**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ  
по лабораторной работе № 3**

**по дисциплине «Операционные системы»**

**Тема: «Исследование организации управления основной памятью»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8304 |  | Сергеев А.Д. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2019

**Цель работы.**

Исследование структур данных и работы функций управления памятью операционной системы.

**Постановка задачи:**

**Шаг 1.**

Необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1. Количество доступной памяти.

2. Размер расширенной памяти.

3. Выводит цепочку блоков управления памятью.

**Шаг 2.**

Изменить программу таким образом, чтобы она освобождала память, которую не занимает.

**Шаг 3.**

Изменить программу ещё раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти.

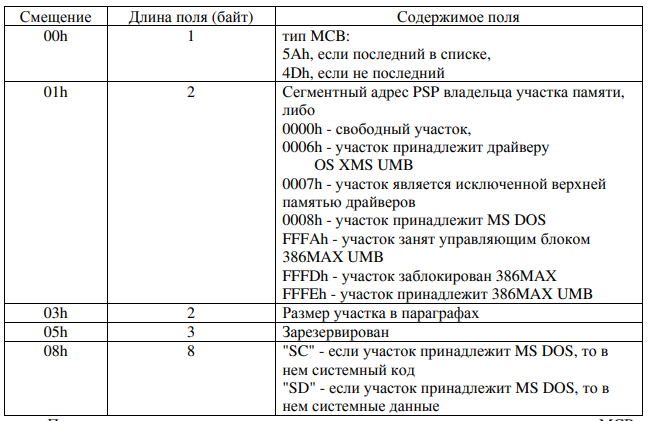
**Шаг 2.**

Изменить первоначальный вариант программы, запросив 64Кб памяти до освобождения памяти. Обработать завершение функций ядра, проверяя флаг CF.

**Необходимые сведения для составления программы.**

Учёт занятой и свободной памяти ведётся при помощи списка блоков управления памятью MCB (Memory Control Block). MCB занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адреса сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

MCB имеет следующую структуру:

По сегментному адресу и размеру участка памяти, контролируемого этим MCB можно определить местоположение следующего MCB в списке.

Адрес первого MCB хранится во внутренней структуре MS DOS, называемой «List of Lists» (список списков).

Размер расширенной памяти находится в ячейках 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ.

**Описание программы.**

В результате выполнения лабораторной работы была написана программа, описание функций которой представлено ниже.

- DEC\_WORD\_PRINT - вывод содержимого AX в десятичной системе счисления;

- HEX\_BYTE\_PRINT - вспомогательная для HEX\_WORD\_PRINT процедура;

- HEX\_WORD\_PRINT - вывод содержимого AX в шестнадцатиричной системе счисления;

- PRINT\_STRING - вывод строки из DX на экран;

- PRINT\_AVL\_MEMORY - вывод количества доступной памяти (в килобайтах);

- PRINT\_EXT\_MEMORY - вывод количества расширенной памяти (в килобайтах).

- PRINT\_MCB - вывод списка MCB;

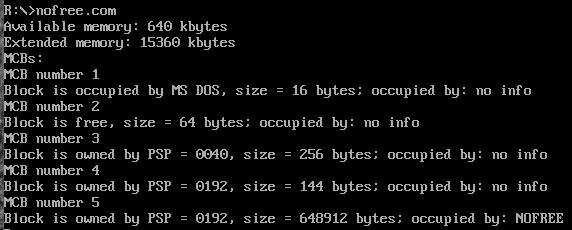
- FREE\_MEM (для пунктов 2 - 4) - освобождение незанятой памяти;

- ALLOC\_MEM (для пунктов 3 - 4) - выделение 64Кб памяти.

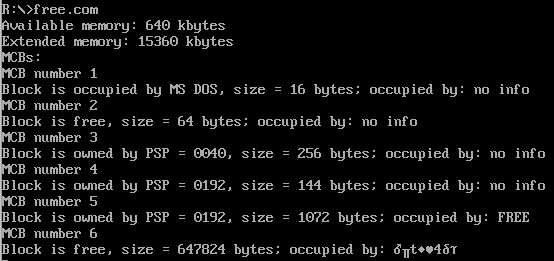
**Ход работы**

Написание исходного кода производилось в редакторе Atom на базе операционной системы Windows 10, сборка и отладка производились в эмуляторе DOSBox.

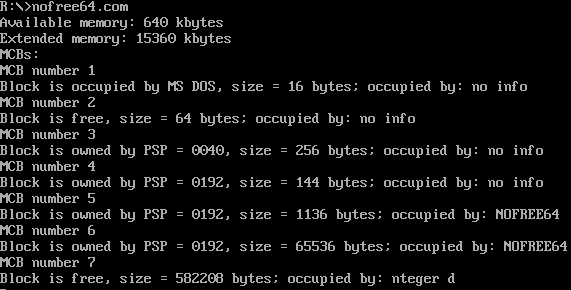
Модуль под названием NOFREE.COM выводит на экран размер доступной и расширенной памяти, а также данные MCB блоков.

Программа занимает всю доступную память.

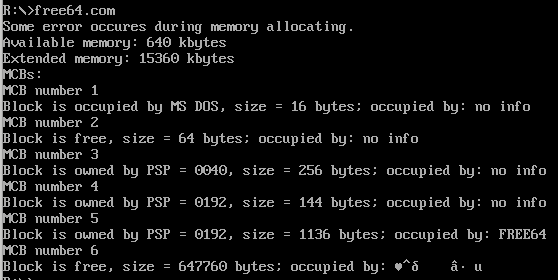
Модуль под названием FREE.COM освобождает память, которая не занята программой.

Теперь программа занимает не всю память, освобождённая память относится к новому, свободному блоку.

Модуль под названием NOFREE64.COM освобождает память, которая не занята программой, а потом запрашивает 64Кб памяти.

Дополнительная память была выделена отдельным блоком, а блок, который появился после освобождения незанятой памяти, остался неизменным.

Модуль под названием FREE64.COM сначала запрашивает 64 Кб памяти, а потом освобождает незанятое.

Дополнительная память не была выделена, так как на момент попытки выделения вся доступная память уже принадлежит программе, о чём свидетельствует сообщение в начале вывода. Под программу выделено меньше 64Кб.

**Вывод.**

В результате выполнения данной лабораторной работы был изучен список блоков управления памятью, а также методы выделения и освобождения памяти для программы.

**Контрольные вопросы.**

**Что означает «доступный объём» памяти:**

Максимальный объём памяти, доступный для запуска и выполнения программ.

**Где MCB блок Вашей программы в списке:**

Принадлежность блока памяти можно определить, по адресу его владельца (расположенному со смещением в один байт в MCB). Также название модуля-владельца может содержаться в последних восьми байтах MCB. Как видно из вывода программы, ей обычно принадлежит два блока, первый из которых имеет фиксированный размер в 144 байта, а второй зависит от размера исходного кода. Также программе будет принадлежать запрошенная и выделенная память.

**Какой размер памяти программа занимает в каждом случае:**

Без освобождения памяти программа занимает 144 байта и всю свободную память. После освобождения она занимает144 байта + около 1000 байтов (в каждом случае это зависит от исходного кода), для измерения размера вычисляется ближайший к концу модуля адрес конца параграфа и к нему добавляется ещё 16 байт. Также при запросе и успешном выделении дополнительной памяти, программа также занимает и её.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. NOFREE.ASM**

LAB segment

ASSUME cs: LAB, ds: LAB, es: NOTHING, ss: NOTHING

org 100h

START: jmp BEGIN

AVL\_MEMORY\_INFO db "Available memory: $"

EXT\_MEMORY\_INFO db "Extended memory: $"

MCB\_INFO db "MCBs:", 0Dh, 0Ah, "$"

MCB\_NUM\_INFO db "MCB number $"

AREA\_SIZE\_INFO db ", size = $"

OCCUPANT\_INFO db "; occupied by: $"

END\_LINE db 0Dh, 0Ah, "$"

BYTES db " bytes$"

KBYTES db " kbytes", 0Dh, 0Ah, "$"

OWNER\_INFO db 0Dh, 0Ah, "Block is $"

OWNER\_UNKNOWN db "owned by PSP = $"

OWNER\_FREE db "free$"

OWNER\_XMS db "occupied by OS XMS UMB$"

OWNER\_TM db "occupied by driver's top memory$"

OWNER\_DOS db "occupied by MS DOS$"

OWNER\_386CB db "busy by 386MAX UMB$"

OWNER\_386B db "blocked by 386MAX$"

OWNER\_386 db "occupied by 386MAX UMB$"

OCCUPANT\_UNKNOWN db "no info$"

;-------------------------------------------------------------------------------

PRINT\_STRING PROC NEAR ; (DX : String)

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT\_STRING ENDP

DEC\_WORD\_PRINT PROC NEAR ; (AX : short)

push ax

push cx

push dx

push bx

mov bx, 10

xor cx, cx

NUM:

div bx

push dx

xor dx, dx

inc cx

cmp ax, 0h

jnz NUM

PRINT\_NUM:

pop dx

or dl, 30h

mov ah, 02h

int 21h

loop PRINT\_NUM

pop bx

pop dx

pop cx

pop ax

ret

DEC\_WORD\_PRINT ENDP

HEX\_BYTE\_PRINT PROC NEAR ; (AL : byte)

push ax

push bx

push dx

mov ah, 0

mov bl, 10h

div bl

mov dx, ax

mov ah, 02h

cmp dl, 0Ah

jl PRINT

add dl, 07h

PRINT:

add dl, '0'

int 21h;

mov dl, dh

cmp dl, 0Ah

jl PRINT\_EXT

add dl, 07h

PRINT\_EXT:

add dl, '0'

int 21h;

pop dx

pop bx

pop ax

ret

HEX\_BYTE\_PRINT ENDP

HEX\_WORD\_PRINT PROC NEAR ; (AX : short)

push ax

push ax

mov al, ah

call HEX\_BYTE\_PRINT

pop ax

call HEX\_BYTE\_PRINT

pop ax

ret

HEX\_WORD\_PRINT ENDP

;-------------------------------------------------------------------------------

PRINT\_AVL\_MEMORY PROC NEAR

push ax

push bx

push dx

mov dx, offset AVL\_MEMORY\_INFO

call PRINT\_STRING

xor ax, ax

int 12h

xor dx, dx

call DEC\_WORD\_PRINT

mov dx, offset KBYTES

call PRINT\_STRING

pop dx

pop bx

pop ax

ret

PRINT\_AVL\_MEMORY ENDP

PRINT\_EXT\_MEMORY PROC NEAR

push ax

push bx

push dx

mov dx, offset EXT\_MEMORY\_INFO

call PRINT\_STRING

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

mov ah, al

mov al, bl

xor dx, dx

call DEC\_WORD\_PRINT

mov dx, offset KBYTES

call PRINT\_STRING

pop dx

pop bx

pop ax

ret

PRINT\_EXT\_MEMORY ENDP

PRINT\_MCB PROC near

push ax

push bx

push cx

push dx

push es

push si

mov dx, offset MCB\_INFO

call PRINT\_STRING

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

xor cx, cx

push cx

NEXT\_MCB:

pop cx

mov dx, offset MCB\_NUM\_INFO

call PRINT\_STRING

inc cx

mov ax, cx

xor dx, dx

call DEC\_WORD\_PRINT

push cx

OWNER\_START:

mov dx, offset OWNER\_INFO

call PRINT\_STRING

mov ax, es:[1h]

cmp ax, 0h

je PRINT\_FREE

cmp ax, 6h

je PRINT\_XMS

cmp ax, 7h

je PRINT\_TM

cmp ax, 8h

je PRINT\_DOS

cmp ax, 0FFFAh

je PRINT\_386CB

cmp ax, 0FFFDh

je PRINT\_386B

cmp ax, 0FFFEh

je PRINT\_386

mov dx, offset OWNER\_UNKNOWN

call PRINT\_STRING

call HEX\_WORD\_PRINT

jmp AREA\_SIZE

PRINT\_FREE:

mov dx, offset OWNER\_FREE

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_XMS:

mov dx, offset OWNER\_XMS

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_TM:

mov dx, offset OWNER\_TM

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_DOS:

mov dx, offset OWNER\_DOS

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386CB:

mov dx, offset OWNER\_386CB

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386B:

mov dx, offset OWNER\_386B

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386:

mov dx, offset OWNER\_386

call PRINT\_STRING

AREA\_SIZE:

mov dx, offset AREA\_SIZE\_INFO

call PRINT\_STRING

mov ax, es:[3h]

mov bx, 10h

mul bx

call DEC\_WORD\_PRINT

mov dx, offset BYTES

call PRINT\_STRING

mov dx, offset OCCUPANT\_INFO

call PRINT\_STRING

mov ax, es:[8h]

cmp ax, 0h

je NO\_OCCUPANT

mov cx, 8

xor si, si

OCCUPANT:

mov dl, es:[si + 8h]

mov ah, 02h

int 21h

inc si

loop OCCUPANT

jmp PROCEEDING

NO\_OCCUPANT:

mov dx, offset OCCUPANT\_UNKNOWN

call PRINT\_STRING

PROCEEDING:

xor ax, ax

mov al, es:[0h]

cmp al, 5Ah

je ENDING

mov ax, es:[3h]

mov bx, es

add bx, ax

inc bx

mov es, bx

mov dx, offset END\_LINE

call PRINT\_STRING

jmp NEXT\_MCB

ENDING:

pop cx

pop si

pop es

pop dx

pop cx

pop bx

pop ax

ret

PRINT\_MCB ENDP

BEGIN:

call PRINT\_AVL\_MEMORY

call PRINT\_EXT\_MEMORY

call PRINT\_MCB

xor al, al

mov ah, 4Ch

int 21h

PROGRAMM\_END:

LAB ends

end START

**ПРИЛОЖЕНИЕ Б**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. FREE.ASM**

LAB segment

ASSUME cs: LAB, ds: LAB, es: NOTHING, ss: NOTHING

org 100h

START: jmp BEGIN

AVL\_MEMORY\_INFO db "Available memory: $"

EXT\_MEMORY\_INFO db "Extended memory: $"

MCB\_INFO db "MCBs:", 0Dh, 0Ah, "$"

MCB\_NUM\_INFO db "MCB number $"

AREA\_SIZE\_INFO db ", size = $"

OCCUPANT\_INFO db "; occupied by: $"

END\_LINE db 0Dh, 0Ah, "$"

BYTES db " bytes$"

KBYTES db " kbytes", 0Dh, 0Ah, "$"

OWNER\_INFO db 0Dh, 0Ah, "Block is $"

OWNER\_UNKNOWN db "owned by PSP = $"

OWNER\_FREE db "free$"

OWNER\_XMS db "occupied by OS XMS UMB$"

OWNER\_TM db "occupied by driver's top memory$"

OWNER\_DOS db "occupied by MS DOS$"

OWNER\_386CB db "busy by 386MAX UMB$"

OWNER\_386B db "blocked by 386MAX$"

OWNER\_386 db "occupied by 386MAX UMB$"

OCCUPANT\_UNKNOWN db "no info$"

FREEING\_ERROR db "Some error occures during memory freeing.", 0Dh, 0Ah, "$"

;-------------------------------------------------------------------------------

PRINT\_STRING PROC NEAR ; (DX : String)

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT\_STRING ENDP

DEC\_WORD\_PRINT PROC NEAR ; (AX : short)

push ax

push cx

push dx

push bx

mov bx, 10

xor cx, cx

NUM:

div bx

push dx

xor dx, dx

inc cx

cmp ax, 0h

jnz NUM

PRINT\_NUM:

pop dx

or dl, 30h

mov ah, 02h

int 21h

loop PRINT\_NUM

pop bx

pop dx

pop cx

pop ax

ret

DEC\_WORD\_PRINT ENDP

HEX\_BYTE\_PRINT PROC NEAR ; (AL : byte)

push ax

push bx

push dx

mov ah, 0

mov bl, 10h

div bl

mov dx, ax

mov ah, 02h

cmp dl, 0Ah

jl PRINT

add dl, 07h

PRINT:

add dl, '0'

int 21h;

mov dl, dh

cmp dl, 0Ah

jl PRINT\_EXT

add dl, 07h

PRINT\_EXT:

add dl, '0'

int 21h;

pop dx

pop bx

pop ax

ret

HEX\_BYTE\_PRINT ENDP

HEX\_WORD\_PRINT PROC NEAR ; (AX : short)

push ax

push ax

mov al, ah

call HEX\_BYTE\_PRINT

pop ax

call HEX\_BYTE\_PRINT

pop ax

ret

HEX\_WORD\_PRINT ENDP

;-------------------------------------------------------------------------------

FREE\_MEM PROC NEAR

push ax

push bx

push cx

push dx

mov bx, offset PROGRAMM\_END

mov cl, 4

shr bx, cl

add bx, 1

mov ah, 4Ah

int 21h

jnc FREE\_SUCCESS

FREE\_ERROR:

mov dx, offset FREEING\_ERROR

call PRINT\_STRING

FREE\_SUCCESS:

pop dx

pop cx

pop bx

pop ax

ret

FREE\_MEM ENDP

PRINT\_AVL\_MEMORY PROC NEAR

push ax

push bx

push dx

mov dx, offset AVL\_MEMORY\_INFO

call PRINT\_STRING

xor ax, ax

int 12h

xor dx, dx

call DEC\_WORD\_PRINT

mov dx, offset KBYTES

call PRINT\_STRING

pop dx

pop bx

pop ax

ret

PRINT\_AVL\_MEMORY ENDP

PRINT\_EXT\_MEMORY PROC NEAR

push ax

push bx

push dx

mov dx, offset EXT\_MEMORY\_INFO

call PRINT\_STRING

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

mov ah, al

mov al, bl

xor dx, dx

call DEC\_WORD\_PRINT

mov dx, offset KBYTES

call PRINT\_STRING

pop dx

pop bx

pop ax

ret

PRINT\_EXT\_MEMORY ENDP

PRINT\_MCB PROC near

push ax

push bx

push cx

push dx

push es

push si

mov dx, offset MCB\_INFO

call PRINT\_STRING

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

xor cx, cx

push cx

NEXT\_MCB:

pop cx

mov dx, offset MCB\_NUM\_INFO

call PRINT\_STRING

inc cx

mov ax, cx

xor dx, dx

call DEC\_WORD\_PRINT

push cx

OWNER\_START:

mov dx, offset OWNER\_INFO

call PRINT\_STRING

mov ax, es:[1h]

cmp ax, 0h

je PRINT\_FREE

cmp ax, 6h

je PRINT\_XMS

cmp ax, 7h

je PRINT\_TM

cmp ax, 8h

je PRINT\_DOS

cmp ax, 0FFFAh

je PRINT\_386CB

cmp ax, 0FFFDh

je PRINT\_386B

cmp ax, 0FFFEh

je PRINT\_386

mov dx, offset OWNER\_UNKNOWN

call PRINT\_STRING

call HEX\_WORD\_PRINT

jmp AREA\_SIZE

PRINT\_FREE:

mov dx, offset OWNER\_FREE

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_XMS:

mov dx, offset OWNER\_XMS

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_TM:

mov dx, offset OWNER\_TM

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_DOS:

mov dx, offset OWNER\_DOS

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386CB:

mov dx, offset OWNER\_386CB

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386B:

mov dx, offset OWNER\_386B

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386:

mov dx, offset OWNER\_386

call PRINT\_STRING

AREA\_SIZE:

mov dx, offset AREA\_SIZE\_INFO

call PRINT\_STRING

mov ax, es:[3h]

mov bx, 10h

mul bx

call DEC\_WORD\_PRINT

mov dx, offset BYTES

call PRINT\_STRING

mov dx, offset OCCUPANT\_INFO

call PRINT\_STRING

mov ax, es:[8h]

cmp ax, 0h

je NO\_OCCUPANT

mov cx, 8

xor si, si

OCCUPANT:

mov dl, es:[si + 8h]

mov ah, 02h

int 21h

inc si

loop OCCUPANT

jmp PROCEEDING

NO\_OCCUPANT:

mov dx, offset OCCUPANT\_UNKNOWN

call PRINT\_STRING

PROCEEDING:

xor ax, ax

mov al, es:[0h]

cmp al, 5Ah

je ENDING

mov ax, es:[3h]

mov bx, es

add bx, ax

inc bx

mov es, bx

mov dx, offset END\_LINE

call PRINT\_STRING

jmp NEXT\_MCB

ENDING:

pop cx

pop si

pop es

pop dx

pop cx

pop bx

pop ax

ret

PRINT\_MCB ENDP

BEGIN:

call FREE\_MEM

call PRINT\_AVL\_MEMORY

call PRINT\_EXT\_MEMORY

call PRINT\_MCB

xor al, al

mov ah, 4Ch

int 21h

PROGRAMM\_END:

LAB ends

end START

**ПРИЛОЖЕНИЕ В**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. NOFREE64.ASM**

LAB segment

ASSUME cs: LAB, ds: LAB, es: NOTHING, ss: NOTHING

org 100h

START: jmp BEGIN

AVL\_MEMORY\_INFO db "Available memory: $"

EXT\_MEMORY\_INFO db "Extended memory: $"

MCB\_INFO db "MCBs:", 0Dh, 0Ah, "$"

MCB\_NUM\_INFO db "MCB number $"

AREA\_SIZE\_INFO db ", size = $"

OCCUPANT\_INFO db "; occupied by: $"

END\_LINE db 0Dh, 0Ah, "$"

BYTES db " bytes$"

KBYTES db " kbytes", 0Dh, 0Ah, "$"

OWNER\_INFO db 0Dh, 0Ah, "Block is $"

OWNER\_UNKNOWN db "owned by PSP = $"

OWNER\_FREE db "free$"

OWNER\_XMS db "occupied by OS XMS UMB$"

OWNER\_TM db "occupied by driver's top memory$"

OWNER\_DOS db "occupied by MS DOS$"

OWNER\_386CB db "busy by 386MAX UMB$"

OWNER\_386B db "blocked by 386MAX$"

OWNER\_386 db "occupied by 386MAX UMB$"

OCCUPANT\_UNKNOWN db "no info$"

FREEING\_ERROR db "Some error occures during memory freeing.", 0Dh, 0Ah, "$"

ALLOCATING\_ERROR db "Some error occures during memory allocating.", 0Dh, 0Ah, "$"

;-------------------------------------------------------------------------------

PRINT\_STRING PROC NEAR ; (DX : String)

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT\_STRING ENDP

DEC\_WORD\_PRINT PROC NEAR ; (AX : short)

push ax

push cx

push dx

push bx

mov bx, 10

xor cx, cx

NUM:

div bx

push dx

xor dx, dx

inc cx

cmp ax, 0h

jnz NUM

PRINT\_NUM:

pop dx

or dl, 30h

mov ah, 02h

int 21h

loop PRINT\_NUM

pop bx

pop dx

pop cx

pop ax

ret

DEC\_WORD\_PRINT ENDP

HEX\_BYTE\_PRINT PROC NEAR ; (AL : byte)

push ax

push bx

push dx

mov ah, 0

mov bl, 10h

div bl

mov dx, ax

mov ah, 02h

cmp dl, 0Ah

jl PRINT

add dl, 07h

PRINT:

add dl, '0'

int 21h;

mov dl, dh

cmp dl, 0Ah

jl PRINT\_EXT

add dl, 07h

PRINT\_EXT:

add dl, '0'

int 21h;

pop dx

pop bx

pop ax

ret

HEX\_BYTE\_PRINT ENDP

HEX\_WORD\_PRINT PROC NEAR ; (AX : short)

push ax

push ax

mov al, ah

call HEX\_BYTE\_PRINT

pop ax

call HEX\_BYTE\_PRINT

pop ax

ret

HEX\_WORD\_PRINT ENDP

;-------------------------------------------------------------------------------

FREE\_MEM PROC NEAR

push ax

push bx

push cx

push dx

mov bx, offset PROGRAMM\_END

mov cl, 4

shr bx, cl

add bx, 1

mov ah, 4Ah

int 21h

jnc FREE\_SUCCESS

FREE\_ERROR:

mov dx, offset FREEING\_ERROR

call PRINT\_STRING

FREE\_SUCCESS:

pop dx

pop cx

pop bx

pop ax

ret

FREE\_MEM ENDP

ALLOC\_MEM PROC NEAR

push ax

push bx

push cx

push dx

mov bx, 1000h

mov ah, 48h

int 21h

jnc FREE\_SUCCESS

ALLOC\_ERROR:

mov dx, offset ALLOCATING\_ERROR

call PRINT\_STRING

ALLOC\_SUCCESS:

pop dx

pop cx

pop bx

pop ax

ret

ALLOC\_MEM ENDP

PRINT\_AVL\_MEMORY PROC NEAR

push ax

push bx

push dx

mov dx, offset AVL\_MEMORY\_INFO

call PRINT\_STRING

xor ax, ax

int 12h

xor dx, dx

call DEC\_WORD\_PRINT

mov dx, offset KBYTES

call PRINT\_STRING

pop dx

pop bx

pop ax

ret

PRINT\_AVL\_MEMORY ENDP

PRINT\_EXT\_MEMORY PROC NEAR

push ax

push bx

push dx

mov dx, offset EXT\_MEMORY\_INFO

call PRINT\_STRING

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

mov ah, al

mov al, bl

xor dx, dx

call DEC\_WORD\_PRINT

mov dx, offset KBYTES

call PRINT\_STRING

pop dx

pop bx

pop ax

ret

PRINT\_EXT\_MEMORY ENDP

PRINT\_MCB PROC near

push ax

push bx

push cx

push dx

push es

push si

mov dx, offset MCB\_INFO

call PRINT\_STRING

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

xor cx, cx

push cx

NEXT\_MCB:

pop cx

mov dx, offset MCB\_NUM\_INFO

call PRINT\_STRING

inc cx

mov ax, cx

xor dx, dx

call DEC\_WORD\_PRINT

push cx

OWNER\_START:

mov dx, offset OWNER\_INFO

call PRINT\_STRING

mov ax, es:[1h]

cmp ax, 0h

je PRINT\_FREE

cmp ax, 6h

je PRINT\_XMS

cmp ax, 7h

je PRINT\_TM

cmp ax, 8h

je PRINT\_DOS

cmp ax, 0FFFAh

je PRINT\_386CB

cmp ax, 0FFFDh

je PRINT\_386B

cmp ax, 0FFFEh

je PRINT\_386

mov dx, offset OWNER\_UNKNOWN

call PRINT\_STRING

call HEX\_WORD\_PRINT

jmp AREA\_SIZE

PRINT\_FREE:

mov dx, offset OWNER\_FREE

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_XMS:

mov dx, offset OWNER\_XMS

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_TM:

mov dx, offset OWNER\_TM

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_DOS:

mov dx, offset OWNER\_DOS

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386CB:

mov dx, offset OWNER\_386CB

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386B:

mov dx, offset OWNER\_386B

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386:

mov dx, offset OWNER\_386

call PRINT\_STRING

AREA\_SIZE:

mov dx, offset AREA\_SIZE\_INFO

call PRINT\_STRING

mov ax, es:[3h]

mov bx, 10h

mul bx

call DEC\_WORD\_PRINT

mov dx, offset BYTES

call PRINT\_STRING

mov dx, offset OCCUPANT\_INFO

call PRINT\_STRING

mov ax, es:[8h]

cmp ax, 0h

je NO\_OCCUPANT

mov cx, 8

xor si, si

OCCUPANT:

mov dl, es:[si + 8h]

mov ah, 02h

int 21h

inc si

loop OCCUPANT

jmp PROCEEDING

NO\_OCCUPANT:

mov dx, offset OCCUPANT\_UNKNOWN

call PRINT\_STRING

PROCEEDING:

xor ax, ax

mov al, es:[0h]

cmp al, 5Ah

je ENDING

mov ax, es:[3h]

mov bx, es

add bx, ax

inc bx

mov es, bx

mov dx, offset END\_LINE

call PRINT\_STRING

jmp NEXT\_MCB

ENDING:

pop cx

pop si

pop es

pop dx

pop cx

pop bx

pop ax

ret

PRINT\_MCB ENDP

BEGIN:

call FREE\_MEM

call ALLOC\_MEM

call PRINT\_AVL\_MEMORY

call PRINT\_EXT\_MEMORY

call PRINT\_MCB

xor al, al

mov ah, 4Ch

int 21h

PROGRAMM\_END:

LAB ends

end START

**ПРИЛОЖЕНИЕ Г**

**ИСХОДНЫЙ КОД ПРОГРАММЫ. FREE64.ASM**

LAB segment

ASSUME cs: LAB, ds: LAB, es: NOTHING, ss: NOTHING

org 100h

START: jmp BEGIN

AVL\_MEMORY\_INFO db "Available memory: $"

EXT\_MEMORY\_INFO db "Extended memory: $"

MCB\_INFO db "MCBs:", 0Dh, 0Ah, "$"

MCB\_NUM\_INFO db "MCB number $"

AREA\_SIZE\_INFO db ", size = $"

OCCUPANT\_INFO db "; occupied by: $"

END\_LINE db 0Dh, 0Ah, "$"

BYTES db " bytes$"

KBYTES db " kbytes", 0Dh, 0Ah, "$"

OWNER\_INFO db 0Dh, 0Ah, "Block is $"

OWNER\_UNKNOWN db "owned by PSP = $"

OWNER\_FREE db "free$"

OWNER\_XMS db "occupied by OS XMS UMB$"

OWNER\_TM db "occupied by driver's top memory$"

OWNER\_DOS db "occupied by MS DOS$"

OWNER\_386CB db "busy by 386MAX UMB$"

OWNER\_386B db "blocked by 386MAX$"

OWNER\_386 db "occupied by 386MAX UMB$"

OCCUPANT\_UNKNOWN db "no info$"

FREEING\_ERROR db "Some error occures during memory freeing.", 0Dh, 0Ah, "$"

ALLOCATING\_ERROR db "Some error occures during memory allocating.", 0Dh, 0Ah, "$"

;-------------------------------------------------------------------------------

PRINT\_STRING PROC NEAR ; (DX : String)

push ax

mov ah, 09h

int 21h

pop ax

ret

PRINT\_STRING ENDP

DEC\_WORD\_PRINT PROC NEAR ; (AX : short)

push ax

push cx

push dx

push bx

mov bx, 10

xor cx, cx

NUM:

div bx

push dx

xor dx, dx

inc cx

cmp ax, 0h

jnz NUM

PRINT\_NUM:

pop dx

or dl, 30h

mov ah, 02h

int 21h

loop PRINT\_NUM

pop bx

pop dx

pop cx

pop ax

ret

DEC\_WORD\_PRINT ENDP

HEX\_BYTE\_PRINT PROC NEAR ; (AL : byte)

push ax

push bx

push dx

mov ah, 0

mov bl, 10h

div bl

mov dx, ax

mov ah, 02h

cmp dl, 0Ah

jl PRINT

add dl, 07h

PRINT:

add dl, '0'

int 21h;

mov dl, dh

cmp dl, 0Ah

jl PRINT\_EXT

add dl, 07h

PRINT\_EXT:

add dl, '0'

int 21h;

pop dx

pop bx

pop ax

ret

HEX\_BYTE\_PRINT ENDP

HEX\_WORD\_PRINT PROC NEAR ; (AX : short)

push ax

push ax

mov al, ah

call HEX\_BYTE\_PRINT

pop ax

call HEX\_BYTE\_PRINT

pop ax

ret

HEX\_WORD\_PRINT ENDP

;-------------------------------------------------------------------------------

FREE\_MEM PROC NEAR

push ax

push bx

push cx

push dx

mov bx, offset PROGRAMM\_END

mov cl, 4

shr bx, cl

add bx, 1

mov ah, 4Ah

int 21h

jnc FREE\_SUCCESS

FREE\_ERROR:

mov dx, offset FREEING\_ERROR

call PRINT\_STRING

FREE\_SUCCESS:

pop dx

pop cx

pop bx

pop ax

ret

FREE\_MEM ENDP

ALLOC\_MEM PROC NEAR

push ax

push bx

push cx

push dx

mov bx, 1000h

mov ah, 48h

int 21h

jnc FREE\_SUCCESS

ALLOC\_ERROR:

mov dx, offset ALLOCATING\_ERROR

call PRINT\_STRING

ALLOC\_SUCCESS:

pop dx

pop cx

pop bx

pop ax

ret

ALLOC\_MEM ENDP

PRINT\_AVL\_MEMORY PROC NEAR

push ax

push bx

push dx

mov dx, offset AVL\_MEMORY\_INFO

call PRINT\_STRING

xor ax, ax

int 12h

xor dx, dx

call DEC\_WORD\_PRINT

mov dx, offset KBYTES

call PRINT\_STRING

pop dx

pop bx

pop ax

ret

PRINT\_AVL\_MEMORY ENDP

PRINT\_EXT\_MEMORY PROC NEAR

push ax

push bx

push dx

mov dx, offset EXT\_MEMORY\_INFO

call PRINT\_STRING

mov al, 30h

out 70h, al

in al, 71h

mov bl, al

mov al, 31h

out 70h, al

in al, 71h

mov ah, al

mov al, bl

xor dx, dx

call DEC\_WORD\_PRINT

mov dx, offset KBYTES

call PRINT\_STRING

pop dx

pop bx

pop ax

ret

PRINT\_EXT\_MEMORY ENDP

PRINT\_MCB PROC near

push ax

push bx

push cx

push dx

push es

push si

mov dx, offset MCB\_INFO

call PRINT\_STRING

mov ah, 52h

int 21h

mov ax, es:[bx-2]

mov es, ax

xor cx, cx

push cx

NEXT\_MCB:

pop cx

mov dx, offset MCB\_NUM\_INFO

call PRINT\_STRING

inc cx

mov ax, cx

xor dx, dx

call DEC\_WORD\_PRINT

push cx

OWNER\_START:

mov dx, offset OWNER\_INFO

call PRINT\_STRING

mov ax, es:[1h]

cmp ax, 0h

je PRINT\_FREE

cmp ax, 6h

je PRINT\_XMS

cmp ax, 7h

je PRINT\_TM

cmp ax, 8h

je PRINT\_DOS

cmp ax, 0FFFAh

je PRINT\_386CB

cmp ax, 0FFFDh

je PRINT\_386B

cmp ax, 0FFFEh

je PRINT\_386

mov dx, offset OWNER\_UNKNOWN

call PRINT\_STRING

call HEX\_WORD\_PRINT

jmp AREA\_SIZE

PRINT\_FREE:

mov dx, offset OWNER\_FREE

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_XMS:

mov dx, offset OWNER\_XMS

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_TM:

mov dx, offset OWNER\_TM

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_DOS:

mov dx, offset OWNER\_DOS

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386CB:

mov dx, offset OWNER\_386CB

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386B:

mov dx, offset OWNER\_386B

call PRINT\_STRING

jmp AREA\_SIZE

PRINT\_386:

mov dx, offset OWNER\_386

call PRINT\_STRING

AREA\_SIZE:

mov dx, offset AREA\_SIZE\_INFO

call PRINT\_STRING

mov ax, es:[3h]

mov bx, 10h

mul bx

call DEC\_WORD\_PRINT

mov dx, offset BYTES

call PRINT\_STRING

mov dx, offset OCCUPANT\_INFO

call PRINT\_STRING

mov ax, es:[8h]

cmp ax, 0h

je NO\_OCCUPANT

mov cx, 8

xor si, si

OCCUPANT:

mov dl, es:[si + 8h]

mov ah, 02h

int 21h

inc si

loop OCCUPANT

jmp PROCEEDING

NO\_OCCUPANT:

mov dx, offset OCCUPANT\_UNKNOWN

call PRINT\_STRING

PROCEEDING:

xor ax, ax

mov al, es:[0h]

cmp al, 5Ah

je ENDING

mov ax, es:[3h]

mov bx, es

add bx, ax

inc bx

mov es, bx

mov dx, offset END\_LINE

call PRINT\_STRING

jmp NEXT\_MCB

ENDING:

pop cx

pop si

pop es

pop dx

pop cx

pop bx

pop ax

ret

PRINT\_MCB ENDP

BEGIN:

call ALLOC\_MEM

call FREE\_MEM

call PRINT\_AVL\_MEMORY

call PRINT\_EXT\_MEMORY

call PRINT\_MCB

xor al, al

mov ah, 4Ch

int 21h

PROGRAMM\_END:

LAB ends

end START