

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

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

Студент гр. 8383

Бабенко Н.С.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

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

Сведения о функциях и структурах данных управляющей программы.

- 1) PRINT – процедура печати, вызывает функцию 09h. Выводит содержимое сегмента DS со смещением из регистра DX.
- 2) TETRTOHEX – процедура вывода байта AL в 16-ричной системе счисления.
- 3) BYTETOHEX – процедура перевода байта в регистре AL в два символа шестнадцатеричного числа в AX.
- 4) WRDTOHEX – процедура перевода в 16-ричную систему счисления 16-ти разрядного числа в регистре AX. Из регистра DI берется адрес последнего символа.
- 5) BYTETODEC – процедура перевода в 10-тичную систему счисления байта в регистре AL. Из регистра SI берется адрес поля младшей цифры.

Последовательность действий, выполняемых утилитой.

Программа читает содержимое предпоследнего байта ROM BIOS и по таблице, сравнивая коды, определяет тип РС. Далее выводит строку с названием модели. Если код не совпал ни с одним значением, то двоичный код переводится в символьную строку, содержащую запись шестнадцатеричного числа, строка выводится. Затем определяется версия системы, используя регистры AL, AH, по которым формируется текстовая строка, формируются строки с серийным номером OEM и серийным номером пользователя. Полученные строки выводятся на экран.

Ход работы.

1) Написан текст исходного .COM модуля. При линковке получаем «плохой» .EXE модуль и наблюдаем предупреждение линковщика о том, что не объявлен сегмент стека. Получим из него «хороший» .COM модуль при помощи EXE2BIN и запустим.

```
Run File [LAB1_COM.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>

C:\>LAB1_COM.EXE

0>EPC type:

0>EPC type:          5 0

0>EPC type:          0

0>EPC type:          000000

0>EPC type:
```

Рисунок 1 – Результат выполнения плохого .EXE модуля

```
C:\>MASM\EXE2BIN.EXE LAB1_COM.EXE LAB1_COM.COM

C:\>LAB1_COM.COM
PC type: AT
Version MS-DOS: 05.00
DEM: 0
User serial number: 000000
```

Рисунок 2 – Результат выполнения хорошего .COM модуля

2) Написан текст исходного .EXE модуля, который выполняет те же функции, что и модуль в шаге 1. На этот раз линковщик не выдает предупреждений.

```

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

49948 + 455265 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>MASM\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]:
Libraries [.LIB]:

C:\>LAB1_EXE.EXE
PC type: AT
Version MS-DOS: 05.00
OEM: 0
User serial number: 0000000

```

Рисунок 3 – Результат выполнения хорошего .EXE модуля

3) Путём сравнения исходных текстов для .COM и .EXE модулей были получены ответы на контрольные вопросы:

Отличия исходных текстов .com и .exe программ

- 1) COM-программа должна содержать ровно один сегмент.
- 2) EXE-программа должна содержать хотя бы один сегмент, может содержать и больше одного.
- 3) В тексте COM-программы должны быть директивы ASSUME (указывает к какому сегментному регистру привязан сегмент), ORG 100h (устанавливает значение IP в 100h - смещение от начала PSP).
- 4) В COM-программах нельзя использовать команды вида mov <регистр> <сегмент>, так как в такой программе всего один сегмент. Из-за того, что в .COM файлах отсутствует заголовок, в отличие от .EXE файлов, линковщик не может связать смещение до сегмента от адреса загрузки программы и возникает ошибка линковки.


```

C:\Users\Acer\Downloads\LabOS\LAB1_COM.EXE |h|1252|1242|Co1 0|0%|19:56
000000000: 4D 5A DA 00 03 00 00 00 20 00 00 00 FF FF 00 00 MZU ♥ yy
000000010: 00 00 C0 DE 00 01 00 00 1E 00 00 00 01 00 00 00 Ab ☺ ▲ ☹
000000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

```

0000002D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000300: E9 10 01 50 43 20 74 79 70 65 3A 20 24 0D 0A 56 é»PC type: $N
000000310: 65 72 73 69 6F 6E 20 4D 53 2D 44 4F 53 3A 20 30 version MS-DOS: 0
000000320: 2A 2E 30 2A 20 20 20 24 0D 0A 4F 45 4D 3A 20 20 *.0* $N
000000330: 20 20 24 0D 0A 55 73 65 72 73 65 72 69 61 6C $N
000000340: 20 6E 75 6D 62 65 72 3A 20 20 20 20 20 20 24 number: $
000000350: 50 43 24 50 43 2F 58 54 24 41 54 24 50 43 20 43 PC$PC/XT$AT$PC C
000000360: 6F 6E 76 65 72 74 69 62 6C 65 24 50 53 32 20 6D onvertible$PS2 m
000000370: 6F 64 65 6C 20 33 30 24 50 53 32 20 6D 6F 64 65 ode1 30$PS2 mode
000000380: 6C 20 35 30 20 6F 72 20 36 30 24 50 53 32 20 6D l 50 or 60$PS2 m
000000390: 6F 64 65 6C 20 38 30 24 50 43 6A 72 24 4E 6F 74 ode1 80$PCjr$Not
0000003A0: 20 66 6F 75 6E 64 2C 20 79 6F 75 72 20 74 79 70 found, your typ
0000003B0: 65 3A 20 20 24 50 B4 09 CD 21 58 C3 24 0F 3C 09 e: $P
0000003C0: 76 02 04 07 04 30 C3 51 8A E0 E8 EF FF 86 C4 B1 ve♦♦0AQ$aeiÿtA±
0000003D0: 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 E9 FF 88 25 ♦OëæÿYAS$üëÿ%
0000003E0: 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F 88 05 5B C3 O♦o$Cëÿ%O♦[A
0000003F0: 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 88 14 QR2ä30'Q ÷ñE0.ñ
000000400: 4E 33 D2 3D 0A 73 F1 3C 00 74 04 0C 30 88 04 N30=Q sn< t♦90 ♦
000000410: 5A 59 C3 B8 00 F0 8E C0 26 A0 FE FF BA 03 01 E8 ZYÄ ðŽA& þÿ♥è
000000420: 93 FF 3C FF 74 23 3C FE 74 25 3C FB 74 21 3C FC "ÿ<ÿt#<pt%<ÿt!<ü
000000430: 74 29 3C FC 74 1F 3C FA 74 27 3C F8 74 29 3C FD t)<ÿt<ÿt!<ot)<ÿ
000000440: 74 2B 3C F9 74 2D EB 31 90 BA 50 01 EB 3C 90 BA t+<ÿt-ë1P°Pëë<P°
000000450: 53 01 EB 36 90 BA 78 01 EB 30 90 BA 59 01 EB 2A Sëë6P°xëë0P°Yëë*
000000460: 90 BA 68 01 EB 24 90 BA 8B 01 EB 1E 90 BA 98 01 P°këë$P°<ëë▲P°°°°
000000470: EB 18 90 BA 5C 01 EB 12 90 BF 9D 01 83 C7 18 E8 ëtP°\ëëtP°ofCtë
000000480: 45 FF 89 05 BA 9D 01 EB 01 90 E8 28 FF B4 30 CD Eÿ%♦°PëëPë(ÿ OI
000000490: 21 BE 0D 01 83 C6 13 E8 56 FF 83 C6 04 8A C4 E8 l%hofA!!ëÿÿfA♦SÄë
0000004A0: 4E FF BA 0D 01 E8 0D FF BE 28 01 83 C6 08 8A C7 NY°Jëëÿÿ%ofA♦SC
0000004B0: E8 3D FF BA 28 01 E8 FC FE BF 33 01 83 C7 1B 8B è=ÿ°(ëëüþ;3ëfC-<
0000004C0: C1 E8 14 FF 8A C3 E8 FE FE 83 EF 02 89 05 BA 33 AeÿÿSÄëþÿfio%♦°3
0000004D0: 01 E8 E1 FE 32 C0 B4 4C CD 21 eëap2A LI!

```

Рисунок 5 – Содержимое файла LAB1_COM.EXE («плохой» EXE)

C:\Users\Acer\Downloads\LabOS\LAB1_EXE.EXE										h	1252	1033	Col 0	0%	19:58								
0000000000:	4D	5A	09	00	03	00	01	00		20	00	00	00	FF	FF	00	00	MZ	o	♥	⊕		yy
0000000010:	18	00	11	A8	5E	00	0E	00		1E	00	00	00	01	00	63	00	↑	◀	^	♫	▲	⊕ c
0000000020:	0E	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00	♫					
0000000030:	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						
0000000050:	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						
0000000070:	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						
0000000090:	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						
00000000B0:	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						
00000000D0:	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						
00000000F0:	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						
0000000110:	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						
0000000130:	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						
0000000150:	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						
0000000170:	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						
0000000190:	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						
00000001B0:	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						
00000001D0:	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						
00000001F0:	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						
0000000210:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00						
0000000220:	50	43	20	74	79	70	65	3A		20	24	0D	0A	56	65	72	73	PC type: \$Jov ersion MS-DOS: 0*.0 * \$JBOEM: \$ JUser serial nu mber: \$PC\$ PC/XT\$AT\$PC Conv ertible\$PS2 mode l 30\$PS2 model 5 0 or 60\$PS2 mode l 80\$PCjr\$Not fo und, your type: \$					
0000000230:	69	6F	6E	20	4D	53	2D	44		4F	53	3A	20	30	2A	2E	30						
0000000240:	2A	20	20	20	24	0D	0A	4F		45	4D	3A	20	20	20	20	24						
0000000250:	0D	0A	55	73	65	72	20	73		65	72	69	61	6C	20	6E	75						
0000000260:	6D	62	65	72	3A	20	20	20		20	20	20	20	24	50	43	24						
0000000270:	50	43	2F	58	54	24	41	54		24	50	43	20	43	6F	6E	76						
0000000280:	65	72	74	69	62	6C	65	24		50	53	32	20	6D	6F	64	65						
0000000290:	6C	20	33	30	24	50	53	32		20	6D	6F	64	65	6C	20	35						
00000002A0:	30	20	6F	72	20	36	30	24		50	53	32	20	6D	6F	64	65						
00000002B0:	6C	20	38	30	24	50	43	6A		72	24	4E	6F	74	20	66	6F						
00000002C0:	75	6E	64	2C	20	79	6F	75		72	20	74	79	70	65	3A	20						
00000002D0:	20	24	00	00	00	00	00	00		00	00	00	00	00	00	00	00						
00000002E0:	50	B4	09	CD	21	58	C3	24		0F	3C	09	76	02	04	07	04	P oI!X\$*<ove♦♦♦ 0AQ\$aeiŷtA±♦Oèèæ ŷYAS\$üèèŷ%o~♦oS Çèŷ'‰o~†[AQ2ä3 0! ÷ñèE0¶IN30=Ç sn< t♦90♦ZYA+ AP.⊙ Žo ðŽA& þŷ ° èšŷ<ŷt#<pt%û t!<üt)<üt#<üt'<ø t)<ŷt+<üt-è1□°M è<□°P è6□°u è0□° V è*□°h èš□°~ è▲ □°• è†□°Y è†□°š fÇtè<ŷ%†°š èø□èŷ ŷ 0I!%□ fA!!èMŷfA ♦\$AèEŷ°□ è♦ŷ% f A\$Çè4ŷ°% èøþ;0 fÇ-<Aèŷ\$Aèøþfŷè %†°ø èøþÈ					
00000002F0:	30	C3	51	8A	E0	E8	EF	FF		86	C4	B1	04	D2	E8	E8	E6						
0000000300:	FF	59	C3	53	8A	FC	E8	E9		FF	88	25	4F	88	05	4F	8A						
0000000310:	C7	E8	DE	FF	88	25	4F	88		05	5B	C3	51	52	32	E4	33						
0000000320:	D2	B9	0A	00	F7	F1	80	CA		30	88	14	4E	33	D2	3D	0A						
0000000330:	00	73	F1	3C	00	74	04	0C		30	88	04	5A	59	C3	1E	2B						
0000000340:	C0	50	B8	02	00	8E	D8	B8		00	F0	8E	C0	26	A0	FE	FF						
0000000350:	BA	00	00	E8	8A	FF	3C	FF		74	23	3C	FE	74	25	3C	FB						
0000000360:	74	21	3C	FC	74	29	3C	FC		74	1F	3C	FA	74	27	3C	F8						
0000000370:	74	29	3C	FD	74	2B	3C	F9		74	2D	EB	31	90	BA	4D	00						
0000000380:	EB	3C	90	BA	50	00	EB	36		90	BA	75	00	EB	30	90	BA						
0000000390:	56	00	EB	2A	90	BA	68	00		EB	24	90	BA	88	00	EB	1E						
00000003A0:	90	BA	95	00	EB	18	90	BA		59	00	EB	12	90	BF	9A	00						
00000003B0:	83	C7	18	E8	3C	FF	89	05		BA	9A	00	EB	01	90	E8	1F						
00000003C0:	FF	B4	30	CD	21	BE	0A	00		83	C6	13	E8	4D	FF	83	C6						
00000003D0:	04	8A	C4	E8	45	FF	BA	0A		00	E8	04	FF	BE	25	00	83						
00000003E0:	C6	08	8A	C7	E8	34	FF	BA		25	00	E8	F3	FE	BF	30	00						
00000003F0:	83	C7	1B	8B	C1	E8	0B	FF		8A	C3	E8	F5	FE	83	EF	02						
0000000400:	89	05	BA	30	00	E8	D8	FE		CB													

Рисунок 6 – Содержимое файла LAB1_EXE.EXE («хороший» EXE)

5) Был открыт отладчик TD.EXE и загружен .COM. Получены ответы на вопросы:

Загрузка COM модуля в основную память

- 1) Когда управление передается программе типа .COM, все сегментные регистры указывают на префикс. IP = 0100H. Код располагается с адреса 100H.
- 2) С адреса 0H располагается PSP.
- 3) Сегментные регистры имеют одинаковые значения 48DD и указывают на начало PSP.
- 4) Стек занимает 64кб (все доступное для него место). Регистр SS устанавливается на начало PSP, регистр SP на конец сегмента PSP и указывает на адрес FFFEH.

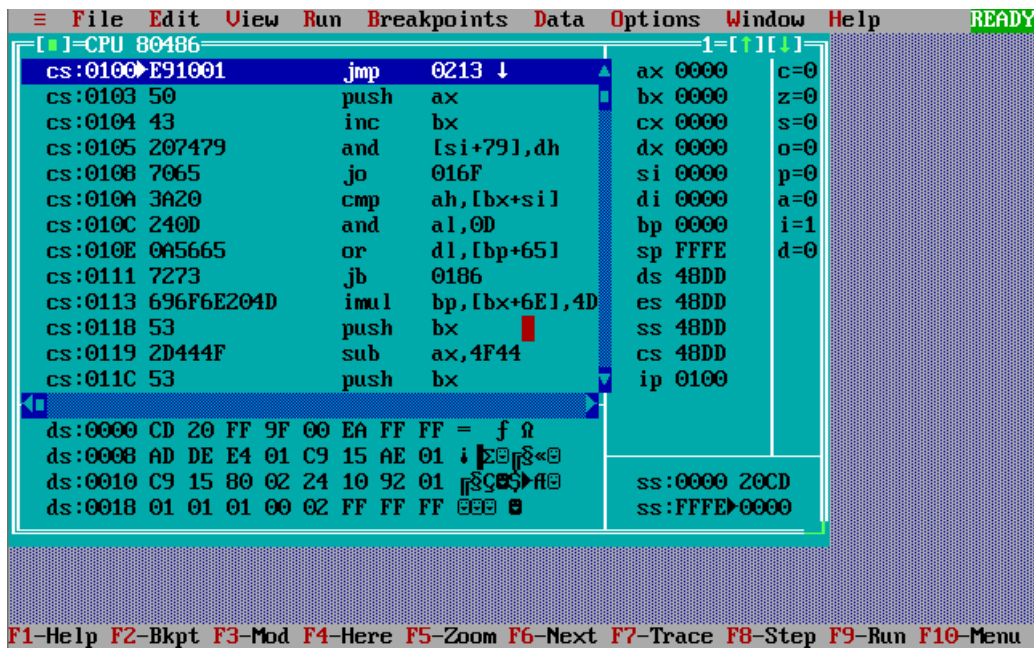


Рисунок 7 – Окно отладчика TD для COM файла

6) Был открыт отладчик TD.EXE и загружен хороший .EXE. Получены ответы на вопросы:

Загрузка «хорошего» EXE модуля в основную память

- 1) Сегментные регистры ES и DS указывают на PSP = 48DDH. CS указывает на начало сегмента кода, значение 48FBH, SP указывает на начало стека, значение 0018H.
- 2) Регистры ES и DS указывают на начало PSP.
- 3) Сегментом стека и выделенным размером под него.
- 4) Директива END сопоставляет точке входа метку, написанную после директивы.

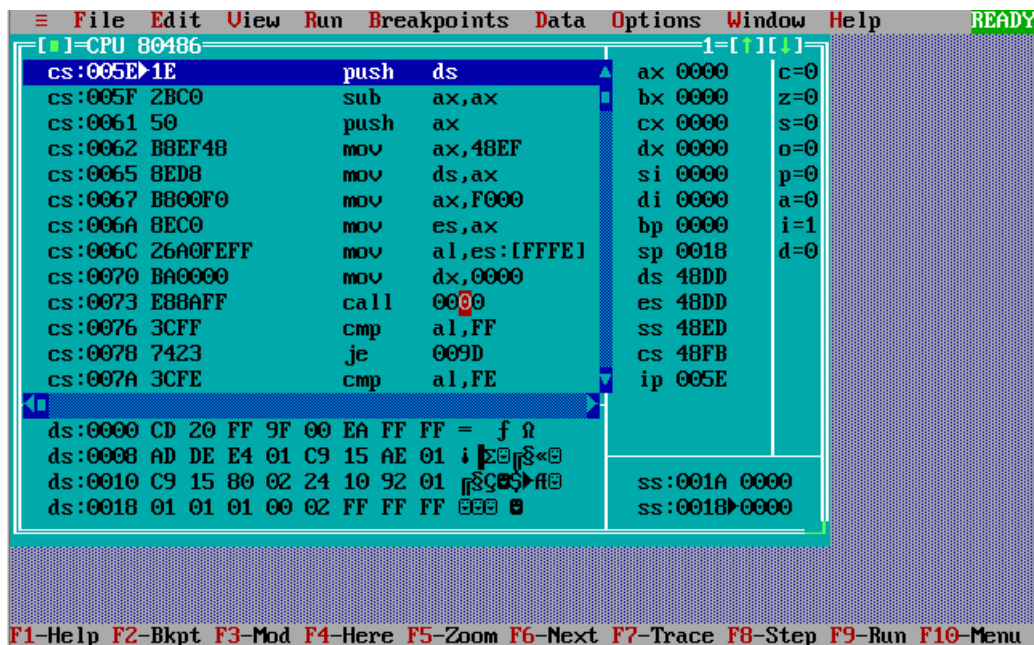


Рисунок 8 – Окно отладчика TD для хорошего EXE файла

Выводы.

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

ПРИЛОЖЕНИЕ А

lab1_exe.asm

```
STACK SEGMENT      STACK

                DW 12 DUP(?)

STACK           ENDS

DATA            SEGMENT

STR_TYPE       DB "PC type: $"
STR_VERSION    DB 13, 10, "Version MS-DOS: 0*.0*  $"
STR_OEM        DB 13, 10, "OEM:  $"
STR_NUMBER     DB 13, 10, "User serial number:  $"
STR_PC         DB "PC$"
STR_PCXT       DB "PC/XT$"
STR_AT         DB "AT$"
STR_PCCon      DB "PC Convertible$"
STR_PS2m30     DB "PS2 model 30$"
STR_PS2m50m60  DB "PS2 model 50 or 60$"
STR_PS2m80     DB "PS2 model 80$"
STR_PCjr       DB "PCjr$"
STR_ERROR      DB "Not found, your type:  $"

DATA            ENDS

CODE            SEGMENT

ASSUME         CS:CODE, DS:DATA, SS:STACK

PRINT          PROC  NEAR

                PUSH  AX
                MOV   AH, 09H
                INT   21H
                POP   AX
                RET

PRINT          ENDP

TETRTOHEX      PROC  NEAR

                AND   AL, 0FH
                CMP   AL, 09H
                JBE   NEXT
                ADD   AL, 07H

                NEXT:
                    ADD   AL, 30H
                    RET

TETRTOHEX      ENDP

BYTETOHEX      PROC  NEAR

                PUSH  CX
                MOV   AH, AL
```

```

        CALL  TETRTOHEX
        XCHG  AL, AH
        MOV   CL, 4H
        SHR   AL, CL
        CALL  TETRTOHEX
        POP   CX
        RET

BYTETOHEX  ENDP

WRDTOHEX   PROC   NEAR

        PUSH  BX
        MOV   BH, AH
        CALL  BYTETOHEX
        MOV   [DI], AH
        DEC   DI
        MOV   [DI], AL
        DEC   DI
        MOV   AL, BH
        CALL  BYTETOHEX
        MOV   [DI], AH
        DEC   DI
        MOV   [DI], AL
        POP   BX
        RET

WRDTOHEX   ENDP

BYTETODEC  PROC   NEAR

        PUSH  CX
        PUSH  DX
        XOR   AH, AH
        XOR   DX, DX
        MOV   CX, 0AH

        LOOP_BD:
                DIV   CX
                OR    DL, 30H
                MOV   [SI], DL
                DEC   SI
                XOR   DX, DX
                CMP   AX, 0AH
                JAE   LOOP_BD
                CMP   AL, 00H
                JE    END_L
                OR    AL, 30H
                MOV   [SI], AL

        END_L:
                POP   DX
                POP   CX
                RET

BYTETODEC  ENDP

MAIN       PROC   FAR

        PUSH  DS

```

```

SUB    AX, AX
PUSH   AX
MOV     AX, DATA
MOV     DS, AX

MOV     AX, 0F000H
MOV     ES, AX
MOV     AL, ES:[0FFFFEH]

MOV     DX, OFFSET STR_TYPE
CALL    PRINT

CMP     AL, 0FFH
JZ      PC

CMP     AL, 0FEH
JZ      PCXT

CMP     AL, 0FBH
JZ      PCXT

CMP     AL, 0FCH
JZ      AT

CMP     AL, 0FCH
JZ      PC2m50or60

CMP     AL, 0FAH
JZ      PC2m30

CMP     AL, 0F8H
JZ      PC2m80

CMP     AL, 0FDH
JZ      PCjr

CMP     AL, 0F9H
JZ      PCCon

JMP     ELS

PC:
    MOV     DX, OFFSET STR_PC
    JMP     PRINT_THIS

PCXT:
    MOV     DX, OFFSET STR_PCXT
    JMP     PRINT_THIS

PC2m50or60:
    MOV     DX, OFFSET STR_PS2m50m60
    JMP     PRINT_THIS

AT:
    MOV     DX, OFFSET STR_AT
    JMP     PRINT_THIS

PC2m30:
    MOV     DX, OFFSET STR_PS2m30
    JMP     PRINT_THIS

PC2m80:
    MOV     DX, OFFSET STR_PS2m80

```

```

        JMP     PRINT_THIS

PCjr:
        MOV     DX, OFFSET STR_PCjr
        JMP     PRINT_THIS

PCCon:
        MOV     DX, OFFSET STR_PCCon
        JMP     PRINT_THIS

ELS:
        MOV     DI, OFFSET STR_ERROR
        ADD     DI, 18H
        CALL    BYTETOHEX
        MOV     [DI], AX
        MOV     DX, OFFSET STR_ERROR
        JMP     PRINT_THIS

PRINT_THIS:
        CALL    PRINT

        MOV     AH, 30H
        INT     21H

        MOV     SI, OFFSET STR_VERSION
        ADD     SI, 13H
        CALL    BYTETODEC
        ADD     SI, 4H
        MOV     AL, AH
        CALL    BYTETODEC
        MOV     DX, OFFSET STR_VERSION
        CALL    PRINT

        MOV     SI, OFFSET STR_OEM
        ADD     SI, 8H
        MOV     AL, BH
        CALL    BYTETODEC
        MOV     DX, OFFSET STR_OEM
        CALL    PRINT

        MOV     DI, OFFSET STR_NUMBER
        ADD     DI, 1BH
        MOV     AX, CX
        CALL    WRDTOHEX
        MOV     AL, BL
        CALL    BYTETOHEX
        SUB     DI, 2H
        MOV     [DI], AX
        MOV     DX, OFFSET STR_NUMBER
        CALL    PRINT

RET

MAIN    ENDP
CODE    ENDS
END     MAIN

```

lab1_com.asm

```
LAB1          SEGMENT

ASSUME        CS:LAB1, DS:LAB1, ES:NOTHING, SS:NOTHING

ORG           100H

START:        JMP     START_PROG

STR_TYPE      DB "PC type: $"
STR_VERSION   DB 13, 10, "Version MS-DOS: 0*.0*  $"
STR_OEM       DB 13, 10, "OEM:  $"
STR_NUMBER    DB 13, 10, "User serial number:  $"
STR_PC        DB "PC$"
STR_PCXT      DB "PC/XT$"
STR_AT        DB "AT$"
STR_PCCon     DB "PC Convertible$"
STR_PS2m30    DB "PS2 model 30$"
STR_PS2m50m60 DB "PS2 model 50 or 60$"
STR_PS2m80    DB "PS2 model 80$"
STR_PCjr      DB "PCjr$"
STR_ERROR     DB "Not found, your type:  $"

PRINT         PROC    NEAR

                PUSH    AX
                MOV     AH, 09H
                INT     21H
                POP     AX
                RET

PRINT         ENDP

TETRTOHEX     PROC    NEAR

                AND     AL, 0FH
                CMP     AL, 09H
                JBE     NEXT
                ADD     AL, 07H

                NEXT:
                ADD     AL, 30H
                RET

TETRTOHEX     ENDP

BYTETOHEX     PROC    NEAR

                PUSH    CX
                MOV     AH, AL
                CALL    TETRTOHEX
                XCHG    AL, AH
                MOV     CL, 4H
                SHR     AL, CL
                CALL    TETRTOHEX
                POP     CX
                RET

BYTETOHEX     ENDP
```

```

WRDTOHEX    PROC    NEAR

    PUSH    BX
    MOV     BH, AH
    CALL    BYTETOHEX
    MOV     [DI], AH
    DEC     DI
    MOV     [DI], AL
    DEC     DI
    MOV     AL, BH
    CALL    BYTETOHEX
    MOV     [DI], AH
    DEC     DI
    MOV     [DI], AL
    POP     BX
    RET

WRDTOHEX    ENDP

BYTETODEC   PROC    NEAR

    PUSH    CX
    PUSH    DX
    XOR     AH, AH
    XOR     DX, DX
    MOV     CX, 0AH

    LOOP_BD:
        DIV     CX
        OR      DL, 30H
        MOV     [SI], DL
        DEC     SI
        XOR     DX, DX
        CMP     AX, 0AH
        JAE     LOOP_BD
        CMP     AL, 00H
        JE      END_L
        OR      AL, 30H
        MOV     [SI], AL

    END_L:
        POP     DX
        POP     CX
        RET

BYTETODEC   ENDP

START_PROG:

    MOV     AX, 0F000H
    MOV     ES, AX
    MOV     AL, ES:[0FFFEH]

    MOV     DX, OFFSET STR_TYPE
    CALL    PRINT

    CMP     AL, 0FFH
    JZ      PC

    CMP     AL, 0FEH

```



```

        JZ      PCXT

        CMP AL, 0FBH
        JZ      PCXT

        CMP AL, 0FCH
        JZ      AT

        CMP AL, 0FCH
        JZ      PC2m50or60

        CMP AL, 0FAH
        JZ      PC2m30

        CMP AL, 0F8H
        JZ      PC2m80

        CMP AL, 0FDH
        JZ      PCjr

        CMP AL, 0F9H
        JZ      PCCon

        JMP     ELS

PC:
        MOV     DX, OFFSET STR_PC
        JMP     PRINT_THIS

PCXT:
        MOV     DX, OFFSET STR_PCXT
        JMP     PRINT_THIS

PC2m50or60:
        MOV     DX, OFFSET STR_PS2m50m60
        JMP     PRINT_THIS

AT:
        MOV     DX, OFFSET STR_AT
        JMP     PRINT_THIS

PC2m30:
        MOV     DX, OFFSET STR_PS2m30
        JMP     PRINT_THIS

PC2m80:
        MOV     DX, OFFSET STR_PS2m80
        JMP     PRINT_THIS

PCjr:
        MOV     DX, OFFSET STR_PCjr
        JMP     PRINT_THIS

PCCon:
        MOV     DX, OFFSET STR_PCCon
        JMP     PRINT_THIS

ELS:
        MOV     DI, OFFSET STR_ERROR
        ADD     DI, 18H
        CALL    BYTETOHEX
        MOV     [DI], AX
        MOV     DX, OFFSET STR_ERROR

```

```

        JMP     PRINT_THIS

PRINT_THIS:
        CALL    PRINT

        MOV     AH, 30H
        INT     21H

        MOV     SI, OFFSET STR_VERSION
        ADD     SI, 13H
        CALL    BYTETODEC
        ADD     SI, 4H
        MOV     AL, AH
        CALL    BYTETODEC
        MOV     DX, OFFSET STR_VERSION
        CALL    PRINT

        MOV     SI, OFFSET STR_OEM
        ADD     SI, 8H
        MOV     AL, BH
        CALL    BYTETODEC
        MOV     DX, OFFSET STR_OEM
        CALL    PRINT

        MOV     DI, OFFSET STR_NUMBER
        ADD     DI, 1BH
        MOV     AX, CX
        CALL    WRDTOHEX
        MOV     AL, BL
        CALL    BYTETOHEX
        SUB     DI, 2H
        MOV     [DI], AX
        MOV     DX, OFFSET STR_NUMBER
        CALL    PRINT

        XOR     AL, AL
        MOV     AH, 4CH
        INT     21H

LAB1    ENDS
END      START

```