**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Организация ЭВМ и систем»**

Тема: Трансляции, отладка и выполнение программы на языке Ассемблера

|  |  |  |
| --- | --- | --- |
| Студент гр. 9383 |  | Гордон Д.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2020

**Цель работы.**

Применить на практике знания о работе с регистрами процессора и познакомиться с основами программирования на языке ассемблер в операционной системе DOS.

**Текст задания.**

Лабораторная работа 1 использует 2 готовых программы на ассемблере: hello1 – составлена с использованием сокращенного описания сегментов и hello2 – составлена с полным описанием сегментов и выводом строки, оформленным как процедура. Выполнение работы состоит из двух частей, по каждой из которых необходимо представить протокол с фиксацией всех выполняемых действий и полученных результатов, и подписать его у преподавателя.

Уточнение задания следует посмотреть в файле lr1\_comp.txt каталога Задания.

Часть 1

1. Просмотреть программу hello1.asm, которая формирует и выводит на экран приветствие пользователя с помощью функции ОС MSDOS, вызываемой через прерывание с номером 21H (команда Int 21h).

Выполняемые функцией действия и задаваемые ей параметры - следующие:

- обеспечивается вывод на экран строки символов, заканчивающейся знаком "$";

- требуется задание в регистре ah номера функции, равного 09h, а в регистре dx - смещения адреса выводимой строки;

- используется регистр ax и не сохраняется его содержимое.

2. Разобраться в структуре и реализации каждого сегмента программы. Непонятные фрагменты прояснить у преподавателя. Строку-приветствие преобразовать в соответствии со своими личными данными.

3. Загрузить файл hello1.asm из каталога Задания в каталог Masm.

4. Протранслировать программу с помощью строки

> masm hello1.asm

c созданием объектного файла и файла диагностических сообщений (файла листинга). Объяснить и исправить синтаксические ошибки, если они будут обнаружены транслятором. Повторить трансляцию программы до получения объектного модуля.

5. Скомпоновать загрузочный модуль с помощью строки

> link hello1.obj

с созданием карты памяти и исполняемого файла hello1.exe.

6. Выполнить программу в автоматическом режиме путем набора строки

> hello1.exe

убедиться в корректности ее работы и зафиксировать результат выполнения в протоколе.

7. Запустить выполнение программы под управлением отладчика с помощью команды

> afd hello1.exe

Часть 2

Выполнить пункты 1 - 7 части 1 настоящего задания применительно к программе hello2.asm, приведенной в каталоге Задания, которая выводит на экран приветствие пользователя с помощью процедуры WriteMsg, а также использует полное определение сегментов. Сравнить результаты прогона под управлением отладчика программ hello1 и hello2 и объяснить различия в размещении сегментов.

Протокол

Часть 1.

|  |
| --- |
| Hello1.asm |
| Программа просмотрена. |
| Разобрался в структуре программы, данные строки-приветствия были изменены. |
| Файл загружен. |
| Ошибки обнаружены не были. |
| Загрузочный модуль скомпонован, карта памяти записана в файл hello1.map. |
| Программа завершилась корректно, на экран было выведено сообщение: «Вас приветствует ст.гр.7303 - Иванов И.И.» |

|  |
| --- |
| Hello2.asm |
| Программа просмотрена. |
| Разобрался в структуре программы, данные строки-приветствия были изменены. |
| Файл загружен. |
| Ошибки были в строчке 28 — отсутствовали запятые при многократном вызове директивы ASSUME. |
| Загрузочный модуль скомпонован, карта памяти записана в файл hello2.map. |
| Программа завершилась корректно, на экран было выведено сообщение: «Hello Worlds!  Student from 4350 - ». |

Часть 2.

**hello1.exe**

CS = 1A05, DS = 19F5, ES = 19F5, SS = 1A0A, SP = 0100

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 16-ричный код команды | Содержимое регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 0010 | MOV AX, 1A07 | B8071A | AX = 0000  IP = 0010 | AX = 1A07  IP = 0013 |
| 0013 | MOV DS, AX | 8ED8 | AX = 1A07  DS = 19F5  IP = 0013 | AX = 1A07  DS = 1A07  IP = 0015 |
| 0015 | MOV DX. 0000 | BA0000 | DX = 0000  IP = 0015 | DX = 0000  IP = 0018 |
| 0018 | MOV AH, 09 | B409 | AH = 1A  IP = 0018 | AH = 09  IP = 001A |
| 001A | INT 21 | CD21 | IP = 001A | IP = 001C |
| 001C | MOV AH, 4C | B44C | AH = 09  IP = 001C | AH = 4C  IP = 001E |
| 001E | INT 21 | CD21 | IP = 001E |  |

**hello2.exe**

CS = 1A0A, DS = 19F5, ES = 19F5, SS = 1A05, SP = 0018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 16-ричный код команды | Содержимое регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 0005 | PUSH DS | 1E | DS = 19F5  SP = 0018 | DS = 19F5  SP = 0016 |
| 0006 | SUB AX, AX | 2BC0 | AX = 0000 | AX = 0000 |
| 0008 | PUSH AX | 50 | AX = 0000  SP = 0016 | AX = 0000  SP = 0014 |
| 0009 | MOV AX, 1A07 | B8071A | AX = 0000 | AX = 1A07 |
| 000C | MOV DS, AX | 8ED8 | DS = 19F5 | DS = 1A07 |
| 000E | MOV DX, 0000 | BA0000 | DX = 0000 | DX = 0000 |
| 0011 | CALL 0000 | E8ECFF | AX = 1A07  SP = 0014 | AX = 0907  SP = 0012 |
| 0000 | MOV AH, 09 | B409 | AH = 1A | AH = 09 |
| 0002 | INT 21 | CD21 |  |  |
| 0004 | RET | C3 | SP = 0012 | SP = 0014 |
| 0014 | MOV DX, 0010 | BA1000 | DX = 0000 | DX = 0010 |
| 0017 | CALL 0000 | E8E6FF | SP = 0014 | SP = 0012 |
| 0000 | MOV AH, 09 | B409 | AH = 09 | AH = 09 |
| 0002 | INT 21 | CD21 |  |  |
| 0004 | RET | C3 | SP = 0012 | SP = 0014 |
| 001A | RET Far | CB | CS = 1A0A | CS = 19F5 |
| 0000 | INT 20 | CD20 |  |  |

PUSH – занести значение регистров и ячеек памяти в стек по адресу SS:SP

MOV A, B – переместить значение из B → A

RET – возврат в программу

RET FAR – процедуры FAR вы можете вызывать  
вне сегмента, в котором они определяются. Вызов FAR заносит в  
стек адрес в виде сегмента и смещения, а затем устанавливает  
CS:IP в адрес процедуры. Когда процессор обнаруживает возврат  
дальнего типа, он извлекает из стека сегмент и смещение адреса возврата и устанавливает в него CS:IP .

RET NEAR – возвращает в тот сегмент, где была определена процедура. Вызов ближнего типа заносит адрес возврата в стек и устанавливает IP в значение смешения процедуры.

SUB A, B – из A вычесть B и записать в A.

CALL x – передаёт управление команде, находящейся по адресу x.

DB – байт.

PSP – структура данных, где хранится состояние программы. Адрес хранится в DS.

INT 21h – функция DOS -- вызов прерывания (считывает из AH номер прерывания).

OFFSET– адрес переменной

**ВЫВОДЫ**

Познакомился с основами программирования на языке Assembly.