

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

Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по практической работе № 1
по дисциплине «Операционные системы»

Тема: Исследование структур загрузочных модулей

Студент гр. 0381

Соколов Д. В.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2022

Цель работы.

Исследование различий в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

Постановка задачи.

Написать код исходного .COM модуля, который определяет тип РС и версию системы.

Ассемблерная программа должна читать содержимое предпоследнего байта ROM BIOS, сравнивать коды по таблице (таблица 1), определять тип РС и выводить строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код должен переводиться в символьную строку, содержащую запись шестнадцатеричного числа и выводиться на экран в виде соответствующего сообщения.

Затем программа определяет версию системы. Она должна по значениям регистров AL и AH формировать текстовую строку в формате xx.уу, где xx – номер основной версии, а уу - номер модификации в десятичной системе счисления, формировать строки с серийным номером OEM (Original Equipment Manufacturer) и серийным номером пользователя. Полученные строки выводятся на экран.

За основу были взяты шаблоны процедур, представленных в методических указаниях к данной лабораторной работе: TETR_TO_HEX, BYTE_TO_HEX, WRD_TO_HEX, BYTE_TO_DEC. А также написаны собственные функции. Описание процедур программы приведено в таблице 2.

Таблица 1. Сопоставление типа IBM PC с шестнадцатеричным кодом.

Тип IBM PC	Шестнадцатеричный код
PC	FF
PC/XT	FE, FB
AT	FC
PS 2 модель 30	FA
PS 2 модель 50/60	FC
PS 2 модель 80	F8
PCjr	FD
PC convertible	F9

Таблица 2. Функции в программе

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа
BYTE_TO_HEX	Перевод байта в 16-ной с/с в символьный код
WRD_TO_HEX	Перевод слова в 16-ной с/с в символьный код
BYTE_TO_DEC	Перевод байта в 16-ной с/с в символьный код в 10-ной с/с
print	Вывод строки на экран
pc_type_definition	Определение типа PC
version_defenition	Определение характеристик OS

Выполнение работы.

Были объявлены строки для вывода информации:

- PC_TYPE db 'Type: PC',0DH,0AH,'\$';
 - XT_TYPE db 'Type: PC/XT',0DH,0AH,'\$';
 - AT_TYPE db 'Type: AT',0DH,0AH,'\$';
 - PS_2_30 db 'Type: PS2 модель 30',0DH,0AH,'\$';
 - PS_2_80 db 'Type: PS2 модель 80',0DH,0AH,'\$';
 - JR_TYPE db 'Type: PCjr',0DH,0AH,'\$';
 - PC_CONVERT db 'Type: PC Convertible',0DH,0AH,'\$';
 - UNDEF db 'Undefined IBM PC type code: h', 0Dh, 0Ah, '\$'
-
- VER db 'Version MS-DOS: . ',0DH,0AH,'\$';
 - OEM db 'Serial number OEM: ',0DH,0AH,'\$';
 - USER db 'User serial number: H \$'.

Были составлены функция для определения типа ПК pc_type_defenition в соответствии с таблицей 1.

А также функция для определения характеристик ОС version_defenition:

- номер основной версии системы и её модификации;
- номер OEM;
- серийный номер пользователя.

В результате выполнения были получены следующие значения(рис.1-3):

```
S:\>lab1_com.com
IBM PC type: AT or PS2 (50 or 60)
MS-DOS version: 5.0
OEM serial number: 0
User serial number: 000000h
S:\>S
```

Рисунок 1. Результат работы «хорошего» .СОМ модуля

```
S:\>lab1_com.exe
5 0
0          ev@IBM PC type: PC
ev@IBM PC type: PC
ev@IBM PC type: P000000M PC type: PC/XT
S:\>S_
```

Рисунок 2. Результат работы «плохого» .EXE модуль

```
S:\>lab1_exe.exe
IBM PC type: AT or PS2 (50 or 60)
MS-DOS version: 5.0
OEM serial number: 0
User serial number: 000000h
S:\>S
```

Рисунок 3 – «хороший» .EXE модуль

Выводы.

В ходе лабораторной работы были исследованы различия в структурах исходных текстов модулей типов .СОМ и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

ПРИЛОЖЕНИЕ А

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

Отличия исходных текстов СОМ и EXE программ:

1. Сколько сегментов должна содержать СОМ-программа?

СОМ-программа должна содержать ровно один сегмент. Код и данные находятся в одном сегменте, а стек генерируется автоматически.

2. EXE-программа?

EXE-программа должна содержать не менее одного сегмента. Сегменты кода, данных и стека описываются отдельно друг от друга, но есть возможность не описывать сегмент стека, в таком случае будет использоваться стек DOS.

3. Какие директивы должны быть обязательно в тексте СОМ-программы?

Должна быть обязательна директива ORG 100h, так как при загрузке модуля все сегментные регистры содержат адрес префикса программного сегмента (PSP), который является 256-байтовым(100H) блоком, поэтому адресация имеет смещение в 256 байт от нулевого адреса. Также необходима процедура ASSUME для того, чтобы сегмент данных и сегмент кода указывали на один общий сегмент. (ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING)

4. Все ли форматы команд можно использовать в СОМ-программе?

Не все форматы поддерживаются. Нельзя использовать команды вида mov <регистр>, seg <имя сегмента>, так как в .com-программе отсутствует таблица настроек (содержит описание адресов, которые зависят от размещения загрузочного модуля в ОП).

Отличия форматов файлов .COM и .EXE программ:

1. Какова структура файла .COM? С какого адреса располагается код?

COM-файл состоит из одного сегмента, состоящего из сегмент кода и сегмент данных, сегмент стека генерируется автоматически при создании COM-программы. COM-файл ограничен размером одного сегмента и не превышает 64 Кб

Код начинается с адреса 0h, но при загрузке модуля устанавливается смещение в 100h.

D:\Универ\0C\lab1\LAB1_COM.COM	
0000000000: E9 F5 01 54 79 70 65 3A	20 50 43 0D 0A 24 54 79 щї>Type: PC\$Tu
0000000010: 70 65 3A 20 50 43 2F 58	54 0D 0A 24 54 79 70 65 ре: PC/XT\$Type
0000000020: 3A 20 41 54 0D 0A 24 54	79 70 65 3A 20 50 53 32 : AT\$Type: PS2
0000000030: 20 D0 BC D0 BE D0 B4 D0	B5 D0 BB D1 8C 20 33 30 ШШЩЦЛГМ 30
0000000040: 0D 0A 24 54 79 70 65 3A	20 50 53 32 20 D0 BC D0 №\$Type: PS2 ШШ
0000000050: BE D0 B4 D0 B5 D0 BB D1	8C 20 35 30 20 D0 B8 D0 ШШЩЦЛГМ 50 ШШ
0000000060: BB D0 B8 20 36 30 0D 0A	24 54 79 70 65 3A 20 50 ШШЩЦЛГМ 60№\$Type: P
0000000070: 53 32 20 D0 BC D0 BE D0	B4 D0 B5 D0 BB D1 8C 20 S2 ШШЩЦЛГМ
0000000080: 38 30 0D 0A 24 54 79 70	65 3A 20 50 D0 A1 6A 72 80№\$Type: P6jr
0000000090: 0D 0A 24 54 79 70 65 3A	20 50 43 20 43 6F 6E 76 №\$Type: PC Conv
00000000A0: 65 72 74 69 62 6C 65 0D	0A 24 56 65 72 73 69 6F ertible№\$Versio
00000000B0: 6E 20 4D 53 2D 44 4F 53	3A 20 20 2E 20 20 0D 0A n MS-DOS: . №
00000000C0: 24 53 65 72 69 61 6C 20	6E 75 6D 62 65 72 20 4F \$Serial number 0
00000000D0: 45 4D 3A 20 20 0D 0A 24	55 73 65 72 20 73 65 72 EM: №\$User ser
00000000E0: 69 61 6C 20 6E 75 6D 62	65 72 3A 20 20 20 20 20 ial number:
00000000F0: 20 20 48 20 24 24 0F 3C	09 76 02 04 07 04 30 C3 Н \$\$\$\$\$\$ovе♦♦♦♦♦
0000000100: 51 8A E0 E8 EF FF 86 C4	B1 04 D2 E8 E8 E6 FF 59 QКршя Ж-ШШЩЦУ
0000000110: C3 53 8A FC E8 E9 FF 88	25 4F 88 05 4F 8A C7 E8 SKN%шш И%ОИ+OK ш
0000000120: DE FF 88 25 4F 88 05 5B	C3 51 52 32 E4 33 D2 B9 И%ОИ+ [QR2Ф3П
0000000130: 0A 00 F7 F1 80 CA 30 88	14 4E 33 D2 3D 0A 00 73 ☐ ѿА=0И9Н3П=с s
0000000140: F1 3C 00 74 04 0C 30 88	04 5A 59 C3 B4 09 CD 21 ё< т♦90И♦ZY о!=
0000000150: C3 B8 00 F0 8E C0 26 A0	FE FF 3C FF 74 1C 3C FE й0Л&aи < тЛ<
0000000160: 74 1E 3C FB 74 1A 3C FC	74 1C 3C FA 74 1E 3C F8 тА<vт><NтL<-тА<°
0000000170: 74 26 3C FD 74 28 3C F9	74 2A BA 03 01 EB 2B 90 т&<#t(<-t* ^θы+P
0000000180: BA 0E 01 EB 25 90 BA 1C	01 EB 1F 90 BA 27 01 EB θы%P θыvP θы
0000000190: 19 90 BA 43 01 EB 13 90	BA 69 01 EB 0D 90 BA 85 ↓P Сθы!!P iθыJР θы
00000001A0: 01 EB 07 90 BA 93 01 EB	01 90 E8 9F FF C3 B4 30 0ы•P θыθРшя 0
00000001B0: CD 21 50 BE AA 01 83 C6	10 E8 6D FF 58 8A C4 83 =!P кθГ шш ХК-Г
00000001C0: C6 03 E8 64 FF BA AA 01	E8 81 FF BE C1 01 83 C6 ^θшd кθшБ ^θГ
00000001D0: 13 8A C7 E8 53 FF BA C1	01 E8 70 FF BF D8 01 83 !!K шS θшр ^θГ
00000001E0: C7 19 8B C1 E8 2A FF 8A	C3 E8 14 FF 83 EF 02 89 θш* K шS Гяой
00000001F0: 05 BA D8 01 E8 55 FF C3	E8 56 FF E8 B0 FF 32 C0 ♦ ^θшU шV ш 2
0000000200: B4 4C CD 21	L=!

2. Какова структура файла «плохого» EXE? С какого адреса располагается код?
Что располагается с адреса 0?

В «плохого» EXE данные и код располагаются в одном сегменте, что для EXE файла некорректно, так как код и данные должны быть разделены на отдельные сегменты. Код располагается с адреса 300h, а с адреса 0h идёт таблица настроек.

0000000000: 4D 5A 49 00 03 00 00 00 00	20 00 00 00 FF FF 00 00 MZI ♥ яя
0000000010: 00 00 00 00 00 01 00 00 00	3E 00 00 00 01 00 FB 50 θ > θ ыР
0000000020: 6A 72 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 jr
0000000030: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000040: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000050: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000060: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000070: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000080: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000090: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000A0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000B0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000C0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000D0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000E0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000000F0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000100: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000110: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000120: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000130: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000140: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000150: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000160: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000170: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000180: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000190: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000001A0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000001B0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000001C0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000001D0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000001E0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000001F0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000200: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000210: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000220: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000230: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000240: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000250: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000260: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000270: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000280: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000290: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000002A0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000002B0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000002C0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000002D0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000002E0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000002F0: 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0000000300: E9 34 01 D0 97 D0 BD D0	B0 D1 87 D0 B5 D0 BD D0 й4ӨР–PSP°C‡PµPSP
0000000310: B8 D0 B5 20 D1 80 D0 B5	D0 B3 D0 B8 D1 81 D1 82 ёРµ СЂPµPiPёCЃC,

3. Какова структура «хорошего» EXE? Чем он отличается от файла «плохого» EXE?

В EXE-программе код, данные и стек поделены на сегменты. Программа в формате EXE может иметь любой размер. EXE-файл имеет заголовок, который используется при его загрузке. Заголовок состоит из форматированной части, содержащей сигнатуру и данные, необходимые для загрузки EXE-файла, и таблицы для настройки адресов. В отличии от «плохого» EXE в «хорошем» EXE присутствуют три сегмента: сегмент кода, сегмент данных и сегмент стека, а «плохой» EXE содержит один сегмент, совмещающий код и данные. Также в «плохом» EXE адресация кода начинается с 300h, так как он получается из .COM файла, в котором изначально сегмент кода смешён на 100h, а при создании «плохого» EXE к этому смещению добавляется размер PSP модуля(200h). А в «хорошем» EXE присутствует только смещение для PSP модуля, поэтому код начинается с 200h. В данной случае смещение кода 400h так как выделяется память под стек (200h), память под стек находится между PSP и кодом.

Загрузка СОМ модуля в основную память:

1. Какой формат загрузки модуля СОМ? С какого адреса располагается код?

Определяется сегментный адрес участка ОП, у которого достаточно места для загрузки программы, образ СОМ-файла считывается с диска и помещается в память, начиная с PSP:0100h. После загрузки двоичного образа СОМ-программы сегментные регистры CS, DS, ES и SS указывают на PSP(в данном случае сегментные регистры указывают на 48DD), SP указывает на конец сегмента PSP(обычно FFFE), слово 00H помещено в стек, IP содержит 100H в результате команды JMP PSP:100H.

The screenshot shows the DOSBox interface with the CPU 80486 window active. The assembly code pane displays the following instructions:

Address	OpCode	Mnemonic	Operands
cs:0100	E9F501	jmp	02F8 ↓
cs:0103	54	push	sp
cs:0104	7970	jns	0176
cs:0106	653A20	cmp	ah,gs:[bx+si]
cs:0109	50	push	ax
cs:010A	43	inc	bx
cs:010B	0D0A24	or	ax,240A
cs:010E	54	push	sp
cs:010F	7970	jns	0181
cs:0111	653A20	cmp	ah,gs:[bx+si]
cs:0114	50	push	ax
cs:0115	43	inc	bx
cs:0116	2F	das	

The registers pane shows the following values:

Register	Value	Description
ax	0000	c=0
bx	0000	z=0
cx	0000	s=0
dx	0000	o=0
si	0000	p=0
di	0000	a=0
bp	0000	i=1
sp	FFFE	d=0
ds	48DD	
es	48DD	
ss	48DD	
cs	48DD	
ip	0100	

The memory dump pane shows the following bytes at address ds:0000:

Address	Value
ds:0000	CD 20 FF 9F 00 EA FF FF
ds:0008	AD DE E4 01 C9 15 AE 01
ds:0010	C9 15 80 02 24 10 92 01
ds:0018	01 01 01 00 02 FF FF FF

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

2. Что располагается с адреса 0?

Программный сегмент PSP, размером 256 байт (100h), зарезервируемый операционной системой.

3. Какие значения имеют сегментные регистры? На какие области памяти они указывают?

Ссегментные регистры CS, DS, ES и SS указывают на PSP и имеют значения 48DD.

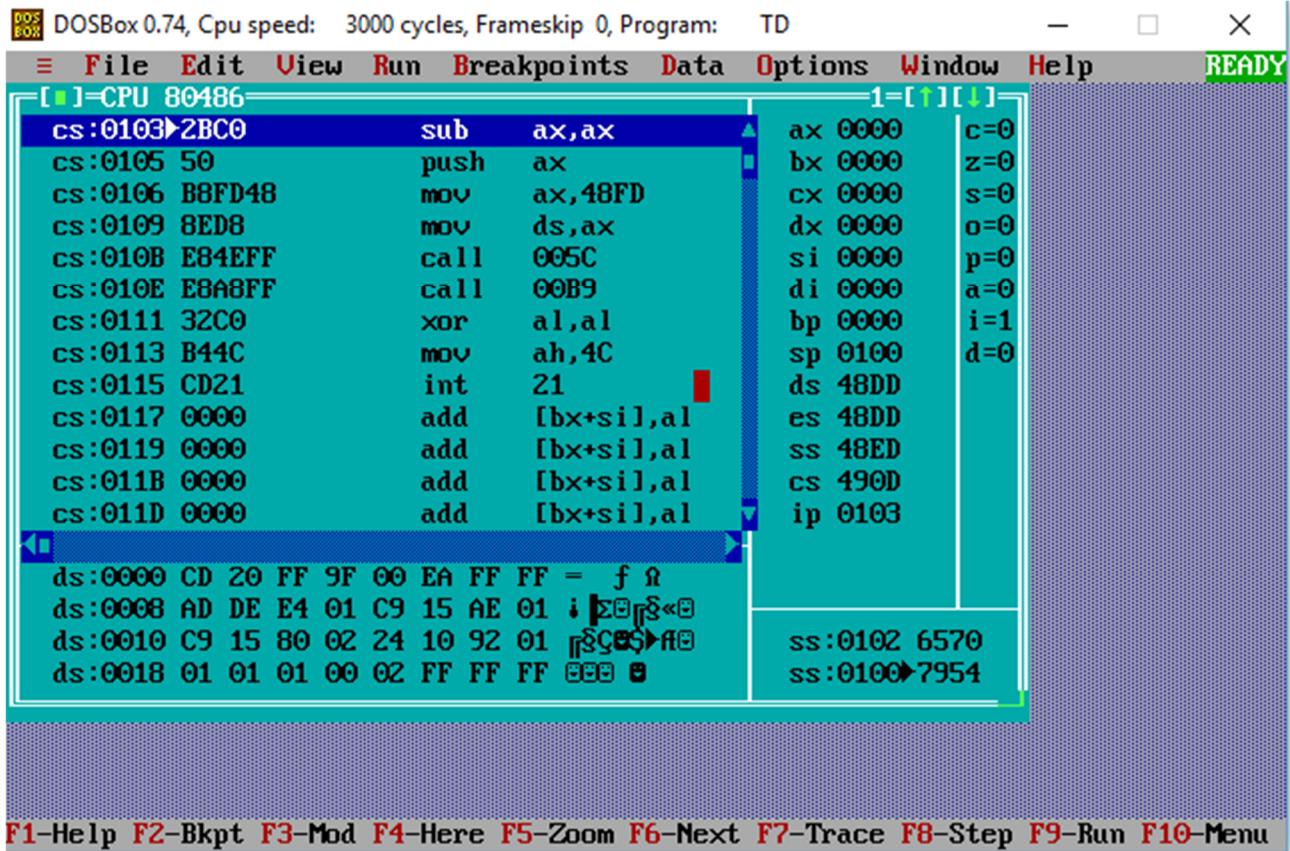
4. Как определяется стек? Какую область памяти он занимает? Какие адреса?

Стек генерируется автоматически при создании СОМ-программы. SS – на начало (0h), регистр SP указывает на конец стека (FFFEh), Адреса стека расположены в диапазоне 0h – FFFEh (FFFEh, – последний адрес, кратный двум).

Загрузка «хорошего» EXE модуля в основную память:

1. Как загружается «хороший» .EXE? Какие значения имеют сегментные регистры?

EXE-файл загружается, начиная с адреса PSP:0100h. В процессе загрузки считывается информация заголовка (PSP) EXE в начале файла и выполняется перемещение адресов сегментов, то есть DS и ES устанавливаются на начало сегмента PSP(DS=ES=48DD), SS(SS=48ED) – на начало сегмента стека, CS(CS=490D) – на начало сегмента команд. В IP загружается смещение точки входа в программу, которая берётся из метки после директивы END. Причём дополнительный программный сегмент (PSP) присутствует в каждом EXE-файле.



2. На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало сегмента PSP.

3. Как определяется стек?

Стек определяется с помощью директивы .stack, после которой задаётся размер стека. При исполнение регистр SS указывает на начало сегмента стека, а SP на конца стека(его смещение).

4. Как определяется точка входа?

Точка входа определяется при помощи директивы END.