

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

Студентка гр. 9383

Чебесова И.Д.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

ПОСТАНОВКА ЗАДАЧИ

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. Написать текст исходно **.COM** модуля, который определяет тип РС и версию системы. Построить «плохой» **.EXE** модуль, полученный из исходного текста для **.COM** модуля.

Шаг 2. Написать текст исходного **.EXE** модуля, который выполняет те же функции, что и модуль в Шаге 1, построить и отладить его. Таким образом будет получен «хороший» **.EXE**.

Шаг 3. Сравнить исходные тексты для **.COM** и **.EXE** модулей. Ответить на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

Шаг 4. Запустить FAR и открыть файлы загрузочного модуля **.COM** и файл «плохого» **.EXE** в шестнадцатеричном виде. Затем открыть файл загрузочного модуля «хорошего» **.EXE** и сравнить его с предыдущими файлами. Ответить на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

Шаг 5. Открыть отладчик TD.EXE и загрузить **.COM**. Ответить на контрольные вопросы «Загрузка COM модуля в основную память». Представить в отчете план загрузки модуля **.COM** в основную память.

Шаг 6. Открыть отладчик TD.EXE и загрузить «хороший» **.EXE**. Ответить на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».

Функции, используемые в программе.

TETR_TO_HEX – переводит десятичную цифру в код ее символа в таблице ASCII.

BYTE_TO_HEX – байт AL переводится в два символа шестнадцатеричного числа. Ответ в AX.

WRD_TO_HEX – переводит в 16 ш/б 16-ти разрядного числа.

BYTE_TO_DEC – перевод в 10 ш/б, SI – адрес поля младшей цифры.

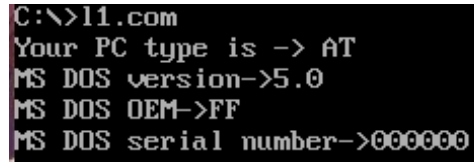
PRINT_MESSAGE – функция печати на экран.

PC_TYPE_TASK – определяет тип PC и выводит строку с названием модели.

DOS_TASK – определяет версию систему, серийный номер OEM и серийный номер пользователя и выводит их на экран.

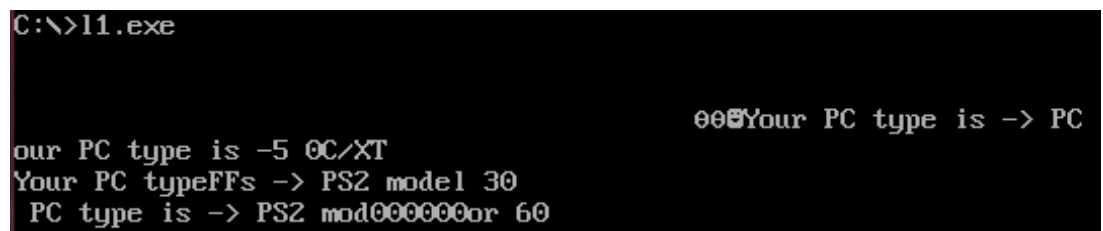
РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ ПРОБЛЕМ

Шаг 1. Был написан текст и построен .COM модуль, а также построен «плохой» .EXE, полученный из исходного текста для .COM модуля.



```
C:\>l1.com
Your PC type is -> AT
MS DOS version->5.0
MS DOS OEM->FF
MS DOS serial number->000000
```

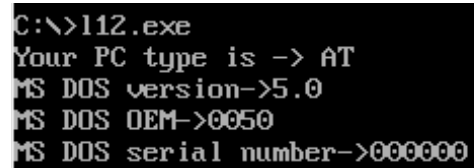
Рисунок 1. Демонстрация корректной работы .COM модуля.



```
C:\>l1.exe
000000Your PC type is -> PC
our PC type is -5 0C/XT
Your PC typeFFs -> PS2 model 30
PC type is -> PS2 mod000000or 60
```

Рисунок 2. Демонстрация некорректной работы «плохого» .EXE модуля.

Шаг 2. Был написан текст и построен «хороший» .EXE модуль, который выполняет те же функции, что и модуль в Шаге 1.



```
C:\>l12.exe
Your PC type is -> AT
MS DOS version->5.0
MS DOS OEM->0050
MS DOS serial number->000000
```

Рисунок 3. Демонстрация корректной работы «хорошего» .EXE модуля.

Шаг 3. Произведено сравнения исходных текстов для .COM и .EXE модулей. По результатам сравнения можно ответить на вопросы «Отличия исходных текстов COM и EXE программ»:

1. Сколько сегментов должна содержать COM-программа?

Ответ: Такая программа содержит только один сегмент, в который включены и данные программы и код. В то же время стек устанавливается с конца этого сегмента.

2. EXE-программа?

Ответ: Такая программа в свою очередь содержит минимум один сегмент (т.е. один и более).

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

Ответ: Директива `org 100h`, она позволяет сместить адресацию на размер PSP, а именно на 256 байтов. Если не прописать или закомментировать другую директиву — `ASSUME`, то компилятор выдаст ошибку, т.к. без этой директивы он не будет знать какой сегмент к какому сегментному регистру привязан.

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

Ответ: Нет, не все. В число запрещенных входят команды, которые непосредственно берут адрес сегмента, так как в COM-файле отсутствует таблица настроек. Такой проблемы нет в EXE файле, т.к. таблица состоит из значений в формате сегмент: смещение; к смещениям в загрузочном модуле, на которые указывают значения в таблице, после загрузки программы в память должен быть прибавлен сегментный адрес, с которого загружена программа.

Шаг 4. Произведено сравнение загрузочных модулей .COM и «плохого» .EXE с загрузочным модулем «хорошего» .EXE в шестнадцатеричном виде с помощью FAR. По итогам сравнения можно ответить на контрольные вопросы «Отличия форматов файлов COM и EXE модулей»:

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

Ответ: В COM-файле все данные хранятся в одном сегменте, адресация которого с учетом установки смещения для PSP, начинается с `0h`. При этом важно помнить, что размер программы не может превышать 64Кб.

00000000	E9	E9 02 59 6F 75 72 20	50 43 20 74 79 70 65 20	Ⓢ.Your PC type
00000010	69 73 20 2D 3E 20 50 43	0D 0A 24 59 6F 75 72 20	is -> PC..\$Your	
00000020	50 43 20 74 79 70 65 20	69 73 20 2D 3E 20 50 43	PC type is -> PC	
00000030	2F 58 54 0D 0A 24 59 6F	75 72 20 50 43 20 74 79	/XT..\$Your PC ty	
00000040	70 65 20 69 73 20 2D 3E	20 41 54 0D 0A 24 59 6F	pe is -> AT..\$Yo	
00000050	75 72 20 50 43 20 74 79	70 65 20 69 73 20 2D 3E	ur PC type is ->	
00000060	20 50 53 32 20 6D 6F 64	65 6C 20 33 30 0D 0A 24	PS2 model 30..\$	
00000070	59 6F 75 72 20 50 43 20	74 79 70 65 20 69 73 20	Your PC type is	
00000080	2D 3E 20 50 53 32 20 6D	6F 64 65 6C 20 35 30 20	-> PS2 model 50	
00000090	6F 72 20 36 30 0D 0A 24	59 6F 75 72 20 50 43 20	or 60..\$Your PC	
000000A0	74 79 70 65 20 69 73 20	2D 3E 20 50 53 32 20 6D	type is -> PS2 m	
000000B0	6F 64 65 6C 20 38 30 0D	0A 24 59 6F 75 72 20 50	odel 80..\$Your P	
000000C0	43 20 74 79 70 65 20 69	73 20 2D 3E 20 50 43 6A	C type is -> PCj	
000000D0	72 0D 0A 24 59 6F 75 72	20 50 43 20 74 79 70 65	r..\$Your PC type	
000000E0	20 69 73 20 2D 3E 20 50	43 20 43 6F 6E 76 65 72	is -> PC Conver	
000000F0	74 69 62 6C 65 0D 0A 24	59 6F 75 72 20 50 43 20	tible..\$Your PC	
00000100	54 79 70 65 20 69 73 20	75 6E 6B 6E 6F 77 6E 2E	Type is unknown.	
00000110	20 43 6F 64 65 20 2D 3E	20 0D 0A 24 4D 53 20 44	Code -> ..\$MS D	
00000120	4F 53 20 76 65 72 73 69	6F 6E 2D 3E 20 2E 20 0D	OS version-> . .	
00000130	0A 24 59 6F 75 72 20 4D	53 20 44 4F 53 20 76 65	.\$Your MS DOS ve	
00000140	72 73 69 6F 6E 20 3C 20	32 2E 30 0D 0A 24 4D 53	rsion < 2.0..\$MS	
00000150	20 44 4F 53 20 4F 45 4D	2D 3E 20 20 20 20 20 20	DOS OEM->	
00000160	20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 20		
00000170	20 0D 0A 24 4D 53 20 44	4F 53 20 73 65 72 69 61	..\$MS DOS seria	
00000180	6C 20 6E 75 6D 62 65 72	2D 3E 20 20 20 20 20 20	l number->	
00000190	0D 0A 24 24 0F 3C 09 76	02 04 07 04 30 C3 51 8A	..\$.<.v....0 Qè	
000001A0	E0 E8 EF FF 86 C4 B1 04	D2 E8 E8 E6 FF 59 C3 53	αΦΠ â— .тΦΦμ Y S	
000001B0	8A FC E8 E9 FF 88 25 4F	88 05 4F 8A C7 E8 DE FF	è"ΦΘ è%0è.0è Φ	
000001C0	88 25 4F 88 05 5B C3 51	52 32 E4 33 D2 B9 0A 00	è%0è. [QR2Σ3т ..	
000001D0	F7 F1 80 CA 30 88 14 4E	33 D2 3D 0A 00 73 F1 3C	≈±Ç—0è.N3т=.s±<	
000001E0	00 74 04 0C 30 88 04 5A	59 C3 50 B4 09 CD 21 58	.t..0è.ZY P .!=!X	
000001F0	C3 50 53 52 06 57 B8 00	F0 8E C0 BF FE FF 26 8A	PSR.ŵ.≡ÄŴ. &è	
00000200	05 3C FF 74 34 3C FE 74	39 3C FB 74 35 3C FC 74	.< t4<.t9<√t5<^t	
00000210	3A 3C FA 74 3F 3C FC 74	44 3C F8 74 49 3C FD 74	:<.t?<^tD<^tI<^2t	
00000220	4E 3C F9 74 53 E8 76 FF	BF F8 01 88 45 1A 88 65	N<.tSΦv γ°.èE.êe	
00000230	1B 8B D7 E8 B4 FF EB 49	90 BA 03 01 E8 AB FF EB	.î Φ δIE ..Φ½ δ	
00000240	40 90 BA 1B 01 E8 A2 FF	EB 37 90 BA 36 01 E8 99	@É ..Φó δ7É 6.ΦÖ	
00000250	FF EB 2E 90 BA 4E 01 E8	90 FF EB 25 90 BA 70 01	δ.É N.ΦÉ δ%É p.	
00000260	E8 87 FF EB 1C 90 BA 98	01 E8 7E FF EB 13 90 BA	Φç δ.É ÿ.Φ~ δ.É	
00000270	BA 01 E8 75 FF EB 0A 90	BA D4 01 E8 6C FF EB 01	.Φu δ.É Œ.Φl δ.	
00000280	90 5F 07 5A 5B 58 C3 50	53 52 06 57 B4 30 CD 21	É_.Z[X PSR.ŵ θ=!	
00000290	3C 00 74 1D 50 E8 2F FF	AD BF 1C 02 88 65 10 58	<.t.PΦ/ iγ..èe.X	
000002A0	86 E0 E8 22 FF AD 88 65	12 8B D7 E8 3C FF EB 07	âαΦ" jêe.î Φ< δ.	
000002B0	90 BA 32 02 E8 33 FF 8A	C7 E8 E2 FE BF 4E 02 88	É 2.Φ3 è ΦΓ.γN.è	
000002C0	45 0C 88 65 0D 8B D7 E8	20 FF 8A C3 E8 CF FE BF	E.èe.î Φ è Φ—γ	
000002D0	74 02 88 45 16 88 65 17	8B C1 83 C7 1B E8 CF FE	t.èE.èe.î—â Φ—.	
000002E0	BA 74 02 E8 04 FF 5F 07	5A 5B 58 C3 E8 02 FF E8	t.Φ. _ .Z[X Φ. Φ	
000002F0	95 FF 32 C0 B4 4C CD 21	+	â γŒ =	

Рисунок 4. Загрузочный модуль .COM в шестнадцатеричном виде.

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

Ответ: Данные и код располагаются в одном сегменте, а сегмент стека и вовсе отсутствует, что мешает корректной работе программы. Код располагается, начиная с адреса 300h. Начиная с адреса 0h располагается заголовок с MZ байтами, а также таблица настроек.

```

00000000  4D 5A F8 01 03 00 00 00 20 00 00 00 FF FF 00 00 MZ°..... ..
00000010  00 00 F6 A3 00 01 00 00 1E 00 00 00 01 00 00 00 ..÷ú.....
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

```

00000300  E9 E9 02 59 6F 75 72 20 50 43 20 74 79 70 65 20 @@.Your PC type
00000310  69 73 20 2D 3E 20 50 43 0D 0A 24 59 6F 75 72 20 is -> PC..$Your
00000320  50 43 20 74 79 70 65 20 69 73 20 2D 3E 20 50 43 PC type is -> PC
00000330  2F 58 54 0D 0A 24 59 6F 75 72 20 50 43 20 74 79 /XT..$Your PC ty
00000340  70 65 20 69 73 20 2D 3E 20 41 54 0D 0A 24 59 6F pe is -> AT..$Yo
00000350  75 72 20 50 43 20 74 79 70 65 20 69 73 20 2D 3E ur PC type is ->
00000360  20 50 53 32 20 6D 6F 64 65 6C 20 33 30 0D 0A 24 PS2 model 30..$
00000370  59 6F 75 72 20 50 43 20 74 79 70 65 20 69 73 20 Your PC type is
00000380  2D 3E 20 50 53 32 20 6D 6F 64 65 6C 20 35 30 20 -> PS2 model 50
00000390  6F 72 20 36 30 0D 0A 24 59 6F 75 72 20 50 43 20 or 60..$Your PC
000003A0  74 79 70 65 20 69 73 20 2D 3E 20 50 53 32 20 6D type is -> PS2 m
000003B0  6F 64 65 6C 20 38 30 0D 0A 24 59 6F 75 72 20 50 odel 80..$Your P
000003C0  43 20 74 79 70 65 20 69 73 20 2D 3E 20 50 43 6A C type is -> PCj
000003D0  72 0D 0A 24 59 6F 75 72 20 50 43 20 74 79 70 65 r..$Your PC type
000003E0  20 69 73 20 2D 3E 20 50 43 20 43 20 43 6F 6E 76 65 72 is -> PC Conver
000003F0  74 69 62 6C 65 0D 0A 24 59 6F 75 72 20 50 43 20 tible..$Your PC
00000400  54 79 70 65 20 69 73 20 75 6E 6B 6E 6F 77 6E 2E Type is unknown.
00000410  20 43 6F 64 65 20 2D 3E 20 0D 0A 24 4D 53 20 44 Code -> ..$MS D
00000420  4F 53 20 76 65 72 73 69 6F 6E 2D 3E 20 2E 20 0D OS version-> . .
00000430  0A 24 59 6F 75 72 20 4D 53 20 44 4F 53 20 76 65 .$.Your MS DOS ve
00000440  72 73 69 6F 6E 20 3C 20 32 2E 30 0D 0A 24 4D 53 rsion < 2.0..$MS
00000450  20 44 4F 53 20 4F 45 4D 2D 3E 20 20 20 20 20 20 DOS OEM->
00000460  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00000470  20 0D 0A 24 4D 53 20 44 4F 53 20 73 65 72 69 61 ..$MS DOS seria
00000480  6C 20 6F 75 6D 62 65 72 2D 3F 20 20 20 20 20 20 l number->

```

00000490	0D 0A 24 24 0F 3C 09 76	02 04 07 04 30 C3 51 8A	..\$.<.v....0 Qè
000004A0	E0 E8 EF FF 86 C4 B1 04	D2 E8 E8 E6 FF 59 C3 53	αφη à-тфм Y S
000004B0	8A FC E8 E9 FF 88 25 4F	88 05 4F 8A C7 E8 DE FF	è"ф0 è%0è.0è ф
000004C0	88 25 4F 88 05 5B C3 51	52 32 E4 33 D2 B9 0A 00	è%0è. [QR23т ..
000004D0	F7 F1 80 CA 30 88 14 4E	33 D2 3D 0A 00 73 F1 3C	≈±ç 0è.N3т=..s±<
000004E0	00 74 04 0C 30 88 04 5A	59 C3 50 B4 09 CD 21 58	.t..0è.ZY P .!=!X
000004F0	C3 50 53 52 06 57 B8 00	F0 8E C0 BF FE FF 26 8A	PSR.W .≡Ä .* &è
00000500	05 3C FF 74 34 3C FE 74	39 3C FB 74 35 3C FC 74	<.t4<.t9<√t5<^nt
00000510	3A 3C FA 74 3F 3C FC 74	44 3C F8 74 49 3C FD 74	:<.t?<^ntD<°tI<^t
00000520	4E 3C F9 74 53 E8 76 FF	BF F8 01 88 45 1A 88 65	N<.tSφv γ°.èE.èe
00000530	1B 8B D7 E8 B4 FF EB 49	90 BA 03 01 E8 AB FF EB	.ï ф δIE ..φ½ δ
00000540	40 90 BA 1B 01 E8 A2 FF	EB 37 90 BA 36 01 E8 99	@E ..φó δ7E 6.φ0
00000550	FF EB 2E 90 BA 4E 01 E8	90 FF EB 25 90 BA 70 01	δ.É N.φÉ δ%É p.
00000560	E8 87 FF EB 1C 90 BA 98	01 E8 7E FF EB 13 90 BA	φç δ.É ÿ.φ~ δ.É
00000570	BA 01 E8 75 FF EB 0A 90	BA D4 01 E8 6C FF EB 01	.φu δ.É L.φL δ.
00000580	90 5F 07 5A 5B 58 C3 50	53 52 06 57 B4 30 CD 21	É_.Z[X PSR.W θ=!
00000590	3C 00 74 1D 50 E8 2F FF	AD BF 1C 02 88 65 10 58	<.t.Pφ/ iγ..ée.X
000005A0	86 E0 E8 22 FF AD 88 65	12 8B D7 E8 3C FF EB 07	άαφ" ièe.ï φ< δ.
000005B0	90 BA 32 02 E8 33 FF 8A	C7 E8 E2 FE BF 4E 02 88	É 2.φ3 è φΓ.γN.è
000005C0	45 0C 88 65 0D 8B D7 E8	20 FF 8A C3 E8 CF FE BF	E.èe.ï φ è-φ±.γ
000005D0	74 02 88 45 16 88 65 17	8B C1 83 C7 1B E8 CF FE	t.èE.èe.ï ä .φ±.
000005E0	BA 74 02 E8 04 FF 5F 07	5A 5B 58 C3 E8 02 FF E8	t.φ. _Z[X φ. φ
000005F0	95 FF 32 C0 B4 4C CD 21	+	ò 2L L=!

Рисунок 5. Загрузочный модуль «плохого» .EXE в шестнадцатеричном виде.

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

Ответ: Файл состоит из трех сегментов – стека, данных и кода. Код начинается с адреса 280h после таблицы настроек и стека. Из отличий можно выделить следующие: у «плохого» EXE файла и код и данные и стек располагаются в одном сегменте, а у «хорошего» - в разных. Также в начале com файла мы специально делаем смещение на 100h для последующего расположения там PSP, а в ехе файле этого не происходит.

00000000	4D 5A 7A 01 03 00 01 00	20 00 00 00 FF FF 00 00	MZz..... ..
00000010	80 00 70 1D 59 01 21 00	1E 00 00 00 01 00 5A 01	Ç.p.Y.!.....Z.
00000020	21 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	!.....
00000030	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000040	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000050	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000060	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000070	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000080	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000090	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000A0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000B0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000C0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000D0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000E0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000F0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000100	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000110	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000120	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000130	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000140	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000150	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000160	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000170	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000180	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

00000190	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000001A0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000001B0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000001C0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000001D0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000001E0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000001F0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000200	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000210	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000220	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000230	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000240	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000250	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000260	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000270	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000280	59 6F 75 72 20 50 43 20	74 79 70 65 20 69 73 20	Your PC type is
00000290	2D 3E 20 50 43 0D 0A 24	59 6F 75 72 20 50 43 20	-> PC..\$Your PC
000002A0	74 79 70 65 20 69 73 20	2D 3E 20 50 43 2F 58 54	type is -> PC/XT
000002B0	0D 0A 24 59 6F 75 72 20	50 43 20 74 79 70 65 20	..\$Your PC type
000002C0	69 73 20 2D 3E 20 41 54	0D 0A 24 59 6F 75 72 20	is -> AT..\$Your
000002D0	50 43 20 74 79 70 65 20	69 73 20 2D 3E 20 50 53	PC type is -> PS
000002E0	32 20 6D 6F 64 65 6C 20	33 30 0D 0A 24 59 6F 75	2 model 30..\$You
000002F0	72 20 50 43 20 74 79 70	65 20 69 73 20 2D 3E 20	r PC type is ->
00000300	50 53 32 20 6D 6F 64 65	6C 20 35 30 20 6F 72 20	PS2 model 50 or
00000310	36 30 0D 0A 24 59 6F 75	72 20 50 43 20 74 79 70	60..\$Your PC typ

000003F0	24 4D 53 20 44 4F 53 20	73 65 72 69 61 6C 20 6E	\$MS DOS serial n
00000400	75 6D 62 65 72 2D 3E 20	20 20 20 20 20 0D 0A 24	umber-> ..\$
00000410	24 0F 3C 09 76 02 04 07	04 30 C3 51 8A E0 E8 EF	\$.<.v....0 QèαΦΠ
00000420	FF 86 C4 B1 04 D2 E8 E8	E6 FF 59 C3 53 8A FC E8	â—¶.ΤΦμ Υ Sè"Φ
00000430	E9 FF 88 25 4F 88 05 4F	8A C7 E8 DE FF 88 25 4F	⊙ è%0è.0è Φ è%0
00000440	88 05 5B C3 51 52 32 E4	33 D2 B9 0A 00 F7 F1 80	è. [QR2Σ3¶ .~±Ç
00000450	CA 30 88 14 4E 33 D2 3D	0A 00 73 F1 3C 00 74 04	Ⓐ0è.N3¶=..s±<.t.
00000460	0C 30 88 04 5A 59 C3 50	B4 09 CD 21 58 C3 50 53	.0è.ZY P .!= X PS
00000470	52 06 57 B8 00 F0 8E C0	BF FE FF 26 8A 05 3C FF	R.¶.≡Ä¶. &è.<
00000480	74 34 3C FE 74 39 3C FB	74 35 3C FC 74 3A 3C FA	t4<.t9<√t5<^t:<.
00000490	74 3F 3C FC 74 44 3C F8	74 49 3C FD 74 4E 3C F9	t?<^tD<°tI<^tN<.
000004A0	74 53 E8 76 FF BF F5 00	88 45 1A 88 65 1B 8B D7	tSΦv ¶ .êE.ée.ï¶
000004B0	E8 B4 FF EB 49 90 BA 00	00 E8 AB FF EB 40 90 BA	¶ δIE ..φ% δøE
000004C0	18 00 E8 A2 FF EB 37 90	BA 33 00 E8 99 FF EB 2E	..φó δ7É 3.φÖ δ.
000004D0	90 BA 4B 00 E8 90 FF EB	25 90 BA 6D 00 E8 87 FF	É K.φÉ δ%E m.Φç
000004E0	EB 1C 90 BA 95 00 E8 7E	FF EB 13 90 BA B7 00 E8	δ.É δ.Φ~ δ.É ¶.Φ
000004F0	75 FF EB 0A 90 BA D1 00	E8 6C FF EB 01 90 5F 07	u δ.É ¶.¶.δ.É_ .
00000500	5A 5B 58 C3 50 53 52 06	57 B4 30 CD 21 3C 00 74	Z[X PSR.W ø= <.t
00000510	1D 50 E8 2F FF AD BF 19	01 88 65 10 58 86 E0 E8	.PΦ/ i¶..ée.XâαΦ
00000520	22 FF AD 88 65 12 8B D7	E8 3C FF EB 07 90 BA 2F	" iée.ï¶¶< δ.É /
00000530	01 E8 33 FF 8A C7 E8 E2	FE BF 4B 01 88 45 0C 88	.Φ3 è ΦΓ.γK.êE.é
00000540	65 0D 8B D7 E8 20 FF 8A	C3 E8 CF FE BF 71 01 88	e.ï¶¶ è Φ=..γq.é
00000550	45 16 88 65 17 8B C1 83	C7 1B E8 CF FE BA 71 01	E.ée.ï â ¶.Φ= q.
00000560	E8 04 FF 5F 07 5A 5B 58	C3 B8 08 00 8E D8 E8 FD	Φ. _ Z[X ¶..Ä¶Φ²
00000570	FE E8 90 FF 32 C0 B4 4C	CD 21 +	•ΦÉ 2¶ L=!

Рисунок 6. Загрузочный модуль «хорошего» .EXE в шестнадцатеричном виде.

Шаг 5. Модуль .COM был открыт в отладчике TD.EXE. По итогам открытия файла в отладчике можно ответить на контрольные вопросы по теме «Загрузка .COM модуля в основную память».

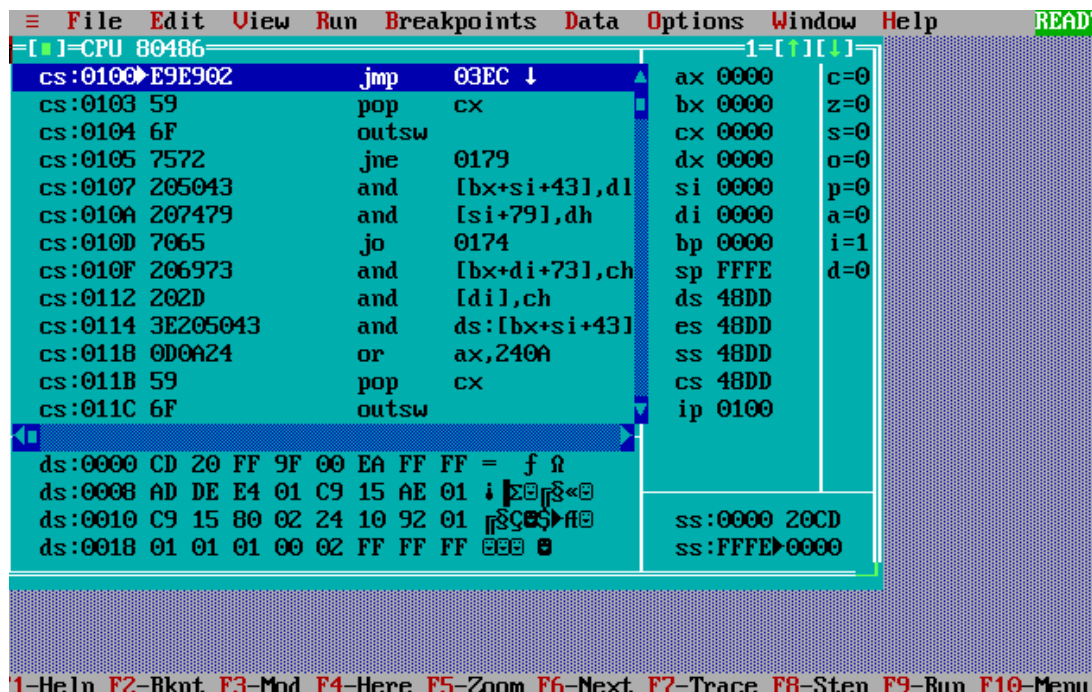


Рисунок 7. Модуль .COM, открытый в отладчике TD.EXE.

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

Ответ: В начале находится свободное место, в которое по адресу 0 помещается PSP, а по смещению 100h загружается сам код. Код располагается, начиная с адреса 100h, сразу после PSP.

2. Что располагается с адреса 0?

Ответ: с нулевого адреса располагается PSP – Program Segment Prefix (префикс программного сегмента).

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

Ответ: CS, ES, DS и SS (все сегментные регистры) имеют значение 48DD. А указывают они на начало PSP.

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

Ответ: COM-программа генерирует стек автоматически при создании исполняемого файла. Регистр SP указывает на конец стека – FFFE, а значит его адрес FFFEx. Стек может увеличиваться и таким образом даже дойти до сегмента кода, а это приведет к некорректной работе программы.

Шаг 6. «Хороший» модуль .EXE был открыт в отладчике TD.EXE. По итогам открытия файла в отладчике можно ответить на контрольные вопросы по теме «Загрузка «хорошего» .EXE модуля в основную память».

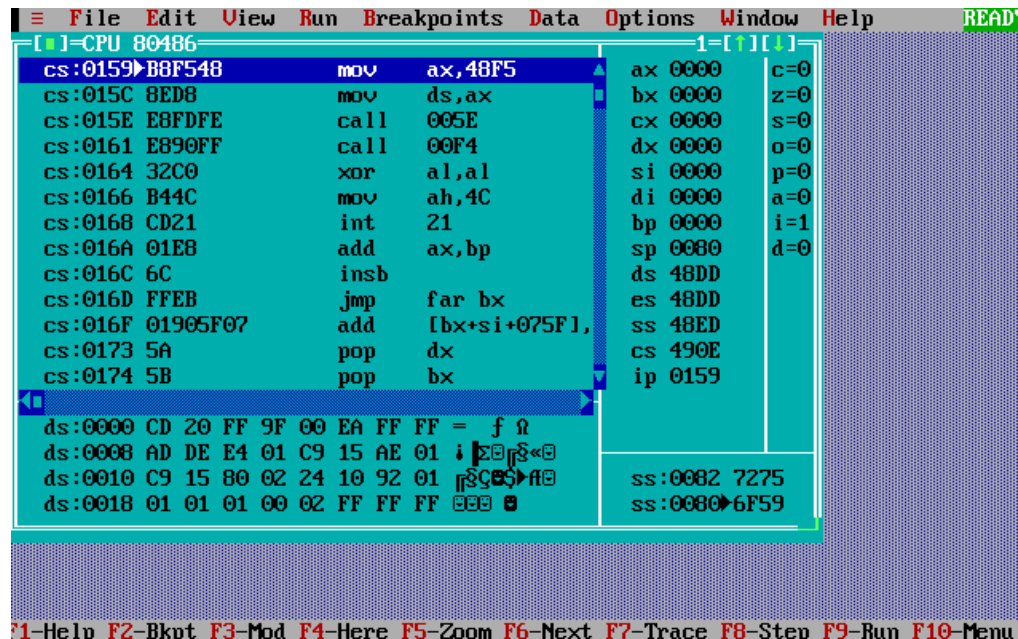


Рисунок 8. «Хороший» модуль .EXE, открытый в отладчике TD.EXE.

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

Ответ: Аналогично .COM модулю .EXE загружается со смещением относительно PSP - 100h. Значения сегментных регистров следующие:

DS и ES имеют значения 48DD, SS – 48ED, CS – 490E.

2. На что указывают регистры DS и ES?

Ответ: Они указывают на начало сегмента PSP.

3. Как определяется стек?

Ответ: Он определяется вручную с помощью директивы SEGMENT STACK, в которой указывается размер стека.

4. Как определяется точка входа?

Ответ: С помощью директивы END.

Выводы.

Исследованы различия в структурах исходных текстов модулей

типов **.COM** и **.EXE**, структур файлов загрузочных модулей и способов их загрузки в основную память.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

l1_com.asm:

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN

PC_TYPE_PC DB 'Your PC type is -> PC', 0DH, 0AH, '$'
PC_TYPE_PCXT DB 'Your PC type is -> PC/XT', 0DH, 0AH, '$'
PC_TYPE_AT DB 'Your PC type is -> AT', 0DH, 0AH, '$'
PC_TYPE_PS2MODEL30 DB 'Your PC type is -> PS2 model 30', 0DH, 0AH,
'$'
PC_TYPE_PS2MODEL50OR60 DB 'Your PC type is -> PS2 model 50 or 60',
0DH, 0AH, '$'
PC_TYPE_PS2MODEL80 DB 'Your PC type is -> PS2 model 80', 0DH, 0AH,
'$'
PC_TYPE_PCJR DB 'Your PC type is -> PCjr', 0DH, 0AH, '$'
PC_TYPE_PCCONVERTIBLE DB 'Your PC type is -> PC Convertible', 0DH,
0AH, '$'
PC_TYPE_UNKNOWN DB 'Your PC Type is unknown. Code -> ', 0DH, 0AH,
'$'

DOS_VERSION DB 'MS DOS version-> . ', 0DH, 0AH, '$'
DOS_VERSION_LESS2 DB 'Your MS DOS version < 2.0', 0DH, 0AH, '$'
DOS_OEM DB 'MS DOS OEM-> ', 0DH, 0AH, '$'
DOS_SERIAL DB 'MS DOS serial number-> ', 0DH, 0AH, '$'

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
    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
    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
    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

PRINT_MESSAGE PROC near

```

    push AX
    mov AH, 9
    int 21h
    pop AX
    ret

```

PRINT_MESSAGE ENDP

PC_TYPE_TASK PROC near

```

    push AX
    push BX
    push DX
    push ES
    push DI

```

```

    mov AX, 0F000h
    mov ES, AX
    mov DI, 0FFFEh
    mov AL, ES:[DI]

```

;TYPE PC

```
cmp AL, 0FFh
je pc
```

```
    ;TYPE PC/XT
cmp AL, 0FEh
je pc_xt
```

```
    ;TYPE PC/XT
cmp AL, 0FBh
je pc_xt
```

```
    ;TYPE AT
cmp AL, 0FCh
je pc_at
```

```
    ;TYPE PS2 MODEL 30
cmp AL, 0FAh
je ps2_model_30
```

```
    ;TYPE PS2 MODEL 50 OR 60
cmp AL, 0FCh
je ps2_model_50_or_60
```

```
    ;TYPE PS2 MODEL 80
cmp AL, 0F8h
je ps2_model_80
```

```
    ;TYPE PCJR
cmp AL, 0FDh
je pcjr
```

```
    ;TYPE PC CONVERTIBLE
cmp AL, 0F9h
je pc_convertible
```

```
    ;TYPE PC UNKNOWN
call BYTE_TO_HEX
mov DI, offset PC_TYPE_UNKNOWN
mov [DI + 26], AL
```

```
    mov [DI + 27], AH
    mov DX, DI
    call PRINT_MESSAGE
    jmp end_pc_type_task
```

```
pc:
    mov DX, offset PC_TYPE_PC
    call PRINT_MESSAGE
    jmp end_pc_type_task
```

```
pc_xt:
    mov DX, offset PC_TYPE_PCXT
    call PRINT_MESSAGE
    jmp end_pc_type_task
```

```
pc_at:
    mov DX, offset PC_TYPE_AT
    call PRINT_MESSAGE
    jmp end_pc_type_task
```

```
ps2_model_30:
    mov DX, offset PC_TYPE_PS2MODEL30
    call PRINT_MESSAGE
    jmp end_pc_type_task
```

```
ps2_model_50_or_60:
    mov DX, offset PC_TYPE_PS2MODEL500R60
    call PRINT_MESSAGE
    jmp end_pc_type_task
```

```
ps2_model_80:
    mov DX, offset PC_TYPE_PS2MODEL80
    call PRINT_MESSAGE
    jmp end_pc_type_task
```

```
pcjr:
    mov DX, offset PC_TYPE_PCJR
    call PRINT_MESSAGE
```



```

        jmp end_pc_type_task

pc_convertible:
    mov DX, offset PC_TYPE_PCCONVERTIBLE
    call PRINT_MESSAGE
    jmp end_pc_type_task

end_pc_type_task:
    pop DI
    pop ES
    pop DX
    pop BX
    pop AX
    ret
PC_TYPE_TASK ENDP

DOS_TASK PROC NEAR
    push AX
    push BX
    push DX
    push ES
    push DI

    mov AH, 30h
    int 21h

    ;AL - version number
    ;AH - modification number
    ;DH - OEM number
    ;BL:CX - users serial number

    cmp AL, 0h
    je less_2

    push AX
    call BYTE_TO_DEC
    lodsw
    mov DI, offset DOS_VERSION
    mov [DI + 16], AH

```

```

pop AX

xchg AH, AL
call BYTE_TO_DEC
lodsw
mov [DI + 18], ah
mov DX, DI
call PRINT_MESSAGE
jmp version

```

```

less_2:
    mov DX, offset DOS_VERSION_LESS2
    call PRINT_MESSAGE

```

```

version:
    mov AL, BH
    call BYTE_TO_HEX
    mov di, offset DOS_OEM
    mov [DI + 12], AL
    mov [DI + 13], AH
    mov DX, DI
    call PRINT_MESSAGE

    mov AL, BL
    call BYTE_TO_HEX
    mov DI, offset DOS_SERIAL
    mov [DI + 22], AL
    mov [DI + 23], AH
    mov AX, CX
    add DI, 27
    call WRD_TO_HEX
    mov DX, offset DOS_SERIAL
    call PRINT_MESSAGE

pop DI
pop ES
pop DX
pop BX
pop AX

```

```

        ret
DOS_TASK ENDP

BEGIN:
    call PC_TYPE_TASK
    call DOS_TASK

```

```

; Выход в DOS
xor AL, AL
mov AH, 4ch
int 21h

```

```

TESTPC ENDS
END START

```

l1_exe.asm

```

MYSTACK SEGMENT STACK
    DW 64 DUP(?)
MYSTACK ENDS

```

```

DATA SEGMENT

```

```

PC_TYPE_PC DB 'YOUR PC TYPE IS -> PC', 0DH, 0AH, '$'
PC_TYPE_PCXT DB 'YOUR PC TYPE IS -> PC/XT', 0DH, 0AH, '$'
PC_TYPE_AT DB 'YOUR PC TYPE IS -> AT', 0DH, 0AH, '$'
PC_TYPE_PS2MODEL30 DB 'YOUR PC TYPE IS -> PS2 MODEL 30', 0DH,
0AH, '$'
PC_TYPE_PS2MODEL50OR60 DB 'YOUR PC TYPE IS -> PS2 MODEL 50 OR
60', 0DH, 0AH, '$'
PC_TYPE_PS2MODEL80 DB 'YOUR PC TYPE IS -> PS2 MODEL 80', 0DH,
0AH, '$'
PC_TYPE_PCJR DB 'YOUR PC TYPE IS -> PCJR', 0DH, 0AH, '$'
PC_TYPE_PCCONVERTIBLE DB 'YOUR PC TYPE IS -> PC CONVERTIBLE',
0DH, 0AH, '$'
PC_TYPE_UNKNOWN DB 'YOUR PC TYPE IS UNKNOWN. CODE -> ', 0DH, 0AH,
'$'

```

```

DOS_VERSION DB 'MS DOS VERSION-> . ', 0DH, 0AH, '$'

```

```
DOS_VERSION_LESS2 DB 'YOUR MS DOS VERSION < 2.0', 0DH, 0AH, '$'
DOS_OEM DB 'MS DOS OEM-> ', 0DH, 0AH, '$'
DOS_SERIAL DB 'MS DOS SERIAL NUMBER-> ', 0DH, 0AH, '$'
```

```
DATA ENDS
```

```
CODE SEGMENT
```

```
    ASSUME CS:CODE, DS:DATA, SS:MYSTACK
```

```
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
```

```
    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
```

```
    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

```

```

    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_L
    OR AL, 30H
    MOV [SI], AL

```

```

END_L:
    POP DX
    POP CX
    RET

```

```

BYTE_TO_DEC ENDP

```

```

PRINT_MESSAGE PROC NEAR

```

```

    PUSH AX
    MOV AH, 9
    INT 21H

```

```
        POP AX
        RET
PRINT_MESSAGE ENDP
```

```
PC_TYPE_TASK PROC NEAR
```

```
    PUSH AX
    PUSH BX
    PUSH DX
    PUSH ES
    PUSH DI
```

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

```
        ;TYPE PC
    CMP AL, 0FFH
    JE PC
```

```
        ;TYPE PC/XT
    CMP AL, 0FEH
    JE PC_XT
```

```
        ;TYPE PC/XT
    CMP AL, 0FBH
    JE PC_XT
```

```
        ;TYPE AT
    CMP AL, 0FCH
    JE PC_AT
```

```
        ;TYPE PS2 MODEL 30
    CMP AL, 0FAH
    JE PS2_MODEL_30
```

```
        ;TYPE PS2 MODEL 50 OR 60
    CMP AL, 0FCH
```

```
JE PS2_MODEL_50_OR_60
```

```
    ;TYPE PS2 MODEL 80  
CMP AL, 0F8H  
JE PS2_MODEL_80
```

```
    ;TYPE PCJR  
CMP AL, 0FDH  
JE PCJR
```

```
    ;TYPE PC CONVERTIBLE  
CMP AL, 0F9H  
JE PC_CONVERTIBLE
```

```
    ;TYPE PC UNKNOWN  
CALL BYTE_TO_HEX  
MOV DI, OFFSET PC_TYPE_UNKNOWN  
MOV [DI + 26], AL  
MOV [DI + 27], AH  
MOV DX, DI  
CALL PRINT_MESSAGE  
JMP END_PC_TYPE_TASK
```

PC:

```
MOV DX, OFFSET PC_TYPE_PC  
CALL PRINT_MESSAGE  
JMP END_PC_TYPE_TASK
```

PC_XT:

```
MOV DX, OFFSET PC_TYPE_PCXT  
CALL PRINT_MESSAGE  
JMP END_PC_TYPE_TASK
```

PC_AT:

```
MOV DX, OFFSET PC_TYPE_AT  
CALL PRINT_MESSAGE  
JMP END_PC_TYPE_TASK
```

```

PS2_MODEL_30:
    MOV DX, OFFSET PC_TYPE_PS2MODEL30
    CALL PRINT_MESSAGE
    JMP END_PC_TYPE_TASK

PS2_MODEL_50_OR_60:
    MOV DX, OFFSET PC_TYPE_PS2MODEL50OR60
    CALL PRINT_MESSAGE
    JMP END_PC_TYPE_TASK

PS2_MODEL_80:
    MOV DX, OFFSET PC_TYPE_PS2MODEL80
    CALL PRINT_MESSAGE
    JMP END_PC_TYPE_TASK

PCJR:
    MOV DX, OFFSET PC_TYPE_PCJR
    CALL PRINT_MESSAGE
    JMP END_PC_TYPE_TASK

PC_CONVERTIBLE:
    MOV DX, OFFSET PC_TYPE_PCConvertible
    CALL PRINT_MESSAGE
    JMP END_PC_TYPE_TASK

END_PC_TYPE_TASK:
    POP DI
    POP ES
    POP DX
    POP BX
    POP AX
    RET

PC_TYPE_TASK ENDP

DOS_TASK PROC NEAR
    PUSH AX
    PUSH BX

```



```
PUSH DX
PUSH ES
PUSH DI
```

```
MOV AH, 30H
INT 21H
```

```
;AL - VERSION NUMBER
;AH - MODIFICATION NUMBER
;DH - OEM NUMBER
;BL:CX - USERS SERIAL NUMBER
```

```
CMP AL, 0H
JE LESS_2
```

```
PUSH AX
CALL BYTE_TO_DEC
LODSW
MOV DI, OFFSET DOS_VERSION
MOV [DI + 16], AH
POP AX
```

```
XCHG AH, AL
CALL BYTE_TO_DEC
LODSW
MOV [DI + 18], AH
MOV DX, DI
CALL PRINT_MESSAGE
JMP VERSION
```

```
LESS_2:
MOV DX, OFFSET DOS_VERSION_LESS2
CALL PRINT_MESSAGE
```

```
VERSION:
MOV AL, BH
CALL BYTE_TO_HEX
MOV DI, OFFSET DOS_OEM
```

```

    MOV [DI + 12], AL
    MOV [DI + 13], AH
    MOV DX, DI
    CALL PRINT_MESSAGE

    MOV AL, BL
    CALL BYTE_TO_HEX
    MOV DI, OFFSET DOS_SERIAL
    MOV [DI + 22], AL
    MOV [DI + 23], AH
    MOV AX, CX
    ADD DI, 27
    CALL WRD_TO_HEX
    MOV DX, OFFSET DOS_SERIAL
    CALL PRINT_MESSAGE

    POP DI
    POP ES
    POP DX
    POP BX
    POP AX
    RET
DOS_TASK ENDP

MAIN PROC FAR
    MOV AX, DATA
    MOV DS, AX
    CALL PC_TYPE_TASK
    CALL DOS_TASK

    ; ВЫХОД В DOS
    XOR AL, AL
    MOV AH, 4CH
    INT 21H

MAIN ENDP

CODE ENDS

```

END MAIN