**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 заголовочных модулей»**

Студент гр. 7381 _____ Трушников А.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

**Цель работы.**

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

**Основные теоретические положения.**

Тип IBM PC узнается путем считывания предпоследнего байта с ROM BIOS. Его значение сравнивается с кодами таблицы, представленной ниже.

| | |
|---|---|
| PC | FF |
| PC/XT | FE,FB |
| AT | FC |
| PS2 модель 30 | FA |
| PS2 модель 50 или 60 | FC |
| PS2 модель 80 | F8 |
| PCjr | FD |
| PC Convertible | F9 |

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
- AH – номер модификации
- BH – серийный номер OEM
- BL:CX – 24-битовый серийный номер пользователя

**Выполнение работы.**

Написан текст исходного .COM модуля, который определяет тип PC и версию системы. Для решения поставленной задачи был использован шаблон ассемблерного текста с функциями управляющей программы и процедурами перевода двоичных кодов в символы шестадцатеричных чисел и десятичное число из раздела “общие сведения” методический указаний. Для того, чтобы

узнать тип IBM PC программа обращается к предпоследнему байту ROM BIOS. Далее полученное значение сравнивается с таблицей. Для определения версии MS DOS используется функция 30h 21h-го прерывания. И в соответствие с полученными данными в регистрах.

Написан текст исходного .EXE модуля с тем же функционалом.

1. Результат выполнения «плохого» .EXE модуля:


2. Результат выполнения «хорошего» .COM модуля:


3. Результат выполнения «хорошего» .EXE модуля:


**Отличия исходных текстов COM и EXE файлов**

1 COM-программа должна содержать один сегмент – сегмент кода.

2 EXE-программа должна содержать три сегмента –сегмент стека, сегмент данных, сегмент кода.

3 В тексте COM-программы обязательно должна быть директива ASSUME, которая должна указывать, что сегмент кода и данных начинается с одного и того же места. Также должна быть директива ORG, указывающая, сколько места в памяти необходимо зарезервировать под PSP.

4 В COM-программе можно использовать все форматы команд.

**Отличия форматов файлов COM и EXE модулей**

1 Структура COM-файла очень компактна (сам файл весит гораздо меньше), код располагается с нулевого адреса.

2 Структура «плохого» EXE-файла менее компактна, чем у COM-файла. Код располагается с адреса 300h, с нулевого адреса располагается таблица настроек, при помощи которых строится данный EXE-файл.

3 Структура «хорошего» EXE-файла несколько компактнее, чем структура «плохого» файла, так как в этом файле отсутствует директива ORG 100h, резервирующая пространство для заголовка. Именно поэтому код располагается с адреса 200h, а не с 300h, как в «плохом» EXE-файле.

**Загрузка COM-модулей в основную память**

1 Порядок загрузки модуля COM: PSP, данные и код, стек. Код начинается с адреса 100h.

2 С нулевого адреса располагается PSP.

3 Все сегментные регистры имеют значение «5BC9» и указывают на начало PSP.

4 Стек занимает всё свободное пространство до конца файла (размер .COM файла не может превышать 64 кб), оставшееся после загрузки данных и кода. В данном случае значение регистра SP=FFF5.

**Загрузка хорошего EXE-модуля в основную память**

1 Порядок загрузки EXE-модуля: PSP, сегмент кода, сегмент данных, сегмент стека. Сегментные регистры на момент загрузки программы имеют значения: ES=119C, CS=11B8, DS=119C, SS=11CE. На начальном этапе ES=DS, так как не были выполнены

команды “mov ax, data; mov ds, ax”, т.е. в регистр данных не был помещён адрес сегмента данных.

2 Регистры DS и ES указывают на начало PSP

3 В регистры SS и SP записываются значения, указанные в заголовке, а к SS прибавляется сегментный адрес начального сегмента.

4 Оператором END start_procedure_name. Эта информация хранится в заголовке модуля.

**Выводы.**

Исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов загрузки в основную память. Реализована программа на языке ассемблера позволяющая определить тип IBM PC и тип системы.

## ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ ТЕКСТ .COM МОДУЛЯ

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

;ДАННЫЕ
PC_TYPE                 DB          'PC TYPE:','$'
PC_UNDEFINED    DB          'PC UNDEFINED:  ',0DH,0AH,'$'
SYSTEM_VERSION DB           'SYSTEM VERSION:  .  ',0DH,0AH,'$'
OEM_NUMBER              DB          'OEM NUMBER:    ',0DH,0AH,'$'
SERIAL_NUMBER   DB          'USER SERIAL NUMBER:   ',0DH,0AH,'$'
PC                      DB          'PC',0DH,0AH,'$'
PCXT            DB          'PC/XT',0DH,0AH,'$'
AT                      DB          'AT',0DH,0AH,'$'
PS2_30                  DB          'PS2 MODEL 30',0DH,0AH,'$'
PS2_50                  DB          'PS2 MODEL 50 OR 60',0DH,0AH,'$'
PS2_80                  DB          'PS2 MODEL 80',0DH,0AH,'$'
PCJR            DB          'PSJR',0DH,0AH,'$'
PC_CONVERTIBLE  DB          'PS 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
;БАЙТ В AL ПЕРЕВОДИТСЯ В ДВА СИМВОЛА ШЕСТН. ЧИСЛА В AX
                PUSH CX
                MOV  AH,AL
                CALL TETR_TO_HEX
                XCHG AL,AH
                MOV  CL,4
                SHR  AL,CL
                CALL TETR_TO_HEX;В AL СТАРШАЯ ЦИФРА
                POP CX                  ;В AH МЛАДШАЯ
                RET
BYTE_TO_HEX ENDP

WRITE_MSG       PROC NEAR
                MOV     AH,09H
                INT     21H
                RET
WRITE_MSG       ENDP

WRD_TO_HEX   PROC NEAR
;ПЕРЕВОД В 16 С/С 16-ТИ РАЗРЯДНОГО ЧИСЛА
```

```
;В AX - ЧИСЛО, DI - АДРЕС ПОСЕЛЕДНЕГО СИМВОЛА
                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

GET_PC_NUMBER  PROC NEAR
; Функция определяющая тип PC
        PUSH    ES
        MOV             BX,0F000H
        MOV     ES,BX
        MOV             BX,0
        MOV     AX,ES:[0FFFEH]
        POP             ES
        RET
GET_PC_NUMBER           ENDP

GET_SYS_VER             PROC NEAR
; Функция определяющая версию системы
        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
        RET
GET_SYS_VER     ENDP

GET_SERIAL_NUM  PROC    NEAR
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    SI
        MOV     AL,BL
        CALL    BYTE_TO_HEX
        LEA     DI,SERIAL_NUMBER
        ADD     DI,22
        MOV     [DI],AX
        MOV     AX,CX
        LEA     DI,SERIAL_NUMBER
        ADD     DI,27
        CALL    WRD_TO_HEX
        POP     SI
        POP     CX
```

```
                POP             BX
                POP             AX
                RET
GET_SERIAL_NUM ENDP

GET_OEM_NUM             PROC NEAR
; ФУНКЦИЯ ОПРЕДЕЛЯЮЩАЯ OEM
                PUSH    AX
                PUSH    BX
                PUSH    SI
                MOV     AL,BH
                LEA             SI,OEM_NUMBER
                ADD             SI,14
                CALL    BYTE_TO_DEC
                POP             SI
                POP             BX
                POP             AX
                RET
GET_OEM_NUM             ENDP

DEFINE_PC_TYPE  PROC NEAR
; Функция, определяющая тип PC
                CMP     AL, 0FFH
                JNE             CMP1
                MOV     DX, OFFSET PC
                RET
```

```
CMP1:  CMP     AL, 0FEH
       JNE     CMP2
       MOV     DX, OFFSET PCXT
       RET

CMP2:  CMP     AL, 0FCH
       JNE     CMP3
       MOV     DX, OFFSET AT
       RET

CMP3:  CMP     AL, 0FAH
       JNE     CMP4
       MOV     DX, OFFSET PS2_30
       RET

CMP4:  CMP     AL, 0FCH
       JNE     CMP5
       MOV     DX, OFFSET PS2_50
       RET

CMP5:  CMP     AL, 0F8H
       JNE     CMP6
       MOV     DX, OFFSET PS2_80
       RET

CMP6:  CMP     AL, 0FDH
       JNE     CMP7
```

```
        MOV     DX, OFFSET PCJR
        RET

CMP7:   CMP     AL, 0F9H
        JNE             CMP8
        MOV     DX, OFFSET PC_CONVERTIBLE
        RET

CMP8:   CALL    BYTE_TO_HEX
        LEA             BX,PC_UNDEFINED
        MOV             [BX+14],AX
        MOV     DX, OFFSET PC_UNDEFINED
        RET

DEFINE_PC_TYPE          ENDP

BEGIN:
        CALL    GET_PC_NUMBER
        MOV     DX, OFFSET PC_TYPE
        CALL    WRITE_MSG
        CALL    DEFINE_PC_TYPE
        CALL    WRITE_MSG
        XOR     AX,AX
        MOV             AH,30H
        INT             21H
        CALL    GET_SYS_VER
        CALL    GET_OEM_NUM
```

```
        CALL    GET_SERIAL_NUM
        MOV     DX, OFFSET SYSTEM_VERSION
        CALL    WRITE_MSG
        MOV     DX, OFFSET OEM_NUMBER
        CALL    WRITE_MSG
        MOV     DX, OFFSET SERIAL_NUMBER
        CALL    WRITE_MSG
        XOR     AL,AL
        MOV     AH,4CH
        INT     21H
TESTPC  ENDS
        END     START
```