ИТМИНОБРНАУКИ 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 организации управления основной памятью

Студент гр. 9381	 Давыдов Д.С.
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.

«Истина познается в сравнении», как говорили древние. К счастью, у нас есть возможность исследовать в одной системе два различных формата загрузочных модулей, сравнить лучше Теноп ИХ И система программирования и управляющая программа обращаются с ними. Система программирования включает компилятор с языка ассемблер (часто называется, просто, ассемблер), который изготавливает объектные модули. Компоновщик (Linker) по совокупности объектных модулей, изготавливает загрузочный модуль, а также, функция ядра – загрузчик, которая помещает программу в основную память и запускает на выполнение. Все эти компоненты согласованно работают для изготовления и выполнения загрузочных модулей разного типа. Для выполнения лабораторной работы сначала нужно изготовить загрузочные модули.

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 хх.уу, где хх номер основной версии, а уу - номер модификации в десятичной системе счисления, формировать строки с серийным номером ОЕМ и серийным номером пользователя. Полученные строки выводятся на экран. Отладьте полученный исходный модуль. Результатом выполнения этого шага будет «хороший» .СОМ модуль, а также необходимо построить «плохой» .ЕХЕ, полученный из исходного текста для .СОМ 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.

Описание функций.

Используя шаблон из методических указаний был написан исходный текст com-модуля. С помощью команд TASM и TLINK был получен «хороший» загрузочный модуль COM и «плохой» модуль EXE.

Результат запуска ехе модуля:

Результат запуска сот модуля:

T:\OS>lb1_com.com type of PC – AT System version: 5.0 OEM: 255 Serial user number: 000000

Также был переписан код для ехе-модуля, который работает идентично com-модулю – «хороший» ехе-модуль:

T:\OS>1b1_exe.exe PC type – AT System version1840 OEM: O Serial user number: 0000FF

Разработанный програмный код смотреть в приложении А.

Ответы на контрольные вопросы:

Отличия исходных текстов СОМ и ЕХЕ программ

1) Сколько сегментов должна содержать СОМ-программа?

Ответ: 1

2) ЕХЕ-программа?

Ответ: любое количество

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

Ответ: Директива ASSUME - сообщение транслятору о том, какие

сегменты использовать в соответствии с какими регистрами, т.к. сегменты сами

по себе равноправны. Директива **ORG 100h**, задающая смещение для всех

адресов программы для префикса программного сегмента. Директива END

необходима для завершения любой программы

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

Ответ: В СОМ-программе можно использовать только near-переходы из-за

наличия только 1 сегмента. Команды с адресом сегмента также нельзя

использовать, тк он до загрузки неизвестен.

Отличия форматов файлов СОМ и ЕХЕ модулей

1) Какова структура файла СОМ? С какого адреса располагается код? –

Ответ: СОМ – файл состоит из одного сегмента включающий в себя

сегменты данных и кода. Код начинается с адреса 0h.

00000000 E9 EF 01 74 79 70 65 20 6F 66 20 50 43 20 2D 20 и́п type of PC -00000010 24 50 43 0D OA 24 41 54 \$PC \$PC/XT \$AT 0A 24 50 43 2F 58 54 0D 00000020 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 33 30 0D \$PS2 model 30 \$PS2 model 50 o 00000030 0A 24 50 53 32 20 6D 6F 64 65 6C 20 35 30 20 6F 00000040 72 20 36 30 0D 0A 24 50 53 32 20 6D 6F 64 65 6C r 60 \$PS2 model 24 50 43 20 80 \$PCjr 00000050 20 38 30 0D 0A 24 50 43 6A 72 0D 0A \$PC 000000060 43 6F 6E 76 65 72 74 69 62 6C 65 0D 0A 24 20 20 Convertible \$ 00000070 20 20 20 65 72 72 6F 72 2E 20 55 6E 6B 6E 6F 77 error. Unknow 080000080 n \$System versi 6E 0D 0A 24 53 79 73 74 65 6D 20 76 65 72 73 69 00000090 6F 6E 3A 20 20 2E 20 20 0D 0A 24 4F 45 4D 3A 20 \$OEM: 000000A0 20 20 0D 0A 24 53 65 72 69 61 6C 20 75 73 65 72 \$Serial user 000000B0 20 6E 75 6D 62 65 72 3A 20 20 20 20 20 20 20 0D number: 000000000 0A 24 24 0F 30 09 76 02 04 07 04 30 C3 51 8A E0 \$\$ < v 0ГОЉа 000000D0 E8 EF FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A ипя‡Д± ТиижяҮГЅЉ 000000E0 FC E8 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 ьийя≣%О≣ ОЉЗиЮя≣ 000000F0 25 4F 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 %0**ш** [ГQR2д3Т№ ч 00000100 F1 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 c⊞KO■ N3T= sc< 88 04 5A 59 C3 50 B4 09 **O■** ZYFP۲ H!XF 00000110 74 04 0C 30 CD 21 58 C3 00000120 50 52 06 B8 00 F0 8E CO 26 A0 FE FF BA 03 01 E8 РК ё рЋА& юяє и 74 23 3C FE 74 25 3C FB зя<яt#<юt%<ыt!<ь 00000130 E7 FF 3C FF 74 21 3C FC 00000140 74 23 3C FA 74 25 3C FC 74 27 3C F8 74 29 3C FD t#<ъt%<ьt'<шt)<э 00000150 74 2B 3C F9 74 2D EB 31 90 BA 11 01 EB 3A 90 BA t+<щt-л1he л:he 00000160 16 01 EB 34 90 BA 1E 01 EB 2E 90 BA 23 01 EB 28 л4he л.he# л(00000170 90 BA 32 01 EB 22 90 BA 47 01 EB 1C 90 BA 56 01 he2 A"heG A heV л не] л неп Ри=я 00000180 EB 16 90 BA 5D 01 EB 10 90 BA 6E 01 50 E8 3D FF <т∎ Ғ∎\$Хи~я ZХГР 8B F2 88 04 46 88 24 58 E8 7E FF 07 00000190 5A 58 C3 50 000001A0 52 B4 30 CD 21 50 56 BE 84 01 83 C6 10 E8 46 FF Rґ0H!PUs" ŕЖ иFя 000001B0 83 C6 03 8A C4 E8 3E FF 5E 58 8A C7 BE 9B 01 83 **ѓЖ ЉДи>я^ХЉЗs> ѓ** C6 07 E8 31 FF 8A C3 E8 ж и1яЉГи яїҐ ѓ3 000001C0 03 FF BF A5 01 83 C7 14 000001D0 89 05 8B C1 BF A5 01 83 C7 19 E8 01 FF BA 84 01 ‰ <ῥ ѓ3 и яє" 000001E0 E8 36 FF BA 9B 01 E8 30 FF BA A5 01 E8 2A FF 5A ибяє> и0яєї и*яZ 000001F0 58 C3 E8 2B FF E8 A7 FF 32 C0 B4 4C CD 21 🗆 🗆 ХГи+яи§я2АҐЬН!□□

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

Ответ: Файл некорректно работает, т. к. он состоит из одного сегмента — данные и код хранятся в одном месте, а сегмента стека нет. Адресация этого сегмента начинается с 300h. Начиная с адреса 0h до 200h располагается EXEмаркер (MZ), заголовок и таблица настройки, а затем с 200h до 300h идет смещение.

Изображение - плохой ехе файл

П	00000000	4D 5A FE 00 03 00 00 00 20 00 00 00 FF FF 00 00	МZю яя	嫡
П	00000010	00 00 00 00 00 01 00 00 3E 00 00 00 01 00 FB 30	> m0	Ā>·
П	00000020	6A 72 00 00 00 00 00 00 00 00 00 00 00 00 00	jr	牪
П	~~~~~~	~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~ ~~	-	

00000300 E9 EF 01 74 79 70 65 20 6F 66 20 50 43 20 2D 20 и́п type of PC -\$PC \$PC/XT \$AT 00000310 24 50 43 0D OA 24 50 43 2F 58 54 0D OA 24 41 54 00000320 \$PS2 model 30 OD OA 24 50 53 32 20 6D 6F 64 65 6C 20 33 30 0D 00000330 \$PS2 model 50 o 0A 24 50 53 32 20 6D 6F 64 65 6C 20 35 30 20 6F 00000340 72 20 36 30 OD 0A 24 50 r 60 \$PS2 model 53 32 20 6D 6F 64 65 6C 00000350 20 38 30 0D 0A 24 50 43 80 \$PCjr \$PC 6A 72 0D 0A 24 50 43 20 62 6C 65 0D 00000360 43 6F 6E 76 65 72 74 69 OA 24 20 20 Convertible 72 72 6F 72 6B 6E 6F 77 00000370 20 20 20 65 2E 20 55 6E error. Unknow 00000380 6E 0D 0A 24 53 79 73 74 65 6D 20 76 65 72 73 69 n \$System versi 00000390 6F 6E 3A 20 20 2E 20 20 OD OA 24 4F 45 4D 3A 20 \$OEM: on: . 20 20 0D 0A 24 53 65 72 000003A0 69 61 6C 20 75 73 65 72 \$Serial user 20 20 20 20 000003B0 20 6E 75 6D 62 65 72 3A 20 20 20 0D number: 000003C0 0A 24 24 0F 30 09 76 02 04 07 04 30 C3 51 8A E0 \$\$ < v **ОГ**ОЉа E8 EF FF 86 C4 B1 04 D2 000003D0 ипя‡Д± ТиижяЧГЅЉ E8 E8 E6 FF 59 C3 53 8A 000003E0 FC E8 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 ьийя≣%О≣ ОЉЗиЮя≣ 000003FF 25 4F 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 %О**ш** [ГQR2д3Т**№** 00000400 F1 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 c⊞KO■ N3T= sc< 74 04 0C 30 88 04 5A 59 C3 50 B4 09 00000410 CD 21 58 C3 t O■ ZYFPr H!XF 00000420 50 52 06 B8 00 F0 8E C0 26 A0 FE FF BA 03 01 E8 PR ё рЋА& юяє и 00000430 E7 FF 3C FF 74 23 3C FE 74 25 3C FB 74 21 3C FC зя<яt#<юt%<ыt!<ь 00000440 74 23 3C FA 74 25 3C FC 74 27 3C F8 74 29 3C FD t#<bt%<bt'<wt)<3 74 2B 3C F9 74 2D EB 31 00000450 90 BA 11 01 EB 3A 90 BA t+<ut-11he 1:he 16 01 EB 34 90 BA 1E 01 EB 2E 90 BA 00000460 23 01 EB 28 л4ђе л.ђе# л(ђе2 л"ђеG л ђе∪ 00000470 90 BA 32 01 EB 22 90 BA 47 01 EB 1C 90 BA 56 01 EB 16 90 BA 5D 01 EB 10 90 BA 6E 01 00000480 50 E8 3D FF л ђе] л ђеп Ри=я ∢τ∎ F∎\$Xи~я ZXΓP 00000490 8B F2 88 04 46 88 24 58 E8 7E FF 07 5A 58 C3 50 000004A0 52 B4 30 CD 21 50 56 BE 84 01 83 C6 10 E8 46 FF Rґ0H!PVs" ѓЖ иFя 000004B0 83 C6 03 8A C4 E8 3E FF 5E 58 8A C7 BE 9B 01 83 **ѓЖ ЉДи>я^ХЉЗs> ѓ** 000004C0 C6 07 E8 31 FF 8A C3 E8 03 FF BF A5 01 83 C7 14 ж и1яЉГи яїҐ ѓ3 ‰ ≺БїҐ ѓ3 и яє" 000004D0 89 05 8B C1 BF A5 01 83 C7 19 E8 01 FF BA 84 01 E8 36 FF BA 9B 01 E8 30 FF BA A5 01 E8 2A FF 5A 000004E0 ибяє> и0яєї и*яZ 000004F0 58 C3 E8 2B FF E8 A7 FF 32 C0 B4 4C CD 21 🗆 🗆 ХГи+яи§я2АҐЬН!□□

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

Ответ: В хорошем ехе-файле стек и код разделены по сегментам и не имеется директивы ORG 100h(под префикс программного сегмента) — код начинается с адреса 200h

Изображение - хороший ехе файл

00000000	4D 5A 1C 00	03 00 01 00	20 00 11 00 FF FF 22 00	МZ яя"
00000010	00 01 00 00	10 00 00 00	3E 00 00 00 01 00 FB 30) > ผูย
00000020	6A 72 00 00	00 00 00 00	00 00 00 00 00 00 00 00	jr
00000030	00 00 00 00	00 00 00 00	00 00 00 00 00 00 44 01	_ D
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	
000000060	00 00 00 00	00 00 00 00	00 00 00 00 00 00 00 00	
00000070				

00000210 E9 30 01 24 OF 3C 09 76 02 04 07 04 30 C3 51 8A и́0 \$ < v 00000220 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 аипя‡Д± ТиижяЧГЅ EØ E8 EF FF 00000230 8A FC E8 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 00000240 88 25 4F 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 **■%0**■ [ГQR2д3Т№ чс™О∎ N3T= sc< 00000250 F7 F1 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C 30 88 04 5A 59 C3 50 B4 09 CD 21 58 00000260 t O■ ZYFPr H!X 00000270 C3 50 52 06 B8 00 F0 8E C0 26 A0 FE FF BA 00 00 ГРК ё рЋА& юяє 00000280 E8 E7 FF 3C FF 74 23 3C FE 74 25 3C FB 74 21 3C usa<at#<nt%<ыt!< FC 74 23 3C FA 74 25 3C FC 74 27 3C ьt#<ъt%<ьt'<шt)< 00000290 F8 74 29 3C 000002A0 FD 74 2B 3C F9 74 2D EB 31 90 BA 0B 00 EB 3A 90 at+<щt-π1hε π:h BA 1D 00 EB 000002B0 BA 10 00 EB 34 90 BA 18 00 EB 2E 90 л4фе л.фе л 1C 90 BA 50 00 EB 22 90 000002C0 28 90 BA 2C BA 41 00 EB (ђе, л"ђеА л ђеР 00 EB 16 90 BA 57 00 EB 10 90 BA 68 00 50 E8 3D л heW л heh Ри= 000002D0 000002E0 FF 8B F2 88 04 46 88 24 58 E8 7E FF 07 5A 58 C3 я<т∎ F∎\$Xи~я ZXГ C6 10 E8 46 000002F0 50 52 B4 30 CD 15 50 56 BE 7E 00 83 PRCOH PUS~ ĆЖ иF яѓЖ ЉДи>я^ХЉЗs∙ 00000300 FF 83 C6 03 8A C4 E8 3E FF 5E 58 8A C7 BE 95 00 **ѓЖ и1яЉГи яїц ѓ3** 00000310 83 C6 07 E8 31 FF 8A C3 E8 03 FF BF 9F 00 83 C7 00000320 14 89 05 8B C1 BF 9F 00 83 C7 19 E8 01 FF BA 7E ‰ <Бїџ ґ3 и яє~ 00000330 BA 95 00 E8 30 FF BA 9F 00 E8 2A FF 00 E8 36 FF ибяє• иОяєц и*я 00000340 5A 58 C3 B8 16 00 8E D8 E8 26 FF E8 A2 FF 32 C0 **ZXГё ЋШи&яиўя2А** 00000350 B4 4C CD 21 00 00 00 00 00 00 00 00 00 00 00 00 L'LH! 00000360 50 43 20 74 79 70 65 20 2D 20 24 50 43 OD OA 24 PC type - \$PC PC/XT \$AT \$PS2 00000370 50 43 2F 58 54 0D 0A 24 41 54 0D 0A 24 50 53 32 model 30 \$PS2 20 6D 6F 64 65 6C 20 33 00000380 30 OD OA 24 50 53 32 20 00000390 6D 6F 64 65 6C 20 35 30 20 6F 72 20 36 30 0D 0A model 50 or 60 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 OD OA 24 \$PS2 model 80 \$ 000003A0 50 43 6A 72 0D 0A 24 50 43 20 43 6F 000003B0 6E 76 65 72 PCjr \$PC Conver 000003C0 74 69 62 6C 65 0D 0A 24 20 20 20 20 20 65 72 72 tible \$ 000003D0 6F 72 2E 20 55 6E 6B 6E 6F 77 6E 0D 0A 24 53 79 or. Unknown \$Sy 000003E0 73 74 65 6D 20 76 65 72 73 69 6F 6E 3A 20 20 2E stem version: 000003F0 20 20 0D 0A 24 4F 45 4D 3A 20 20 20 OD OA 24 53 \$0EM: 00000400 65 72 69 61 6C 20 75 73 65 72 20 6E 75 6D 62 65 erial user numbe 00000410 72 3A 20 20 20 20 20 20 00 0A 24 \$0000

Загрузка СОМ модуля в основную память

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

Ответ: начиная с 0 адреса в оперативную память загружается PSP. Сам код располагается, начиная с адреса 100h, сразу после PSP.

2) Что располагается с адреса 0?

Ответ: Префикс программного сегмента (PSP)

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

Ответ: Все сегментные регистры имеют значение 48DD. Сегментные регистры указывают на начало PSP.

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

Ответ: Указатель стека в конце сегмента, занимает неиспользованную память, адреса меняются от FFFEh к 0000h

Загрузка «хорошего» EXE модуля в основную память

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

Ответ: Создается префикс программного сегмента, в начальный сегмент записывается загрузочный модуль, таблица настройки — в рабочую память. Каждый сегмент прибавляет сегментный адрес начального сегмента данных. Далее определяются значения сегментных регистров. DS и ES — начало PSP(48DD), CS — Начало команд(4932). SS — начало стека(48ED)

2) На что указывают регистры DS и ES?

Ответ: DS и ES – начало PSP(48DD)

3) Как определяется стек?

Ответ: В отличие от СОМ стек определяется при объявлении его сегмента, выделяется указанная память.

4) Как определяется точка входа?

Ответ: директивой END с адресом, с которого начинается выполнение программы.

Выводы.

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lr1_com.asm

```
TESTPC SEGMENT
ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
      ORG 100H
START: JMP MAIN
; ДАННЫЕ
pc type str DB 'type of PC - ', '$'
pc_type_1 DB 'PC', ODH, OAH, '$'
pc type 2 DB 'PC/XT', ODH, OAH, '$'
pc type 3 DB 'AT', ODH, OAH, '$'
pc type 4 DB 'PS2 model 30', 0DH, 0AH, '$'
pc type 5 DB 'PS2 model 50 or 60', 0DH, 0AH, '$'
pc type 6 DB 'PS2 model 80', 0DH, 0AH, '$'
pc type 7 DB 'PCjr', ODH, OAH, '$'
pc type 8 DB 'PC Convertible', ODH, OAH, '$'
unknown pc type DB ' error. Unknown', ODH, OAH ,'$'
System version DB 'System version: . ', ODH, OAH, '$'
OEM DB 'OEM: ', ODH, OAH, '$'
user serial number DB 'Serial user number: ',0DH, 0AH, '$'
;Представление 4 бита регистра al в виде цифры 16ой с.с. и представление
её в символьном виде.
TETR TO HEX PROC near
     and AL, OFh
     cmp AL,09
     jbe NEXT
     add AL,07
NEXT:
     add AL, 30h ; результат в al
     ret
TETR TO HEX ENDP
;Представление al как два числа в 16-ой с.с. и перемещение их в ах
BYTE TO HEX PROC near
     push CX
     mov AH, AL
     call TETR TO HEX
     xchq AL, AH
     mov CL, 4
     shr AL, CL
     call TETR TO HEX ;в AL старшая цифра
     рор СХ ;в АН младшая
     ret
BYTE TO HEX ENDP
; перевод в 16 с.с 16-ти разрядного числа
; в АХ - число, а в DI - адрес последнего символа
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
; перевод в 10 c/c. SI - адрес поля младшей цифры
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_1:
           pop DX
           pop CX
           ret
BYTE TO DEC ENDP
WRITE PROC NEAR
    push ax
     mov AH, 9
    int 21h ; функция DOS по прерыванию
     pop ax
    ret
WRITE ENDP
; получение типа РС
GET PC TYPE PROC NEAR
     push ax
     push dx
     push es
     mov ax, 0F000h
     mov es, ax
     mov al, es:[0FFFEh]
     mov dx, offset pc_type_str
     call WRITE
     cmp al, 0FFh ; распознавание типа РС по специльной технической
таблице
     je pc_type_1_case
     cmp al, OFEh
     je pc type 2 case
```

```
je pc type 2 case
     cmp al, OFCh
     je pc_type_3_case
     cmp al, OFAh
     je pc type 4 case
     cmp al, OFCh
     je pc_type_5_case
     cmp al, 0F8h
     je pc type 6 case
     cmp al, OFDh
     je pc_type_7_case
     cmp al, 0F9h
     je pc type 8 case
     jmp unknown pc type case
pc_type_1_case:
     mov dx, offset pc_type_1; загрузка в зависимости от значения al
смещения нужной строки
     jmp final step
pc_type_2_case:
     mov dx, offset pc type 2
     jmp final step
pc type 3 case:
     mov dx, offset pc type 3
     jmp final_step
pc type 4 case:
     mov dx, offset pc_type_4
     jmp final step
pc_type_5_case:
     mov dx, offset pc_type_5
     jmp final step
pc_type_6 case:
     mov dx, offset pc_type_6
     jmp final step
pc_type_7_case:
     mov dx, offset pc type 7
     jmp final_step
pc_type_8_case:
     mov dx, offset pc_type_8
     jmp final_step
unknown pc type case:
     mov dx, offset unknown pc type
     push ax
     call BYTE TO HEX
     mov si, dx
     mov [si], al
     inc si
     mov [si], ah ; в начало сообщения записывается код ошибки в 16-
ричном виде
     pop ax
final step:
     call write
end of proc:
     pop es
     pop dx
     pop ax
     ret
GET PC TYPE ENDP
```

cmp al, OFBh

```
GET VERSRION PROC NEAR
     push ax
     push dx
     MOV AH, 30h
     INT 21h
     ; write version number
;Сначала надо обработать al, а потом ah и записать в конец System version
           push ax
           push si
           lea si, System_version
           add si, 16
           call BYTE TO DEC
           add si, 3
           mov al, ah
           call BYTE TO DEC
           pop si
          pop ax
;OEM
     mov al, bh
     lea si, OEM
     add si, 7
     call BYTE TO DEC
;user_serial_number
     mov al, bl
     call BYTE TO HEX ;
     lea di, user serial number
     add di, 20
     mov [di], ax
     mov ax, cx
     lea di, user_serial_number
     add di, 25
     call WRD TO HEX
version :
     mov dx, offset System_version
     call WRITE
get OEM:
     mov dx, offset OEM
     call write
get user serial number:
     mov dx, offset user_serial_number
     call write
end of proc 2:
     pop dx
     pop ax
     ret
GET VERSRION ENDP
MAIN: ;функция main
     call GET PC TYPE
     call GET_VERSRION
;выход в ДОС
     xor AL, AL
     mov AH, 4Ch
     int 21H
```

TESTPC ENDS

END START end