Коми Республикаса велӧдан, наука да том йӧз политика министерство
Министерство образования, науки и молодежной политики Республики Коми
Государственное профессиональное образовательное учреждение
«Сыктывкарский целлюлозно – бумажный техникум»

РАССМОТРЕНО
на заседании ПЦК информационных
дисциплин ГПОУ «СЦБТ»
Протокол № __
«__» _____ г.
Председатель ПЦК _____ В.А. Шулепов
Преподаватель _____ Д.Д. Расов

УТВЕРЖДАЮ
Зам.директора по УПР
ГПОУ «СЦБТ»
_____ Е.В. Соколова
«____» _____ 20__ г.

Методические указания по выполнению лабораторных работ

по учебной дисциплине МДК02.01 «Микропроцессорные системы»
для специальности
09.02.01 Компьютерные системы и комплексы

Программирование микропроцессора i8086 на языке Ассемблер

Сыктывкар 2018

ЛАБОРАТОРНАЯ РАБОТА № 1

Ознакомление с работой эмулятора Emu8086

Цель работы: ознакомление со структурой учебной микроЭВМ (эмулятора Emu8086), органами управления и режимами ее работы.

1. Краткие теоретические сведения

1.1 Структура ассемблерной программы

Каждый язык программирования имеет свои особенности. Язык ассемблера - не исключение. Традиционно первая программа выводит приветственное сообщение на экран '**Hello World**'.

В отличие от многих современных языков программирования в ассемблерной программе каждая команда располагается на ОТДЕЛЬНОЙ СТРОКЕ. Нельзя разместить несколько команд на одной строке. Не принято, также, разбивать одну команду на несколько строк.

Язык ассемблера является РЕГИСТРОНЕЧУВСТВИТЕЛЬНЫМ, т.е. в большинстве случаев нет разницы между большими и малыми буквами. Команда может быть ДИРЕКТИВОЙ - указанием транслятору. Они выполняются в процессе превращения программы в машинный код. Многие директивы начинаются с точки. Для удобства чтения программы они обычно пишутся БОЛЬШИМИ БУКВАМИ. Кроме директив еще бывают ИНСТРУКЦИИ - команды процессору. Именно они и будут составлять машинный код программы.

Нужно отметить, что понятие "машинного кода" очень условно. Часто оно обозначает просто содержимое выполняемого файла, хранящего кроме собственно машинных команд еще и данные. В нашем случае это будет текст выводимого сообщения "**Hello**".

1.2 Особенности создания ассемблерной программы в среде DOS средствами TASM и MASM

Язык ассемблера является самым низкоуровневым языком программирования, т.е. он ближе любых других приближен к архитектуре ЭВМ и ее аппаратным возможностям, позволяя получить к ним полный доступ. В отличие от языков высокого уровня (ЯВУ) ассемблерная программа содержит только тот код, который ВВЕЛ ПРОГРАММИСТ. Никаких дополнительных "обвязок". Вся ответственность за "логичность" кода ПОЛНОСТЬЮ лежит на узких плечах ПРОГРАММИСТА.

Простой пример. Обычно подпрограммы заканчиваются командой возврата. Если ее не задать явно, транслятор все равно добавит ее в конец подпрограммы. Ассемблерная подпрограмма без команды возврата НЕ

ВЕРНЕТСЯ в точку вызова, а будет выполнять код, следующий за подпрограммой, как-будто он является ее продолжением. Еще пример. Можно попробовать "выполнить" данные вместо кода. Часто это лишено смысла. Но если программист это сделает, транслятор промолчит. Язык ассемблера позволяет делать все! Тут нет НИКАКИХ ограничений. Но с другой стороны это часто является источником ошибок.

Эти особенности приводят к тому, что ассемблерные программы часто "подвешивают" компьютер, особенно у начинающих программистов.

Выделим три разновидности "зависания" по способу борьбы с ним.

Простое - для выхода из него достаточно нажать **Ctrl+Break** или **Ctrl+C** (сначала нажимается клавиша Ctrl и, НЕ ОТПУСКАЯ ее, нажимается вторая клавиша - C или Break; отпускаются в обратном порядке). Программа при этом аварийно завершается выходом в DOS.

Мягкое - машина не реагирует на **Ctrl+Break**, но клавиатура "дышит", т.е. при нажатии на клавиши, типа NumLock, моргают соответствующие светодиоды. В этом случае машину нужно будет перегрузить, нажав **Ctrl+Alt+Del**. В среде Windows нужно просто "убить" сеанс, закрыв окно.

Жесткое - машина никак не реагирует на клавиатуру и не воспринимает комбинацию **Ctrl+Alt+Del**. В этом случае поможет аппаратный сброс при помощи кнопки "**Reset**", расположенной на передней панели системного блока. Не нужно ВЫКЛЮЧАТЬ и включать ЭВМ. Вы должны знать, что она выходит из строя в основном при включении и выключении.

1.3 Процесс обработки программы на языке ассемблера

Из-за своей специфики, а также по традиции, для программирования на языке ассемблера нет никаких сред-оболочек типа **Turbo C**, **Turbo Pascal** и т.д. Тут приходится пользоваться "утилитами командных строк", как 30 лет назад. Весь процесс технического создания ассемблерной программы можно разбить на 4 шага (исключены этапы создания алгоритма, выбора структур данных и т.д.).

Набор программы в текстовом редакторе и сохранение ее в отдельном файле. Каждый файл имеет имя и тип, называемый иногда расширением. Тип в основном используется для определения назначения файла. Например, программа на **C** имеет тип **C**, на **Pascal - PAS**, на языке **ассемблера - ASM**.

Обработка текста программы транслятором. На этом этапе текст превращается в машинный код, называемый объектным. Кроме того, есть возможность получить листинг программы, содержащий кроме текста программы различную дополнительную информацию и таблицы, созданные транслятором. Тип объектного файла - **OBJ**, файла листинга - **LST**. Этот этап называется ТРАНСЛЯЦИЕЙ.

Обработка полученного объектного кода компоновщиком. Тут программа "привязывается" к конкретным условиям выполнения на ЭВМ. Полученный машинный код называется выполняемым. Кроме того, обычно получается карта загрузки программы в ОЗУ. Выполняемый файл имеет тип

EXE, карта загрузки - **MAP**. Этот этап называется КОМПОНОВКОЙ или ЛИНКОВКОЙ.

Запуск программы. Если программа работает не совсем корректно, перед этим может присутствовать этап ОТЛАДКИ программы при помощи специальной программы - отладчика. При нахождении ошибки приходится проводить коррекцию программы, возвращаясь к шагу 1. Таким образом, процесс создания ассемблерной программы можно изобразить в виде следующей схемы. Конечной целью, напомним, является работоспособный выполняемый файл HELLO. EXE.

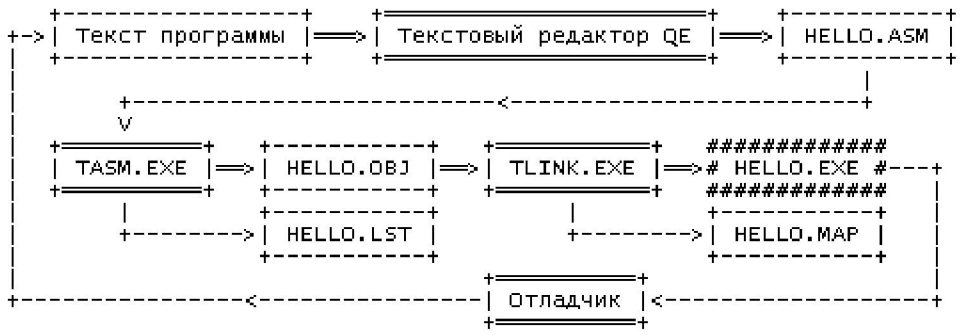

Рис.1

1.4 Особенности создания ассемблерной программы в среде эмулятора EMU8086

Этот программный продукт содержит все необходимое для создания программы на языке **Assembler**.

Пакет **Emu8086** сочетает в себе продвинутый текстовый редактор, assembler, disassembler, эмулятор программного обеспечения (Виртуальную машину) с пошаговым отладчиком, примеры.

В процессе выполнения программы мы можем наблюдать программные регистры, флаги и память, АЛУ показывает работу центрального процессора.

Встроенная виртуальная машина полностью блокирует обращение программы к реальным аппаратным средствам ЭВМ, накопителям памяти, это делает процесс отладки намного более легкой.

1.5 Правила оформления ассемблерных программ

При наборе программ на языке ассемблера придерживайтесь следующих правил:

- директивы набирайте большими буквами, инструкции - малыми;
- пишите текст широко;
- не выходите за край экрана, т.е. не делайте текст шире 80 знаков - его не удобно будет редактировать и печатать;
- для отступов пользуйтесь табуляцией (клавиша TAB);

- блоки комментариев задавайте с одинаковым отступом.

Оптимальной считается такая строка:

<TAB><TAB>mov<TAB>ax,<пробел>bx< (1-3) TAB>; <пробел>текст комментария

Количество табуляций перед комментарием определяется длиной аргументов команды и может быть от 1 до 3. По мере знакомства с синтаксисом языка будут приводиться дополнительные правила.

2. Задание для выполнения

2.1 Запустить эмулятор **EMU8086**.

2.2 Пользуясь правилами оформления ассемблерных программ, исправьте слова "Please Register." на любые понравившиеся (Не забудьте заключить их в апострофы).

2.3 Запустите приложение, нажав кнопку ‘Emulate’ или клавишу F5.

2.4 Запустите полученный код на выполнение, используя кнопку “RUN” или нажмите функциональную клавишу F9.

2.5 Откомпилируйте программу. Вернитесь в главное окно формы, предварительно закрыв все открытые окна, далее нажмите кнопку “Compile”.

2.6 Полученный com-файл запустите во встроенной командной строке WINDOWS на выполнение или запустите сеанс dos в total commander’e.

2.7 Поэкспериментируйте с другими примерами которые открываются при нажатие клавиши “Samples” в главном окне эмулятора.

2.8 Ознакомитесь со встроенной в эмулятор EMU8086 справкой. В ней содержится вся необходимая информация для работы с программой, азы написания программ на языке assembler и др.

3. Контрольные вопросы

3.1 Каковы основные отличия ассемблерных программ от ЯВУ?

3.2 Какова структура ассемблерной программы?

3.3 В чем отличие инструкции от директивы?

3.4 Каковы правила оформления программ на языке ассемблера?

3.5 Каковы этапы получения выполняемого файла?

3.6. Для чего нужен этап отладки программы?

3.7. Опишите основные моменты создания исполняемого файла и эмуляции работы программы?

3.8. Каковы шаги технического создания ассемблерной программы в программах TASM и MASM?

3.9 Основные возможности эмулятора EMU8086?

3.10 Методы борьбы с зависанием в DOS’e?