

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

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Операционные системы»
Тема: Исследование структур загрузочных модулей.

Студент гр. 0381

Дзаппала Д.

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.

Задание.

Написать текст исходного .COM модуля, который определяет тип PC и версию системы. Ассемблерная программа должна читать содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определять тип PC и выводить строку с названием модели. Затем определяется версия системы. Ассемблерная программа должна по значениям регистров AL и AH формировать текстовую строку в формате xx.yy, где xx — номер основной версии, а yy - номер модификации в десятичной системе счисления, формировать строки с серийным номером OEM и серийным номером пользователя.

Написать текст исходного .EXE модуля, который выполняет те же функции, что и модуль .COM и постройте и отладьте его. Таким образом, будет получен «хороший» .EXE.

Сравнить исходные тексты для .COM и .EXE модулей. Ответить на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

Основные теоретические положения.

Шестнадцатеричные коды типов IBM PC: PC – **FF**, PC/XT – **FE (FB)**, AT – **FC**, PS2 (model 30) – **FA**, PS2 (model 50 or 60) – **FC**, PS2 (model 80) – **F8**, PCjr – **FD**, PC Convertible – **F9**.

Для определения версии MS-DOS используется функция 30h прерывания 21h. Входным параметром является номер функции в AH. Выходными параметрами являются: AL – номер основной версии, AH – номер модификации, BH – серийный номер OEM, BL:CH – 24-битовый серийный номер пользователь.

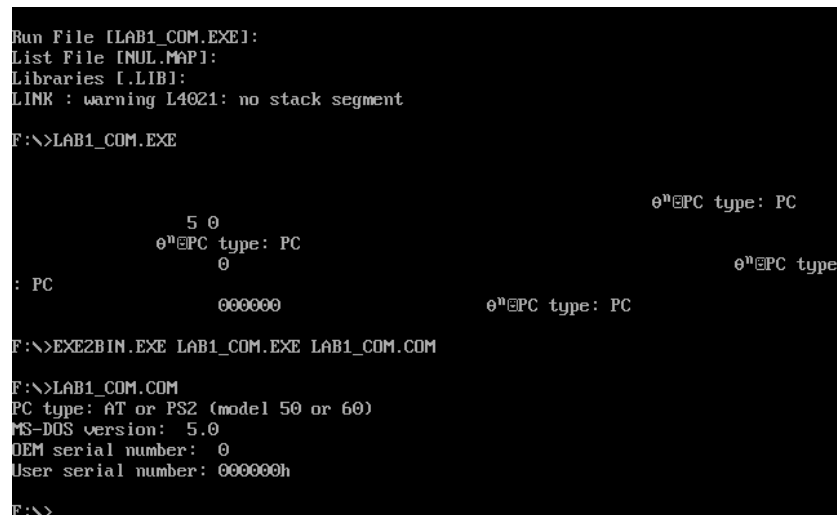
Выполнение работы.

Для выполнения задания были определены две процедуры: для определения типа IBM PC – DefPCType, для определения характеристик MS-DOS – DefSysVer. Были объявлены строки для вывода нужной информации:

- pc db 'PC type: PC', 0dh, 0ah, '\$'
- pc_xt db 'PC type: PC/XT', 0dh, 0ah, '\$'
- pc_at db 'PC type: AT or PS2 (model 50 or 60)', 0dh, 0ah, '\$'
- pc_ps2_30 db 'PC type: PS2 model 30', 0dh, 0ah, '\$'
- pc_ps2_80 db 'PC type: PS2 model 80', 0dh, 0ah, '\$'
- pc_pcjr db 'PC type: PCjr', 0dh, 0ah, '\$'
- pc_conv db 'PC type: PC Convertible', 0dh, 0ah, '\$'
- ms_dos_ver db 'MS-DOS version: . ', 0dh, 0ah, '\$'
- oem_sn db 'OEM serial number: ', 0dh, 0ah, '\$'
- user_sn db 'User serial number: h', 0dh, 0ah, '\$'

Тип PC AT и PS2 (модели 50 или 60) были объединены так как у них одинаковый код.

При запуске .COM модуля в виде «плохого» .EXE модуля получаем следующую информацию:



```
Run File [LAB1.COM.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

F:\>LAB1.COM.EXE

                    5 0
                    0"PC type: PC
                    0
                    0"PC type
: PC
                    000000
                    0"PC type: PC

F:\>EXE2BIN.EXE LAB1.COM.EXE LAB1.COM.COM

F:\>LAB1.COM.COM
PC type: AT or PS2 (model 50 or 60)
MS-DOS version: 5.0
OEM serial number: 0
User serial number: 000000h

F:\>
```

Используя утилиту EXE2BIN, получаем правильный .COM модуль, который выводит корректную информацию.

Для того, чтобы написать «хороший» .EXE модуль, нужно будет отделить данные от сегмента кода, добавив сегмент данных и сегмент стека. После сборки, получаем следующую информацию:

```
Object filename [LAB1_EXE.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

49950 + 453215 Bytes symbol space free

0 Warning Errors
0 Severe Errors

F:\>link LAB1_EXE.OBJ

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run File [LAB1_EXE.EXE]:
List File [NUL.MAP]: lab1EXE
Libraries [LIB1]:

F:\>LAB1_EXE.EXE
PC type: AT or PS2 (model 50 or 60)
MS-DOS version: 5.0
OEM serial number: 0
User serial number: 0000000h

F:\>_
```

Выводы.

Были исследованы различия в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов загрузки в основную память.

ПРИЛОЖЕНИЕ А

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ ПО ЛАБ.РАБОТЕ №1

Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM программа?

COM модуль должен содержать только один сегмент — сегмент кода (данные также находятся в этом сегменте), а стек формируется автоматически.

2. EXE программа?

Как минимум сегмент кода, в лучшем случае, определить отдельно сегмент кода, данных и стека.

3. Какие директивы должны обязательно быть в тексте COM программы?

ORG 100h — для смещения 256 байт, в которых размещается PSP (префикс программного сегмента). Assume — для указания, что сегмент кода является общим для кода и сегмента данных.

4. Все ли форматы команд можно использовать в COM программе?

Нет, так как в COM модуле нету таблицы настройки адресов (таблица, созданная транслятором, с указателями на сегменты), нельзя использовать команды, аргументы которых будут сегменты.

Отличия форматов файлов COM и EXE модулей

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

COM модуль состоит из одного сегмента — кода, совмещенного с данными. Размер исполняемого COM модуля не может превышать 64 Кб. Код начинается с адреса 0, но при загрузке программы устанавливается смещение 100h.

```
C:\Users\danie\Desktop\OS\lab1\LAB1_COM.COM
00000000: E9 E8 01 50 43 20 74 79 70 65 3A 20 50 43 2F 58 54 0D 0A 00 PC type: PC
00000010: 24 50 43 20 74 79 70 65 3A 20 50 43 2F 58 54 0D 0A 00 $PC type: PC/XT
00000020: 0A 24 50 43 20 74 79 70 65 3A 20 41 54 20 6F 72 0A 00 $PC type: AT or
00000030: 20 50 53 32 20 28 6D 6F 64 65 6C 20 35 30 20 6F 0A 00 PS2 (model 50 o
00000040: 72 20 36 30 29 0D 0A 24 50 43 20 74 79 70 65 3A 0A 00 r 60) $PC type:
00000050: 20 50 53 32 20 6D 6F 64 65 6C 20 33 30 0D 0A 24 0A 00 PS2 model 30 $
00000060: 50 43 20 74 79 70 65 3A 20 50 53 32 20 6D 6F 64 0A 00 PC type: PS2 mod
00000070: 65 6C 20 38 30 0D 0A 24 50 43 20 74 79 70 65 3A 0A 00 el 80 $PC type:
00000080: 20 50 43 6A 72 0D 0A 24 50 43 20 74 79 70 65 3A 0A 00 PCjr $PC type:
00000090: 20 50 43 20 43 6F 6E 76 65 72 74 69 62 6C 65 0D 0A 00 PC Convertible
000000A0: 0A 24 4D 53 2D 44 4F 53 20 76 65 72 73 69 6F 6E 0A 00 $MS-DOS version
000000B0: 3A 20 20 20 2E 20 20 0D 0A 24 4F 45 4D 20 73 65 0A 00 : . $OEM se
000000C0: 72 69 61 6C 20 6E 75 6D 62 65 72 3A 20 20 20 0D 0A 00 rial number:
000000D0: 0A 24 55 73 65 72 20 73 65 72 69 61 6C 20 6E 75 0A 00 $User serial nu
000000E0: 6D 62 65 72 3A 20 20 20 20 20 20 68 0D 0A 24 0A 00 mber: h $
000000F0: 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 EF 0A 00 $*ov ♦♦♦ÃQŠàèi
00000100: FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 0A 00 ŷ†Ã±♦ðèèæŷYÄŠšüè
00000110: E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F 0A 00 éŷ~%0^+0ŠÇèbŷ~%0
00000120: 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80 0A 00 ^+ [ÃQR2ä30¹ ÷ñ€
00000130: CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0A 00 Ê0^JN30= sñ< t♦
00000140: 0C 30 88 04 5A 59 C3 B8 00 F0 8E C0 26 A0 FE FF 0A 00 90^♦ZYÃ, ðŽÀ& pŷ
00000150: 3C FF 74 1C 3C FE 74 1E 3C FB 74 1A 3C FC 74 1C 0A 00 <ŷtL<pt▲<ût-><ütL
00000160: 3C FA 74 1E 3C F8 74 20 3C FD 74 22 3C F9 74 24 0A 00 <üt▲<øt <ŷt"<üt$
00000170: BA 03 01 EB 25 90 BA 11 01 EB 1F 90 BA 22 01 EB 0A 00 °♥0ë%0°◀0ë▼0°"0ë
00000180: 19 90 BA 48 01 EB 13 90 BA 60 01 EB 0D 90 BA 78 0A 00 ↓0°H0ë!!0°`0ë0°x
00000190: 01 EB 07 90 BA 88 01 EB 01 90 E8 49 00 C3 B4 30 0A 00 0ë•0°^0ë0°èI Ã´0
000001A0: CD 21 BE A2 01 83 C6 11 E8 79 FF 8A E0 83 C6 03 0A 00 Í!%¢0fÆ◀èŷŠšàfÆ♥
000001B0: E8 71 FF BA A2 01 E8 2D 00 BE BA 01 83 C6 14 8A 0A 00 èqŷ°¢0è- %°0fÆJŠ
000001C0: E7 E8 60 FF BA BA 01 E8 1C 00 BF D2 01 83 C7 19 0A 00 çè`ŷ°°0èL ¿00fÇ↓
000001D0: 8B C1 E8 37 FF 8A C3 E8 21 FF 83 EF 02 89 05 BA 0A 00 <Àè7ŷŠÀè!ŷfi0%+°
000001E0: D2 01 E8 01 00 C3 B4 09 CD 21 C3 E8 59 FF E8 AD 0A 00 00è0 Ã´oÍ!ÀèYè-
```

```
1Help 2Dump 3Quit 4Text 5 6Edit 7Search 8AN
```

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

Код и данные находятся в одном сегменте, что неправильно. Код располагается с адреса 300h, с адреса 0h располагается заголовок .EXE модуля.

```
C:\Users\danie\Desktop\OS\lab1\LAB1_COM.EXE
00000002C0: 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
00000002E0: 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
0000000300: E9 E8 01 50 43 20 74 79 70 65 3A 20 50 43 0D 0A  éèPC type: PC
0000000310: 24 50 43 20 74 79 70 65 3A 20 50 43 2F 58 54 0D  $PC type: PC/XT
0000000320: 0A 24 50 43 20 74 79 70 65 3A 20 41 54 20 6F 72  PC type: AT or
0000000330: 20 50 53 32 20 28 6D 6F 64 65 6C 20 35 30 20 6F  PS2 (model 50 o
0000000340: 72 20 36 30 29 0D 0A 24 50 43 20 74 79 70 65 3A  r 60)
0000000350: 20 50 53 32 20 6D 6F 64 65 6C 20 33 30 0D 0A 24  PC type:
0000000360: 50 43 20 74 79 70 65 3A 20 50 53 32 20 6D 6F 64  PS2 model 30
0000000370: 65 6C 20 38 30 0D 0A 24 50 43 20 74 79 70 65 3A  PC type: PS2 mod
0000000380: 20 50 43 6A 72 0D 0A 24 50 43 20 74 79 70 65 3A  el 80
0000000390: 20 50 43 20 43 6F 6E 76 65 72 74 69 62 6C 65 0D  PCjr
00000003A0: 0A 24 4D 53 2D 44 4F 53 20 76 65 72 73 69 6F 6E  PC Convertible
00000003B0: 3A 20 20 20 2E 20 20 0D 0A 24 4F 45 4D 20 73 65  MS-DOS version
00000003C0: 72 69 61 6C 20 6E 75 6D 62 65 72 3A 20 20 20 0D  : . OEM se
00000003D0: 0A 24 55 73 65 72 20 73 65 72 69 61 6C 20 6E 75  rial number:
00000003E0: 6D 62 65 72 3A 20 20 20 20 20 20 68 0D 0A 24  User serial nu
00000003F0: 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 EF  mber: h
0000000400: FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8  $<ov
0000000410: E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F  y+Ä±
0000000420: 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80  éy^%0^OSÇèbÿ^%0
0000000430: CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04  ^+[ÄQR2ä30± ÷ñ€
0000000440: 0C 30 88 04 5A 59 C3 B8 00 F0 8E C0 26 A0 FE FF  Ê0^JN30=sñ< t
0000000450: 3C FF 74 1C 3C FE 74 1E 3C FB 74 1A 3C FC 74 1C  90^ZYÃ, õŽÀ& pÿ
0000000460: 3C FA 74 1E 3C F8 74 20 3C FD 74 22 3C F9 74 24  <ÿtL<pt▲<Ût-><ÛtL
0000000470: BA 03 01 EB 25 90 BA 11 01 EB 1F 90 BA 22 01 EB  <Ût▲<øt <ÿt"<Ût$
0000000480: 19 90 BA 48 01 EB 13 90 BA 60 01 EB 0D 90 BA 78  °♥ö%◻°◻ö◻°◻ö
0000000490: 01 EB 07 90 BA 88 01 EB 01 90 E8 49 00 C3 B4 30  ↓°Höë!!°◻°ö°◻°x
00000004A0: CD 21 BE A2 01 83 C6 11 E8 79 FF 8A E0 83 C6 03  öë°◻°^öë°◻èI Ä^0
Í!%øøfÄ-èÿÿŠàfA♥
```

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

Такой файл не имеет лимита в размере; содержит три отдельных друг от друга сегмента: сегмент кода, сегмент данных и сегмент стека. Структура: заголовок (с информацией), сегмент стека (200h-280h, 128 байт), сегмент кода (280h – 386h) и сегмент данных (с 390h до конца файла).

```
C:\Users\danie\Desktop\OS\lab1\LAB1_EXE.EXE
00000001F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000200: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
0000000210: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
0000000220: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
0000000230: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
0000000240: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
0000000250: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
0000000260: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
0000000270: 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31
0000000280: 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 EF $<ov000000290: FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 yT±0000002A0: E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F éy^%0^0Šçèbÿ^%0
0000002B0: 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80 ^+[ÄQR2ä30¹ ÷ñ€
0000002C0: CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 Ê0^ñN30= sñ< t
0000002D0: 0C 30 88 04 5A 59 C3 B8 00 F0 8E C0 26 A0 FE FF 90^ZYÄ, öZ& pÿ
0000002E0: 3C FF 74 1C 3C FE 74 1E 3C FB 74 1A 3C FC 74 1C <yT<pT<ûT-><ûT
0000002F0: 3C FA 74 1E 3C F8 74 20 3C FD 74 22 3C F9 74 24 <ûT<pT <yT" <ûT$
000000300: BA 00 00 EB 25 90 BA 0E 00 EB 1F 90 BA 1F 00 EB 0 ẽ%00 ẽ▼00 ẽ
000000310: 19 90 BA 45 00 EB 13 90 BA 5D 00 EB 0D 90 BA 75 ↓00E ẽ!!00] ẽ!00u
000000320: 00 EB 07 90 BA 85 00 EB 01 90 E8 49 00 C3 B4 30 ẽ•00... ẽ00èI Ä´0
000000330: CD 21 BE 9F 00 83 C6 11 E8 79 FF 8A E0 83 C6 03 Í!%ÿ fÄ-èÿÿŠäfA♥
000000340: E8 71 FF BA 9F 00 E8 2D 00 BE B7 00 83 C6 14 8A èqÿ0ÿ è- %• fÄŠ
000000350: E7 E8 60 FF BA B7 00 E8 1C 00 BF CF 00 83 C7 19 çè`ÿ0• èL ẽİ fÇ↓
000000360: 8B C1 E8 37 FF 8A C3 E8 21 FF 83 EF 02 89 05 BA <Äè7ÿŠÄè!ÿfi0%•0
000000370: CF 00 E8 01 00 C3 B4 09 CD 21 C3 B8 19 00 8E D8 İ è0 Ä´oÍ!Ä,↓ Ž0
000000380: E8 54 FF E8 A8 FF 32 C0 B4 4C CD 21 00 00 00 00 èTÿè"ÿ2Ä`LÍ!
000000390: 50 43 20 74 79 70 65 3A 20 50 43 0D 0A 24 50 43 PC type: PC)0$PC
0000003A0: 20 74 79 70 65 3A 20 50 43 2F 58 54 0D 0A 24 50 type: PC/XT)0$P
0000003B0: 43 20 74 79 70 65 3A 20 41 54 20 6F 72 20 50 53 C type: AT or PS
0000003C0: 32 20 28 6D 6F 64 65 6C 20 35 30 20 6F 72 20 36 2 (model 50 or 6
0000003D0: 30 29 0D 0A 24 50 43 20 74 79 70 65 3A 20 50 53 0)0$PC type: PS
1Help 2Dump 3Quit 4Text 5 6Edit 7Search 8ANS
```


Загрузка COM модуля в основную память

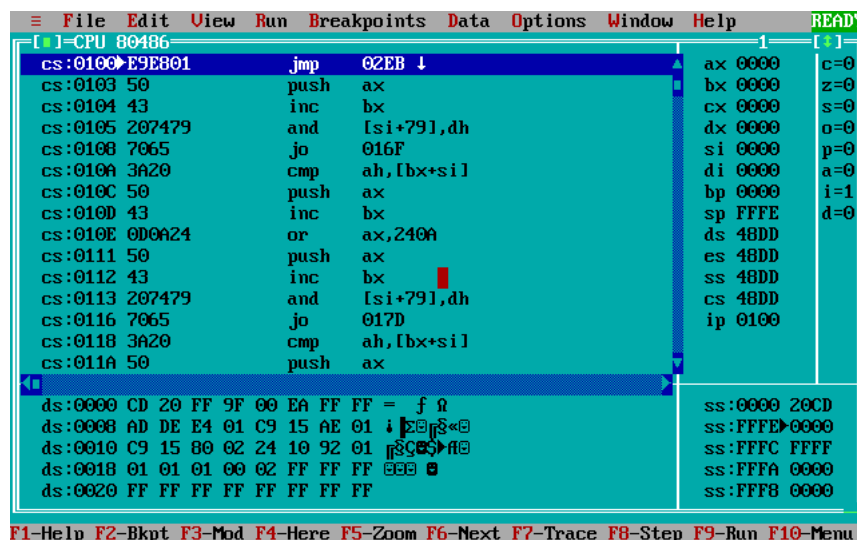
1. Какой формат загрузки COM модуля? С какого адреса располагается код?

Выделяется свободный кусок ОП и его адрес заносится в сегментные регистры. Далее содержится PSP, которое занимает 256 байт. После загружается содержимое COM модуля. Указатель на стек (SP) указывает на конец сегмента PSP, в стек записывается адрес возврата (PSP = 0000h), IP = 100h.

2. Что располагается с адреса 0? - PSP.

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

Все сегментные регистры имеют значение равное 48DD (в данном случае), это и есть адрес сегмента в который была загружена программа.



The screenshot shows a debugger window titled "CPU 80486". The main pane displays assembly code with addresses and instructions. The right pane shows the state of registers and segment registers.

| Address | Instruction |
|---------|------------------------|
| cs:0100 | jmp 02EB ↓ |
| cs:0103 | 50 push ax |
| cs:0104 | 43 inc bx |
| cs:0105 | 207479 and [si+791,dh] |
| cs:0108 | 7065 jo 016F |
| cs:010A | 3A20 cmp ah,[bx+si] |
| cs:010C | 50 push ax |
| cs:010D | 43 inc bx |
| cs:010E | 0D0A24 or ax,240A |
| cs:0111 | 50 push ax |
| cs:0112 | 43 inc bx |
| cs:0113 | 207479 and [si+791,dh] |
| cs:0116 | 7065 jo 017D |
| cs:0118 | 3A20 cmp ah,[bx+si] |
| cs:011A | 50 push ax |

| Register | Value |
|----------|-------|
| ax | 0000 |
| bx | 0000 |
| cx | 0000 |
| dx | 0000 |
| si | 0000 |
| di | 0000 |
| bp | 0000 |
| sp | FFFE |
| ds | 48DD |
| es | 48DD |
| ss | 48DD |
| cs | 48DD |
| ip | 0100 |

| Segment | Address | Value |
|---------|---------|-------------------------------------|
| ds | 0000 | CD 20 FF 9F 00 EA FF FF = f 0 |
| ds | 0008 | AD DE E4 01 C9 15 AE 01 i 20 p S 0 |
| ds | 0010 | C9 15 80 02 24 10 92 01 p S 00 ff 0 |
| ds | 0018 | 01 01 01 00 02 FF FF FF 000 0 |
| ds | 0020 | FF FF FF FF FF FF FF FF |
| ss | 0000 | 20CD |
| ss | FFFE | 0000 |
| ss | FFFC | FFFF |
| ss | FFFA | 0000 |
| ss | FFF8 | 0000 |

At the bottom, there is a status bar with function key shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, F10-Menu.

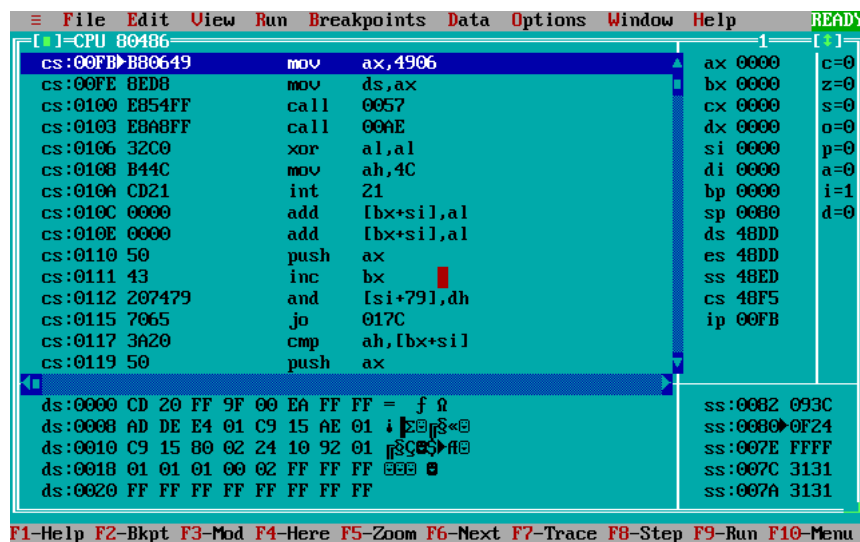
4. Как определяется стек? Какую область памяти он занимает? Какие адреса?

Стек в COM модуле формируется автоматически и находится в том же сегменте, что и остальные. SS указывает на начало PSP, SP указывает на конец стека (FFFEh).

Загрузка «хорошего» EXE модуля в основную память

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

EXE модуль загружается с адреса PSP:0100h. В процессе загрузки считывается информация заголовка (PSP) в начале файла и выполняется перемещение адресов сегментов (т.е. DS и ES указывают на начало сегмента PSP = 48DD), SS указывает на начало сегмента стека, CS на начало сегмента кода. В IP загружается смещение точки входа в программу, которая берется из метки после директивы END.



The screenshot shows a debugger window with the following content:

| Address | Disassembly | Comment | Register/Value |
|----------------|----------------|---------|----------------|
| cs:00FB B80649 | mov ax,4906 | | ax 0000 |
| cs:00FE 8ED8 | mov ds,ax | | bx 0000 |
| cs:0100 E854FF | call 0057 | | cx 0000 |
| cs:0103 E8A8FF | call 00AE | | dx 0000 |
| cs:0106 32C0 | xor al,al | | si 0000 |
| cs:0108 B44C | mov ah,4C | | di 0000 |
| cs:010A CD21 | int 21 | | bp 0000 |
| cs:010C 0000 | add [bx+si],al | | sp 0000 |
| cs:010E 0000 | add [bx+si],al | | ds 48DD |
| cs:0110 50 | push ax | | es 48DD |
| cs:0111 43 | inc bx | | ss 48ED |
| cs:0112 207479 | and [si+79],dh | | cs 48F5 |
| cs:0115 7065 | jo 017C | | ip 00FB |
| cs:0117 3A20 | cmp ah,[bx+si] | | |
| cs:0119 50 | push ax | | |

| Segment | Address | Value |
|---------|---------|---|
| ds | 0000 | CD 20 FF 9F 00 EA FF FF = f 2 |
| ds | 0008 | AD DE E4 01 C9 15 AE 01 i 2 3 4 5 6 7 8 9 A B C D E F |
| ds | 0010 | C9 15 80 02 24 10 92 01 i 2 3 4 5 6 7 8 9 A B C D E F |
| ds | 0018 | 01 01 01 00 02 FF FF FF 00 00 00 00 00 00 00 00 |
| ds | 0020 | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF |

| Segment | Address | Value |
|---------|---------|-------|
| ss | 0002 | 093C |
| ss | 0008 | 0F24 |
| ss | 007E | FFFF |
| ss | 007C | 3131 |
| ss | 007A | 3131 |

At the bottom, there is a status bar with keyboard shortcuts: F1-Help, F2-Bkpt, F3-Mod, F4-Here, F5-Zoom, F6-Next, F7-Trace, F8-Step, F9-Run, F10-Menu.

2. На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало сегмента PSP.

3. Как определяется стек?

Стек можно определить директивой segment, написав:

segment_name segment stack

<*db/dw/dd/dq/dt> <count_of_*> (заполнитель dup (чем_заполнить))

segment_name ends

Также, можно воспользоваться директивой .stack <size>

4. Как определяется точка входа?

Точка входа определяется директивой END.