Министерство образования Республики Беларусь

Учреждение Образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №1

по дисциплине «Конструирование программ и языки программирования»

на тему «Создание простой программы на языке ассемблера»

вариант №2

|  |  |  |
| --- | --- | --- |
| Выполнил  Лагодич И.Р. №250504 |  | Проверил  Туровец Н.О. |

Минск 2023

**Цель работы:** Ознакомиться с программным обеспечением, предназначенным для сборки, отладки и запуска программ на языке ассемблера. Ознакомиться с основными особенностями архитектуры процессора и общей структурой программы..

**Теоретические сведения**

Ассемблер – это программа, которая переводит мнемонические текстовые команды в машинный код. Для формирования программ на языке ассемблера можно использовать разные среды разработки:

1) TASM (turbo assembler) – система программирования фирмы Borland, предназначенная для создания 16-битных программ для DOS и процессоров семейства Intel x86.

TASM включает в себя следующие основные компоненты:

-- компилятор языка ассемблер – *tasm.exe* – программа, предназначенная для компиляции файла написанного на языке ассемблера (*.asm*) в объектные модули (*.obj*).

-- компоновщик (linker) – *tlink.exe* – программа, предназначенная для сборки исполняемого файла из объектных модулей и библиотек.

-- отладчик (debugger) – *td.exe* – программа, предназначенная для отладки, созданных исполняемых файлов.

Все указанные выше программы получают опции через командную строку (все доступные функции можно получить, запустив программу без параметров). Кроме этого, Borland предоставляет интегрированную среду асемблера *ta.exe* – программу, объединяющую текстовый редактор и указанные выше компоненты и похожую по своим функциональным возможностям на среду «*Turbo C*».

1. MASM –система программирования от Microsoft, схожая с TASM.
2. FASM (flat assembler) – свободно распространяемый многопроходной ассемблер.
3. Эмулятор Emu8086 – система программирования, предназначенная для создания и отладки ассемблерных программ под Windows, имеет более дружественный интерфейс, чем DOS-среды, но работает достаточно медленно, а также имеет некоторые ограничения на работу с ресурсами.
4. Среда разработки программ для языка C\С++ (например, Borland C++ или MS Visual Studio) – в таких средах можно использовать команды встроенного ассемблера, что позволит изучить некоторые аспекты программирования на ассемблере без создания полноценной программы полностью на ассемблере.

Также следует понимать, что создание программ на ассемблере должно в требуемой мере использовать ресурсы и возможности операционной системы для взаимодействия с аппаратными средствами компьютера. Для лабораторных работ актуально использование операционной системы MS-DOS или FreeDOS – это будет актуально при работе с DOS-средами разработки ассемблерных программ, а также для создания резидентных программ. Для того чтобы не устанавливать отдельно DOS на компьютере c Windows (Unix) можно воспользоваться виртуальной машиной (например, Oracle VM VirtualBox) или средой DOSBox.

Для создания простейших программ на языке ассемблера требуется знание следующих понятий: 1. *Регистры процессора.*

В состав процессора Intel 8086 входят 16-ти битовые регистры:

-- общего назначения – АХ (аккумулятор), ВХ (база), СХ (счетчик), DX (данные), также можно обращаться отдельно к их старшей или младшей 8-ми битовой половине: AH, AL, BH, BL, CH, CL, DH и DL;

-- сегментные – CS (кода), DS (данных), SS (стека) и ES (расширенных данных);

-- адресные – SI (индекс источника), DI (индекс приемника) и BP (указатель базы); также к ним относят и регистр BX (база);

-- указатель стека – SP;

-- указатель команд – IP;

-- регистр флагов – FLAGS (биты признаков, см. рисунок 1).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  | OF | DF | IF | TF | SF | ZF |  | AF |  | PF |  | CF |

Рисунок 1 – Формат регистра флагов

Важные флаги:

-- CF – флаг переноса;

-- ZF – флаг нуля;

-- SF – флаг знака;

-- OF – флаг переполнения. 2. *Сегментная адресация памяти.*

Для адресации памяти процессор Intel 8086 использует 16-разрядные адресные регистры, что обеспечивает доступ к 64 Кбайт (000016–FFFF16) основной памяти. Т.к. все возможное адресное пространство памяти данного процессора составляет 1 Мбайт (0000016–FFFFF16), то при этом память приходится делить на сегменты объемом 64 Кбайт. Каждый сегмент начинается на границе параграфа (16 байт) от начала памяти, поэтому стартовыми для сегментов могут быть следующие адреса памяти – 0000016, 0001016, 0002016, …, FFFE016, FFFF016. Физический адрес байта памяти (20 бит) определяется суммой значения, заданного в сегментном регистре (16-бит) и умноженного на 16, со значением смещения (16-бит).

Существуют три основных типа сегментов:

-- сегмент кода (.code) – набор команд (CS:IP);

-- сегмент данных (.data) – данные программы ((DS | ES):(BX | SI | DI)); -- сегмент стека (.stack) – временные данные (SS:(BP | SP)). 3. *Структура программы.*

Программа на языке ассемблера состоит из строк общего формата:

метка команда/директива операнды ; комментарий

Метка может быть любой комбинацией букв английского алфавита, цифр и символов «\_», «$», «@», «?», при этом цифра не может быть первым символом метки, а символы «$» и «?» иногда имеют специальные значения и не рекомендуются к использованию. Регистр символов по умолчанию не различается.

Команда процессора – мнемоника, которая транслируется в исполняемый код. Директива – мнемоника, которая не приводит к появлению нового кода, а управляет работой самого ассемблера. В поле операндов располагаются требуемые командой или директивой операнды (то есть нельзя указать операнды и не указать команду или директиву).

Метка, стоящая перед командой, всегда заканчивается символом «:» (двоеточие) и фактически представляет собой адрес данной команды в программе. Метка, стоящая перед директивой, является одним из операндов директивы и символа двоеточия не имеет.

Комментарий – текст от символа «;» до конца строки, который не анализируется ассемблером.

Метки и комментарии в строке программы могут отсутствовать. 4. *Программные вызовы прерываний.*

Прерывание – это системная функция, которая обычно обеспечивает доступ к ресурсам компьютера и операционной системы (работа с вводомвыводом данных, переключение режимов, завершение программы и т.п.). Прерывания делятся на BIOS-прерывания (низкий уровень) и DOS-прерывания (высокий уровень). Для вызова прерывания внутри программы используется команда INT число.

5. *Расположение директив и форматы исполняемых файлов.*

Основные два формата исполняемых файлов в DOS — *com* и *exe*.

Файлы типа *com* содержат только скомпилированный код без какой-либо дополнительной информации о программе. Весь код, данные и стек такой программы располагаются в одном сегменте и не могут превышать 64 килобайта.

Файлы типа *exe* содержат заголовок, в котором описывается размер файла, требуемый объем памяти, список команд в программе, использующих абсолютные адреса, которые зависят от расположения программы в памяти, и т.д. Размер *exe*-файла может быть значительно больше *com*-файла, за счет использования нескольких сегментов памяти.

Ниже приведены два примера программ «Hello, world!» с использованием упрощенных директив описания сегментов, поясненных в комментариях, которые демонстрируют общую структуру программ написанных полностью на языке ассемблера.

**Код программы (.com)**

.model tiny

.code

org 100h

start: mov ah, 09h

mov dx, offset str1

int 21h

ret

str1 db "String", 0Ah, 0Dh, '$'

end start

**Код программы (.exe)**

.model small

.stack 100h

.code

start: mov ax, @data

mov ds, ax

mov dx, offset str1

mov ah, 09h

int 21h

mov ax, 4C00H

int 21h

.data

str1 db "String", 0Ah, 0Dh, '$'

end start

**Вывод программы**

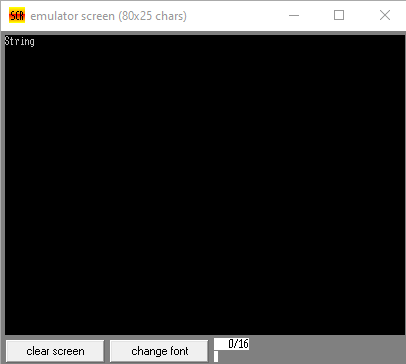
****

Рисунок 1 – Результат работы программы