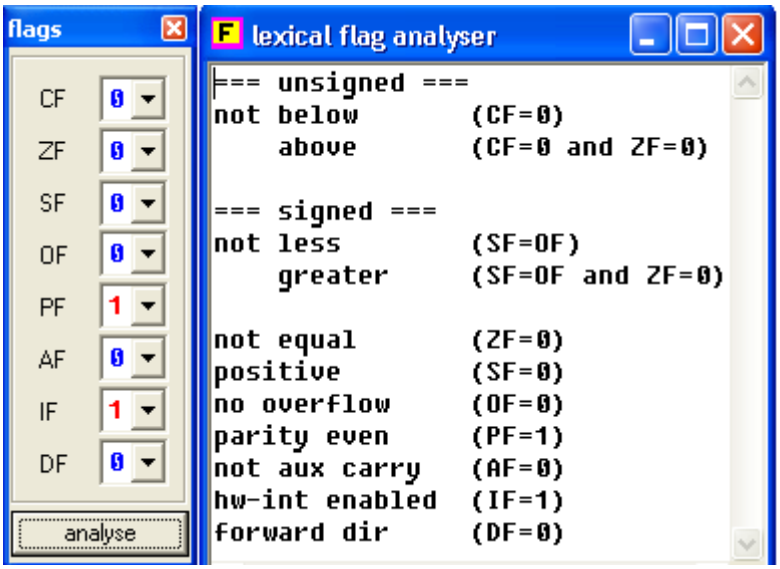



Рисунок 3.13 – Окно просмотра состояния флагов

## 4. Программа работы

4.1. Изучить архитектуру МП 8086, состав регистров и работу процессора с использованием временных диаграмм (выполняется в процессе домашней подготовки к лабораторной работе).

4.2. Изучить основные команды МП 8086 (выполняется в процессе домашней подготовки к лабораторной работе).

4.3. Исследовать процесс выполнения программы (т.е. проследить изменение содержимого регистров) сложения-вычитания, взятой из примера программ отладчика (code examples).

4.4. Исследовать процесс выполнения программы сложения элементов двух массивов (add_two_arrays), взятой из примера программ отладчика (code examples).

4.5. Рассчитать время выполнения программ.

**ПРИМЕЧАНИЕ**: В этих программах используются прерывания DOS и BIOS, а сами программы представлены в форматах com или exe. Поэтому, прежде чем начать исследование примеров ассемблерных программ, внимательно ознакомьтесь с методическими указаниями по выполнению данной работы.

## 5. Методические указания по выполнению работы

Исполняемые модули ассемблерных программ могут быть оформлены в виде exe- com-файлов. Эти файлы, кроме собственно исполняемого кода приложения, должны содержать информацию для операционной системы (DOS) о размещении программы в памяти компьютера, а также команды. которые обеспечат возврат в операционную систему после завершения приложения. Отличие форматов состоит в следующем.

**Размер программы**. Программа в EXE может иметь любой размер, в то время как COM-файл ограничен размером одного сегмента (≤64к). Размер COM-файла всегда меньше, чем размер соответствующего EXE-файла, так как в COM-файле отсутствует заголовок (512 байт) EXE-файла.

Заголовок хранится на диске в начале exe-файла. В заголовке содержится информация о размере выполняемого модуля, области загрузки в памяти, адрес стека и относительное смещение. В нем также указывается число байтов в последнем блоке exe-файла, число настраиваемых параметров, количество параграфов в заголовке и некоторые другие данные.

**Сегмент стека.** В EXE программисту следует задавать сегмент стека, в то время как COM-программа генерирует стек автоматически.

**Сегмент данных.** В EXE-программах обычно определяется сегмент данных, а регистр DS инициализируется адресом этого сегмента. В COM-программах все данные должны быть определены в сегменте кода.

**Инициализация.** В EXE-программе следует записать 0-слово в стек и инициализировать регистр DS. Так как в COM стек и сегмент данных не определены, то эти шаги отсутствуют.

При запуске COM-программы DOS заносит в сегментные регистры адрес префикса программного сегмента (256 байт=100H), который резервируется DOS непосредственно перед COM- или EXE –программой в памяти. Так как адресация начинается со смещения 100H от начала префикса, то в программе после директивы SEGMENT следует вносить директиву ORG 100H.

Для преобразования EXE-файла в COM-файл используется программа EXE2BIN.

Для облегчения работы программиста разработан ряд стандартных служебных системных программ взаимодействия с клавиатурой, дисплеем и др. устройствами ввода/вывода данных. Эти программы (функции) входят в состав операционной системы (функции DOS) и функции BIOS ─ базовой системы ввода/вывода (Basic Input/Output System). Вызов этих функций осуществляется путем команд прерывания INT.

Все функции DOS вызываются с помощью прерывания 21h (в десятичной нотации 33). Выбор конкретной функции осуществляется путем записи соответствующего номера в регистр **AH**.

**Функция 02: вывод одного символа на экран дисплея**

Для вывода одного символа на экран ПК используется функция 02 прерывания 21h:

```
mov DL, <код выводимого символа>
mov AH, 2
int 21h
```

Выводимый символ высвечивается в позиции курсора (что бы там ни было записано), после чего курсор сдвигается на одну позицию вправо.

**Функция 9: вывод строки на экран дисплея**

Для вывода на экран строки (последовательности символов) можно, конечно, использовать функцию 02, однако сделать это можно и за один прием с помощью функции 09 прерывания 21h:

```
DS:DX := начальный адрес строки
mov AH, 9
int 21h.
```

Перед обращением к этой функции в регистр DS должен быть помещен номер того сегмента памяти, в котором находится выводимая строка, а в регистр DX ─ смещение строки внутри этого сегмента. При этом в конце строки должен находиться символ $ (код 24h), который служит признаком конца строки и сам не выводится.

**Функция 4Ch: завершение программы**

Завершив все свои действия, прикладная программа обязана вернуть управление операционной системе, чтобы пользователь мог продолжить работу на ПК. Такой возврат реализуется функцией 4Ch прерывания 21h, которую помещают в конце программы:

mov AL, <код завершения>
mov AH, 4Ch
int 21h

Каждая программа обязана сообщить, успешно или нет она завершила свою работу. Так как любая программа вызывается из какой-то другой программы (например, из операционной системы), и иногда вызвавшей программе, чтобы правильно продолжить работу, надо знать, выполнила ли вызванная программа задачу верно, или она проработала с ошибкой. Такая информация передается в виде кода завершения программы (некоторого целого числа), который должен быть нулевым, если программа проработала правильно, и ненулевым (каким именно - оговаривается в каждом случае особо) в противном случае. (Узнать код завершения вызванной программы можно с помощью функции 4Dh прерывания 21h.) Потребуется этот код или нет, программа все равно должна выдать его.

Для работы с клавиатурой используются функции BIOS, вызываемые прерыванием INT 16h. Они включает в себя функции для выборки кода нажатого символа из буфера с ожиданием нажатия, функции для проверки содержимого буфера и для управления содержимым буфера, функции для изменения скоростных характеристик клавиатуры.

Функция 00h выполняет чтение кода символа из буфера клавиатуры, если он там есть. Если *буфер клавиатуры пуст*, программа переводится в состояние *ожидания* до тех пор, *пока не будет нажата какая-нибудь клавиша*. Скан-код и ASCII-код нажатой клавиши передаются программе.

## 6. Содержание отчета

6.1. Цель и программа работы.
6.2. Структурная схема МП 8086 и временные диаграммы его функционирования.
6.3. Описание взаимодействия блоков микропроцессора при выполнении команд различной длины и различных типов.
6.4. Тексты исследуемых программ с построчными комментариями.
6.5. Результаты проведенных исследований и расчетов.
6.6. Выводы по работе с анализом результатов выполненных исследований и расчетов.

## 7. Контрольные вопросы

7.1. Расскажите о составе и назначении основных блоков процессора Intel 8086.

7.2. Поясните, за счет чего повышено быстродействие процессора 8086 по сравнению с его предшественником.

7.3. Объясните понятие машинного цикла, перечислите виды машинных циклов МП 8086 и поясните, какие сигналы и в какой последовательности появляются на выводах процессора в каждом из циклов.

7.4. Перечислите основные внешние выходы МП КР1810, расскажите об их назначении.

7.5. Расскажите о флагах процессора и особенностях их использования.

7.6. В чем состоит отличие логического и физического адресов и как формируется физический адрес?

7.7. Какова роль указателя стека в организации выполнения программы и каково его значение при выполнении первой команде Push?

7.8. Поясните целесообразность включение в состав процессора индексных регистров и приведите пример программы с их использованием.

7.9. Расскажите подробно о работе процессора после включения питания.

7.10.В чем состоит особенность работы процессора при поступлении сигнала прерывания от внешнего устройства?

7.11.Проведите сравнительную характеристику различных способов адресации, используемых в МП 8086.

7.12.В чем состоит отличие работы процессора в минимальном и максимальном режимах?

7.13.Расскажите об основных возможностях экранного отладчика emu8086.

7.14.Расскажите о режимах исполнения отдельных команд и целых программ в экранном отладчике emu8086.

7.15.Прокомментируйте результат действия каждой из команд в программе сложение-вычитание операндов.

7.16.Опишите возможности взаимодействия микропроцессора с внешними устройствами реализованные в экранном отладчике emu8086.

7.17. Для чего используются внутренние прерывания DOS и BIOS?

7.18. Расскажите об основных функциях BIOS и DOS и их назначении.

7.19. В чем состоит особенность оформления ассемблерных программ в виде exe- и com-файлов?

## Список рекомендованной литературы

1. Абель П. Язык Ассемблера для IBM PC и программирования / Пер. с англ. Ю.В.Сальникова. — М.: Высш. школа, 1992. — 447 с.

2. Новиков Ю.В. Основы микропроцессорной техники: Учебное пособие/Ю.В. Новиков, П.К. Скоробогатов. — М.: Интернет-университет информационных технологий; БИНОМ, 2006. — 359 с.

3. Программирование на ассемблере для процессоров персонального компьютера / М.К. Маркелов, Л.С. Гурьянова, А.С. Ишков, А.С. Колдов, С.В. Волков.— Пенза: ПГУ, 2013 .— ISBN 978 -5-94170-537-5 http://rucont.ru/efd/210624?cldren=0

4. Федотова Д.Э. Архитектура ЭВМ и систем [Электронный ресурс]: лабораторная работа. Учебное пособие/ Федотова Д.Э.— Электрон. текстовые данные.— М.: Российский новый университет, 2009.— 124 с.— Режим доступа: http://www.iprbookshop.ru/21263

5. Чернега В.С. Архитектура вычислительных устройств информационных систем. Конспект лекций / В.С. Чернега. – Севастополь: Изд-во СевГУ, 2020 – 160 с.

# ПРИЛОЖЕНИЕ А

1. Сформировать последовательность целых чисел соответствующих Числам Фибоначчи длиной N элементов. Параметр N задается преподавателем.

2. Осуществить сортировку массива натуральных чисел, используя алгоритм сортировки подсчетом. Размер массива и направление сортировки задается преподавателем.

3. Суммировать элементы массива расположенные на позициях кратных трем. Размер массива задается преподавателем.

4. Переписать элементы массива в обратном порядке. Размер массива задается преподавателем.

5. Поменять местами две половины массива. Размер массива задается преподавателем.

6. Из исходно массива скопировать в результирующий массив те элементы, которые удовлетворяют следующим условиям: являются четными, лежат в диапазоне от X1 до X2, имеют в двоичном представлении единицу в третьем разряде. Размер массива и границы диапазона задаются преподавателем.

7. Найти номера L минимальных элементов массива. Размер массива и количество минимальных элементов L задаются преподавателем.

8. Найти минимальный и максимальный элементы в массиве. Размер массива задается преподавателем.

9. Каждый элемент массива модифицировать следующим образом: если нечетный — заменить элемент на его обратный код, если четный или в двоичном представлении разряды 5,4,3,2 равны единице заменить элемент на ноль, иначе оставить без изменений. Размер массива задается преподавателем.

10. Сформировать массив из номеров разрядов двоичного представления заданного числа равных нулю. Исследуемое число задается преподавателем.

11. Из исходного массива сформировать результирующий массив из номеров позиций исходного массива, элементы которых равны заданной величине. Размер массива задается преподавателем.

12. Из исходно массива скопировать в результирующий массив те элементы, которые содержат больше трех единиц в двоичном представлении. Размер массива задается преподавателем.

13. Найти суммы по модулю два всех элементов массива стоящих на четных и нечетных позициях. Большее из полученных значений записать по адресу расположенному после массива. Размер массива задается преподавателем.

14. Методом табличного преобразования осуществить перекодирование массива исходных данных.

# ПРИЛОЖЕНИЕ Б

1. Начертить структурную схему подключения N светодиодов к портам вывода микропроцессорного модуля. Написать программу осуществляющую одновременное мигание всех N светодиодов с частотой F герц. Значение параметров N и F задается преподавателем.

2. Начертить структурную схему подключения 4 светодиодов к порту вывода микропроцессорного модуля. Написать программу осуществляющую мигание светодиодов с частотами F, 2*F, 4*F, 8*F Герц соответственно. Значение параметра F задается преподавателем.

3. Начертить структурную схему подключения 8 светодиодов к порту вывода микропроцессорного модуля. Написать программу осуществляющую вывод значения заданной величины в двоичном коде. Значение отображаемой величины задается преподавателем.

4. Начертить структурную схему подключения 8 светодиодов к порту вывода микропроцессорного модуля. Написать программу реализующую эффект «бегущий огонек» с частотой F Герц. Значение параметра F задается преподавателем.

5. Начертить структурную схему подключения динамика к порту вывода микропроцессорного модуля. Написать программу осуществляющую генерацию звука с частотой F Герц. Значение параметра F задается преподавателем.

6. Начертить структурную схему подключения трех динамиков к портам вывода микропроцессорного модуля. Написать программу генерации звука частотами F1, F2, F3 соответственно одновременно. Значения параметров F1, F2, F3 задаются преподавателем.

7. Начертить структурную схему подключения динамика к порту вывода, а также двух кнопок к портам ввода микропроцессорного модуля. Написать программу осуществляющую опрос нажатия кнопок и при нажатии одной кнопки генерация звука частотой F1, при нажатии двух кнопок частотой F2 Герц. Значения параметров F1, F2 задаются преподавателем.

8. Начертить структурную схему подключения 8-ми сегментного индикатора к порту вывода микропроцессорного модуля. Написать программу осуществляющую вывод чисел от 0 до 9. Длительность задержки каждого числа равна t секунд. Значение параметра t задается преподавателем.

9. Начертить структурную схему подключения 8-ми сегментного индикатора к порту вывода микропроцессорного модуля. Написать програм-

му осуществляющую вывод числа, предварительно введенного из порта ввода. Значение вводимого числа и номер порта ввода задаются преподавателем.

10. Начертить структурную схему подключения восьми 8-ми сегментного индикаторов к портам вывода микропроцессорного модуля, с использованием статического и динамического режима вывода. Написать программу осуществляющую вывод текущей даты в статическом и динамическом режимах.

11. Начертить структурную схему подключения 8-ми сегментного индикатора к порту вывода, а также восьми кнопок к порту ввода микропроцессорного модуля. Написать программу осуществляющую вывод номера нажатой кнопки. При нажатии нескольких кнопок одновременно применить приоритетный режим шифрации.

12. Начертить структурную схему подключения 16 кнопок к портам ввода микропроцессорного модуля. Написать программу считывания данных с портов и определения, в каком из портов больше нажатых кнопок.

13. Начертить структурную схему подключения 8 кнопок к порту ввода микропроцессорного модуля. Написать программу считывания данных с порта и определения количества нажатых кнопок, сохранить результат в памяти.

14. Начертить структурную схему подключения 8 кнопок к порту ввода микропроцессорного модуля. Написать программу считывания данных с порта и определения четное или нечетное количество кнопок нажато, сохранить результат в памяти.

15. Начертить структурную схему подключения 8 кнопок к порту ввода микропроцессорного модуля. Написать программу считывания данных с порта и определения факта нажатия кнопок с заданными номерами, сохранить результат в памяти. Номера анализируемых кнопок и их количество задаются преподавателем

Заказ № _____ от «___» _____ 2021 г. Тираж _____ экз.

Изд-во СевГУ