

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(ка) гр. 9382

Русинов Д.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованный в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается нестраничная память и способ управления динамическими разделами. Для реализации управления памятью в этом случае строится список занятых и свободных участков памяти. Функции ядра, обеспечивающие управление основной памятью, просматривают и преобразуют этот список.

В лабораторной работе исследуются структуры данных и работа функций управления памятью ядра операционной системы.

Задание.

Шаг 1. Для выполнения лабораторной работы 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.

Адреса при выводе представляются шестнадцатеричными числами. Объем памяти функциями управления памятью выводится в параграфах. Необходимо преобразовать его в байты и выводить в виде десятичных чисел. Последние восемь байт MSB 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.

Шаг 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.

Шаг 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 предыдущих шагах. Сохраните результаты, полученные программой, и включите их в отчет в виде скриншота.

Шаг 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.

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

Необходимые сведения для составления программы

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

МСВ имеет следующую структуру:

Смещение	Длина поля (байт)	Содержимое поля
00h	1	тип MCB: 5Ah, если последний в списке, 4Dh, если не последний
01h	2	Сегментный адрес PSP владельца участка памяти, либо 0000h - свободный участок, 0006h - участок принадлежит драйверу OS XMS UMB 0007h - участок является исключенной верхней памятью драйверов 0008h - участок принадлежит MS DOS FFFAh - участок занят управляющим блоком 386MAX UMB FFFDh - участок заблокирован 386MAX FFFEh - участок принадлежит 386MAX UMB
03h	2	Размер участка в параграфах
05h	3	Зарезервирован
08h	8	"SC" - если участок принадлежит MS DOS, то в нем системный код "SD" - если участок принадлежит MS DOS, то в нем системные данные

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). Доступ к указателю на эту структуру можно получить используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX—2] и есть адрес самого первого MCB.

Размер расширенной памяти находится в ячейках 3011, 3111 CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить обращаясь к ячейкам CMOS следующим образом:

```

mov AL,30h ; запись адреса ячейки CMOS
out 70h,AL
in AL,71h ; чтение младшего байта
mov BL,AL ; размера расширенной памяти
mov AL,31h ; запись адреса ячейки CMOS
out 70h,AL

```

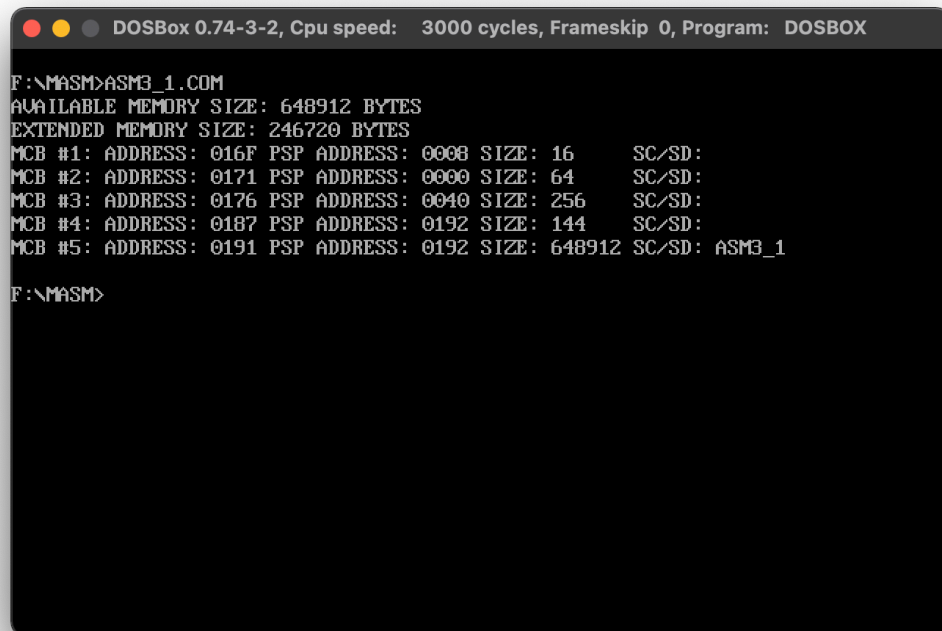
in AL,7lh ; чтение старшего байта
; размера расширенной памяти

Выполнение работы.

1. Был написан модуль типа .COM. Данный модуль выполняет следующие действия:

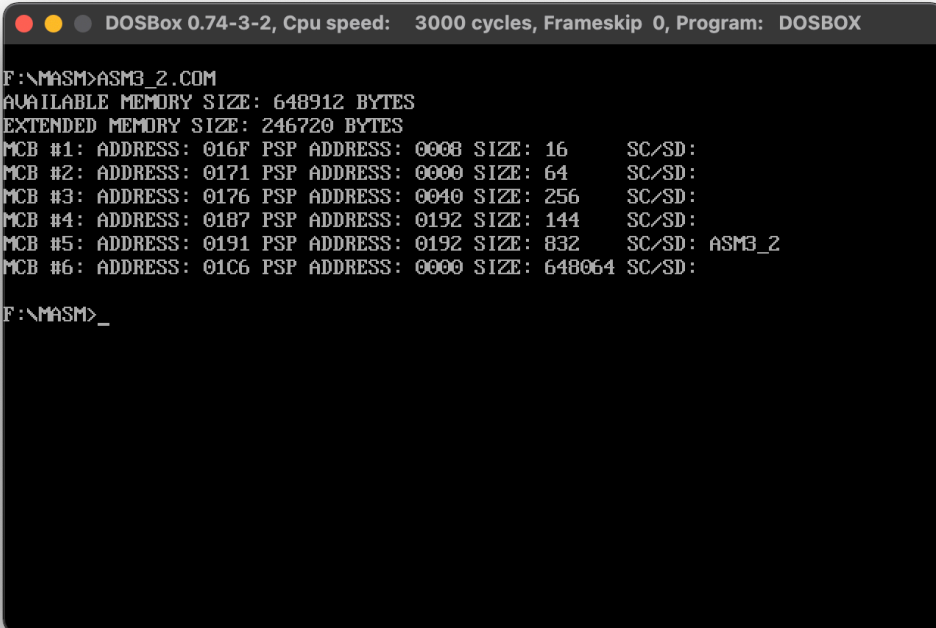
- a) Печать количества доступной памяти
- b) Печать размера расширенной памяти
- c) Печать последовательности блоков управления памятью

Адреса представлены в шестнадцатеричном виде. Размер памяти представлен в десятичной СС. При выводе МСВ записи печатаются ее последние 8 байт в символьном виде.



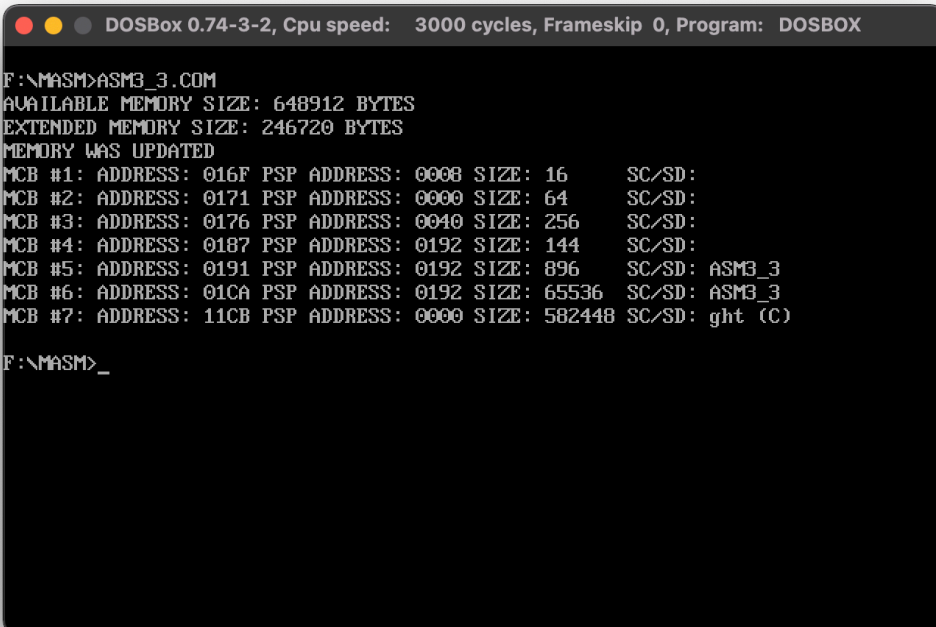
```
DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
F:\MASM>ASM3_1.COM
AVAILABLE MEMORY SIZE: 648912 BYTES
EXTENDED MEMORY SIZE: 246720 BYTES
MCB #1: ADDRESS: 016F PSP ADDRESS: 0000 SIZE: 16 SC/SD:
MCB #2: ADDRESS: 0171 PSP ADDRESS: 0000 SIZE: 64 SC/SD:
MCB #3: ADDRESS: 0176 PSP ADDRESS: 0040 SIZE: 256 SC/SD:
MCB #4: ADDRESS: 0187 PSP ADDRESS: 0192 SIZE: 144 SC/SD:
MCB #5: ADDRESS: 0191 PSP ADDRESS: 0192 SIZE: 648912 SC/SD: ASM3_1
F:\MASM>
```

2. Программа была изменена таким образом, чтобы она освобождала неиспользуемую программой память.



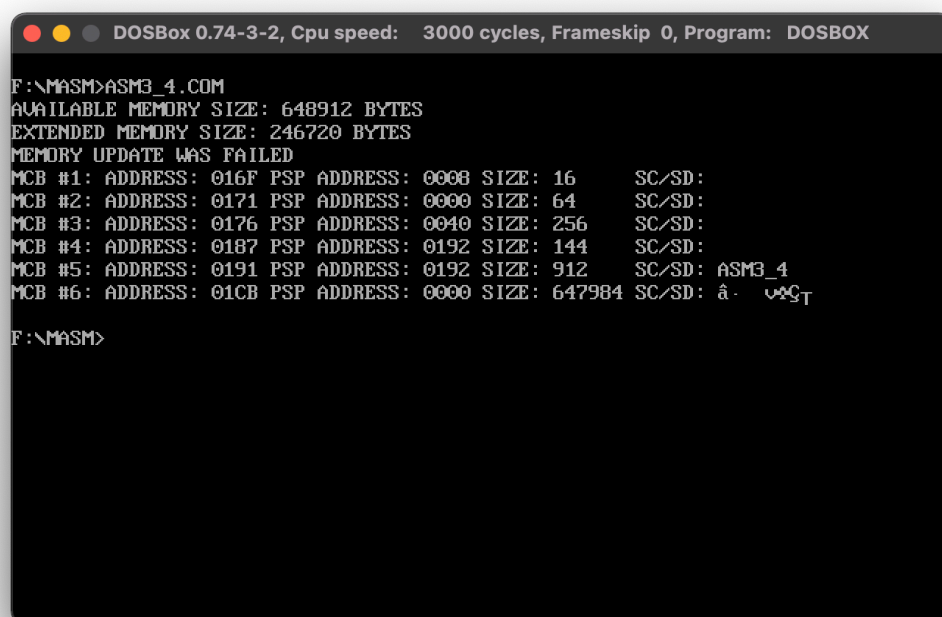
```
DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
F:\MASM>ASM3_2.COM
AVAILABLE MEMORY SIZE: 648912 BYTES
EXTENDED MEMORY SIZE: 246720 BYTES
MCB #1: ADDRESS: 016F PSP ADDRESS: 0008 SIZE: 16 SC/SD:
MCB #2: ADDRESS: 0171 PSP ADDRESS: 0000 SIZE: 64 SC/SD:
MCB #3: ADDRESS: 0176 PSP ADDRESS: 0040 SIZE: 256 SC/SD:
MCB #4: ADDRESS: 0187 PSP ADDRESS: 0192 SIZE: 144 SC/SD:
MCB #5: ADDRESS: 0191 PSP ADDRESS: 0192 SIZE: 832 SC/SD: ASM3_2
MCB #6: ADDRESS: 01C6 PSP ADDRESS: 0000 SIZE: 648064 SC/SD:
F:\MASM>_
```

3. Программа была изменена. После освобождения памяти, программа запрашивает 64Кб памяти функцией 48h прерывания 21h.



```
DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
F:\MASM>ASM3_3.COM
AVAILABLE MEMORY SIZE: 648912 BYTES
EXTENDED MEMORY SIZE: 246720 BYTES
MEMORY WAS UPDATED
MCB #1: ADDRESS: 016F PSP ADDRESS: 0008 SIZE: 16 SC/SD:
MCB #2: ADDRESS: 0171 PSP ADDRESS: 0000 SIZE: 64 SC/SD:
MCB #3: ADDRESS: 0176 PSP ADDRESS: 0040 SIZE: 256 SC/SD:
MCB #4: ADDRESS: 0187 PSP ADDRESS: 0192 SIZE: 144 SC/SD:
MCB #5: ADDRESS: 0191 PSP ADDRESS: 0192 SIZE: 896 SC/SD: ASM3_3
MCB #6: ADDRESS: 01CA PSP ADDRESS: 0192 SIZE: 65536 SC/SD: ASM3_3
MCB #7: ADDRESS: 11CB PSP ADDRESS: 0000 SIZE: 582448 SC/SD: ght (C)
F:\MASM>_
```

4. Программа была модифицирована таким образом, чтобы она запрашивала 64Кб памяти до освобождения неиспользуемой памяти.



```
DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
F:\MASM>ASM3_4.COM
AVAILABLE MEMORY SIZE: 648912 BYTES
EXTENDED MEMORY SIZE: 246720 BYTES
MEMORY UPDATE WAS FAILED
MCB #1: ADDRESS: 016F PSP ADDRESS: 0008 SIZE: 16 SC/SD:
MCB #2: ADDRESS: 0171 PSP ADDRESS: 0000 SIZE: 64 SC/SD:
MCB #3: ADDRESS: 0176 PSP ADDRESS: 0040 SIZE: 256 SC/SD:
MCB #4: ADDRESS: 0187 PSP ADDRESS: 0192 SIZE: 144 SC/SD:
MCB #5: ADDRESS: 0191 PSP ADDRESS: 0192 SIZE: 912 SC/SD: ASM3_4
MCB #6: ADDRESS: 01CB PSP ADDRESS: 0000 SIZE: 647984 SC/SD: â· ºT
F:\MASM>
```

Ответы на контрольные вопросы

Контрольные вопросы по лабораторной работе №3

1) Что означает "доступный объем памяти"?

Ответ: Область оперативной памяти, которая выделяется для использования программе.

2) Где MCB блок Вашей программы в списке?

Ответ: на первой фотографии MCB-блок программы – 5-ый в таблице.

На второй фотографии MCB-блок программы – 5-ый в таблице.

На третьей фотографии MCB-блок программы – 5-ый и 6-ой в таблице.

На четвертой фотографии MCB-блок программы – 5-ый в списке.

3) Какой размер памяти занимает программа в каждом случае?

Ответ: на первой фотографии программа занимает весь доступный ей объем памяти.

На второй фотографии после освобождения памяти, программа занимает 832 байт.

На третьей фотографии программа освобождает неиспользуемую память, а затем запрашивает 64Кб памяти. Объем занимаемой ею памяти в конечном итоге – 66432 байт.

На четвертой фотографии программа занимает необходимый ей объем памяти – 912 байт.

Выводы.

Были изучены и применены на практике способы управления динамическими разделами памяти. Была рассмотрена структура МСВ-таблицы и ее блоков соответственно. Была освоена работа функций управления памятью ядра операционной системы

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: LAB3_1.COM

```
TESTPC  SEGMENT

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

ORG 100H

STARTUP: JMP START

SC_OR_SD_MSG          DB          "SC/SD:  ",
'$'
RECORD_SIZE            DB          "SIZE:      ",
'$'
ADDRESS                DB          "ADDRESS:    ",
'$'
PSP_ADDRESS            DB          "PSP ADDRESS:  ",
'$'
MCB_NUMBER             DB          "MCB  #",
'$'
DECIMAL_NUMBER         DB          "          ",
'$'
EXTENDED_MEMORY_SIZE   DB          "EXTENDED MEMORY SIZE:  BYTES",
0DH, 0AH, '$'
AVAILABLE_MEM_SIZE     DB          "AVAILABLE MEMORY SIZE:  BYTES",
0DH, 0AH, '$'
NEWLINE                DB          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

PARAGRAPHS_TO_BYTES PROC

```
PUSH AX
PUSH BX
PUSH CX
PUSH DX
PUSH SI
```

```
MOV BX, 10H
MUL BX
MOV BX, 0AH
XOR CX, CX
```

```

DIVISION:
    DIV BX
    PUSH DX
    INC CX
    XOR DX, DX
    CMP AX, 0H
    JNZ DIVISION

WRITE_SYMBOL:
    POP DX
    OR DL, 30H
    MOV [SI], DL
    INC SI
    LOOP WRITE_SYMBOL

    POP SI
    POP DX
    POP CX
    POP BX
    POP AX

    RET

PARAGRAPHS_TO_BYTES ENDP

PRINT_NEWLINE PROC NEAR
    PUSH AX
    PUSH DX

    MOV DX, OFFSET NEWLINE
    MOV AH, 9H
    INT 21H

    POP DX
    POP AX

    RET
PRINT_NEWLINE ENDP

```

WRITE_STRING PROC NEAR

PUSH AX

MOV AH, 9H

INT 21H

POP AX

RET

WRITE_STRING ENDP

PRINT_AVAILABLE_MEM_SIZE PROC NEAR

PUSH AX

PUSH BX

PUSH SI

MOV AH, 4AH

MOV BX, 0FFFFH

INT 21H

MOV AX, BX

MOV SI, OFFSET AVAILABLE_MEM_SIZE

ADD SI, 23

CALL PARAGRAPHS_TO_BYTES

MOV DX, OFFSET AVAILABLE_MEM_SIZE

CALL WRITE_STRING

POP SI

POP BX

POP AX

RET

PRINT_AVAILABLE_MEM_SIZE ENDP

PRINT_EXTENDED_MEM_SIZE PROC NEAR

PUSH AX

PUSH BX

PUSH SI

```

MOV AL, 30H
OUT 70H, AL
IN AL, 71H

MOV AL, 31H
OUT 70H, AL
IN AL, 71H
MOV AH, AL

MOV SI, OFFSET EXTENDED_MEMORY_SIZE
ADD SI, 22
CALL PARAGRAPHS_TO_BYTES

MOV DX, OFFSET EXTENDED_MEMORY_SIZE
CALL WRITE_STRING

POP SI
POP BX
POP AX

RET
PRINT_EXTENDED_MEM_SIZE ENDP

PRINT_MCB_RECORD PROC NEAR
    PUSH AX
    PUSH DX
    PUSH SI
    PUSH DI
    PUSH CX

    MOV AX, ES
    MOV DI, OFFSET ADDRESS
    ADD DI, 12
    CALL WRD_TO_HEX
    MOV DX, OFFSET ADDRESS
    CALL WRITE_STRING

```

```

    MOV AX, ES:[1]
    MOV DI, OFFSET PSP_ADDRESS
    ADD DI, 16
    CALL WRD_TO_HEX
    MOV DX, OFFSET PSP_ADDRESS
    CALL WRITE_STRING

    MOV AX, ES:[3]
    MOV SI, OFFSET RECORD_SIZE
    ADD SI, 6
    CALL PARAGRAPHS_TO_BYTES
    MOV DX, OFFSET RECORD_SIZE
    CALL WRITE_STRING

    MOV BX, 8
    MOV DX, OFFSET SC_OR_SD_MSG
    CALL WRITE_STRING
    MOV CX, 7

PRINT_SCSD_LOOP:
    MOV DL, ES:[BX]
    MOV AH, 02H
    INT 21H
    INC BX
    LOOP PRINT_SCSD_LOOP

    POP CX
    POP DI
    POP SI
    POP DX
    POP AX

    RET
PRINT_MCB_RECORD ENDP

OFFSET_DECIMAL_NUMBER PROC NEAR
OFFSET_LOOP:
    CMP BYTE PTR [SI], ' '

```

```

    JNE EXIT_OFFSET_DECIMAL
    INC SI
    JMP OFFSET_LOOP

EXIT_OFFSET_DECIMAL:
    RET

OFFSET_DECIMAL_NUMBER ENDP


PRINT_MCB_TABLE PROC NEAR
    PUSH AX
    PUSH BX
    PUSH ES
    PUSH DX

    MOV AH, 52H
    INT 21H
    MOV AX, ES:[BX-2]
    MOV ES, AX
    MOV CL, 1

PRINT_MCB_INFO:
    MOV DX, OFFSET MCB_NUMBER
    CALL WRITE_STRING

    MOV AL, CL
    MOV SI, OFFSET DECIMAL_NUMBER
    ADD SI, 2
    CALL BYTE_TO_DEC
    CALL OFFSET_DECIMAL_NUMBER
    MOV DX, SI
    CALL WRITE_STRING

    MOV DL, ':'
    MOV AH, 02H
    INT 21H
    MOV DL, ' '
    MOV AH, 02H
    INT 21H

```



```

CALL PRINT_MCB_RECORD
CALL PRINT_NEWLINE

MOV AL, ES:[0]
CMP AL, 5AH
JE EXIT

MOV BX, ES:[3]
MOV AX, ES
ADD AX, BX
INC AX
MOV ES, AX

INC CL
JMP PRINT_MCB_INFO

EXIT:
POP DX
POP ES
POP BX
POP AX

RET
PRINT_MCB_TABLE ENDP

START:
CALL PRINT_AVAILABLE_MEM_SIZE
CALL PRINT_EXTENDED_MEM_SIZE
CALL PRINT_MCB_TABLE

XOR AL, AL
MOV AH, 4CH
INT 21H

TESTPC ENDS
END START

```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: LAB3_2.ASM

```
TESTPC  SEGMENT

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

ORG 100H

STARTUP: JMP START

        MCB_NUMBER                      DB          "MCB  #",
'$'
        NEWLINE                          DB          0DH,
0AH, '$'
        DECIMAL_NUMBER                   DB          "      ",
'$'
        MCB_RECORD_SIZE                  DB          "SIZE:      ",
'$'
        SC_SD                            DB          "SC/SD:  ",
'$'
        MCB_ADDRESS                      DB          "ADDRESS:      ",
'$'
        AVAILABLE_MEM_SIZE               DB          "AVAILABLE MEMORY SIZE:      BYTES",
0DH, 0AH, '$'
        PSP_ADDRESS                      DB          "PSP  ADDRESS:      ",
'$'
        EXTENDED_MEMORY_SIZE            DB          "EXTENDED MEMORY SIZE:      BYTES",
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

PARAGRAPH2BYTES PROC

PUSH AX
PUSH BX
PUSH CX
PUSH DX
PUSH SI

MOV BX, 10H
MUL BX
MOV BX, 0AH
XOR CX, CX

DIVISION:

```
DIV BX
PUSH DX
INC CX
XOR DX, DX
CMP AX, 0H
JNZ DIVISION
```

WRITE_SYMBOL:

```
POP DX
OR DL, 30H
MOV [SI], DL
INC SI
LOOP WRITE_SYMBOL
```

POP SI

POP DX

POP CX

POP BX

POP AX

RET

PARAGRAPH2BYTES ENDP

PRINT_NEWLINE PROC NEAR

PUSH AX

PUSH DX

MOV DX, OFFSET NEWLINE

MOV AH, 9H

INT 21H

POP DX

POP AX

RET

PRINT_NEWLINE ENDP

WRITE_STRING PROC NEAR

PUSH AX

MOV AH, 9H

INT 21H

POP AX

RET

WRITE_STRING ENDP

PRINT_MEM_SIZE PROC NEAR

MOV AH, 4AH

MOV BX, 0FFFFH

INT 21H

MOV AX, BX

MOV SI, OFFSET AVAILABLE_MEM_SIZE

ADD SI, 23

CALL PARAGRAPH2BYTES

MOV DX, OFFSET AVAILABLE_MEM_SIZE

CALL WRITE_STRING

MOV AL, 30H

OUT 70H, AL

IN AL, 71H

MOV AL, 31H

OUT 70H, AL

IN AL, 71H

MOV AH, AL

MOV SI, OFFSET EXTENDED_MEMORY_SIZE

ADD SI, 22

CALL PARAGRAPH2BYTES

MOV DX, OFFSET EXTENDED_MEMORY_SIZE

CALL WRITE_STRING

```

    RET

PRINT_MEM_SIZE ENDP


PRINT_MCB_RECORD PROC NEAR
    PUSH AX
    PUSH DX
    PUSH SI
    PUSH DI
    PUSH CX

    MOV AX, ES
    MOV DI, OFFSET MCB_ADDRESS
    ADD DI, 12
    CALL WRD_TO_HEX
    MOV DX, OFFSET MCB_ADDRESS
    CALL WRITE_STRING

    MOV AX, ES:[1]
    MOV DI, OFFSET PSP_ADDRESS
    ADD DI, 16
    CALL WRD_TO_HEX
    MOV DX, OFFSET PSP_ADDRESS
    CALL WRITE_STRING

    MOV AX, ES:[3]
    MOV SI, OFFSET MCB_RECORD_SIZE
    ADD SI, 6
    CALL PARAGRAPH2BYTES
    MOV DX, OFFSET MCB_RECORD_SIZE
    CALL WRITE_STRING

    MOV BX, 8
    MOV DX, OFFSET SC_SD
    CALL WRITE_STRING
    MOV CX, 7

PRINT_SCSD:
    MOV DL, ES:[BX]

```

```

MOV AH, 02H
INT 21H
INC BX
LOOP PRINT_SCSD

POP CX
POP DI
POP SI
POP DX
POP AX

RET

PRINT_MCB_RECORD ENDP

OFFSET_DECIMAL_NUMBER PROC NEAR
OFFSET_LOOP:
    CMP BYTE PTR [SI], ' '
    JNE EXIT_OFFSET_DECIMAL
    INC SI
    JMP OFFSET_LOOP

EXIT_OFFSET_DECIMAL:
    RET
OFFSET_DECIMAL_NUMBER ENDP

PRINT_MCB_TABLE PROC NEAR

    PUSH ES

    MOV AH, 52H
    INT 21H
    MOV ES, ES:[BX-2]
    MOV CL, 1

PRINT_MCB_INFO:
    MOV DX, OFFSET MCB_NUMBER
    CALL WRITE_STRING

```



```

MOV AL, CL
MOV SI, OFFSET DECIMAL_NUMBER
ADD SI, 2
CALL BYTE_TO_DEC
CALL OFFSET_DECIMAL_NUMBER
MOV DX, SI
CALL WRITE_STRING

MOV DL, ':'
MOV AH, 02H
INT 21H
MOV DL, ' '
MOV AH, 02H
INT 21H

CALL PRINT_MCB_RECORD
CALL PRINT_NEWLINE

MOV AL, ES:[0]
CMP AL, 5AH
JE EXIT

MOV BX, ES:[3]
MOV AX, ES
ADD AX, BX
INC AX
MOV ES, AX

INC CL
JMP PRINT_MCB_INFO

EXIT:

POP ES

RET
PRINT_MCB_TABLE ENDP

```

FREE_MEME PROC NEAR

LEA AX, EOF

MOV BX, 10H

XOR DX, DX

DIV BX

INC AX

MOV BX, AX

MOV AL, 0

MOV AH, 4AH

INT 21H

RET

FREE_MEME ENDP

START:

CALL PRINT_MEM_SIZE

CALL FREE_MEME

CALL PRINT_MCB_TABLE

XOR AL, AL

MOV AH, 4CH

INT 21H

EOF:

TESTPC ENDS

END STARTUP

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: LAB3_3.ASM

```
TESTPC  SEGMENT

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

ORG 100H

STARTUP: JMP START

        MCB_NUMBER                      DB          "MCB  #",
'$'
        NEWLINE                          DB          0DH,
0AH, '$'
        DECIMAL_NUMBER                   DB          "      ",
'$'
        MCB_RECORD_SIZE                  DB          "SIZE:      ",
'$'
        SC_SD                            DB          "SC/SD:  ",
'$'
        MCB_ADDRESS                      DB          "ADDRESS:      ",
'$'
        AVAILABLE_MEM_SIZE               DB          "AVAILABLE MEMORY SIZE:      BYTES",
0DH, 0AH, '$'
        PSP_ADDRESS                      DB          "PSP  ADDRESS:      ",
'$'
        EXTENDED_MEMORY_SIZE             DB          "EXTENDED MEMORY SIZE:      BYTES",
0DH, 0AH, '$'
        MEMORY_UPDATED                   DB          "MEMORY  WAS  UPDATED",
0DH, 0AH, '$'
        MEMORY_UPD_FAILED                 DB          "MEMORY UPDATE WAS FAILED",
'$'

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
```

```
PARAGRAPH2BYTES PROC
```

```
    PUSH AX
```

```
    PUSH BX
```

```
    PUSH CX
```

```
    PUSH DX
```

```
    PUSH SI
```

```
    MOV BX, 10H
```

```

        MUL  BX
        MOV  BX,  0AH
        XOR  CX,  CX

DIVISION:
        DIV  BX
        PUSH DX
        INC  CX
        XOR  DX,  DX
        CMP  AX,  0H
        JNZ  DIVISION

WRITE_SYMBOL:
        POP  DX
        OR   DL,  30H
        MOV  [SI], DL
        INC  SI
        LOOP WRITE_SYMBOL

        POP  SI
        POP  DX
        POP  CX
        POP  BX
        POP  AX
        RET

PARAGRAPH2BYTES ENDP

PRINT_NEWLINE PROC NEAR
        PUSH AX
        PUSH DX

        MOV  DX,  OFFSET NEWLINE
        MOV  AH,  9H
        INT  21H

        POP  DX
        POP  AX

```

```
RET
PRINT_NEWLINE ENDP
```

```
WRITE_STRING PROC NEAR
    PUSH AX
    MOV AH, 9H
    INT 21H
    POP AX
    RET
WRITE_STRING ENDP
```

```
PRINT_MEM_SIZE PROC NEAR
    MOV AH, 4AH
    MOV BX, 0FFFFH
    INT 21H

    MOV AX, BX
    MOV SI, OFFSET AVAILABLE_MEM_SIZE
    ADD SI, 23
    CALL PARAGRAPH2BYTES

    MOV DX, OFFSET AVAILABLE_MEM_SIZE
    CALL WRITE_STRING

    MOV AL, 30H
    OUT 70H, AL
    IN AL, 71H

    MOV AL, 31H
    OUT 70H, AL
    IN AL, 71H
    MOV AH, AL

    MOV SI, OFFSET EXTENDED_MEMORY_SIZE
    ADD SI, 22
    CALL PARAGRAPH2BYTES
```

```

MOV DX, OFFSET EXTENDED_MEMORY_SIZE
CALL WRITE_STRING

RET

PRINT_MEM_SIZE ENDP

PRINT_MCB_RECORD PROC NEAR
    PUSH AX
    PUSH DX
    PUSH SI
    PUSH DI
    PUSH CX

    MOV AX, ES
    MOV DI, OFFSET MCB_ADDRESS
    ADD DI, 12
    CALL WRD_TO_HEX
    MOV DX, OFFSET MCB_ADDRESS
    CALL WRITE_STRING

    MOV AX, ES:[1]
    MOV DI, OFFSET PSP_ADDRESS
    ADD DI, 16
    CALL WRD_TO_HEX
    MOV DX, OFFSET PSP_ADDRESS
    CALL WRITE_STRING

    MOV AX, ES:[3]
    MOV SI, OFFSET MCB_RECORD_SIZE
    ADD SI, 6
    CALL PARAGRAPH2BYTES
    MOV DX, OFFSET MCB_RECORD_SIZE
    CALL WRITE_STRING

    MOV BX, 8
    MOV DX, OFFSET SC_SD
    CALL WRITE_STRING
    MOV CX, 7

```



```

PRINT_SCSD:
    MOV DL, ES:[BX]
    MOV AH, 02H
    INT 21H
    INC BX
    LOOP PRINT_SCSD

    POP CX
    POP DI
    POP SI
    POP DX
    POP AX

    RET
PRINT_MCB_RECORD ENDP

OFFSET_DECIMAL_NUMBER PROC NEAR
OFFSET_LOOP:
    CMP BYTE PTR [SI], ' '
    JNE EXIT_OFFSET_DECIMAL
    INC SI
    JMP OFFSET_LOOP

EXIT_OFFSET_DECIMAL:
    RET
OFFSET_DECIMAL_NUMBER ENDP

PRINT_MCB_TABLE PROC NEAR

    PUSH ES

    MOV AH, 52H
    INT 21H
    MOV ES, ES:[BX-2]
    MOV CL, 1

```

```

PRINT_MCB_INFO:
    MOV DX, OFFSET MCB_NUMBER
    CALL WRITE_STRING

    MOV AL, CL
    MOV SI, OFFSET DECIMAL_NUMBER
    ADD SI, 2
    CALL BYTE_TO_DEC
    CALL OFFSET_DECIMAL_NUMBER
    MOV DX, SI
    CALL WRITE_STRING

    MOV DL, ':'
    MOV AH, 02H
    INT 21H
    MOV DL, ' '
    MOV AH, 02H
    INT 21H

    CALL PRINT_MCB_RECORD
    CALL PRINT_NEWLINE

    MOV AL, ES:[0]
    CMP AL, 5AH
    JE EXIT

    MOV BX, ES:[3]
    MOV AX, ES
    ADD AX, BX
    INC AX
    MOV ES, AX

    INC CL
    JMP PRINT_MCB_INFO

EXIT:

    POP ES

```

```
RET
PRINT_MCB_TABLE ENDP
```

```
FREE_MEME PROC NEAR
```

```
    LEA AX, EOF
    MOV BX, 10H
    XOR DX, DX
    DIV BX
    INC AX
    MOV BX, AX
    MOV AL, 0
    MOV AH, 4AH
    INT 21H
```

```
RET
FREE_MEME ENDP
```

```
UPDATE_MEME PROC NEAR
```

```
    MOV BX, 1000H
    MOV AH, 48H
    INT 21H
```

```
JC FAILED
```

```
    MOV DX, OFFSET MEMORY_UPDATED
    CALL WRITE_STRING
    JMP EXIT_UPD_MEME
```

```
FAILED:
```

```
    MOV DX, OFFSET MEMORY_UPD_FAILED
    CALL WRITE_STRING
    JMP EXIT_UPD_MEME
```

```
EXIT_UPD_MEME:
```

```
    RET
UPDATE_MEME ENDP
```

```
START:
    CALL PRINT_MEM_SIZE
    CALL FREE_MEME
    CALL UPDATE_MEME
    CALL PRINT_MCB_TABLE

    XOR AL, AL
    MOV AH, 4CH
    INT 21H

EOF:
TESTPC ENDS
END STARTUP
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: LAB3_4.ASM

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

ORG 100H

STARTUP: JMP START

        MCB_NUMBER          DB          "MCB  #",
'$'
        NEWLINE              DB          0DH,
0AH, '$'
        DECIMAL_NUMBER      DB          "      ",
'$'
        MCB_RECORD_SIZE     DB          "SIZE:      ",
'$'
        SC_SD                DB          "SC/SD:  ",
'$'
        MCB_ADDRESS         DB          "ADDRESS:      ",
'$'
        AVAILABLE_MEM_SIZE   DB          "AVAILABLE MEMORY SIZE:      BYTES",
0DH, 0AH, '$'
        PSP_ADDRESS         DB          "PSP ADDRESS:      ",
'$'
        EXTENDED_MEMORY_SIZE DB          "EXTENDED MEMORY SIZE:      BYTES",
0DH, 0AH, '$'
        MEMORY_UPDATED       DB          "MEMORY WAS UPDATED",
0DH, 0AH, '$'
        MEMORY_UPD_FAILED    DB          "MEMORY UPDATE WAS FAILED",
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

PARAGRAPH2BYTES PROC

```
PUSH AX
PUSH BX
PUSH CX
PUSH DX
PUSH SI
```

```
MOV BX, 10H
MUL BX
MOV BX, 0AH
XOR CX, CX
```

DIVISION:

```
DIV BX
PUSH DX
INC CX
XOR DX, DX
CMP AX, 0H
JNZ DIVISION
```

WRITE_SYMBOL:

```
POP DX
OR DL, 30H
MOV [SI], DL
INC SI
LOOP WRITE_SYMBOL
```

POP SI

POP DX

POP CX

POP BX

POP AX

RET

PARAGRAPH2BYTES ENDP

PRINT_NEWLINE PROC NEAR

PUSH AX

PUSH DX

MOV DX, OFFSET NEWLINE

MOV AH, 9H

INT 21H

POP DX

POP AX

RET

PRINT_NEWLINE ENDP

WRITE_STRING PROC NEAR

PUSH AX

MOV AH, 9H

INT 21H

POP AX

RET

WRITE_STRING ENDP

PRINT_MEM_SIZE PROC NEAR

MOV AH, 4AH

MOV BX, 0FFFFH

INT 21H

MOV AX, BX

MOV SI, OFFSET AVAILABLE_MEM_SIZE

ADD SI, 23

CALL PARAGRAPH2BYTES

MOV DX, OFFSET AVAILABLE_MEM_SIZE

CALL WRITE_STRING

MOV AL, 30H

OUT 70H, AL

IN AL, 71H

MOV AL, 31H

OUT 70H, AL

IN AL, 71H

MOV AH, AL

MOV SI, OFFSET EXTENDED_MEMORY_SIZE

ADD SI, 22

CALL PARAGRAPH2BYTES

MOV DX, OFFSET EXTENDED_MEMORY_SIZE

CALL WRITE_STRING

```

    RET

PRINT_MEM_SIZE ENDP


PRINT_MCB_RECORD PROC NEAR
    PUSH AX
    PUSH DX
    PUSH SI
    PUSH DI
    PUSH CX

    MOV AX, ES
    MOV DI, OFFSET MCB_ADDRESS
    ADD DI, 12
    CALL WRD_TO_HEX
    MOV DX, OFFSET MCB_ADDRESS
    CALL WRITE_STRING

    MOV AX, ES:[1]
    MOV DI, OFFSET PSP_ADDRESS
    ADD DI, 16
    CALL WRD_TO_HEX
    MOV DX, OFFSET PSP_ADDRESS
    CALL WRITE_STRING

    MOV AX, ES:[3]
    MOV SI, OFFSET MCB_RECORD_SIZE
    ADD SI, 6
    CALL PARAGRAPH2BYTES
    MOV DX, OFFSET MCB_RECORD_SIZE
    CALL WRITE_STRING

    MOV BX, 8
    MOV DX, OFFSET SC_SD
    CALL WRITE_STRING
    MOV CX, 7

PRINT_SCSD:
    MOV DL, ES:[BX]

```

```

MOV AH, 02H
INT 21H
INC BX
LOOP PRINT_SCSD

POP CX
POP DI
POP SI
POP DX
POP AX

RET

PRINT_MCB_RECORD ENDP

OFFSET_DECIMAL_NUMBER PROC NEAR
OFFSET_LOOP:
    CMP BYTE PTR [SI], ' '
    JNE EXIT_OFFSET_DECIMAL
    INC SI
    JMP OFFSET_LOOP

EXIT_OFFSET_DECIMAL:
    RET
OFFSET_DECIMAL_NUMBER ENDP

PRINT_MCB_TABLE PROC NEAR

    PUSH ES

    MOV AH, 52H
    INT 21H
    MOV ES, ES:[BX-2]
    MOV CL, 1

PRINT_MCB_INFO:
    MOV DX, OFFSET MCB_NUMBER
    CALL WRITE_STRING

```

```

MOV AL, CL
MOV SI, OFFSET DECIMAL_NUMBER
ADD SI, 2
CALL BYTE_TO_DEC
CALL OFFSET_DECIMAL_NUMBER
MOV DX, SI
CALL WRITE_STRING

MOV DL, ':'
MOV AH, 02H
INT 21H
MOV DL, ' '
MOV AH, 02H
INT 21H

CALL PRINT_MCB_RECORD
CALL PRINT_NEWLINE

MOV AL, ES:[0]
CMP AL, 5AH
JE EXIT

MOV BX, ES:[3]
MOV AX, ES
ADD AX, BX
INC AX
MOV ES, AX

INC CL
JMP PRINT_MCB_INFO

EXIT:

POP ES

RET
PRINT_MCB_TABLE ENDP

```

FREE_MEME PROC NEAR

LEA AX, EOF

MOV BX, 10H

XOR DX, DX

DIV BX

INC AX

MOV BX, AX

MOV AL, 0

MOV AH, 4AH

INT 21H

RET

FREE_MEME ENDP

UPDATE_MEME PROC NEAR

MOV BX, 1000H

MOV AH, 48H

INT 21H

JC FAILED

MOV DX, OFFSET MEMORY_UPDATED

CALL WRITE_STRING

JMP EXIT_UPD_MEME

FAILED:

MOV DX, OFFSET MEMORY_UPD_FAILED

CALL WRITE_STRING

JMP EXIT_UPD_MEME

EXIT_UPD_MEME:

RET

UPDATE_MEME ENDP

START:

CALL PRINT_MEM_SIZE

```
CALL UPDATE_MEME  
CALL FREE_MEME  
CALL PRINT_MCB_TABLE
```

```
XOR AL, AL  
MOV AH, 4CH  
INT 21H
```

```
EOF:  
TESTPC ENDS  
END STARTUP
```