МИНОБРНАУКИ РОССИИ

ФГБОУ ВО «Ижевский государственный технический университет имени М.Т.Калашникова»

В.Г. Тарасов

Методические указания

к выполнению лабораторной работы № 1

на тему «Знакомство с программированием на языке ассемблера»

по дисциплине «Машинно-зависимые языки программирования»

|  |
| --- |
| Ижевск 2018  Рег. номер |

Методические рекомендации по выполнению лабораторной работы составлены в соответствии с рабочей программой учебной дисциплины, разработанной на основе Федерального государственного образовательного стандарта по направлению 09.03.04 «Программная инженерия» профиль «Разработка программно-информационных систем» программа подготовки «Академический бакалавриат» при изучении дисциплины «Машинно-зависимые языки программирования»

Рецензент: *П.В. Лекомцев, канд. техн. наук, доцент*

Составитель: *В.Г.Тарасов, канд. техн. наук, профессор*

Рекомендовано Советом института ИВТ для использования в учебном процессе в качестве учебно-методических материалов для студентов, обучающихся по направлению 09.03.04 «Программная инженерия» профиль «Разработка программно-информационных систем» программа подготовки «Академический бакалавриат» при изучении дисциплины «Машинно-зависимые языки программирования(протокол № \_\_ от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.)

**Лабораторное занятие 1. Знакомство с программированием на языке ассемблера**

ЦЕЛЬ РАБОТЫ: изучение структуры программы на языке ассемблера, синтаксиса ассемблерных команд и структуры соответствующих команд на машинном языке.

1. **Общие положения**

Изучение методов и алгоритмов трансляции программ на языке ассемблера предполагает знакомство с языком ассемблера, поскольку соответствующие тексты будут входными данными для разрабатываемых трансляторов. Сами эти трансляторы также называются ассемблерами. Поэтому понимание смысла термина «Ассемблер» зависит от контекста в его употреблении.

В качестве аппаратной платформы в лабораторном практикуме выбран 16-разрядный микропроцессор I8086, возможности которого являются малой частью современных 64-разрядных микропроцессоров. Тем не менее изучение основных особенностей алгоритмов трансляции и компоновки происходит на реальных моделях I8086, позволяющих производить проверку результатов как путем сравнения с результатами промышленных трансляторов и компоновщиков, так и путем запуска полученного кода на работающей модели микропроцессора. Этот вариант – замечательная альтернатива использованию сложных современных 32- и 64-битных трансляторов и компоновщиков. С другой стороны, требуется работающая модель 16-битного I808, что достигается использованием 16-битной виртуальной машины DOSBox.

1. **Порядок выполнения работы**

Лабораторная работа начинается с установки виртуальной машины DOSBox, удобной оболочкой для которой является пакет D-Fend Reloaded. После установки пользователь получает в распоряжение 16-битную виртуальную машину с предустановленной операционной системой DOS и файловой оболочкой FreeDOS Shell.

Для работы с ассемблером рекомендуются инструменты из пакета Turbo   
Pascal 7: транслятор, компоновщик, отладчик. Они не требуют установки, достаточно простого копирования.

На основе рассмотренных примеров студент выполняет индивидуальное задание, полученное от преподавателя.

Предлагаемые ниже задания 1-4 определяют содержание лабораторной работы и задают порядок ее выполнения.

3. УСТАНОВКА И НАСТРОЙКА ВИРТУАЛЬНОЙ МАШИНЫ DOSBox  
 (**Задание 1**)

Краткое описание задачи: подготовим среду для работы с языком ассемблера, установим и настроим виртуальную машину.

Скачать VirtualBOX можно из официального источника по ссылке <http://dfendreloaded.sourceforge.net/>.

После установки виртуальной машины на диске С: для конкретного пользователя создается информационная структура в каталоге D-Fend Reloaded. При этом одна из папок называется VirtualHD – она является диском С: виртуальной машины. Таким образом, эту папку можно заполнять как с помощью инструментов исходной операционной системы Windows, так и работая внутри виртуальной машины.

Для работы с ассемблером можно создать папку TASM, сбросить в нее 6 файлов из [http://box.cs.istu.ru/public/kafedra/taras/pakm/tasm/](http://box.cs.istu.ru/public/kafedra/taras/pakm/tasm/?), добавить файл DPMI16BI.OVL – см. рис.1.

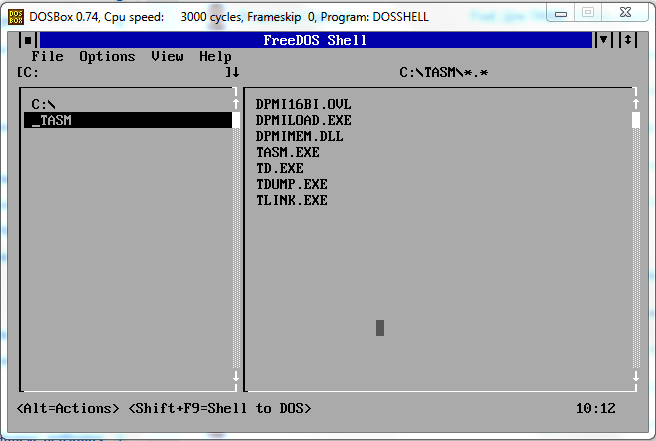


Рис.1. Внешний вид виртуальной машины DOSBox с инструментами для работы с ассемблером

4. Разработка программы на ассемблере (**Задание 2**)

Краткое описание задачи: рассмотрим типовую структуру программы на ассемблере для простейших задач: «Hello world» и «А + В», познакомимся с правилами запуска транслятора, компоновщика, загрузчика и отладчика.

Согласно тому, что в микропроцессоре I8086 (180286) при доступе к основной памяти применяется сегментный механизм, структуры программы состоят из одного или более сегментов. Логично смысловое разделение сегментов по содержащейся в них информации: сегменты с данными, командами, стековые. Рассмотрим пример печати строки-приветствия «Hello world» - рис.2.

dseg segment

s db 'Hello, world$'

endl db 10,13,'$'

dseg ends

cseg segment

assume cs:cseg, ds:dseg

start:

mov ax,dseg

mov ds,ax

mov ah,9

mov dx,offset s

int 21h

mov dx, offset endl

int 21h

lea dx,s

int 21h

lea dx,endl

int 21h

mov ax,04C00h

int 21h

cseg ends

end start

Рис. 2. Программа печати строки «Hello world»

Программа на ассемблере состоит из предложений двух типов, записываемых на отдельной строке. Предложения могут быть директивами ассемблера и служить для управления ходом ассемблирования или для описания данных. Второй тип предложений – ассемблерные команды, причем каждая команда впоследствии превращается в одну команду на машинном языке (одна команда процессора).

Каждый сегмент оформляется директивами segment (обозначает начало) и ends (обозначает конец сегмента). Обе они должны иметь одинаковое имя, придуманное программистом. Имена надо записывать латинскими буквами, цифрами, символом '\_', считающимся буквой; первый символ в имени должен быть буквой, длина имени – не более 8 символов. Строчные и прописные буквы не различаются. В нашем примере два сегмента, названные dseg и cseg.

В сегменте данных (назван dseg) обычно размещаются директивы описания данных. В простых случаях нам достаточно задавать данные в виде набора байтов (db – define bytes), либо в виде набора целых чисел, для каждого из которых в памяти компьютера будут отводиться по два байта (dw – define words). Предложения для записи директив db, dw имеют следующий формат:

**имя db 'символы'**

**имя dw числа\_через\_запятую**

Также в директиве db допускается символы записывать их порядковыми номерами, разделяя запятыми. Символ '$', применяемый в примерах, используется как символ-ограничитель при выводе строк. Имя не является обязательным элементом при записи директив db, dw.

В сегменте команд (в примере назван cseg) большую часть занимают команды ассемблера, состоящие из полей метки, названия\_операции, операндов, комментариев – из них обязательным является только название\_операции.

В микропроцессоре доступ к сегментам обеспечивается тем, что положение сегмента (номер сегмента в I8086) хранится в одном из четырех сегментных регистров CS, DS, SS и ES. В начале сегмента команд программист с помощью директивы **assume** (предположить) сообщает, номера каких сегментов должны оказаться в соответствующих сегментных регистрах. Причем указание относительно регистров CS и SS обеспечивает операционная система (в лице ее загрузчика). А заполнение регистра DS (и ES) – обязанность программиста. Именно для этого и нужны команды **mov ax,dseg** и **mov ds,ax**. Команда **mov ax,dseg** является первой выполняемой командой программы, поэтому она помечена меткой **start:** - здесь символ ':' показывает, что соответствующее имя является именно меткой. То, что метка **start:** обозначает первую выполняемую команду программы, прописано в последней строке программы – это директива **end start**.

Сложные и часто повторяющиеся последовательности действий в компьютере часто поручается выполнять с помощью операционной системы (ОС). В языке ассемблера для обращения к ОС служит команда **int 21h** – запрос на прерывание с номером 21h, где номер задан в 16-ричной системе счисления. Сервис ОС с номером 21h имеет много функций. В примере мы пользуемся им для того, чтобы передать строку символов из основной памяти на экран.

Для указания, что нужно вывести строку символов, мы в регистр ah заносим 9 – команда **mov ah,9**. Сама строка «Hello world» размещена в сегменте ds, ей присвоено имя s. Сервис 21h должен в регистре dx получить смещение этой строки от начала сегмента – команда **mov dx,offset s**. Теперь при вызове сервиса ОС из памяти на экран будут переданы символы до первого символа '$', который уже не отображается. Чтобы перенести курсор в начало следующей строки применяются команды:

mov dx, offset endl

int 21h

Другой способ поместить в регистр dx смещение строки в сегменте – команда **lea dx,s**.

Для завершения программы с освобождением памяти и возвратом управления ОС служат команды

mov ax,04C00h

int 21h

- снова вызов сервиса 21h, но его функции с номером 4Ch (суффикс 'h' – признак записи числа в 16-ричной системе счисления).

5. Перенос программы на электронный носитель (**Задание 3**)

Краткое описание задачи: произведем набор программы в память компьютера и сохраним ее в файле.

Набор (редактирование) текстов программ удобнее всего выполнять средствами ОС Windows. При этом файлы должны создаваться в упомянутой выше папке VirtualHD. Поскольку мы работаем с ассемблером, то удобно сосредоточить информацию на виртуальном диске в папке TASM. Файлы с программами на языке ассемблера должны иметь расширение .asm. После сохранения рассмотренной программы в файле t.asm содержимое папки TASM приобретает вид – рис. 2.

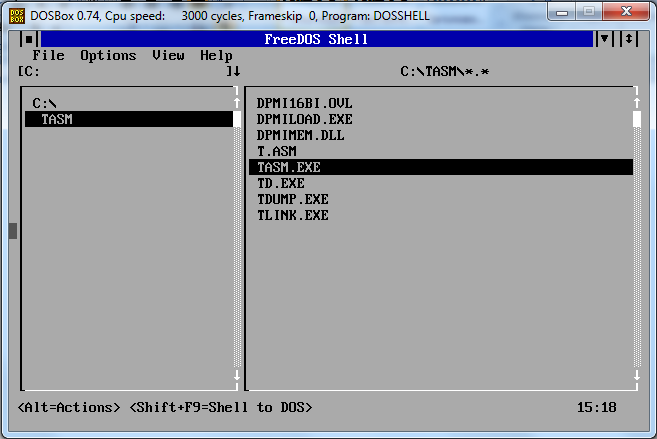


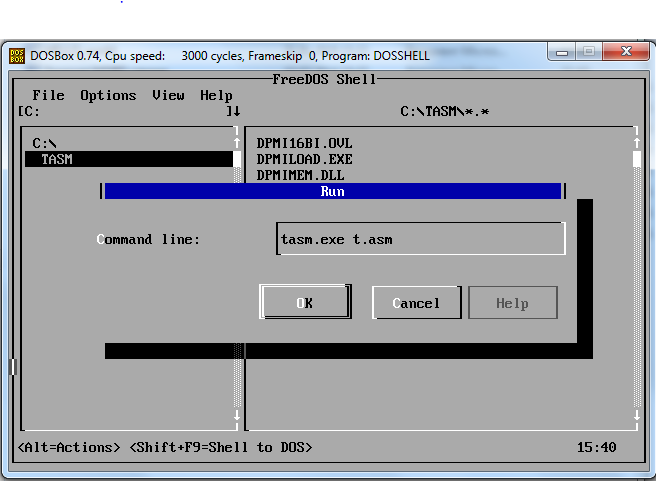
Рис. 2. Папка TASM с файлом T.ASM

6. Трансляция, компоновка и запуск программ в DOSBox (**Задание 4)**

Краткое описание задачи: изучение команд для трансляции, компоновки, запуска и отладки программ.

Для трансляции необходима команда **tasm.exe t.acm**.

Для формирования любой команды в оболочке FreeDOS Shell меню File выбираем пункт Run - рис. 3.

Рис. 3. Формирование командной строки в FreeDOS Shell

Если программа не содержит синтаксических ошибок, то транслятор построит объектный файл **t.obj**. К объектному файлу применяем компоновку для получения исполняемого файла (**t.exe**):

**tlink t.obj**.

Запускаемый файл **t.exe** дает необходимые результаты – рис. 4.

Если программа содержит синтаксические ошибки, то информацию о них можно прочитать в выходном документе транслятора **t.lst**. Листинг получаем после выполнения следующей команды:

**tasm t.asm /l**.

Здесь конструкция /l – ключ, задающий формирование листинга.

Поиск логических ошибок в программе производится с помощью программы отладчика, вызываемой командной строкой вида:

**td t.exe**.

Окно отладчика показано на рис. 4.

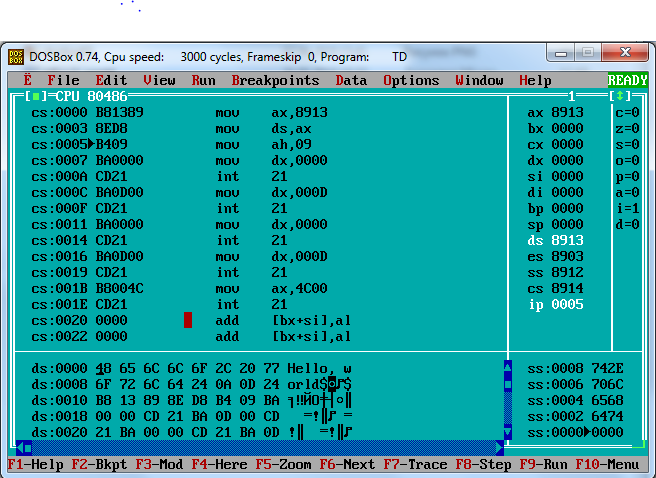


Рис. 4. Окно отладчика td.exe с загруженной программой печати «Hello world»

7. ЛАБОРАТОРНОЕ ЗАДАНИЕ

Получить от преподавателя задание на разработку алгоритма и программы на языке ассемблера.

Подготовить наборы тестовых начальных значений и соответствующих результатов, необходимых для проверки правильности разработанной программы.

Используя представленные в описании примеры работы с текстовыми редакторами и справочные материалы, произвести необходимые вычисления на виртуальной машине под управлением установленной ОС

.

8. СОДЕРЖАНИЕ ОТЧЕТА

* Цель работы.
* Текст варианта задания.
* Расчет тестовых начальных значений и соответствующих результатов, необходимых для проверки правильности разработанной программы.
* Текст программы с подробными комментариями.
* Результаты экспериментирования на ЭВМ.
* Выводы.

9. КОНТРОЛЬНЫЕ ВОПРОСЫ

Что такое виртуальная машина и как она реализована?

Какие этапы включает в себя процесс запуска на выполнение программ, спроектированных на языке ассемблера?

Какую структуру имеет программа на языке ассемблера?

Как сформировать командную строку для запуска процесса трансляции (компоновки, отладки, запуска на выполнение)?

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Морс С.П., Алберт Д.Д. Архитектура микропроцессора 80286: Пер. с англ. – М.: Радио и связь, 1990. – 304 с.