Министерство науки и высшего образования Российской Федерации

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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине

"Технические средства информационных систем"

для студентов, обучающихся по направлению

09.03.02 “Информационные системы и технологии”

очной и заочной форм обучения

Севастополь
2019

УДК 004.732

Методические указания к лабораторным занятиям по дисциплине "Технические средства информационных систем". Часть 2 / Сост. Чернега В.С., Дрозин А.Ю. — Севастополь: Изд-во СевГУ, 2019— 26 с.

Методические указания предназначены для проведения лабораторных работ по дисциплине “Технические средства информационных систем“. Целью методических указаний является помощь студентам в выполнении лабораторных работ по исследованию архитектуры 16-разрядных процессоров и персональных ЭВМ, а также по программированию различных задач на языке ассемблера процессора Intel 8086. Излагаются краткие теоретические и практические сведения, необходимые для выполнения лабораторных работ, примеры составления программ, требования к содержанию отчета.

Методические указания рассмотрены и утверждены на методическом семинаре и заседании кафедры информационных систем
(протокол № 1 от 30 августа 2019 г.)

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

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

Содержание

Лабораторная работа 6

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

1. Цель работы

Исследовать 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 хра-

нения 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 адресное 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.

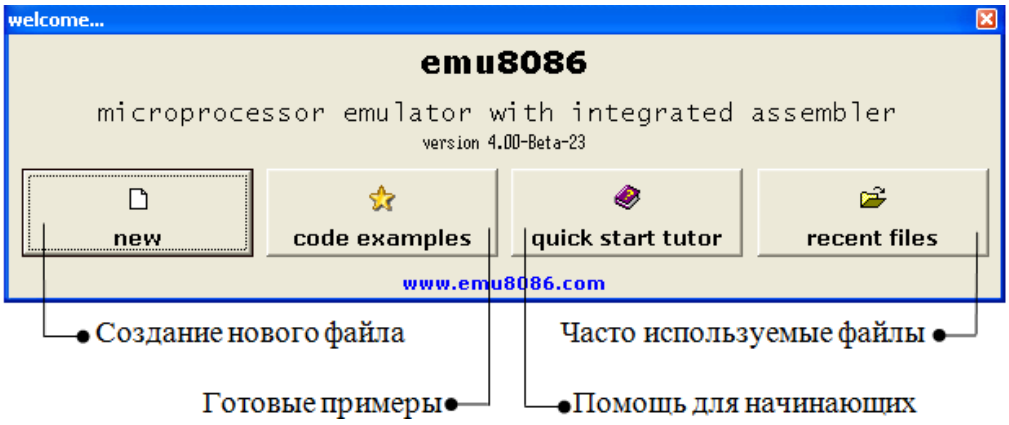


Рисунок 3.1 — Окно приглашения

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

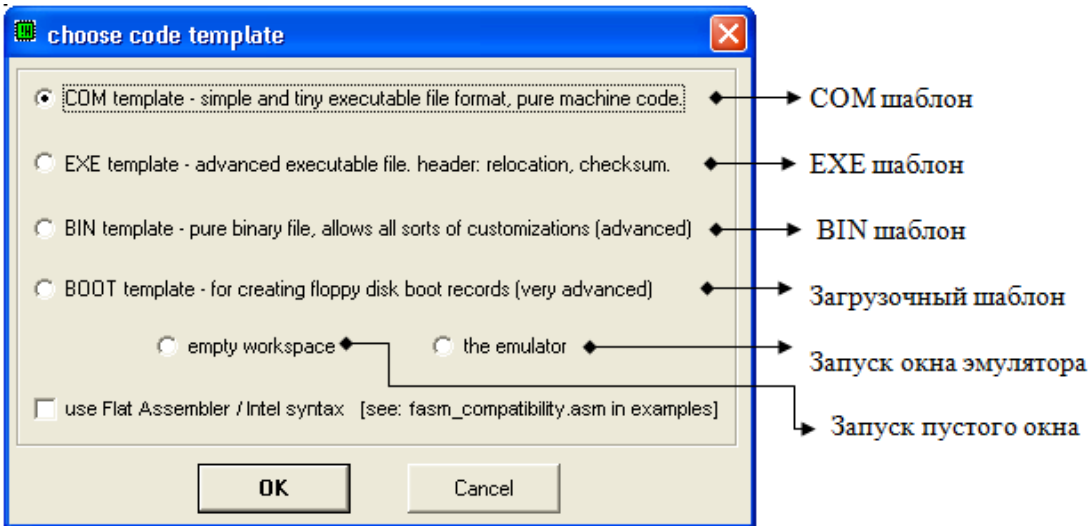


Рисунок 3.2 — Окно выбора шаблона

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.

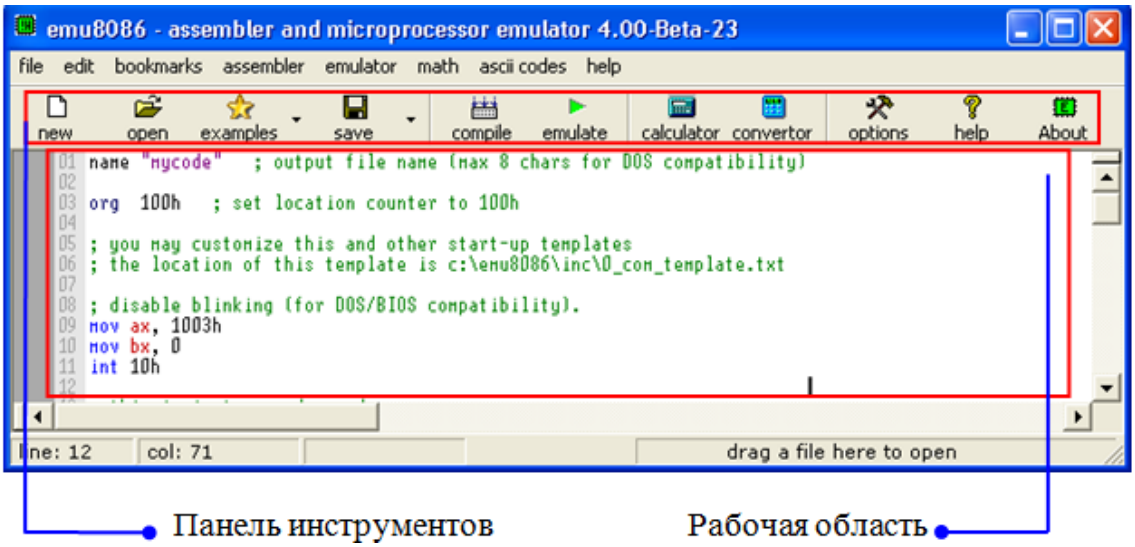


Рисунок 3.3 — Окно создания и редактирования кода

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 регистров, 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 показан на рисунке 3.5.

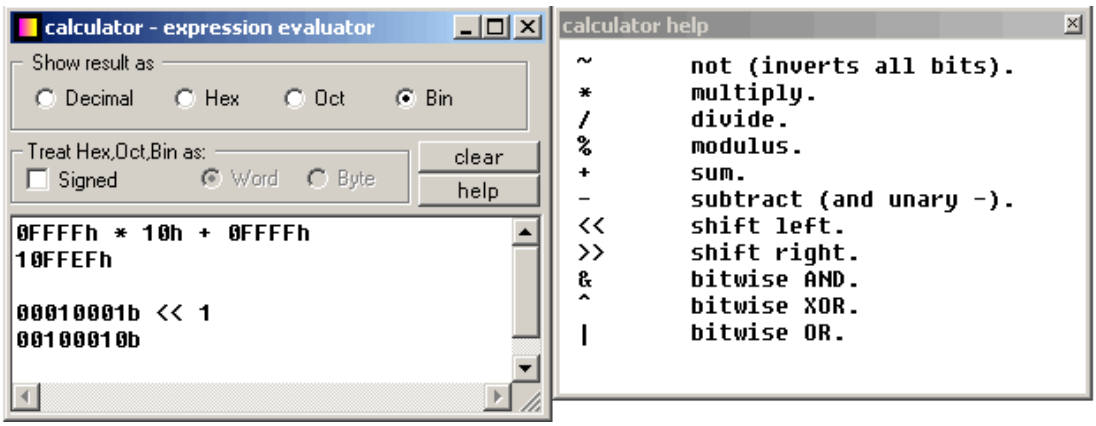


Рисунок 3.4 — Окно калькулятора и окно помощи для работы с ним

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 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.

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

- 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.

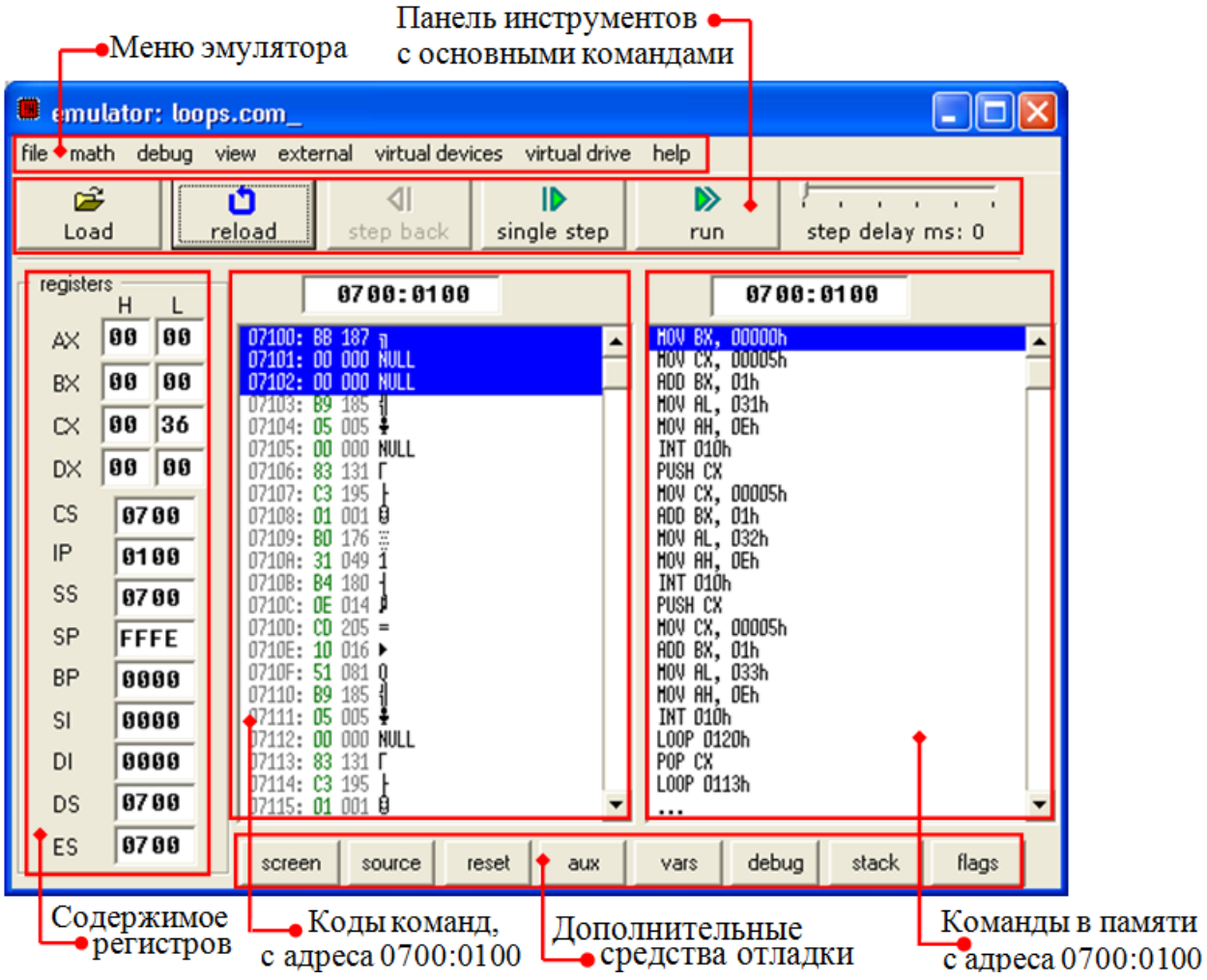


Рисунок 3.6 — Окно эмулятора

Окно эмулятора также предоставляет дополнительные возможности для отладки программ, в частности:

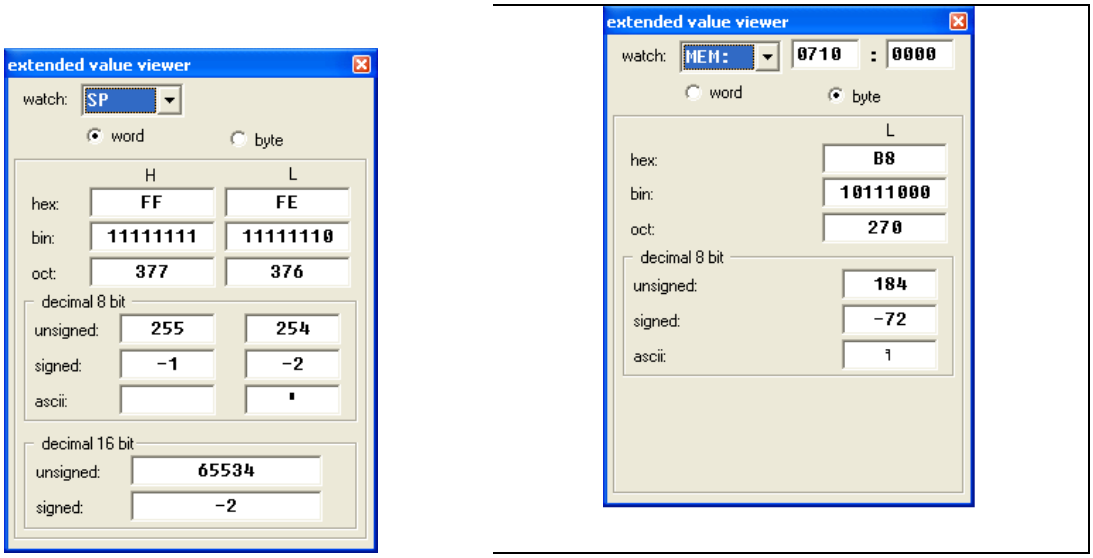
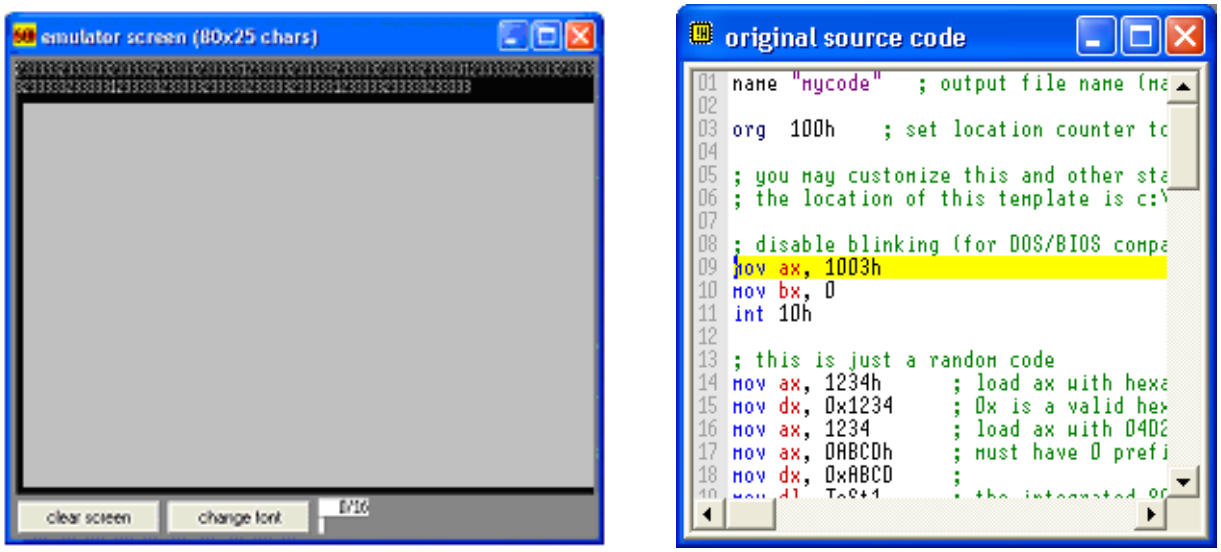


Рисунок 3.7 — Пример окон просмотра значений регистров

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

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)

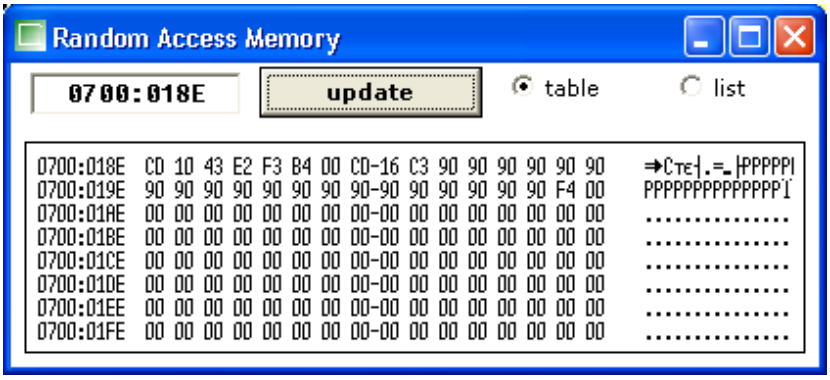


Рисунок 2.9 — Окно просмотра содержимого памяти

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.

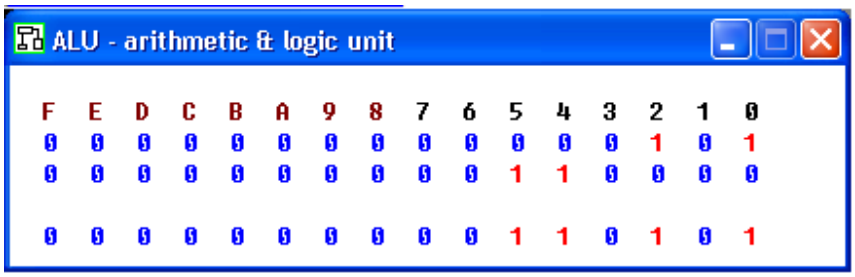


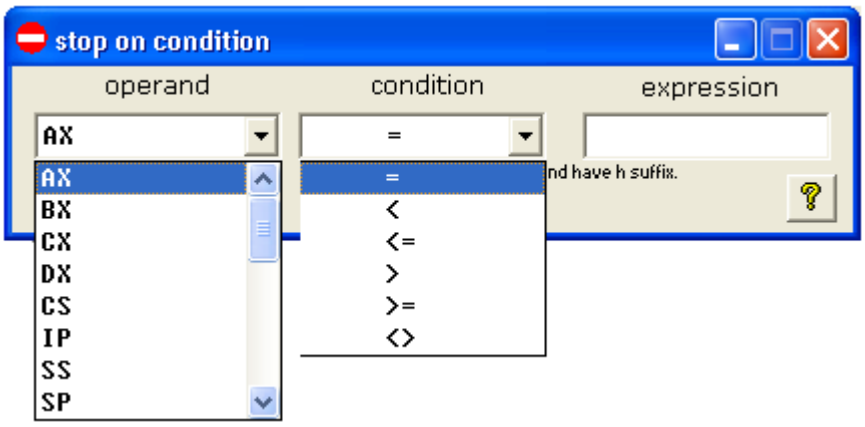
Рисунок 3.10— Окно просмотра содержимого ALU

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 окна показан на рисунке 3.11. Таблица символов **symbol table**. В этой таблице отображаются названия и параметры символов.

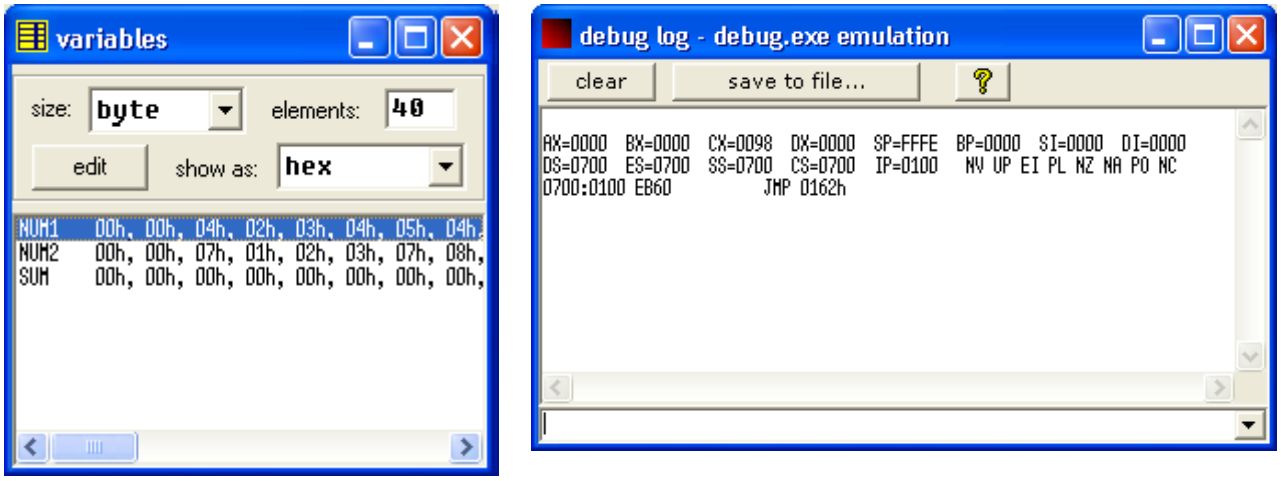
В окне **listing** — отображается программа в машинных кодах. Вся эмулируемая программа представляется в виде кодов команд и адресов, по которым они располагаются.

var — Просмотр переменных. Можно осуществить быстрый и прямой доступ ко всем переменным. В случае необходимости можно изменить их значение двойным щелчком мыши по необходимой переменной



.

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

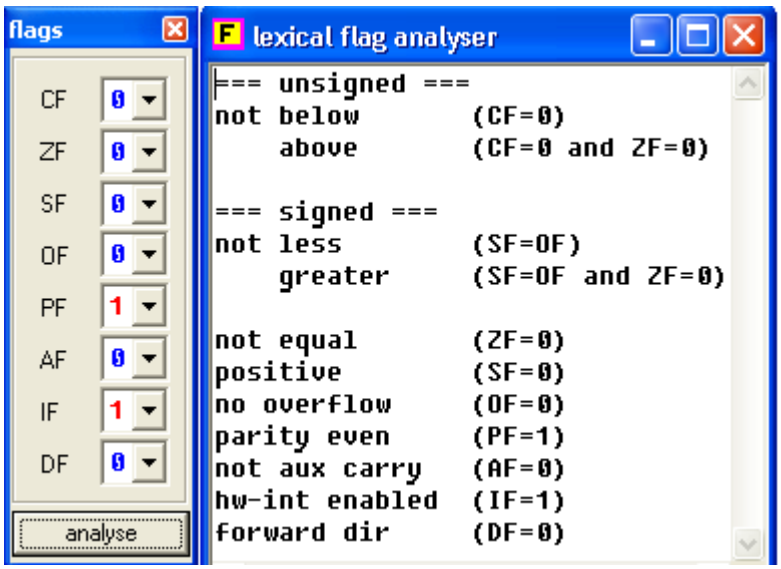


а) б)

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

Окно stack — позволяет 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.



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. Составить исследуемую программу на языке ассемблера в соответствии с заданным вариантом (Приложение А).

4.4. Запустить разработанную программу в среде отладчика и исследовать взаимодействие блоков процессора в ходе выполнения команд различных типов (команды пересылки данных, арифметические и логические команды, команды управления и другие).

4.5. Рассчитать время выполнения программ.

5. Содержание отчета

5.1. Цель и программа работы.

5.2. Структурная схема МП 8086 и временные диаграммы его функционирования.

5.3. Описание взаимодействия блоков микропроцессора при выполнении команд различной длины и различных типов.

5.4. Результаты проведенных исследований и расчетов.

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

6. Контрольные вопросы

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

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

6.3. 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.

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

6.5. Расскажите о флагах процессора и особенностях их использования.

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

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

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

6.9. Расскажите подробно о работе процессора после включения питания.

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

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

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

6.13. Расскажите об основных возможностях экранного отладчика emu8086.

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

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

Лабораторная работа 8

“Исследование методов адресации и программирования арифметических и логических операций ”

1. Цель работы

Изучить основные директивы языка ассемблера, исследовать их воздействие на процесс ассемблирования и формирования листинга программы.

Исследовать особенности функционирования блоков 16-разрядного микро- процессора при выполнении арифметических и логических операций и при использовании различных способов адресации. Приобрести практические навыки программирования на языке ассемблера МП 8086 арифметических и логических операций с применением различных способов адресации.

2. Общие теоретические положения

3.1 Основные директивы ассемблера

Операторы языка "Ассемблер", которые позволяют управлять процессом ассемблирования и формирования листинга называются директивами или псевдокомандами. Они действуют только в процессе ассемблирования и не генерируют машинных кодов. Наиболее употребимыми являются следующие директивы.

PAGE строк, символов в строке — задает формат распечатки программы, т.е. количество строк на листе и число символов в строке. Максимальное количество строк 255, символов - 132. По умолчанию в ассемблере установлено PAGE 60,80.

TITLE текст — задает текст, который будет печататься на каждой странице листинга. В качестве текста рекомендуется указывать имя программы, под которым она находится в каталоге на диске.

SEGMENT [параметры] — указывает используемый в программе сегмент. Любые программы содержат по крайней мере один сегмент кода. Имя сегмента должно обязательно присутствовать, быть уникальным и не совпадать со стандартным набором слов, используемых в языке. Для описания сегмента используется следующий формат:

```
Имя   Директива   Операнд    имя SEGMENT   [параметры]
            :
            :
имя     ENDS.
```

Директива SEGMENT может содержать три типа параметров: выравнивание, объединение и класс. Обычно *выравнивание* сегмента осуществляется по параграфам. Поэтому типовым параметром является PARA, который принимается также по умолчанию. Параметр *объединение* определяет, объединяется ли

данный сегмент в процессе компоновки с другими сегментами. Возможны следующие виды объединений: STACK, COMMON, PUBLIC, AT-выражение и MEMORY. Сегмент стека определяется следующим образом:

имя SEGMENT PARA STACK .

Если отдельно ассемблируемые программы должны объединяться компоновщиком, то можно использовать типы объединений: PUBLIC, COMMON и MEMORY. В других случаях тип объединения можно не указывать.

Параметр *класс* заключается в кавычки используется для группирования относительных сегментов при компоновке.

Директива PROC выделяет отдельные процедуры в сегменте. Если сегмент имеет только одну процедуру, то он оформляется следующим образом:

```
имя_сегмента      SEGMENT     PARA
имя_процедуры     PROC        FAR
                  :
                  RET
имя_процедуры     ENDP
имя_сегмента      ENDS.
```

Директивы ENDP и ENDS указывают соответственно конец процедуры и сегмента. Операнд FAR сообщает загрузчику DOS, что начало данной процедуры является точкой входа для выполнения программы.

Директива ASSUME операнд - указывает Ассемблеру назначение каждого сегмента. Например:

ASSUME SS: им_стек, DS:им_дан, CS:им_код.

Запись SS: им_стек означает, что ассемблер должен инициализировать регистр SS адресом им_стек.

Директива END завершает всю программу. Операнд содержит имя, указанное в директиве PROC, которое было обозначено как FAR.

3.2 Пример EXE-программы

При написании программ на языке ассемблера следует учитывать, что существует два основных типа загрузочных модулей; EXE и COM. В настоящей и ряде последующих работ мы будем использовать EXE-формат. При инициализации ассемблерной EXE-программы система DOS предъявляет следующие требования;

1) указать ассемблеру, какие сегментные регистры должны соответствовать конкретным сегментам;

2) сохранить в стеке адрес, содержащийся в регистре DS;

3) записать в стек нулевой адрес;

4) загрузить в регистр DS адрес сегмента данных.

Пример оформления ассемблерной программы:

```
                        page 60,132
          TITLE  Lab1.asm     Пример  регистровых операций
;-------------------------------------------------------------------------------
STACKSG    SEGMENT  PARA  'Stack"
```

```
                DB      16  DUP('STACKSEG')
STACKSG     ENDS
;-------------------------------------------------------------------------------
DATASG      SEGMENT  PARA  'Data'
ALFA        DW          240
BETA         DW          130
GAMMA       DD         (?)
DATASG      ENDS
;-------------------------------------------------------------------------------
CODESG      SEGMENT  PARA  'Code'
BEGIN       PROC        FAR
ASSUME  SS: STACKSG, CS:CODESG, DS:DATASG, ES:NOTHING
PUSH      DS
SUB        AX,AX
PUSH      AX
MOV     AX,0123H
ADD    AX,ALFA
MOV     BX,AX
SUB     AX,BETA
MOV     CX,AX
ADD    BX,CX
MOV     GAMMA,AX
NOP
RET
BEGIN       ENDP
CODESG      ENDS
END         BEGIN
```

3.2 Методы адресации микропроцессорных систем

В процессе программирования МП 8086 используется 7 режимов адресации.

1. Регистровая адресация. Операнд находится в одном из регистров общего назначения, а в ряде случаев — в сегментном регистре:

```
mov  dx,bx
add   bx,di
```

2. Непосредственная адресация. 8- или 16- битовая константа содержится в команде в качестве источника:

```
mov  al,45
mov  ax,1024
```

3. Прямая адресация. Эффективный адрес берется из поля смещения команды. Применяется в основном, если операндом служит метка:

```
table  db 10, 20, 30
.............
mov  ax, table
```

Микропроцессор добавляет адрес к сдвинутому на 4 разряда содержимому регистра DS и получает 20-разрядный адрес операнда.

4. Косвенная регистровая адресация. Эффективный адрес содержится в базовом регистре BX, регистре указателя базы BP или индексном регистре (DI или SI). Косвенные регистровые операнды заключают в квадратные скобки:

```
mov ax,[bx]
```

По этой команде в регистр AX загружается содержимое ячейки памяти, адресуемой значением регистра BX.

5. Адресация по базе. Эффективный адрес вычисляется с помощью сложения значения сдвига с содержимым одного из базовых регистров BX или BP. Например, при доступе к элементам таблицы адрес начала таблицы предварительно записывается в регистр BX:

```
mov ax,[bx]+8
```

6. Прямая адресация с индексированием. Эффективный адрес вычисляется как сумма значений сдвига и одного из индексных регистров (DI или SI). Такой способ адресации целесообразно применять при доступе к элементам таблицы, когда сдвиг указывает на начало таблицы, а индекс - на элемент таблицы.

```
table db 10, 20, 30, 40
  .............
  mov di,2 ; загрузить в индексный регистр номер
  выбираемого байта минус 1
 mov al,table[di]  ; загрузить 3 байт таблицы в al
```

7. Адресация по базе с индексированием. Эффективный адрес вычисляется как сумма значений базового регистра, индексного регистра и, возможно, сдвига. Это удобно при работе с двухмерными таблицами: базовый регистр содержит начальный адрес массива, значения сдвига и индексного регистра представляют собой смещения по строке и столбцу.

```
table dw  1024, 1048, 2048,3600
      dw  4100, 5000, 600, 2000
      dw  80, 300, 4000, 5000
   .............
value db 2              ; номер элемента в строке-1
   .............
mov bx,table            ;адрес таблицы
mov di,16               ;адрес начала третьей строки
mov ax,value[bx][di] ;       загрузка элемента
                               table(3,3)=4000
```

3. Программа исследований

3.1. Изучить основные директивы ассемблера и их воздействие на процесс ассемблирования и формирования листинга программы. Повторить коман-

ды пересылки данных, а также команды арифметических и логических операций (выполняется в процессе домашней подготовки к лабораторной работе).

3.2. Изучить методы адресации, используемые в 16-разрядных процессорах (выполняется во время домашней подготовке к работе).

3.3. Составит программу, состоящую из следующих процедур обработки строк:

3.3.1. Заполнить 100+10*i* ячеек области памяти, начинающейся с адреса MAS рядом натуральных чисел. Здесь *i* – последняя цифра номера Вашей зачетной книжки.

3.3.2. Переслать массив слов из области памяти, начиная с адреса MAS1 в область с начальным адресом MAS2.

3.3.3. Найти в заданном массиве число, равное двум последним цифрам Вашей зачетной книжки и определить его индекс.

3.4. Переслать в память с адресом 2020:300 диагональные элементы матрицы размером 8×8. Значения элементов матрицы должны быть определены в сегменте данных программы.

3.5. Произвести отладку разработанных программ в пошаговом режиме и проследить за изменениями содержимого регистров

3.6. Произвести ассемблирование программы и получить объектный и исполняемый модуль программы в Exe-формате и ее листинг.

3.7. Рассчитать время выполнения программы.

4. Содержание отчета

4.1 Цель и программа работы.

4.2 Текст и листинг ассемблерной программы для заданного варианта.

4.3 Выводы по работе.

5. Контрольные вопросы

5.1 Каково различие между директивой и командой?

5.2 Назовите директивы определения данных ассемблера и поясните механизм их действия.

5.3 Какие директивы применяются для оформления процедур?

5.4 Какие типы сегментов используются в ассемблерных программах и каково их назначение?

5.5 Поясните назначение параметров выравнивания и объединения, используемых в директивах SEGMENT.

5.6 Когда и в каких случаях применяется директива ORG?

5.7 Что конкретно подразумевает директива END, если она завершает: а) программу, б) процедуру, в) сегмент?

5.8 Какие операции необходимо произвести в процессоре до начала выполнения программы?

5.9 Назовите команды арифметических операций и поясните использование регистров процессора при каждой операции.

5.10 С какой целью в начале кодового сегмента в стек заносится содержимое сегментного регистра DS, а затем нулевое значение?

5.11 Каково назначение директивы ASSUME?

5.12 Расскажите об особенностях размещения в памяти ЭВМ программ с расширениями .exe и .com.

5.13 Нарисуйте схему подключения 16- разрядного порта к МП 1810ВМ86, если в наличии имеются только микросхемы 580ВВ55 или 580ВА86.

5.14 Каково назначение вывода M/IO в МП 8086 и нарисуйте схему подключения устройств с его использованием.

5.15 Какие функции выполняет сопроцессор, каково его устройство и особенности его функционирования?

5.16 С какой целью используется системная и резидентная память и какие способы их организации и команды доступа к ним?

5.17 На каком основании и при каких условиях арбитр шин дает доступ МП к системной памяти?

5.18 Каковы функции системного контроллера и какие сигналы он формирует?

5.19 Как сопроцессор определяет, что команда относится к нему?

5.20 Расскажите о многопроцессорных системах, их назначении и способах организации шин.

5.21 Какая информация хранится в заголовке .exe –программы, его назначение и размер?

5.22 Зачем к исполняемому модулю добавляется префикс программного сегмента, какой его размер и какая информация в нем хранится?

5.23 Расскажите о методах адресации, используемых в МП-системах, и объясните в каких случаях целесообразно использование этих методов?

5.24 Объясните особенности использования строковых команд.

5.25 Каким образом можно изменять направление просмотра строк?

5.26 Чем отличаются команды CMPS и SCAS?

5.27 Как обеспечить ввод данных с группы 16-разрядных портов с максимальной скоростью?

5.28 В чем состоит суть защищенного режима работы процессора и как осуществляется защита?

5.29 Что такое дескриптор сегмента, из каких частей он состоит и как используется при защите памяти?

5.30 Как организуется виртуальная память и как используется дескриптор для ее поддержки?

5.31 Расскажите об архитектуре 16-разрядного процессора второго поколения и приведите его схему.

5.32 Расскажите о регистрах 16-разрядного процессора второго поколения и особенностях их использования в защищенном режиме.

5.33 Расскажите о многозадачном режиме работы процессора и составе и назначении сегмента состояния задачи.

3. Программа работы

2.1 Изучить команды прерывания ПЭВМ и особенности использования их при программировании на ассемблере задач связанных с организацией ввода с клавиатуры и вывода данных на экран монитора, а также при воспроизведении звуковых колебаний.

2.2 Организовать меню выбора для запуска подпрограмм, описанных в пункте 2.4.

2.3 Работу меню организовать следующим образом:

- Вывести на экран краткое описание каждого пункта меню, с центрированием;
- Вывести на экран запрос пользователю ввода номера необходимого пункта меню;
- Осуществить ввод введенного пользователем номера требуемого пункта меню;
- Реализовать обработку введенного номера с запуском соответствующего пункта меню;
- Для пункта меню 2.4.6 организовать ввод значение генерируемой частоты.

2.4 Реализовать следующие подпрограммы для генерации звука в ПЭВМ.

2.4.1 Написать программу, которая при нажатии на различные клавиши клавиатуры ПЭВМ, генерирует звуки различной тональности.

2.4.2 Написать программу, которая воспроизводит звуки со случайной величиной длительности и высоты. Примером случайной последовательности чисел может служить набор данных, хранящийся или в ПЗУ (адрес F600:0000) , или в ОЗУ (например, с адреса 0000:0000).

2.4.3 Написать программу, которая выдаёт последовательность звуков, высота тона которых изменяется от низкой частоты к высокой.

2.4.4 Написать программу, которая выдаёт последовательность звуков, высота тона которых изменяется от высокой частоты к низкой.

2.4.5 Написать программу, которая воспроизводит последовательность звуков одной тональности, при длительности, меняющейся от короткой к длинной.

2.4.6 Написать программу, которая выдает последовательность звуков заданной пользователем частоты.

Примечание! Каждый вариант должен содержать процедуру BEEP, для которой надо передавать следующие параметры: длина и высота звука.

4. Содержание отчета

4.1. Цель и программа работы;

4.2. Алгоритм решаемой задачи;

4.3. Текст программы, согласно варианту;
4.4. Выводы по результатам исследований.

5. Контрольные вопросы

5.1. Расскажите о командах прерывания ПЭВМ и выполняемых операциях во время прерывания.
5.2. Напишите команды для установки курсора по координатам: строка 10, столбец 12.
5.3. Напишите команды для очистки экрана с 12-й по 20-ю строку.
5.4. Составьте фрагмент программы вывода запроса на ввод текущей даты.
5.5. Составьте фрагмент программы вывода сообщения на экран монитора с произвольными координатами.
5.6. Расскажите устройство клавиатуры ПЭВМ и о схеме взаимодействия ее с процессором.
5.7. Расскажите об клавиатурных функциях BIOS.
5.8. Как используется байт состояния клавиатуры?
5.9. Расскажите о системных средствах ввода данных с клавиатуры.
5.10. Расскажите устройство черно-белой и цветной электронно-лучевой трубки дисплея и покажите элементы электронной линзы на образце электронной пушки.
5.11. Поясните отличие организации видеобуферов для текстового и графического режимов.
5.12. Начертите структурную схему текстового видеоадаптера, поясните его работу и расскажите о схемах формирования цветов в различных режимах.
5.13. Приведите характеристику дисплейных функций BIOS.
5.14. Начертите структурную схему программируемого таймера, запишите его программную модель и объясните функционирование устройства и особенности программирования.
5.15. Какое значение регистра константы надо установить, чтобы динамик воспроизвёл ноту «ми» средней октавы (частота ноты «ми» 659.3 Гц).
5.16. Как вы считаете, почему когда бит 1 регистра B равен «0» динамик не издаёт звук даже тогда, когда на выходе канала 2 имеется сигнал определённой частоты.
5.17. Какой минимальной частоты звук может быть воспроизведён динамиком, при генерации звука только каналом 2.
5.18. Звук какой максимальной частоты может быть воспроизведён динамиком, при генерации звука только каналом 2.
5.19. Почему при генерации сигнала каналом 2, труднее создавать звуковые спецэффекты, нежели при использовании бита 1 регистра B.
5.20. Рассчитать число циклов, необходимое для генерации звука с частотой 1000 Гц (методом, использующим бит 1 регистра B), при условии, что тактовая частота процессора 100 МГц, время выполнения 1-го машин-

ного цикла = 4-ём тактам и время выполнения команды = (длина команды в байтах)*(1 машинный цикл).

5.21. Как создается обработчик прерываний от таймера?
5.22. Поясните принцип формирования звука в звуковой карте компьютера.
5.23. Начертите структурную схему звуковой карты и осветите функции, которые выполняет это устройство.
5.24. Расскажите о методах синтеза звука в компьютере.

6. Список рекомендованной литературы

6.1. Абель П. Язык Ассемблера для IBM PC и программирования / Пер. с англ. Ю.В.Сальникова. — М.: Высш. школа, 1992. — 447 с.Новиков Ю.В. Основы микропроцессорной техники: Учебное пособие/Ю.В. Новиков, П.К. Скоробогатов. — М.: Интернет-университет информационных технологий; БИНОМ, 2006. — 359 с.
6.2. 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
6.3. 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
6.4. Чернега В.С. Архитектура информационных систем . Конспект лекций / В.С. Чернега. – Севастополь: Изд-во СевГУ, 2015 – 160 с.

ПРИЛОЖЕНИЕ А

Варианты индивидуальных заданий к лабораторным работам.

В задании к лабораторной работе номер 3 вариант выбирается в соответствии с последней цифрой зачетки студента, i - целое число равное предпоследней цифре номера зачетки студента.

Варианты к заданию 8 приведены в таблице А.1.

Таблица А.1 – Формулы для расчета результирующей матрицы

Вариант	Формула для расчета результирующей матрицы	Вариант	Формула для расчета результирующей матрицы
0	C=(A*i)-B	5	C=(A-i)+B
1	C=A+(B*i)	6	C=A+B+2i
2	C=(A-i)+B	7	C=A+B-i
3	C=(A*i)+(B*i)	8	C=(A*i)-(B*i)
4	C=A+B+i	9	C=(B*i)-A

Примеры результатов выполнения заданий приведены в таблице А.2.

Таблица А.2 – Примеры результатов выполнения заданий

Задание	Файл	Состояние файлов до выполнения программы	Состояние файлов после выполнения программы
6	Ф1	abcde4f	abcde4f
	Ф2	не существует	abcde4ff4edcba
7	Ф1	aAbCedF	aAbCedF
	Вариант1: Ф2	пустой	AABCEDF
	Вариант 2: Ф2	пустой	aabcedf
8	Ф1,матрица А	2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2
	Ф2,матрица В	3 3 3 3 3 3 3 3 3	3 3 3 3 3 3 3 3 3
	Ф3 Вариант 4: C=A+B+i, i=4.	пустой	9 9 9 9 9 9 9 9 9

Заказ № _____ от «___» _____ 2018 г. Тираж _____ экз.
Изд-во СевГУ