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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Операционные системы»

Тема: Исследование структур загрузочных модулей

Студент гр. 8383 _____ Костарев К.В.

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.

Выполнение работы.

В первую очередь был написан текст исходного .COM модуля для программы, определяющей тип PC и версию системы. Для выполнения этой задачи ассемблерная программа читает содержимое предпоследнего байта ROM BIOS, и в соответствие с табл. 1 определяет тип PC и выводит строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код переводится в символьную строку, содержащую запись шестнадцатеричного числа и выводится на экран.

Таблица 1 – Соответствие кода и типа PC

Тип	Код
PC	FF
PC/XT	FE, FB
AT	FC
PS2 модель 30	FA
PS2 модель 50 или 60	FC
PS2 модель 80	F8
PC Convertiable	F9
PCjr	FD

Для решения задачи определения версии системы ассемблерная программа по значению регистров AL и AH, после выполнения функции 30H прерывания 21H, формирует текстовую строку в формате x.y, где x – номер основной версии, а y – номер модификации, числа в десятичной системе счисления. Также по значению регистров BH и BL:CX формируются строки со значением OEM и серийного номера пользователя.

Результатом линковки были получены «хороший» .COM и «плохой» .EXE модули.

Далее был написан текст исходного «хорошего» .EXE модуля, который выполняет те же задачи, что и .COM и «плохой» .EXE, с последующей линковкой.

Код исходных модулей представлен в Приложении А.

Отличия исходных текстов COM и EXE программ:

1. *Сколько сегментов должна содержать COM-программа?*

COM-программа содержит только один сегмент.

2. *EXE-программа?*

Тем временем EXE может содержать произвольное число сегментов.

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

ORG 100H, которая необходима для размещения PSP, и Assume для инициализации регистров.

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

В COM отсутствует таблица настроек, а следовательно для них некорректно указание адреса сегмента.

После этого в FAR были открыты файлы .COM, «плохого» и «хорошего» .EXE модулей в шестнадцатеричном виде с последующим их сравнением. Содержание файлов представлено в Приложении Б.

Отличия форматов файлов COM и EXE модулей:

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

Файл COM содержит только лишь сегмент с кодом и данными, в памяти код начинается с адреса 100H.

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

«Плохой» EXE тоже содержит только сегмент с кодом и данными, но начинается с адреса 300H. С адреса 0H располагается заголовок и таблица настроек.

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

В «плохом» EXE только один сегмент, и при этом файл занимает больше памяти, т.к. по умолчанию память до адреса 300H зарезервирована под таблицу релокации. В начале файлов информация отличается.

Далее в отладчике TD был загружен .COM файл, представленный на рис. 1.

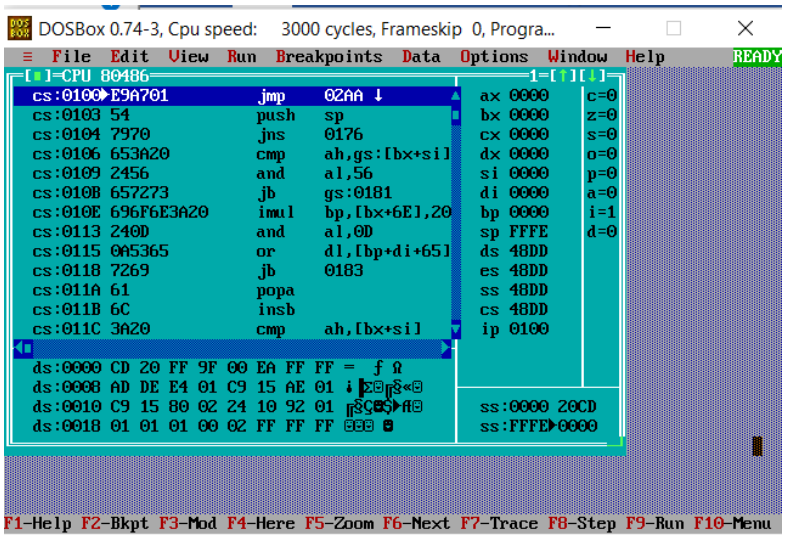


Рисунок 1 – COM файл, запущенный в отладчике TD

Загрузка COM модуля в основную память:

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

Код и данные располагаются с адреса 100H.

2. *Что располагается с адреса 0?*

При загрузке ОС с адреса 0H располагается PSP.

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

При запуске COM файла все сегментные регистры содержат адрес PSP.

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

Стек занимает всю доступную память вместе с кодом и данными. При загрузке в SP записывается FFFE, а в BP 0000.

Также в отладчике TD был загружен и «хороший» EXE файл, который представлен на рис. 2.

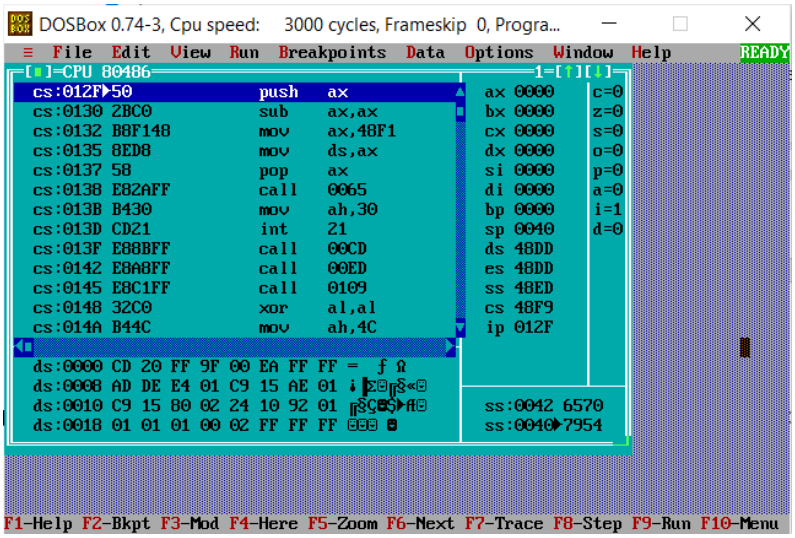


Рисунок 2 – «хороший» EXE файл, запущенный в отладчике TD

Загрузка «хорошего» EXE модуля в основную память:

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

В сегментные регистры CS и SS записываются адреса начала соответствующего сегмента при загрузке, а в DS и ES адрес начала PSP.

2. *На что указывают регистры DS и ES?*

DS и ES указывают на начало PSP.

3. *Как определяется стек?*

Память под стек выделяется в соответствии с моделью памяти в программе, если только это не прописано в самой программе. При ее загрузке инициализируются регистры BP и SP.

4. *Как определяется точка входа?*

Точкой входа является начало сегмента кода CS, если она не указана явно. В программе точка входа указывается в конце директивой END <точка входа>.

Какая дополнительная информация находится в заголовке EXE модуля?

00-01 4D5A — сигнатура файла .EXE;

02-03 Длина образа задачи по модулю 512 (то есть число полезных байт в последнем блоке). Компоновщики версий до 1.10 помещали в это поле 04; если оно имеет такое значение, его рекомендуется игнорировать);

04-05 Длина файла в блоках;

06-07 Число элементов таблицы настройки адресов;

08-09 Длина заголовка в 16-байтных параграфах. Используется для выяснения начала тела загрузочного модуля;

0A-0B Минимальный объём памяти, которую нужно выделить после конца образа задачи (в 16-байтных параграфах);

0C-0D Максимальный объём памяти, которую нужно выделить после конца образа задачи (в 16-байтных параграфах);

0E-0F Сегментный адрес начала стекового сегмента относительно начала образа задачи;

10-11 Значение SP при входе в задачу;

12-13 Контрольная сумма — ноль минус результат сложения без переноса всех слов файла;

14-15 Значение IP (счетчика команд) при входе в задачу;

16-17 Сегментный адрес начала кодового сегмента относительно начала образа задачи;

18-19 Адрес первого элемента таблицы настройки адресов относительно начала файла;

1A-1B Номер сегмента перекрытий (0 для корневого сегмента программы).

Выводы.

В ходе выполнения данной лабораторной работы были исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память. Для исследования были написаны две программы для COM и EXE модулей, решающие одну и ту же задачу определения типа PC, версии и серийных номеров OEM и пользователя.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД РЕАЛИЗОВАННЫХ ПРОГРАММ

COM модуль:

```
TESTPC SEGMENT
ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
ORG 100H
START: JMP BEGIN
TYPE_PC db 'Type: ', '$'
VERSION db 'Version: ', '$'
SERIAL db 0DH, 0AH, 'Serial: ', '$'
OEM db 0DH, 0AH, 'OEM: ', '$'
NEW_STR db "  $"
NEW_STR2 db "        $"
STR_PC db 'PC',0DH,0AH,'$'
STR_PCXT db 'PC/XT',0DH,0AH,'$'
STR_AT db 'AT',0DH,0AH,'$'
STR_PS230 db 'PS2 model 30',0DH,0AH,'$'
STR_PS280 db 'PS2 model 80',0DH,0AH,'$'
STR_PCJR db 'PCjr',0DH,0AH,'$'
STR_PCCO db 'PC Convertible',0DH,0AH,'$'

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_l
    or AL,30h
    mov [SI],AL
end_l: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

WRITE_NUMBER PROC near
    push AX
    mov AH, 02H
    int 21H
    pop AX
    ret
WRITE_NUMBER ENDP

WRITE_STRING PROC near
    push AX
    mov AH, 09H
    int 21H
    pop AX
    ret
WRITE_STRING ENDP
```

```
WRITE_TYPE PROC near
    push   AX
    push   BX
    push   DX
    push   ES
    mov DX,offset TYPE_PC
    call WRITE_STRING
    mov    BX,0F000H
    mov    ES,BX
    mov    AL,ES:[0FFFEH]
    call BYTE_TO_HEX
    cmp AX, 0FFH
    je printPC
    cmp AX, 0FEH
    je printPCXT
    cmp AX, 0FCH
    je printAT
    cmp AX, 0FAH
    je printPS230
    cmp AX, 0F8H
    je printPS280
    cmp AX, 0FDH
    je printPCjr
    cmp AX, 0F9H
    je printPCCo
printPC:
    mov DX,offset STR_PC
    jmp printSTR
printPCXT:
    mov DX,offset STR_PCXT
    jmp printSTR
printAT:
    mov DX,offset STR_AT
    jmp printSTR
printPS230:
    mov DX,offset STR_PS230
    jmp printSTR
printPS280:
    mov DX,offset STR_PS280
    jmp printSTR
printPCjr:
    mov DX,offset STR_PCJR
    jmp printSTR
printPCCo:
    mov DX,offset STR_PCCO
printSTR:
    call WRITE_STRING
    pop    ES
    pop    DX
    pop    BX
```

```
    pop    AX
    ret
WRITE_TYPE ENDP

WRITE_VERSION PROC near
    push AX
    push DX
    mov DX, offset VERSION
    call WRITE_STRING
    mov DL, AL
    add DL, '0'
    call WRITE_NUMBER
    mov DL, '.'
    call WRITE_NUMBER
    mov DL, AH
    add DL, '0'
    call WRITE_NUMBER
    pop DX
    pop AX
    ret
WRITE_VERSION ENDP

WRITE_OEM PROC near
    push AX
    push DX
    mov DX, offset OEM
    call WRITE_STRING
    mov SI, offset NEW_STR
     add SI, 2
     mov AL, BH
     call BYTE_TO_DEC
     mov DX, offset NEW_STR
     call WRITE_STRING
    pop DX
    pop AX
    ret
WRITE_OEM ENDP

WRITE_SERIAL PROC near
    push AX
    push DX
    mov DX, offset SERIAL
    call WRITE_STRING
    mov DI, offset NEW_STR2
     add DI, 6
     mov AX, CX
     call WRD_TO_HEX
     mov AL, BL
     call BYTE_TO_HEX
     sub DI, 2
```

```
      mov [DI], AX
      mov dx, offset NEW_STR2
      call WRITE_STRING
    pop DX
    pop AX
    ret
WRITE_SERIAL ENDP

BEGIN:
    call WRITE_TYPE
    mov AH, 30H
    int 21H
    call WRITE_VERSION
    call WRITE_OEM
    call WRITE_SERIAL
    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
END START
```

EXE модуль:

```
AStack    SEGMENT  STACK
          DW 20h DUP(?)
AStack    ENDS

DATA  SEGMENT
    TYPE_PC db 'Type: ', '$'
    VERSION db 'Version: ', '$'
    SERIAL db 0DH, 0AH, 'Serial: ', '$'
    OEM db 0DH, 0AH, 'OEM: ', '$'
    NEW_STR db "  $"
    NEW_STR2 db "        $"
    STR_PC db 'PC',0DH,0AH,'$'
    STR_PCXT db 'PC/XT',0DH,0AH,'$'
    STR_AT db 'AT',0DH,0AH,'$'
    STR_PS230 db 'PS2 model 30',0DH,0AH,'$'
    STR_PS280 db 'PS2 model 80',0DH,0AH,'$'
    STR_PCJR db 'PCjr',0DH,0AH,'$'
    STR_PCCO db 'PC Convertible',0DH,0AH,'$'
DATA ENDS

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

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

WRITE_NUMBER PROC near
    push AX
    mov AH, 02H
    int 21H
    pop AX
    ret
WRITE_NUMBER ENDP

WRITE_STRING PROC near
    push AX
    mov AH, 09H
    int 21H
    pop AX
    ret
WRITE_STRING ENDP

WRITE_TYPE PROC near
    push   AX
    push   BX
    push   DX
    push   ES
    mov DX,offset TYPE_PC
    call WRITE_STRING
    mov    BX,0F000H
    mov    ES,BX
    mov    AL,ES:[0FFFEH]
    call BYTE_TO_HEX
    cmp AX, 0FFH
    je printPC
    cmp AX, 0FEH
    je printPCXT
    cmp AX, 0FCH
    je printAT
    cmp AX, 0FAH
    je printPS230
    cmp AX, 0F8H
    je printPS280
    cmp AX, 0FDH
    je printPCjr
    cmp AX, 0F9H
    je printPCCo
printPC:
    mov DX,offset STR_PC
    jmp printSTR
printPCXT:
```

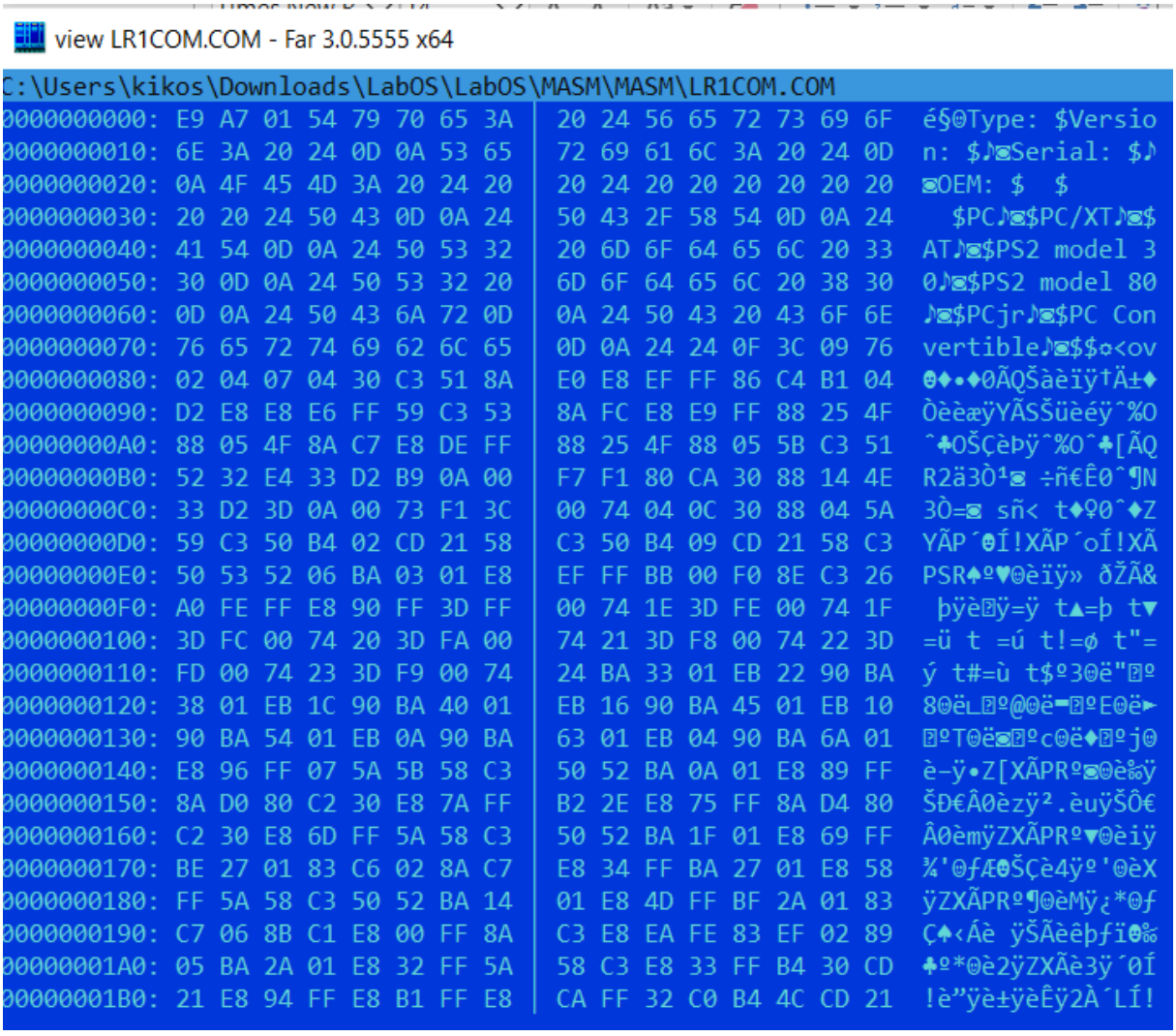
```
    mov DX,offset STR_PCXT
    jmp printSTR
printAT:
    mov DX,offset STR_AT
    jmp printSTR
printPS230:
    mov DX,offset STR_PS230
    jmp printSTR
printPS280:
    mov DX,offset STR_PS280
    jmp printSTR
printPCjr:
    mov DX,offset STR_PCJR
    jmp printSTR
printPCCo:
    mov DX,offset STR_PCCO
printSTR:
    call WRITE_STRING
    pop    ES
    pop    DX
    pop    BX
    pop    AX
    ret
WRITE_TYPE ENDP

WRITE_VERSION PROC near
    push AX
    push DX
    mov DX, offset VERSION
    call WRITE_STRING
    mov DL, AL
    add DL, '0'
    call WRITE_NUMBER
    mov DL, '.'
    call WRITE_NUMBER
    mov DL, AH
    add DL, '0'
    call WRITE_NUMBER
    pop DX
    pop AX
    ret
WRITE_VERSION ENDP

WRITE_OEM PROC near
    push AX
    push DX
    mov DX, offset OEM
    call WRITE_STRING
    mov SI, offset NEW_STR
     add SI, 2
```

```
      mov AL, BH
      call BYTE_TO_DEC
      mov DX, offset NEW_STR
      call WRITE_STRING
    pop DX
    pop AX
    ret
WRITE_OEM ENDP

WRITE_SERIAL PROC near
    push AX
    push DX
    mov DX, offset SERIAL
    call WRITE_STRING
    mov DI, offset NEW_STR2
      add DI, 6
      mov AX, CX
      call WRD_TO_HEX
      mov AL, BL
      call BYTE_TO_HEX
      sub DI, 2
      mov [DI], AX
      mov dx, offset NEW_STR2
      call WRITE_STRING
    pop DX
    pop AX
    ret
WRITE_SERIAL ENDP

MAIN PROC FAR
    push AX
    sub AX,AX
    mov AX,DATA
    mov DS,AX
    pop AX
    call WRITE_TYPE
    mov AH, 30H
    int 21H
    call WRITE_VERSION
    call WRITE_OEM
    call WRITE_SERIAL
    xor AL,AL
    mov AH,4Ch
    int 21H
MAIN ENDP
CODE ENDS
      END MAIN
```

ПРИЛОЖЕНИЕ Б

СОДЕРЖАНИЕ ФАЙЛОВ ПРОГРАММ В ШЕСТНАДЦАТЕРИЧНОМ ВИДЕ

.COM:



«Плохой» .EXE:


view LR1COM.EXE - Far 3.0.5555 x64
C:\Users\kikos\Downloads\LabOS\LabOS\MASM\MASM\LR1COM.EXE
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: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000002A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000002B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000002C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000002D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000002E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000002F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000300: E9 A7 01 54 79 70 65 3A 20 24 56 65 72 73 69 6F й§☺Type: $Versio
0000000310: 6E 3A 20 24 0D 0A 53 65 72 69 61 6C 3A 20 24 0D n: $♪◙Serial: $♪
0000000320: 0A 4F 45 4D 3A 20 24 20 20 24 20 20 20 20 20 20 ◙OEM: $ $
0000000330: 20 20 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 $PC♪◙$PC/XT♪◙$
0000000340: 41 54 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 33 AT♪◙$PS2 model 3
0000000350: 30 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0♪◙$PS2 model 80
0000000360: 0D 0A 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F 6E ♪◙$PCjr♪◙$PC Con
0000000370: 76 65 72 74 69 62 6C 65 0D 0A 24 24 0F 3C 09 76 vertible♪◙$$☼<ov
0000000380: 02 04 07 04 30 C3 51 8A E0 E8 EF FF 86 C4 B1 04 ☻♦•♦0ГQЉаипя†Д±♦
0000000390: D2 E8 E8 E6 FF 59 C3 53 8A FC E8 E9 FF 88 25 4F ТиижяYГSЉьийя€%O
00000003A0: 88 05 4F 8A C7 E8 DE FF 88 25 4F 88 05 5B C3 51 €♣OЉЗиЮя€%O€♣[ГQ
00000003B0: 52 32 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 88 14 4E R2д3ТN№◙ чсЂК0€¶N
00000003C0: 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C 30 88 04 5A 3Т=◙ sc< t♦♀0€♦Z
00000003D0: 59 C3 50 B4 02 CD 21 58 C3 50 B4 09 CD 21 58 C3 YГРґ☻Н!ХГРґоН!ХГ
00000003E0: 50 53 52 06 BA 03 01 E8 EF FF BB 00 F0 8E C3 26 PSR♠є▼☻ипя» рЋГ&
00000003F0: A0 FE FF E8 90 FF 3D FF 00 74 1E 3D FE 00 74 1F юяиЋя=я t▲=ю t▼
0000000400: 3D FC 00 74 20 3D FA 00 74 21 3D F8 00 74 22 3D =ь t =ъ t!=ш t"=
0000000410: FD 00 74 23 3D F9 00 74 24 BA 33 01 EB 22 90 BA э t#=щ t$є3☻л"Ђє
0000000420: 38 01 EB 1C 90 BA 40 01 EB 16 90 BA 45 01 EB 10 8☻лLЂє@☻л━ЂєE☻л►
0000000430: 90 BA 54 01 EB 0A 90 BA 63 01 EB 04 90 BA 6A 01 ЂєT☻л◙Ђєc☻л♦Ђєj☻
0000000440: E8 96 FF 07 5A 5B 58 C3 50 52 BA 0A 01 E8 89 FF и-я•Z[ХГРRє◙☻иљяя
0000000450: 8A D0 80 C2 30 E8 7A FF B2 2E E8 75 FF 8A D4 80 ЉРЂВ0иzяI.иияЉФЂ
0000000460: C2 30 E8 6D FF 5A 58 C3 50 52 BA 1F 01 E8 69 FF В0имяZХГРRє▼☻иія
0000000470: BE 27 01 83 C6 02 8A C7 E8 34 FF BA 27 01 E8 58 s'☻ѓЖ☻ЉЗи4яє'☻иХ
0000000480: FF 5A 58 C3 50 52 BA 14 01 E8 4D FF BF 2A 01 83 яZХГРRє¶☻иМяї*☻ѓ
0000000490: C7 06 8B C1 E8 00 FF 8A C3 E8 EA FE 83 EF 02 89 З♠‹Би яЉГикюѓп☻‰
00000004A0: 05 BA 2A 01 E8 32 FF 5A 58 C3 E8 33 FF B4 30 CD ♣є*☻и2яZХГиЗяґ0Н
00000004B0: 21 E8 94 FF E8 B1 FF E8 CA FF 32 C0 B4 4C CD 21 !и”яи±яиКя2АґLН!

«Хороший» .EXE:

view LR1EXE.EXE - Far 3.0.5555 x64

```
C:\Users\kikos\Downloads\LabOS\LabOS\MASM\MASM\LR1EXE.EXE
0000000000: 4D 5A 0E 00 03 00 01 00  20 00 00 00 FF FF 00 00  MZ♫ ♥ @     ÿÿ
0000000010: 40 00 E5 16 2F 01 0C 00  1E 00 00 00 01 00 33 01  @ å▬/@♀ ▲   @ 3@
0000000020: 0C 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: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0000000230: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0000000240: 54 79 70 65 3A 20 24 56  65 72 73 69 6F 6E 3A 20  Type: $Version:
0000000250: 24 0D 0A 53 65 72 69 61  6C 3A 20 24 0D 0A 4F 45  $♪◙Serial: $♪◙OE
0000000260: 4D 3A 20 24 20 20 24 20  20 20 20 20 20 20 20 24  M: $  $         $
0000000270: 50 43 0D 0A 24 50 43 2F  58 54 0D 0A 24 41 54 0D  PC♪◙$PC/XT♪◙$AT♪
0000000280: 0A 24 50 53 32 20 6D 6F  64 65 6C 20 33 30 0D 0A  ◙$PS2 model 30♪◙
0000000290: 24 50 53 32 20 6D 6F 64  65 6C 20 38 30 0D 0A 24  $PS2 model 80♪◙$
00000002A0: 50 43 6A 72 0D 0A 24 50  43 20 43 6F 6E 76 65 72  PCjr♪◙$PC Conver
00000002B0: 74 69 62 6C 65 0D 0A 24  00 00 00 00 00 00 00 00  tible♪◙$
00000002C0: 24 0F 3C 09 76 02 04 07  04 30 C3 51 8A E0 E8 EF  $☼<○v•♦••♦0ÃQŠàèï
00000002D0: FF 86 C4 B1 04 D2 E8 E8  E6 FF 59 C3 53 8A FC E8  ÿ†Ä±♦Òèèæÿ YÃSŠüè
00000002E0: E9 FF 88 25 4F 88 05 4F  8A C7 E8 DE FF 88 25 4F  éÿˆ%Oˆ♣OŠÇèÞÿˆ%O
00000002F0: 88 05 5B C3 51 52 32 E4  33 D2 B9 0A 00 F7 F1 80  ˆ♣[ÃQR2ä3Ò¹◙ ÷ñ€
0000000300: CA 30 88 14 4E 33 D2 3D  0A 00 73 F1 3C 00 74 04  Ê0ˆ¶N3Ò=◙ sñ< t♦
0000000310: 0C 30 88 04 5A 59 C3 50  B4 02 CD 21 58 C3 50 B4  ♀0ˆ♦ZYÃP´☻Í!XÃP´
0000000320: 09 CD 21 58 C3 50 53 52  06 BA 00 00 E8 EF FF BB  ○Í!XÃPSR♠º  èïÿ»
0000000330: 00 F0 8E C3 26 A0 FE FF  E8 90 FF 3D FF 00 74 1E  ðŽÃ& þÿè▣ÿ=ÿ t▲
0000000340: 3D FE 00 74 1F 3D FC 00  74 20 3D FA 00 74 21 3D  =þ t▼=ü t =ú t!=
0000000350: F8 00 74 22 3D FD 00 74  23 3D F9 00 74 24 BA 30  ø t"=ý t#=ù t$º0
0000000360: 00 EB 22 90 BA 35 00 EB  1C 90 BA 3D 00 EB 16 90  ë"▣º5 ë∟▣º= ë▬▣
0000000370: BA 42 00 EB 10 90 BA 51  00 EB 0A 90 BA 60 00 EB  ºB ë►▣ºQ ë◙▣º` ë
0000000380: 04 90 BA 67 00 E8 96 FF  07 5A 5B 58 C3 50 52 BA  ♦▣ºg è–ÿ•Z[XÃPRº
0000000390: 07 00 E8 89 FF 8A D0 80  C2 30 E8 7A FF B2 2E E8  • è‰ÿŠĐ€Â0èzÿ².è
00000003A0: 75 FF 8A D4 80 C2 30 E8  6D FF 5A 58 C3 50 52 BA  uÿŠÔ€Â0èmÿZXÃPRº
00000003B0: 1C 00 E8 69 FF BE 24 00  83 C6 02 8A C7 E8 34 FF  ∟ èiÿ¾$ ƒÆ☻ŠÇè4ÿ
00000003C0: BA 24 00 E8 58 FF 5A 58  C3 50 52 BA 11 00 E8 4D  º$ èXÿZXÃPRº◄ èM
00000003D0: FF BF 27 00 83 C7 06 8B  C1 E8 00 FF 8A C3 E8 EA  ÿ¿' ƒÇ♠‹Áè ÿŠÃèê
00000003E0: FE 83 EF 02 89 05 BA 27  00 E8 32 FF 5A 58 C3 50  þƒï☻‰♠º' è2ÿZXÃP
00000003F0: 2B C0 B8 04 00 8E D8 58  E8 2A FF B4 30 CD 21 E8  +À¸♦ ŽØXè*ÿ´0Í!è
0000000400: 8B FF E8 A8 FF E8 C1 FF  32 C0 B4 4C CD 21        ‹ÿè¨ÿèÁÿ2À´LÍ!
```