МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Организация ЭВМ и систем»
Тема: Трансляция, отладка и выполнение программ на языке Ассемблера

Студент гр. 0383 Смирнов И. А.
Преподаватель Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Знакомство с процессами трансляции, отладки и выполнения программ на языке Ассемблера

Задание.

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:

   1.a.обеспечивается вывод на экран строки символов, заканчивающейся знаком "$";

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

   1.c.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.

   Загрузить файл hello1.asm из каталога Задания в каталог Masm. Протранслировать программу с помощью строки

   > masm hello1.asm

с 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.

Скомпоновать загрузочный модуль с помощью строки

- link hello1.obj

созданием карты памяти и исполняемого файла hello1.exe.

Выполнить программу в автоматическом режиме путем набора строки

> hello1.exe

убедиться в корректности ее работы и зафиксировать результат выполнения в протоколе

Запустить выполнение программы под управлением отладчика с помощью команды

> afd hello1.exe

Записать начальное содержимое сегментных регистров CS, DS, ES и SS. Выполнить программу в пошаговом режиме с фиксацией используемых регистров и ячеек памяти до и после выполнения каждой команды. Обычные команды выполняются по F1 (Step), а вызовы обработчиков прерываний (Int) - по F2 (StepProc), чтобы не входить внутрь обработчика прерываний. Продвижение по сегментам экранной формы отладчика

выполняется с помощью клавиш F7 – F10 (up, down, left, right). Перезапуск программы в отладчике выполняется клавишей F3 (Retrieve). Выход из отладчика - по команде Quit.

Результаты прогона программы под управлением отладчика должны быть представлены в виде, показанном на примере одной команды в табл.1, и подписаны преподавателем.

Табл.1

Адрес Команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			до выполнения .	После выполнения
0003	Mov DS, AX	8E D8	(AX) = 2D87 (DS) = 2D75 (IP) = 0003	(AX) = 2D87 (DS) = 2D87 (IP) = 0005

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.

Выполнение работы.

Часть 1

1. Программа hello1.asm была просмотрена, структура и реализация каждого сегмента программы были изучены

2. Программа была протранслирована с помощью строки
masm hello1.asm

3. Загрузочный модуль был скомпонован с помощью строки
link hello1.obj

4. Программа была выполнена в автоматическом режиме строкой
hello1.exe

5. Выполнение программы было запущено под управлением отладчика с помощью команды
afdpro.exe hello1.exe

6. Начальное содержимое сегментов:
(CS) = 1A05
(DS) = 19F5
(ES) = 19F5
(SS) = 1A0A

SS.7. Результаты прогона программы под управлением отладчика представлена в Таблице 2

Таблица 2.

Адрес команды	Символический код команды	16-ичный код команды	Содержимое регистров и ячеек памяти	
			до выполнения	после выполнения
0010	Mov AX, 1A07	B8071A	(AX) = 0000 (DS) = 19F5 (IP) = 0010	(AX) = 1A07 (DS) = 19F5 (IP) = 0013
0013	Mov DS, AX	8ED8	(AX) = 1A07 (DS) = 19F5 (IP) = 0013	(AX) = 1A07 (DS) = 1A07 (IP) = 0015
0015	Mov DX, 0000	BA0000	(AX) = 1A07 (DS) = 1A07 (IP) = 0015	(AX) = 1A07 (DS) = 1A07 (IP) = 0018
0018	Mov AH, 09	B409	(AX) = 1A07 (DS) = 1A07 (IP) = 0018	(AX) = 0907 (DS) = 1A07 (IP) = 001A
001A	Int 21	CD21	(AX) = 0907 (DS) = 1A07 (IP) = 001A	(AX) = 0907 (DS) = 1A07 (IP) = 001C
001C	Mov AH, 4C	B44C	(AX) = 0907 (DS) = 1A07 (IP) = 001C	(AX) = 4C07 (DS) = 1A07 (IP) = 001E
001E	Int 21	CD21	(AX) = 4C07 (DS) = 1A07 (IP) = 001E	Завершение работы

Часть 2.

1. Были проделаны пункты 1-5 из 1 части выполнения работы с файлами hello2.xxx

2.Начальное содержимое сегментов:

(CS) = 1A0A

(DS) = 19F5

(ES) = 19F5

(SS) = 1A05

Адрес команды	Символическ ийкод команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0005	push ds	1E	(AX) = 0000 (DX) = 0000 (DS) = 19F5 (CS) = 1A0A	(AX) = 0000 (DX) = 0000 (DS) = 19F5 (CS) = 1A0A
			(SP) = 0018 (IP) = 0005 Stack +0 0000	(SP) = 0016 (IP) = 0006 Stack +0 19F5
0006	sub ax, ax	2BC0	(AX) = 0000 (DX) = 0000 (DS) = 19F5 (CS) = 1A0A (SP) = 0016 (IP) = 0006 Stack +0 19F5	(AX) = 0000 (DX) = 0000 (DS) = 19F5 (CS) = 1A0A (SP) = 0016 (IP) = 0008 Stack +0 19F5
0008	push ax	50	(AX) = 0000 (DX) = 0000 (DS) = 19F5 (CS) = 1A0A (SP) = 0016 (IP) = 0008 Stack +0 19F5	(AX) = 0000 (DX) = 0000 (DS) = 19F5 (CS) = 1A0A (SP) = 0014 (IP) = 0009 Stack +0 0000 Stack +0 19F5
0009	mov ax, 1A07	B8071A	(AX) = 0000 (DX) = 0000 (DS) = 19F5 (CS) = 1A0A (SP) = 0014	(AX) = 1A07 (DX) = 0000 (DS) = 19F5 (CS) = 1A0A (SP) = 0014

			(IP) = 0009 Stack +0 0000 Stack +0 19F5	(IP) = 000C Stack +0 0000 Stack +0 19F5
000C	mov ds, ax	8ED8	(AX) = 1A07 (DX) = 0000 (DS) = 19F5	(AX) = 1A07 (DX) = 0000 (DS) = 1A07
			(CS) = 1A0A (SP) = 0014 (IP) = 000C Stack +0 0000 Stack +0 19F5	(CS) = 1A0A (SP) = 0014 (IP) = 000E Stack +0 0000 Stack +0 19F5
000E	mov dx, 0000	BA0000	(AX) = 1A07 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0014 (IP) = 000E Stack +0 0000 Stack +0 19F5	(AX) = 1A07 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0014 (IP) = 0011 Stack +0 0000 Stack +0 19F5
0011	call 0000	E8ECFF	(AX) = 1A07 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0014 (IP) = 0011 Stack +0 0000 Stack +0 19F5	(AX) = 1A07 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0000 Stack +0 0014 Stack +2 0000 Stack +4 19F5
0000	mov ah 09	B409	(AX) = 1A07 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A	(AX) = 0907 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A

			(SP) = 0012 (IP) = 0000 Stack +0 0014	(SP) = 0012 (IP) = 0002 Stack +0 0014
			Stack +2 0000 Stack +4 19F5	Stack +2 0000 Stack +4 19F5
0002	int 21	CD21	(AX) = 0907 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0002 Stack +0 0014 Stack +2 0000 Stack +4 19F5	(AX) = 0907 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0004 Stack +0 0014 Stack +2 0000 Stack +4 19F5
0004	ret	C3	(AX) = 0907 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0004 Stack +0 0014 Stack +2 0000 Stack +4 19F5	(AX) = 0907 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0014 (IP) = 0014 Stack +0 0000 Stack +0 19F5
0014	mov dx, 0010	BA1000	(AX) = 0907 (DX) = 0000 (DS) = 1A07 (CS) = 1A0A (SP) = 0014 (IP) = 0014 Stack +0 0000 Stack +0 19F5	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 1A0A (SP) = 0014 (IP) = 0017 Stack +0 0000 Stack +0 19F5

0017	call 0000	E8E6FF	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 1A0A (SP) = 0014 (IP) = 0017 Stack +0 0000 Stack +0 19F5	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0000 Stack +0 001A Stack +2 0000 Stack +4 19F5
0000	mov ah, 09	B409	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0000 Stack +0 001A Stack +2 0000 Stack +4 19F5	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0002 Stack +0 001A Stack +2 0000 Stack +4 19F5
0002	int 21	CD21	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0002 Stack +0 001A Stack +2 0000 Stack +4 19F5	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 1A0A (SP) = 0012 (IP) = 0004 Stack +0 001A Stack +2 0000 Stack +4 19F5
0004	ret	C3	(AX) = 0907 (DX) = 0010 (DS) = 1A07	(AX) = 0907 (DX) = 0010 (DS) = 1A07

			(CS) = 1A0A (SP) = 0012 (IP) = 0004 Stack +0 001A Stack +2 0000 Stack +4 19F5	(CS) = 1A0A (SP) = 0014 (IP) = 001A Stack +0 0000 Stack +2 19F5
001A	ret far	CB	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 1A0A (SP) = 0014 (IP) = 001A Stack +0 0000 Stack +2 19F5	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 19F5 (SP) = 0018 (IP) = 0000 Stack +0 0000
0000	int 20	CD20	(AX) = 0907 (DX) = 0010 (DS) = 1A07 (CS) = 19F5 (SP) = 0018 (IP) = 0000 Stack +0 0000	Программа завершилась

Выводы.

Были рассмотрены процессы процессы трансляции, отладки и выполнения программ на языке Ассемблера

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл hello1.asm

```
; HELLO1.ASM -  упрощенная версия учебной программы лаб.раб. N1
;               по дисциплине "Архитектура компьютера"
; ***************************************************************
; Назначение: Программа формирует и выводит на экран приветствие
;             пользователя с помощью функции ДОС "Вывод строки"
;             (номер 09 прерывание 21h), которая:
;              - обеспечивает вывод на экран строки символов,
;                заканчивающейся знаком "$";
;              - требует задания в регистре ah номера функции=09h,
;                а в регистре dx -  смещения  адреса  выводимой
;                строки;
;              - использует регистр  ax  и  не сохраняет его
;                содержимое.
; ***************************************************************
   DOSSEG                                    ; Задание сегментов под ДОС

   .MODEL  SMALL                             ; Модель памяти-SMALL(Малая)
   .STACK  100h                              ; Отвести под Стек 256 байт
   .DATA                                     ; Начало сегмента данных
Greeting  LABEL  BYTE                        ; Текст  приветствия

   .CODE                                   ; Начало сегмента кода
   mov  ax, @data                           ; Загрузка в DS адреса начала
   mov  ds, ax                              ; сегмента данных
   mov  dx, OFFSET Greeting                 ; Загрузка в dx смещения
                                           ; адреса текста приветствия
DisplayGreeting:
   mov  ah, 9                               ; # функции ДОС печати строки
   int  21h                                ; вывод на экран  приветствия
   mov  ah, 4ch                             ; # функции ДОС завершения программы
   int  21h                                ; завершение программы и выход в ДОС
   END
```

Файл hello2.asm

```
; HELLO2 - Учебная программа N2  лаб.раб.#1 по дисциплине "Архитектура компьютера"
;          Программа использует процедуру для печати строки
;
;      ТЕКСТ  ПРОГРАММЫ

EOFLine    EQU    '$'           ; Определение символьной константы
                                ;     "Конец строки"

; Стек    программы


ASSUME CS:CODE, SS:AStack

AStack     SEGMENT  STACK
           DW 12 DUP('!')    ; Отводится 12 слов памяти
AStack     ENDS


; Данные программы

DATA       SEGMENT

;  Директивы описания данных
```

```
HELLO     DB 'Hello Worlds!', 0AH, 0DH,EOFLine
GREETING  DB 'Student from 0383 - Smirnov I.A.$'
DATA      ENDS

; Код программы

CODE      SEGMENT
; Процедура печати строки
WriteMsg  PROC  NEAR
          mov             AH,9
          int             21h  ; Вызов функции DOS по прерыванию
          ret
WriteMsg  ENDP

; Головная процедура
Main      PROC  FAR
          push  DS       ;\  Сохранение адреса начала PSP в стеке
          sub   AX,AX    ; > для последующего восстановления по
          push  AX       ;/  команде ret, завершающей процедуру.
          mov   AX,DATA             ; Загрузка сегментного
          mov   DS,AX               ; регистра данных.
          mov   DX, OFFSET HELLO    ; Вывод на экран первой
          call  WriteMsg            ; строки приветствия.
          mov   DX, OFFSET GREETING ; Вывод на экран второй
          call  WriteMsg            ; строки приветствия.
          ret                       ; Выход в DOS по команде,
                                    ; находящейся в 1-ом слове PSP.
Main      ENDP
CODE      ENDS
          END Main
```