

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**  
**по практической работе № 2**  
**по дисциплине «Операционные системы»**  
**Тема: Исследование структур загрузочных модулей**

Студент гр. 9382

Демин В.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

### **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.**

Шаг 1. Напишите текст исходного .COM модуля, который определяет тип РС и версию системы. Это довольно простая задача и для тех, кто уже имеет опыт программирования на ассемблере, это будет небольшой разминкой. Для тех, кто раньше не сталкивался с программированием на ассемблере, это неплохая задача для первого опыта.

За основу возьмите шаблон, приведенный в разделе «Основные сведения». Необходимые сведения о том, как извлечь требуемую информацию, представлены в следующем разделе.

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

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. Полученные строки выводятся на экран.

Отладьте полученный исходный модуль.

Результатом выполнения этого шага будет «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM модуля.

Шаг 2. Напишите текст исходного .EXE модуля, который выполняет те же функции, что и модуль в Шаге 1 и постройте и отладьте его. Таким образом, будет получен «хороший» .EXE.

Шаг 3. Сравните исходные тексты для .COM и .EXE модулей. Ответьте на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

Шаг 4. Запустите FAR и откройте (F3/F4) файл загрузочного модуля .COM и файл «плохого» .EXE в шестнадцатеричном виде. Затем откройте (F3/F4) файл загрузочного модуля «хорошего» .EXE и сравните его с предыдущими файлами. Ответьте на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

Шаг 5. Откройте отладчик TD.EXE и загрузите .COM. Ответьте на контрольные вопросы «Загрузка COM модуля в основную память». Представьте в отчете план загрузки модуля .COM в основную память.

Шаг 6. Откройте отладчик TD.EXE и загрузите «хороший» .EXE. Ответьте на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».

Шаг 7. Оформление отчета в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей – в отладчике.

Необходимые сведения для составления программы

Тип IBM PC хранится в байте по адресу 0F000:0FFFEh, в предпоследнем байте ROM BIOS. Соответствие кода и типа в таблице:

PC	FF
PC/XT	FE,FB
AT	FC
PS2 модель 30	FA
PS2 модель 50 или 60	FC
PS2 модель 80	F8
PCjr	FD

Для определения версии MS DOS следует воспользоваться функцией 30H прерывания 21H. Входным параметром является номер функции в AH:

MOV AH,30h

INT 21h

Выходными параметрами являются:

AL - номер основной версии. Если 0, то < 2.0

AH - номер модификации


BH - серийный номер OEM (Original Equipment Manufacturer) BL:CH - 24-битовый серийный номер пользователя.

### **Выполнение работы.**

1. Был написан модуль .com , который определяет тип PC и версию системы. Был использован шаблон из методички, в котором были прописаны функции для перевода бинарного числа в десятичное, шестнадцатеричное число в строку из шестнадцатеричных цифр. Эти функции использовались, чтобы вывести на экран в нужном формате тип PC и версию системы, так как этот результат записывался в регистры. А для определения типа PC, мы взяли готовые названия типов и в зависимости от значения в AL, выводили на экран соответствующие наименования.

2. После чего данная программа была переписана для модуля .exe.

Запуск модуля .COM



```
IBM_PC:AT
MS DOS: 5.0
OEM: 0
User Serial Number: 000000
```

Рис.1 – «Хороший» .COM модуль

Запуск плохого .EXE

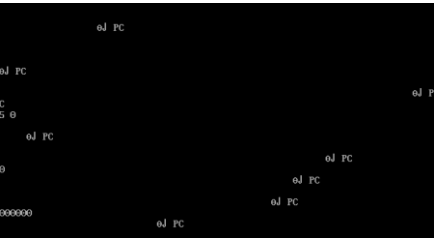


Рис.2 – «Плохой» EXE модуль

Запуск хорошего .EXE

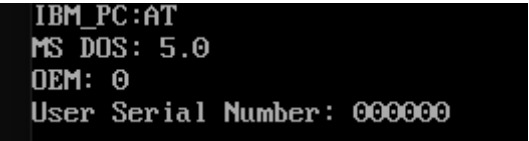


Рис. 3 – «Хороший» EXE модуль

Шаг 4.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
000	4D	5A	D4	00	03	00	00	00	20	00	00	00	FF	FF	00	00	MZ <input type="checkbox"/> <input type="checkbox"/>
010	00	00	4F	37	00	01	00	00	1E	00	00	00	01	00	00	00	<input type="checkbox"/> 07 <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
230	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
260	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
270	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
290	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
2A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
2B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
2C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
2D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
2E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
2F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
300	E9 F5 00 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24	PC \$PC/XT \$
310	41 54 0D 0A 24 50 53 32 20 AC AE A4 A5 AB EC 20	AT \$PS2
320	33 30 0D 0A 24 50 53 32 20 AC AE A4 A5 AB EC 20	30 \$PS2
330	38 30 0D 0A 24 50 43 6A 72 0D 0A 24 50 43 20 43	80 \$PCjr \$PC C
340	6F 6E 76 65 72 74 09 69 62 6C 65 0D 0A 24 49 42	convert ible \$IB
350	4D 5F 50 43 3A 24 20 20 0D 0A 24 20 2E 20 20 20	M_PC:\$ \$ .
360	20 0D 0A 24 4D 53 20 44 4F 53 3A 20 24 3C 32 2E	\$MS DOS: \$<2.
370	30 0D 0A 24 4F 45 4D 3A 20 24 20 0D 0A 24 55 73	0 \$OEM: \$ \$Us
380	65 72 20 53 65 72 69 61 6C 20 4E 75 6D 62 65 72	er Serial Number
390	3A 20 24 20 20 20 20 20 20 0D 0A 24 B4 09 CD 21	: \$ \$!
3A0	C3 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8	\$< v
3B0	EF FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC	
3C0	E8 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25	
3D0	4F 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1	QQR23
3E0	80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74	ON3 s t
3F0	04 0C 30 88 04 5A 59 C3 BA 4E 01 E8 9E FF B8 00	Y
400	F0 8E C0 26 A0 FE FF 3C FE 74 25 3C FB 74 21 3C	
410	FC 74 23 3C FA 74 25 3C F8 74 27 3C FD 74 29 3C	
420	F9 74 2B 3C FF 74 03 EB 2B 90 BA 03 01 EB 33 90	
430	BA 08 01 EB 2D 90 BA 10 01 EB 27 90 BA 15 01 EB	
440	21 90 BA 25 01 EB 1B 90 BA 35 01 EB 15 90 BA 3C	
450	01 EB 0F 90 E8 55 FF BE 56 01 83 C6 01 89 04 BA	
460	56 01 E8 37 FF BA 64 01 E8 31 FF B4 30 CD 21 3C	
470	00 74 17 BE 5B 01 E8 5C FF 83 C6 03 8A C4 E8 54	
480	FF BA 5B 01 E8 15 FF EB 07 90 BA 6D 01 E8 0C FF	

Рис.4 Файл загрузочного модуля «Плохого» .EXE в 16-ном виде

	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	
000	E9 F5 00 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24	PC \$PC/XT \$
010	41 54 0D 0A 24 50 53 32 20 AC AE A4 A5 AB EC 20	AT \$PS2
020	33 30 0D 0A 24 50 53 32 20 AC AE A4 A5 AB EC 20	30 \$PS2
030	38 30 0D 0A 24 50 43 6A 72 0D 0A 24 50 43 20 43	80 \$PCjr \$PC C
040	6F 6E 76 65 72 74 09 69 62 6C 65 0D 0A 24 49 42	convert ible \$IB
050	4D 5F 50 43 3A 24 20 20 0D 0A 24 20 2E 20 20 20	M_PC:\$ \$ .
060	20 0D 0A 24 4D 53 20 44 4F 53 3A 20 24 3C 32 2E	\$MS DOS: \$<2.
070	30 0D 0A 24 4F 45 4D 3A 20 24 20 0D 0A 24 55 73	0 \$OEM: \$ \$Us
080	65 72 20 53 65 72 69 61 6C 20 4E 75 6D 62 65 72	er Serial Number
090	3A 20 24 20 20 20 20 20 20 0D 0A 24 B4 09 CD 21	: \$ \$!
0A0	C3 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8	\$< v
0B0	EF FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC	
0C0	E8 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25	
0D0	4F 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1	QQR23
0E0	80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74	ON3 s t
0F0	04 0C 30 88 04 5A 59 C3 BA 4E 01 E8 9E FF B8 00	Y
100	F0 8E C0 26 A0 FE FF 3C FE 74 25 3C FB 74 21 3C	
110	FC 74 23 3C FA 74 25 3C F8 74 27 3C FD 74 29 3C	
120	F9 74 2B 3C FF 74 03 EB 2B 90 BA 03 01 EB 33 90	
130	BA 08 01 EB 2D 90 BA 10 01 EB 27 90 BA 15 01 EB	
140	21 90 BA 25 01 EB 1B 90 BA 35 01 EB 15 90 BA 3C	
150	01 EB 0F 90 E8 55 FF BE 56 01 83 C6 01 89 04 BA	
160	56 01 E8 37 FF BA 64 01 E8 31 FF B4 30 CD 21 3C	
170	00 74 17 BE 5B 01 E8 5C FF 83 C6 03 8A C4 E8 54	
180	FF BA 5B 01 E8 15 FF EB 07 90 BA 6D 01 E8 0C FF	
190	BA 74 01 E8 06 FF 8A C7 BE 7A 01 E8 37 FF BA 7A	
1A0	01 E8 F8 FE BA 7E 01 E8 F2 FE BE 93 01 8A C3 E8	
1B0	FA FE 89 04 83 C6 02 8A C5 E8 F0 FE 89 04 83 C6	
1C0	02 8A C1 E8 E6 FE 89 04 BA 93 01 E8 CE FE 32 C0	
1D0	B4 4C CD 21	!

Рис.5 Файл загрузочного модуля .COM в 16-ном виде



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
000	4D	5A	72	00	03	00	01	00	20	00	00	00	FF	FF	00	00	M2r □ □ ♦♦
010	80	00	C5	CB	5C	00	13	00	1E	00	00	00	01	00	61	00	♦♦♦ □ □ □ a
020	13	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	□
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
230	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
250	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
260	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
270	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
280	50	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	41	54	0D	PC \$PC/XT \$AT
290	0A	24	50	53	32	20	D0	BC	D0	BE	D0	B4	D0	B5	D0	BB	\$PS2 \$PC Convert
2A0	D1	8C	20	33	30	0D	0A	24	50	53	32	20	D0	BC	D0	BE	ible \$IBM_PC:\$
2B0	D0	B4	D0	B5	D0	BB	D1	8C	20	38	30	0D	0A	24	50	43	\$ . \$MS
2C0	6A	72	0D	0A	24	50	43	20	43	6F	6E	76	65	72	74	09	DOS: \$<2.0 \$OEM
2D0	69	62	6C	65	0D	0A	24	49	42	4D	5F	50	43	3A	24	20	: \$ \$User Seri
2E0	20	0D	0A	24	20	2E	20	20	20	20	0D	0A	24	4D	53	20	al Number: \$
2F0	44	4F	53	3A	20	24	3C	32	2E	30	0D	0A	24	4F	45	4D	\$
300	3A	20	24	20	0D	0A	24	55	73	65	72	20	53	65	72	69	♦♦♦♦♦< v♦♦♦♦♦
310	61	6C	20	4E	75	6D	62	65	72	3A	20	24	20	20	20	20	0000000000000000
320	20	20	0D	0A	24	00	00	00	00	00	00	00	00	00	00	00	0000000000000000
330	B4	09	CD	21	C3	24	0F	3C	09	76	02	04	07	04	30	C3	0000000000000000
340	51	8A	E0	E8	EF	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	0000000000000000
350	C3	53	8A	FC	E8	E9	FF	88	25	4F	88	05	4F	8A	C7	E8	0000000000000000
360	DE	FF	88	25	4F	88	05	5B	C3	51	52	32	E4	33	D2	B9	0000000000000000
370	0A	00	F7	F1	80	CA	30	88	14	4E	33	D2	3D	0A	00	73	0000000000000000
380	F1	3C	00	74	04	0C	30	88	04	5A	59	C3	1E	2B	C0	50	0< tD000000000000
390	B8	08	00	8E	D8	BA	57	00	E8	95	FF	B8	00	F0	8E	C0	0000000000000000
3A0	26	A0	FE	FF	3C	FE	74	25	3C	FB	74	21	3C	FC	74	23	0000000000000000
3B0	3C	FA	74	25	3C	F8	74	27	3C	FD	74	29	3C	F9	74	2B	0000000000000000
3C0	3C	FF	74	03	EB	2B	90	BA	00	00	EB	33	90	BA	05	00	0000000000000000
3D0	EB	2D	90	BA	0D	00	EB	27	90	BA	12	00	EB	21	90	BA	0000000000000000
3E0	28	00	EB	1B	90	BA	3E	00	EB	15	90	BA	45	00	EB	0F	( 0000 0000E 0
3F0	90	E8	4C	FF	BE	5F	00	83	C6	01	89	04	BA	5F	00	E8	000000 000000 0
400	2E	FF	BA	6D	00	E8	28	FF	B4	30	CD	21	3C	00	74	17	0000 00000000< t□
410	BE	64	00	E8	53	FF	83	C6	03	8A	C4	E8	4B	FF	BA	64	0000000000000000
420	00	E8	0C	FF	EB	07	90	BA	76	00	E8	03	FF	BA	7D	00	00000000 000000
430	E8	FD	FE	8A	C7	BE	83	00	E8	2E	FF	BA	83	00	E8	EF	00000000 000000
440	FE	BA	87	00	E8	E9	FE	BE	9C	00	8A	C3	E8	F1	FE	89	00000000 000000
450	04	83	C6	02	8A	C5	E8	E7	FE	89	04	83	C6	02	8A	C1	0000000000000000
460	E8	DD	FE	89	04	BA	9C	00	E8	C5	FE	32	C0	B4	4C	CD	00000000 0000000
470	21	CB															!♦

Рис.6 Файл загрузочного модуля «Хорошего» .EXE в 16-ном виде

## Шаг 5

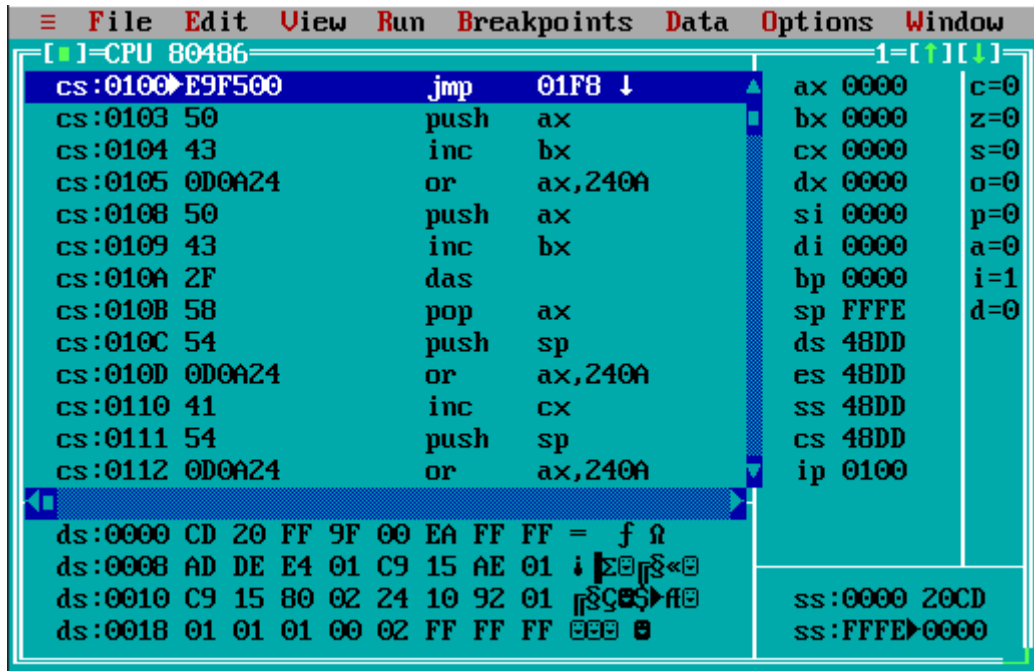


Рис. 7 отладчик TD.EXE для .COM

## Шаг 6

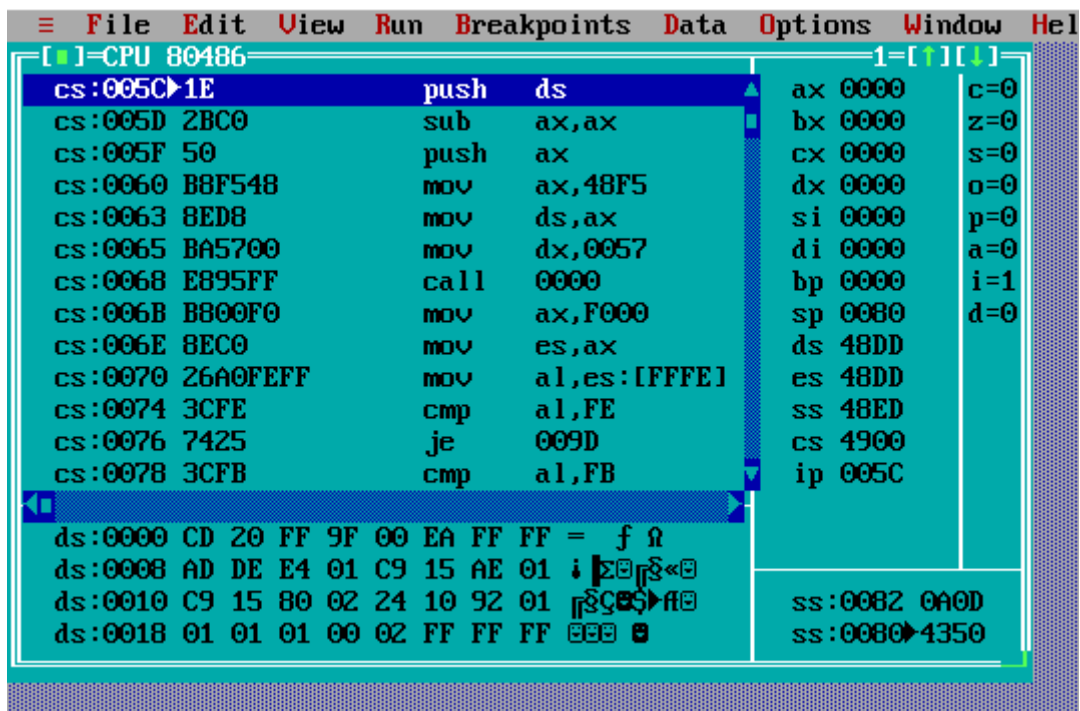


Рис. 8 отладчик TD.EXE для .EXE



### **Выводы.**

Были исследованы модули .COM и .EXE, рассмотрены из различия в исходных текстах и различия готовых модулей. Также были рассмотрены способы загрузки модулей в основную память.

## **ПРИЛОЖЕНИЕ А**

### **ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

#### **Отличия исходных текстов COM и EXE программ:**

1. Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать один сегмент.

2. EXE-программа?

Один сегмент и более.

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

ORG 100h – обязательная директива для COM-программы, так как при загрузке модуля в оперативную память, сначала грузится PSP размером в 100h. И директива нужна, чтобы задать смещение для адресации.

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

Например, нельзя использовать:

mov ax, DATA

Где DATA – адрес сегмента кода

#### **Отличия форматов файлов .COM и .EXE программ:**

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

Файл .COM состоит из одного сегмента, его максимальный размер – 64 кб.

В этот сегмент входит сегмент кода и сегмент данных. Он начинается с адреса 0h.

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

Что располагается с адреса 0?

В «плохом» EXE данные и код содержатся в одном сегменте.

С адреса 0h идёт таблица настроек (Relocation table). Код располагается с адреса 300h.

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

В «хорошем» EXE данные, стек и код разделены по сегментам. Код располагается с адреса 200h в отличие от 300h в «плохом» EXE.

#### **Загрузка COM модуля в основную память:**

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

Код располагается с адреса 100h. При загрузке модуля COM. Сегментные регистры будут указывать на начало PSP.

2. Что располагается с адреса 0?

PSP

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

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

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

Стек генерируется автоматически при загрузке COM-программы. SS – на начало – 0h, SP – на конец – FFFEh, адрес расположен в диапазоне 0h – FFFEh.

#### **Загрузка «хорошего» EXE модуля в основную память:**

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

DS и ES устанавливаются на начало сегмента PSP, SS – на начало сегмента стека, CS – на начало сегмента команд. В IP загружается смещение точки входа в программу, которая берётся из метки после директивы END.

CS = 4900

SS = 48ED

ES = 48DD

DS = 48DD

2. На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало сегмента PSP.

3. Как определяется стек?

При помощи директивы .STACK, также задается размер стека. Регистр SS будет указывать на начало сегмента стека, а SP на конец.

4. Как определяется точка входа?

Точка входа определяется при помощи директивы END.

## ПРИЛОЖЕНИЕ Б

### КОД ПРОГРАММЫ

```
Lab1com.asm:  TESTPC SEGMENT

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

    ORG 100H

    START: JMP BEGIN


STRING_PC db 'PC',0DH,0AH,$'
STRING_PC_XT db 'PC/XT',0DH,0AH,$'
STRING_AT db 'AT',0DH,0AH,$'
STRING_PS2_30 db 'PS2 модель 30',0DH,0AH,$'
STRING_PS2_80 db 'PS2 модель 80',0DH,0AH,$'
STRING_PCjr db 'PCjr',0DH,0AH,$'
STRING_PC_CON db 'PC Convert ible',0DH,0AH,$'
STRING_IMB db 'IBM_PC:','$'
STRING_ANOTHER db ' ',0DH,0AH,$'
STRING_VERSION_DOS db '. ',0DH,0AH,$'
STRING_VERSION_DOS_TEXT db 'MS DOS:','$'
STRING_0 db '<2.0',0DH,0AH,$'
STRING_OEM_TEXT db 'OEM: ', '$'
STRING_OEM db ' ',0DH,0AH,$'
STRING_USER_TEXT db 'User Serial Number: ', '$'
STRING_USER db ' ',0DH,0AH,$'

PRINT PROC near
    mov     AH, 09h        ;Номер функции 09h
    int     21h
    ret
PRINT ENDP
```

TETR\_TO\_HEX PROC near

and AL, 0Fh

cmp AL, 09

jbe NEXT

add AL, 07

NEXT: add AL, 30h

ret

TETR\_TO\_HEX ENDP

;-----

BYTE\_TO\_HEX PROC near

;byte AL translate in two symbols on 16cc numbers in AX

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL, 4

shr AL,CL

call TETR\_TO\_HEX

pop CX

ret

BYTE\_TO\_HEX ENDP

;-----

WRD\_TO\_HEX PROC near

;translate in 16cc a 16 discharge number

;in AL - number, DI - the address of the last symbol

push BX

mov BH,AH

call BYTE\_TO\_HEX

mov [DI],AH



```

    dec DI
    mov [DI],AL
    dec DI
    mov AL,BH
    call BYTE_TO_HEX
    mov [DI],AH
    dec DI
    mov [DI],AL
    pop BX
ret
WRD_TO_HEX ENDP
;-----
BYTE_TO_DEC PROC near
;translate in 10cc, SI - the adress of the field of younger digit
    push CX
    push DX
    xor AH,AH
    xor DX,DX
    mov CX,10
loop_bd: div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae loop_bd
    cmp AL,00h
    je end_1
    or AL,30h

```

```

    mov [SI],AL
end_1: pop DX
    pop CX
ret
BYTE_TO_DEC ENDP
;-----

```

BEGIN:

```

ibm_pc:
    mov DX,offset STRING_IMB
    call PRINT

```

```

MOV AX,0F000H ;указывает ES на ПЗУ
MOV ES,AX ;
MOV AL,ES:[0FFFEH] ;получаем байт

```

```

CMP AL,0FEH
je PC_XT
CMP AL,0FBH
je PC_XT
CMP AL,0FCH
je vAT
CMP AL,0FAH
je PS2_30
CMP AL,0F8H
je PS2_80
CMP AL,0FDH

```

```
je PCjr
CMP AL,0F9H
je PC_CON
CMP AL,0FFH
je PC
```

```
jmp Another
```

PC:

```
mov DX, offset STRING_PC
jmp PRINT_IBM_PC
```

PC\_XT:

```
mov DX, offset STRING_PC_XT
jmp PRINT_IBM_PC
```

vAT:

```
mov DX,offset STRING_AT
jmp PRINT_IBM_PC
```

PS2\_30:

```
mov DX,offset STRING_PS2_30
jmp PRINT_IBM_PC
```

PS2\_80:

```
mov DX,offset STRING_PS2_80
```

```
jmp PRINT_IBM_PC
```

PCjr:

```
mov DX,offset STRING_PCjr
```

```
jmp PRINT_IBM_PC
```

PC\_CON:

```
mov DX,offset STRING_PC_CON
```

```
jmp PRINT_IBM_PC
```

Another:

```
call BYTE_TO_HEX
```

```
mov si, offset STRING_ANOTHER
```

```
add si, 1
```

```
mov [si], ax
```

```
mov dx,offset STRING_ANOTHER
```

PRINT\_IBM\_PC:

```
call PRINT
```

MS\_DOS:

```
mov dx, offset STRING_VERSION_DOS_TEXT
```

```
call PRINT
```

```
MOV AH,30h
```

```
INT 21h
```

```
cmp AL,0
```

```
je NULL_VERSION
```

```
mov si, offset STRING_VERSION_DOS
```

```
call BYTE_TO_DEC
```

```
add si,3
```

```

    mov AL,AH
    call BYTE_TO_DEC
    MOV dx,offset STRING_VERSION_DOS
    call PRINT
    jmp OEM
NULL_VERSION:
    mov dx, offset STRING_0
    call PRINT
OEM:
    mov dx,offset STRING_OEM_TEXT
    call PRINT
    mov AL,BH
    mov si,offset STRING_OEM
    call BYTE_TO_DEC
    mov dx, offset STRING_OEM
    call PRINT
USER_SERIES:
    mov dx,offset STRING_USER_TEXT
    call PRINT
    mov  si, offset STRING_USER
    mov al, bl
    call byte_to_hex
    mov [si], ax
    add si, 2
    mov al, ch
    call byte_to_hex
    mov [si], ax
    add si, 2
    mov al, cl

```

```

        call byte_to_hex
        mov [si], ax
        mov dx, offset STRING_USER
        call PRINT
        xor AL,AL
        mov AH,4Ch
        int 21H
TESTPC  ENDS
END START

```

```

LAB1EXE.ASM: AStack  SEGMENT STACK
                DW 64 DUP(?) ; Žâç®œâĩ 12 á«®ç ¯¬ĩâ
AStack  ENDS

```

```

DATA SEGMENT

```

```

STRING_PC db 'PC',0DH,0AH,$'
STRING_PC_XT db 'PC/XT',0DH,0AH,$'
STRING_AT db 'AT',0DH,0AH,$'
STRING_PS2_30 db 'PS2 модель 30',0DH,0AH,$'
STRING_PS2_80 db 'PS2 модель 80',0DH,0AH,$'
STRING_PCjr db 'PCjr',0DH,0AH,$'
STRING_PC_CON db 'PC Convert ible',0DH,0AH,$'
STRING_IMB db 'IBM_PC:','$'
STRING_ANOTHER db ' ',0DH,0AH,$'
STRING_VERSION_DOS db '. ',0DH,0AH,$'
STRING_VERSION_DOS_TEXT db 'MS DOS:','$'
STRING_0 db '<2.0',0DH,0AH,$'
STRING_OEM_TEXT db 'OEM: ', '$'

```



```

STRING_OEM db ' ',0DH,0AH,'$'
STRING_USER_TEXT db 'User Serial Number: ', '$'
STRING_USER db      '      ',0DH,0AH,'$'

DATA ENDS

CODE      SEGMENT
          ASSUME CS:CODE, DS:DATA, SS:AStack

PRINT PROC near
    mov     AH, 09h          ;Номер функции 09h
    int     21h
    ret
PRINT ENDP

TETR_TO_HEX PROC near
    and AL, 0Fh
    cmp AL, 09
    jbe NEXT
    add AL, 07
NEXT: add AL, 30h
    ret
TETR_TO_HEX ENDP

;-----
BYTE_TO_HEX PROC near
;byte AL translate in two symbols on 16cc numbers in AX
    push CX
    mov AH,AL
    call TETR_TO_HEX

```

```

    xchg AL,AH
    mov CL, 4
    shr AL,CL
    call TETR_TO_HEX
    pop CX
ret
BYTE_TO_HEX ENDP
;-----
WRD_TO_HEX PROC near
;translate in 16cc a 16 discharge number
;in AL - number, DI - the address of the last symbol
    push BX
    mov BH,AH
    call BYTE_TO_HEX
    mov [DI],AH
    dec DI
    mov [DI],AL
    dec DI
    mov AL,BH
    call BYTE_TO_HEX
    mov [DI],AH
    dec DI
    mov [DI],AL
    pop BX
ret
WRD_TO_HEX ENDP
;-----
BYTE_TO_DEC PROC near
;translate in 10cc, SI - the adress of the field of younger digit

```

```

    push CX
    push DX
    xor AH,AH
    xor DX,DX
    mov CX,10
loop_bd: div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae loop_bd
    cmp AL,00h
    je end_1
    or AL,30h
    mov [SI],AL
end_1: pop DX
    pop CX
ret
BYTE_TO_DEC ENDP
;-----

```

```

Main    PROC FAR
    push DS
    sub  AX,AX
    push AX
    mov  AX,DATA
    mov  DS,AX
ibm_pc:

```

```
mov DX,offset STRING_IMB  
call PRINT
```

```
MOV AX,0F000H ;указывает ES на ПЗУ  
MOV ES,AX ;  
MOV AL,ES:[0FFFEH] ;получаем байт
```

```
CMP AL,0FEH
```

```
je PC_XT
```

```
CMP AL,0FBH
```

```
je PC_XT
```

```
CMP AL,0FCH
```

```
je vAT
```

```
CMP AL,0FAH
```

```
je PS2_30
```

```
CMP AL,0F8H
```

```
je PS2_80
```

```
CMP AL,0FDH
```

```
je PCjr
```

```
CMP AL,0F9H
```

```
je PC_CON
```

```
CMP AL,0FFH
```

```
je PC
```

```
jmp Another
```

PC:

```
mov DX, offset STRING_PC  
jmp PRINT_IBM_PC
```

PC\_XT:

```
mov DX, offset STRING_PC_XT  
jmp PRINT_IBM_PC
```

vAT:

```
mov DX,offset STRING_AT  
jmp PRINT_IBM_PC
```

PS2\_30:

```
mov DX,offset STRING_PS2_30  
jmp PRINT_IBM_PC
```

PS2\_80:

```
mov DX,offset STRING_PS2_80  
jmp PRINT_IBM_PC
```

PCjr:

```
mov DX,offset STRING_PCjr  
jmp PRINT_IBM_PC
```

PC\_CON:

```
mov DX,offset STRING_PC_CON  
jmp PRINT_IBM_PC
```

Another:

```
call BYTE_TO_HEX
mov  si, offset STRING_ANOTHER
add  si, 1
mov  [si], ax
mov  dx, offset STRING_ANOTHER
```

PRINT\_IBM\_PC:

```
call PRINT
```

MS\_DOS:

```
mov dx, offset STRING_VERSION_DOS_TEXT
call PRINT
MOV AH,30h
INT 21h
cmp AL,0
je NULL_VERSION
mov  si, offset STRING_VERSION_DOS
call BYTE_TO_DEC
add si,3
mov AL,AH
call BYTE_TO_DEC
MOV dx, offset STRING_VERSION_DOS
call PRINT
jmp OEM
```

NULL\_VERSION:

```
mov dx, offset STRING_0
call PRINT
```

OEM:

```
mov dx, offset STRING_OEM_TEXT
```



```

    call PRINT
    mov AL,BH
    mov si,offset STRING_OEM
    call BYTE_TO_DEC
    mov dx, offset STRING_OEM
    call PRINT
USER_SERIES:
    mov dx,offset STRING_USER_TEXT
    call PRINT
    mov  si, offset STRING_USER
    mov al, bl
    call byte_to_hex
    mov [si], ax
    add si, 2
    mov al, ch
    call byte_to_hex
    mov [si], ax
    add si, 2
    mov al, cl
    call byte_to_hex
    mov [si], ax
    mov dx, offset STRING_USER
    call PRINT
    xor AL,AL
    mov AH,4Ch
    int 21H
    ret
Main    ENDP
CODE    ENDS

```

END Main