**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Операционные системы»**

Тема: **Исследование организации управления основной памятью**

|  |  |  |
| --- | --- | --- |
| Студент гр.0382 |  | Литягин С.М. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2022

## Цель работы.

Исследование структуры данных и работы функций управления памятью ядра операционной системы.

## Задание.

1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

- количество доступной памяти;

- размер расширенной памяти;

- выводит цепочку блоков управления памятью.

Адреса при выводе представляются шестнадцатеричными числами. Объем памяти функциями управления выводится в параграфах. Необходимо преобразовать его в байты и выводить в виде десятичных чисел. Последние восемь байт МСВ выводятся как символы, не следует преобразовывать их в шестнадцатеричные числа.

Запустите программу и внимательно оцените результаты. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

2. Измените программу таким образом, чтобы она освобождала память, которую она не занимает. Для этого используйте функцию 4Ah прерывания 21h (пример в разделе “Использование функции 4Ah”). Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущем шаге. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

3. Измените программу еще раз таким образом, чтобы после освобождения памяти, программа запрашивала 64Кб памяти функцией 48h прерывания 21h. Повторите эксперимент, запустив модифицированную программу. Сравните выходные данные с результатами, полученными на предыдущем шаге. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

4. Измените первоначальный вариант программы, запросив 64Кб памяти функцией 48h прерывания 21h до освобождения памяти. Обязательно обрабатывайте завершение функций ядра, проверяя флаг CF. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

5. Оцените результаты, полученные на предыдущих шагах. Ответьте на контрольные вопросы и оформите отчет.

## Выполнение работы.

1. За основу был взят шаблон .COM модуля из методического пособия, в котором реализованы процедуры преобразования двоичных кодов в символы шестнадцатеричных и десятичных чисел. В программу были добавлены следующие процедуры:

- процедура AM для вывода количества доступной памяти в байтах;

- процедура EM для вывода количества расширенной памяти в байтах;

- процедура MCB для вывода цепочки блоков управления памятью.

Результаты шага представлены на рисунке 1.

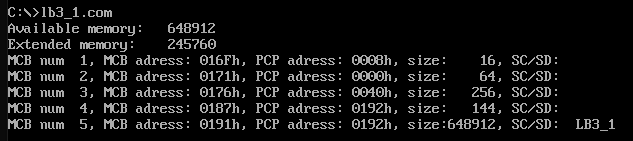


Рисунок 1 – Результаты первого шага

Оценивая полученный результат, делаем вывод, что программа занимает всю доступную память.

2. Для выполнения данного шага в программу была добавлена процедура FREE\_MEM. Результаты шага представлены на рисунке 2. На нем видно, что теперь программа занимает лишь ту область памяти, что необходима для ее хранения.

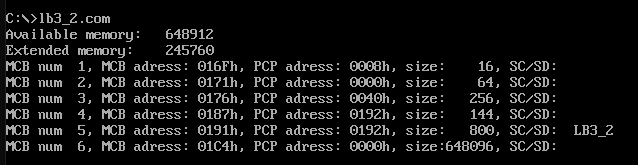


Рисунок 2 – Результаты второго шага

3. Для выполнения данного шага в программу была добавлена процедура GET\_MEM. Процедура вызывается после освобождения памяти процедурой FREE\_MEM. Результаты представлены на рисунке 3.

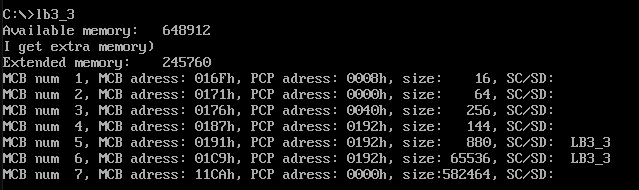


Рисунок 3 – Результаты третьего шага

В результате выполнения этого шага видно, что программе, после освобождения памяти, был выделен еще один блок памяти в размере 64Кб.

4. На этом шаге процедура GET\_MEM вызывается до освобождения памяти процедурой FREE\_MEM. Результаты представлены на рисунке 4.

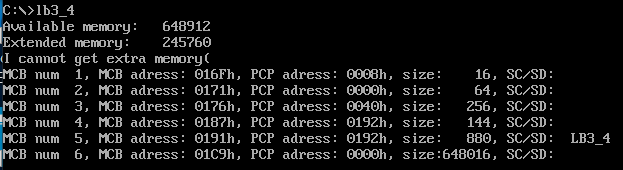


Рисунок 4 – Результаты четвертого шага

В третьей строке вывода сообщается о невозможности получения дополнительной памяти. Это действительно так, поскольку до освобождения памяти программа занимала всю доступную память.

Исходный код программы см. в приложении А.

## Ответы на вопросы.

1. Что означает “доступный объем памяти”?

Это объем памяти, выделенный управляющей программой для модуля

2. Где MCB блок вашей программы в списке?

Блок MCB программы в списке – это тот, у которого в графе SC/SD написано название программы (на третьем шаге таких блока два, т.к. выделена доп. память). На шаге 1 – это последний в списке блок, на шаге 2 – блок памяти программы предпоследний, за ним – блок свободной памяти, на шаге 3 – блоки 5 и 6, последний блок – свободный, на шаге 4 – предпоследний блок, за ним – блок свободной памяти.

3. Какой размер памяти занимает программа в каждом случае?

На шаге 1 – 648912 байта, на шаге 2 – 800 байт, на шаге 3 – 66416 байт (необходимая память после освобождения + дополнительно выделенная), шаг 4 – 880 байт.

## Выводы.

В ходе работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

# Приложение А Исходный код программы

Название файла: lb3\_1.asm

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: jmp BEGIN

; ДАННЫЕ

AVAIL\_MEM db 'Available memory: ', 0DH, 0AH, '$'

EXTEND\_MEM db 'Extended memory: ', 0DH, 0AH, '$'

MCB\_I db 'MCB num , MCB adress: h, PCP adress: h, size: , SC/SD: ', 0DH, '$'

; ПРОЦЕДУРЫ

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 16-го 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

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

WRD\_TO\_HEX PROC near

; перевод в 16 с/с 16-ти разрядного числа

; в AX - число, в DI - адрес последнего 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

; перевод в 10 с/с, в SI - адрес поля младшей цифры

push AX

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

pop AX

ret

BYTE\_TO\_DEC ENDP

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

WORD\_TO\_DEC PROC near

push AX

push BX

push DX

push CX

push SI

mov BX, 10h

mul BX

mov BX, 0Ah

division:

div BX

or DX, 30h

mov [SI], DL

dec SI

xor DX, DX

cmp AX, 0h

jne division

pop SI

pop CX

pop DX

pop BX

pop AX

ret

WORD\_TO\_DEC ENDP

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

PRINT PROC near

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

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

PRINT\_SYM PROC near

push AX

mov AH, 02h

int 21h

pop AX

ret

PRINT\_SYM ENDP

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

AM PROC near

push AX

push BX

push SI

xor AX, AX

mov AH, 4Ah

mov BX, 0FFFFh

int 21h

mov AX, BX

mov SI, offset AVAIL\_MEM

add SI, 25

call WORD\_TO\_DEC

mov DX, offset AVAIL\_MEM

call PRINT

pop SI

pop BX

pop AX

ret

AM ENDP

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

EM PROC near

push AX

push BX

push SI

xor AX, AX

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

mov SI, offset EXTEND\_MEM

add SI, 25

call WORD\_TO\_DEC

mov DX, offset EXTEND\_MEM

call PRINT

pop SI

pop BX

pop AX

ret

EM ENDP

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

MCB PROC near

push AX

push BX

push CX

push DX

push ES

push SI

xor AX, AX

mov AH, 52h

int 21h

mov AX, ES:[BX-2]

mov ES, AX

mov CL, 1

loop\_mcb:

mov AL, CL

mov SI, offset MCB\_I

add SI, 9

call BYTE\_TO\_DEC

mov AX, ES

mov DI, offset MCB\_I

add DI, 27

call WRD\_TO\_HEX

mov AX, ES:[01h]

mov DI, offset MCB\_I

add DI, 46

call WRD\_TO\_HEX

mov AX, ES:[03h]

add SI, 52

call WORD\_TO\_DEC

mov BX, 8

push CX

mov CX, 7

add SI, 11

loop\_sc\_sd:

mov DX, ES:[BX]

mov DS:[SI], DX

inc BX

inc SI

loop loop\_sc\_sd

mov DX, offset MCB\_I

call PRINT

mov AH, ES:[0]

cmp AH, 5Ah

je end\_mcb

mov BX, ES:[3]

mov AX, ES

add AX, BX

inc AX

mov ES, AX

pop CX

inc CL

jmp loop\_mcb

end\_mcb:

pop SI

pop ES

pop DX

pop CX

pop BX

pop AX

ret

MCB ENDP

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

BEGIN:

call AM

call EM

call MCB

xor AL, AL

mov AH, 4Ch

int 21h

TESTPC ENDS

END START

Название файла: lb3\_2.asm

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: jmp BEGIN

; ДАННЫЕ

AVAIL\_MEM db 'Available memory: ', 0DH, 0AH, '$'

EXTEND\_MEM db 'Extended memory: ', 0DH, 0AH, '$'

MCB\_I db 'MCB num , MCB adress: h, PCP adress: h, size: , SC/SD: ', 0DH, '$'

ERROR db 'I can't get extra memory(', 0DH, 0AH, '$'

SUCCEC db 'I get extra memory)', 0DH, 0AH, '$'

; ПРОЦЕДУРЫ

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 16-го 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

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

WRD\_TO\_HEX PROC near

; перевод в 16 с/с 16-ти разрядного числа

; в AX - число, в DI - адрес последнего 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

; перевод в 10 с/с, в SI - адрес поля младшей цифры

push AX

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

pop AX

ret

BYTE\_TO\_DEC ENDP

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

WORD\_TO\_DEC PROC near

push AX

push BX

push DX

push CX

push SI

mov BX, 10h

mul BX

mov BX, 0Ah

division:

div BX

or DX, 30h

mov [SI], DL

dec SI

xor DX, DX

cmp AX, 0h

jne division

pop SI

pop CX

pop DX

pop BX

pop AX

ret

WORD\_TO\_DEC ENDP

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

PRINT PROC near

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

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

PRINT\_SYM PROC near

push AX

mov AH, 02h

int 21h

pop AX

ret

PRINT\_SYM ENDP

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

AM PROC near

push AX

push BX

push SI

xor AX, AX

mov AH, 4Ah

mov BX, 0FFFFh

int 21h

mov AX, BX

mov SI, offset AVAIL\_MEM

add SI, 25

call WORD\_TO\_DEC

mov DX, offset AVAIL\_MEM

call PRINT

pop SI

pop BX

pop AX

ret

AM ENDP

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

EM PROC near

push AX

push BX

push SI

xor AX, AX

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

mov SI, offset EXTEND\_MEM

add SI, 25

call WORD\_TO\_DEC

mov DX, offset EXTEND\_MEM

call PRINT

pop SI

pop BX

pop AX

ret

EM ENDP

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

MCB PROC near

push AX

push BX

push CX

push DX

push ES

push SI

xor AX, AX

mov AH, 52h

int 21h

mov AX, ES:[BX-2]

mov ES, AX

mov CL, 1

loop\_mcb:

mov AL, CL

mov SI, offset MCB\_I

add SI, 9

call BYTE\_TO\_DEC

mov AX, ES

mov DI, offset MCB\_I

add DI, 27

call WRD\_TO\_HEX

mov AX, ES:[01h]

mov DI, offset MCB\_I

add DI, 46

call WRD\_TO\_HEX

mov AX, ES:[03h]

add SI, 52

call WORD\_TO\_DEC

mov BX, 8

push CX

mov CX, 7

add SI, 11

loop\_sc\_sd:

mov DX, ES:[BX]

mov DS:[SI], DX

inc BX

inc SI

loop loop\_sc\_sd

mov DX, offset MCB\_I

call PRINT

mov AH, ES:[0]

cmp AH, 5Ah

je end\_mcb

mov BX, ES:[3]

mov AX, ES

add AX, BX

inc AX

mov ES, AX

pop CX

inc CL

jmp loop\_mcb

end\_mcb:

pop SI

pop ES

pop DX

pop CX

pop BX

pop AX

ret

MCB ENDP

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

FREE\_MEM PROC NEAR

push AX

push BX

push DX

lea AX, end\_programm

mov BX, 10h

xor DX, DX

div BX

inc AX

mov BX, AX

xor AX, AX

mov AH, 4Ah

int 21h

pop DX

pop BX

pop AX

ret

FREE\_MEM ENDP

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

GET\_MEM PROC near

mov BX, 1000h

xor AX, AX

mov AH, 48h

int 21h

GET\_MEM ENDP

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

BEGIN:

call AM

call FREE\_MEM

call EM

call MCB

xor AL, AL

mov AH, 4Ch

int 21h

end\_programm:

TESTPC ENDS

END START

Название файла: lb3\_3.asm

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: jmp BEGIN

; ДАННЫЕ

AVAIL\_MEM db 'Available memory: ', 0DH, 0AH, '$'

EXTEND\_MEM db 'Extended memory: ', 0DH, 0AH, '$'

MCB\_I db 'MCB num , MCB adress: h, PCP adress: h, size: , SC/SD: ', 0DH, '$'

ERROR db 'I cannot get extra memory(', 0DH, 0AH, '$'

SUCCESS db 'I get extra memory)', 0DH, 0AH, '$'

; ПРОЦЕДУРЫ

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 16-го 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

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

WRD\_TO\_HEX PROC near

; перевод в 16 с/с 16-ти разрядного числа

; в AX - число, в DI - адрес последнего 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

; перевод в 10 с/с, в SI - адрес поля младшей цифры

push AX

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

pop AX

ret

BYTE\_TO\_DEC ENDP

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

WORD\_TO\_DEC PROC near

push AX

push BX

push DX

push CX

push SI

mov BX, 10h

mul BX

mov BX, 0Ah

division:

div BX

or DX, 30h

mov [SI], DL

dec SI

xor DX, DX

cmp AX, 0h

jne division

pop SI

pop CX

pop DX

pop BX

pop AX

ret

WORD\_TO\_DEC ENDP

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

PRINT PROC near

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

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

PRINT\_SYM PROC near

push AX

mov AH, 02h

int 21h

pop AX

ret

PRINT\_SYM ENDP

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

AM PROC near

push AX

push BX

push SI

xor AX, AX

mov AH, 4Ah

mov BX, 0FFFFh

int 21h

mov AX, BX

mov SI, offset AVAIL\_MEM

add SI, 25

call WORD\_TO\_DEC

mov DX, offset AVAIL\_MEM

call PRINT

pop SI

pop BX

pop AX

ret

AM ENDP

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

EM PROC near

push AX

push BX

push SI

xor AX, AX

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

mov SI, offset EXTEND\_MEM

add SI, 25

call WORD\_TO\_DEC

mov DX, offset EXTEND\_MEM

call PRINT

pop SI

pop BX

pop AX

ret

EM ENDP

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

MCB PROC near

push AX

push BX

push CX

push DX

push ES

push SI

xor AX, AX

mov AH, 52h

int 21h

mov AX, ES:[BX-2]

mov ES, AX

mov CL, 1

loop\_mcb:

mov AL, CL

mov SI, offset MCB\_I

add SI, 9

call BYTE\_TO\_DEC

mov AX, ES

mov DI, offset MCB\_I

add DI, 27

call WRD\_TO\_HEX

mov AX, ES:[01h]

mov DI, offset MCB\_I

add DI, 46

call WRD\_TO\_HEX

mov AX, ES:[03h]

add SI, 52

call WORD\_TO\_DEC

mov BX, 8

push CX

mov CX, 7

add SI, 11

loop\_sc\_sd:

mov DX, ES:[BX]

mov DS:[SI], DX

inc BX

inc SI

loop loop\_sc\_sd

mov DX, offset MCB\_I

call PRINT

mov AH, ES:[0]

cmp AH, 5Ah

je end\_mcb

mov BX, ES:[3]

mov AX, ES

add AX, BX

inc AX

mov ES, AX

pop CX

inc CL

jmp loop\_mcb

end\_mcb:

pop SI

pop ES

pop DX

pop CX

pop BX

pop AX

ret

MCB ENDP

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

FREE\_MEM PROC NEAR

push AX

push BX

push DX

lea AX, end\_programm

mov BX, 10h

xor DX, DX

div BX

inc AX

mov BX, AX

xor AX, AX

mov AH, 4Ah

int 21h

pop DX

pop BX

pop AX

ret

FREE\_MEM ENDP

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

GET\_MEM PROC near

push AX

push BX

push DX

mov BX, 1000h

xor AX, AX

mov AH, 48h

int 21h

jc CF\_ERROR

mov DX, offset SUCCESS

call PRINT

jmp end\_gm

CF\_ERROR:

mov DX, offset ERROR

call PRINT

jmp end\_gm

end\_gm:

pop DX

pop BX

pop AX

ret

GET\_MEM ENDP

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

BEGIN:

call AM

call EM

call FREE\_MEM

call GET\_MEM

call MCB

xor AL, AL

mov AH, 4Ch

int 21h

end\_programm:

TESTPC ENDS

END START

Название файла: lb3\_4.asm

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: jmp BEGIN

; ДАННЫЕ

AVAIL\_MEM db 'Available memory: ', 0DH, 0AH, '$'

EXTEND\_MEM db 'Extended memory: ', 0DH, 0AH, '$'

MCB\_I db 'MCB num , MCB adress: h, PCP adress: h, size: , SC/SD: ', 0DH, '$'

ERROR db 'I cannot get extra memory(', 0DH, 0AH, '$'

SUCCESS db 'I get extra memory)', 0DH, 0AH, '$'

; ПРОЦЕДУРЫ

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 16-го 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

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

WRD\_TO\_HEX PROC near

; перевод в 16 с/с 16-ти разрядного числа

; в AX - число, в DI - адрес последнего 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

; перевод в 10 с/с, в SI - адрес поля младшей цифры

push AX

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

pop AX

ret

BYTE\_TO\_DEC ENDP

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

WORD\_TO\_DEC PROC near

push AX

push BX

push DX

push CX

push SI

mov BX, 10h

mul BX

mov BX, 0Ah

division:

div BX

or DX, 30h

mov [SI], DL

dec SI

xor DX, DX

cmp AX, 0h

jne division

pop SI

pop CX

pop DX

pop BX

pop AX

ret

WORD\_TO\_DEC ENDP

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

PRINT PROC near

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

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

PRINT\_SYM PROC near

push AX

mov AH, 02h

int 21h

pop AX

ret

PRINT\_SYM ENDP

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

AM PROC near

push AX

push BX

push SI

xor AX, AX

mov AH, 4Ah

mov BX, 0FFFFh

int 21h

mov AX, BX

mov SI, offset AVAIL\_MEM

add SI, 25

call WORD\_TO\_DEC

mov DX, offset AVAIL\_MEM

call PRINT

pop SI

pop BX

pop AX

ret

AM ENDP

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

EM PROC near

push AX

push BX

push SI

xor AX, AX

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

mov SI, offset EXTEND\_MEM

add SI, 25

call WORD\_TO\_DEC

mov DX, offset EXTEND\_MEM

call PRINT

pop SI

pop BX

pop AX

ret

EM ENDP

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

MCB PROC near

push AX

push BX

push CX

push DX

push ES

push SI

xor AX, AX

mov AH, 52h

int 21h

mov AX, ES:[BX-2]

mov ES, AX

mov CL, 1

loop\_mcb:

mov AL, CL

mov SI, offset MCB\_I

add SI, 9

call BYTE\_TO\_DEC

mov AX, ES

mov DI, offset MCB\_I

add DI, 27

call WRD\_TO\_HEX

mov AX, ES:[01h]

mov DI, offset MCB\_I

add DI, 46

call WRD\_TO\_HEX

mov AX, ES:[03h]

add SI, 52

call WORD\_TO\_DEC

mov BX, 8

push CX

mov CX, 7

add SI, 11

loop\_sc\_sd:

mov DX, ES:[BX]

mov DS:[SI], DX

inc BX

inc SI

loop loop\_sc\_sd

mov DX, offset MCB\_I

call PRINT

mov AH, ES:[0]

cmp AH, 5Ah

je end\_mcb

mov BX, ES:[3]

mov AX, ES

add AX, BX

inc AX

mov ES, AX

pop CX

inc CL

jmp loop\_mcb

end\_mcb:

pop SI

pop ES

pop DX

pop CX

pop BX

pop AX

ret

MCB ENDP

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

FREE\_MEM PROC NEAR

push AX

push BX

push DX

lea AX, end\_programm

mov BX, 10h

xor DX, DX

div BX

inc AX

mov BX, AX

xor AX, AX

mov AH, 4Ah

int 21h

pop DX

pop BX

pop AX

ret

FREE\_MEM ENDP

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

GET\_MEM PROC near

push AX

push BX

push DX

mov BX, 1000h

xor AX, AX

mov AH, 48h

int 21h

jc CF\_ERROR

mov DX, offset SUCCESS

call PRINT

jmp end\_gm

CF\_ERROR:

mov DX, offset ERROR

call PRINT

jmp end\_gm

end\_gm:

pop DX

pop BX

pop AX

ret

GET\_MEM ENDP

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

BEGIN:

call AM

call EM

call GET\_MEM

call FREE\_MEM

call MCB

xor AL, AL

mov AH, 4Ch

int 21h

end\_programm:

TESTPC ENDS

END START