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.

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

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 MCB 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.

Теоретические сведения.

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. MCB 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 f52h "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 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта

Выполнение работы.

На основе шаблона .COM модуля, приведённого в методических указаниях, была реализована программа, которая считывает и выводит требуемые данные в различных указанных случаях. Для этого были установлены сообщения, относящиеся к каждому из рассматриваемых параметров. К имеющимся в программе процедурам были добавлены:

- Вспомогательные процедуры:

  PRINT — осуществление вывода;

  WRD_TO_DEC — перевод в десятичную систему счисления;

- _AM — Вывод количества доступной памяти. Реализация основана на обращении к функции 48h прерывания 21h. Полученное значение, измеряемое в параграфах, переводится в байты при помощи WRD_TO_DEC и выводится десятичной системе.
- _EM — Вывод размера расширенной памяти. Осуществляется посредством общения к ячейкам CMOS. Значение выводится в килобайтах.
- _MCB — Вывод цепочки информации о блоке управления памятью. Считывание происходит при помощи функции 52h прерывания 21h и завершается в момент достижения последнего блока в списке.

Шаг 1.

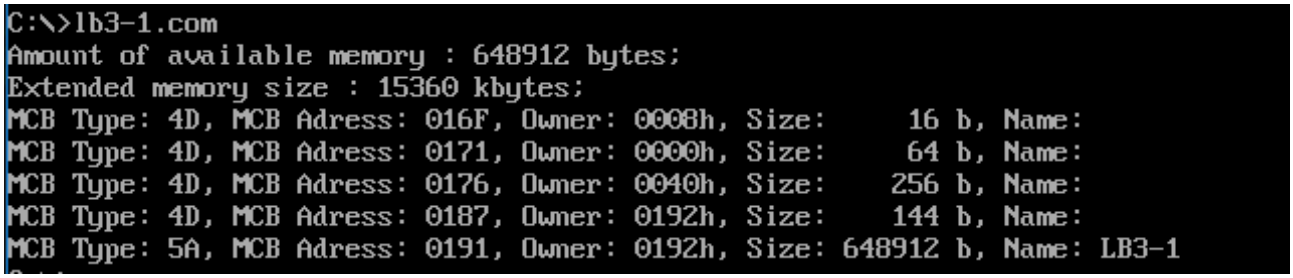

Рисунок 1 — Результат запуска lb3-1.com

Шаг 2.

Далее было добавлено освобождение памяти, которую не занимает программа. Для этого использовалась функция 4Ah прерывания 21h.

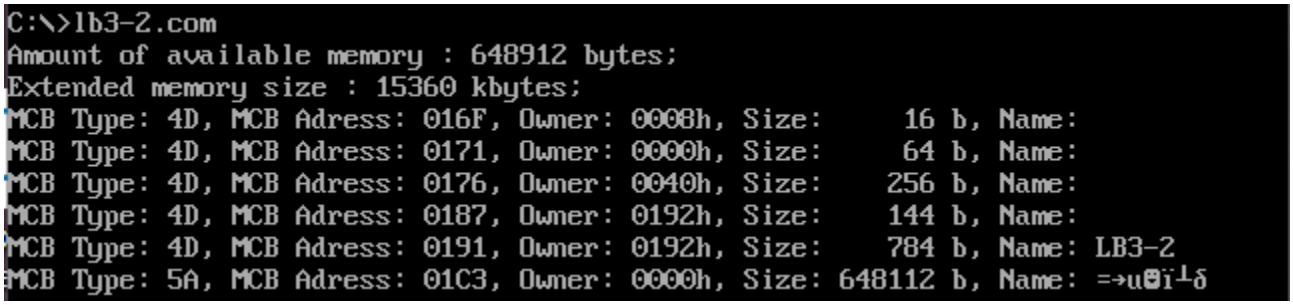

Рисунок 2 — Результат запуска lb3-2.com

Вывод программы демонстрирует то, что освобождённая часть теперь относится к дополнительному (шестому) блоку управления памятью.

Шаг 3.

Следом в программу был добавлен запрос 64Кб памяти посредством функции 48h прерывания 21h (запрос выполняется после освобождения памяти).

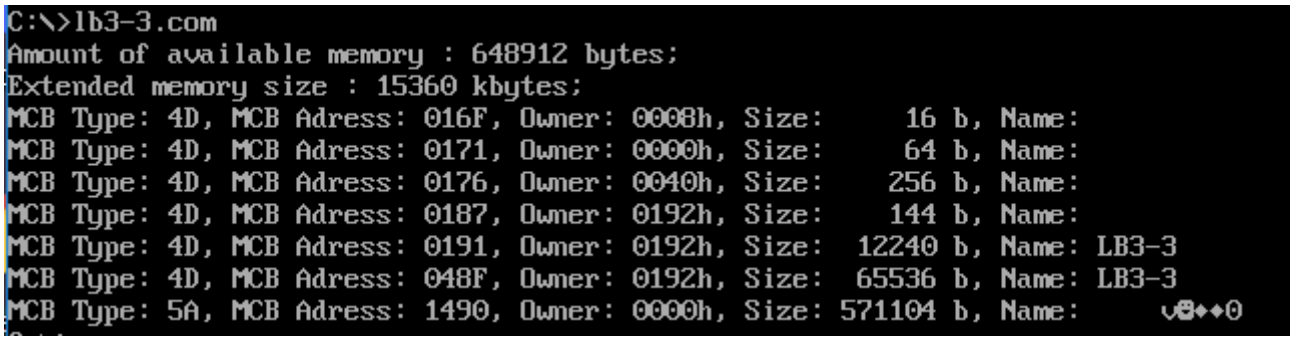

Рисунок 3 — Результат запуска lb3-3.com

По цепочке информации о блоке управления памятью видно, что сразу после блока памяти программы появляется новый (седьмой) блок размером 64 Кб, это и есть та самая выделенная память.

Шаг 4.

В данном случае запрос памяти осуществляется ДО её освобождения.

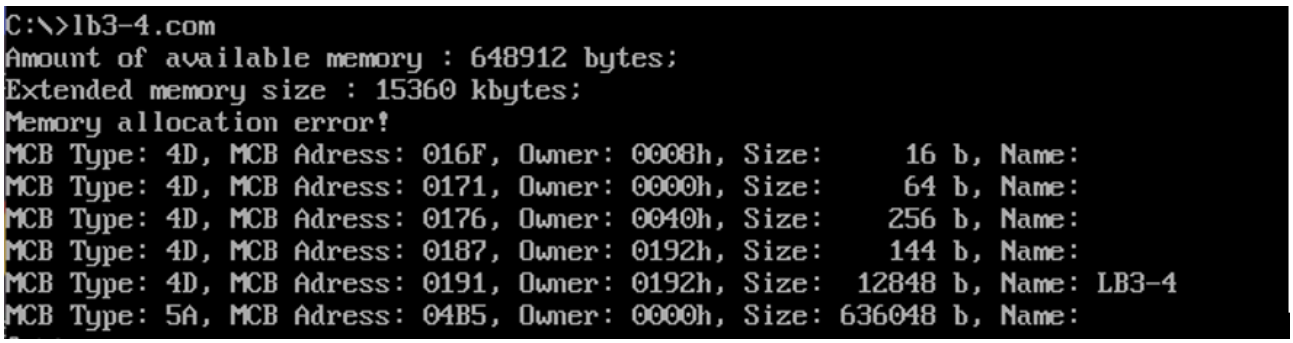

Рисунок 4 — Результат запуска lb3-4.com

В результате видно, что память не была выделена (о чём свидетельствует соответствующее сообщение) и связанно это с тем, что на данном этапе программе уже принадлежит вся свободная память. При этом успешно выполняется след идущее освобождение памяти.

Исходный код программ см. в приложении А

Контрольные вопросы.

1) Что означает "доступный объем памяти"?
   - Ответ: Доступный объем памяти — размер оперативной памяти в системе, который используется для запуска и выполнения программы.
2) Где MCB блок Вашей программы в списке?
   - Ответ: MCB блок в первой, второй и четвёртой версиях программы – это 5 блок в списке, в третьей – 6 (выделенный при выполнении).
3) Какой размер памяти занимает программа в каждом случае?
   - Ответ: Размер памяти программ:
     1. 648912 байт (весь доступный объём памяти);
     2. 784 байт (только объём, занимаемый программой);
     3. 66352 байт (объём памяти программы + дополнительно выделенная память);
     4. 12848 байт (аналогично п. 2);

Выводы.

Были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

Был реализован программный модуль типа .COM, который выбирает и распечатывает: количество доступной памяти, размер расширенной памяти и цепочку блоков управления памятью. Вместе с этим рассмотрены различные случаи работы программы при запрашивании/освобождении памяти.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: lb3-1.asm

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

AM db 'Amount of available memory :       bytes;', 0DH, 0AH, '$'
EM db 'Extended memory size :      kbytes;', 0DH, 0AH, '$'
MCB db 'MCB Type:   , MCB Adress:     , Owner:     h, Size:       b, Name: $'
END_STR db 0DH, 0AH,'$'

; 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 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
        ret
BYTE_TO_DEC ENDP
;----------------------
; КОД
WRD_TO_DEC PROC near
        push CX
        push DX
        mov CX, 10
loop_wd:
        div CX
        or DL, 30h
        mov [SI], DL
        dec SI
        xor DX, DX
        cmp AX, 10
        jae loop_wd
        cmp AL, 00h
```

```
        je end_2
        or AL, 30h
        mov [SI], AL
end_2:
        pop DX
        pop CX
        ret
WRD_TO_DEC ENDP

PRINT PROC near
        mov AH, 09h
        int 21h
        ret
PRINT ENDP

_AM PROC near
        mov SI, offset AM
        add SI, 34
        mov AH, 4Ah
        mov BX, 0FFFFh
        int 21h
        mov AX, 16
        mul BX
        call WRD_TO_DEC
        mov DX, offset AM
        call PRINT
        ret
_AM ENDP

_EM PROC near
        mov SI, offset EM
        add SI, 27
        sub DX, DX

        mov AL, 30h  ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h      ; чтение младшего байта
        mov BL, AL    ; размер расширенной памяти
        mov AL, 31h  ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h      ; чтение старшего байта
                        ; размера расширенной памяти
        mov AH, AL
        mov Al, BL
        call WRD_TO_DEC
        mov DX, offset EM
        call PRINT
        ret
_EM ENDP
```

```
_MCB PROC  near
        mov AH, 52h
        int 21h
        sub BX, 2
        mov AX, ES:[BX]
        mov ES, AX
Chain:
;Type
        mov DI, offset MCB
        add DI, 10
        mov AX, ES:[00h]
        call BYTE_TO_HEX
         mov [DI], AL
         add DI, 1
        mov [DI], AH
;Adress
        mov DI, offset MCB
        add DI, 29
        mov AX, ES
        call WRD_TO_HEX
;Owner
        mov DI, offset MCB
        add DI, 42
         mov AX, ES:[01h]
        call WRD_TO_HEX
;Size
        mov SI, offset MCB
         add SI, 57
        mov AX, ES:[03h]
        mov BX, 16
        mul BX
         call WRD_TO_DEC
;Print
        mov DX, offset MCB
        call PRINT
;Name
          mov DI, offset MCB
        add DI, 58
        mov BX, 8
        mov CX, 7
        cycle:
        mov DL, ES:[BX]
                mov AH, 02h
                int 21h
                add BX, 1
                loop cycle

        mov AL, ES:[0h]
```

```
        cmp AL, 5ah
        je final

        mov BX, ES
        mov AX, ES:[03h]
        add AX, BX
        add AX, 1
        mov ES, AX
        mov DX, offset END_STR
        call PRINT
        jmp Chain
final:
      ret
_MCB ENDP

BEGIN:
        call _AM
        call _EM
        call _MCB
; Выход в DOS
        xor AL, AL
        mov AH, 4Ch
        int 21h
ENDING:
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

AM db 'Amount of available memory :       bytes;', 0DH, 0AH, '$'
EM db 'Extended memory size :      kbytes;', 0DH, 0AH, '$'
MCB db 'MCB Type:   , MCB Adress:     , Owner:     h, Size:       b, Name: $'
END_STR db 0DH, 0AH,'$'

; 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 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
        ret
BYTE_TO_DEC ENDP
;---------------------
; КОД
WRD_TO_DEC PROC near
        push CX
        push DX
        mov CX, 10
loop_wd:
        div CX
        or DL, 30h
        mov [SI], DL
        dec SI
        xor DX, DX
        cmp AX, 10
        jae loop_wd
        cmp AL, 00h
        je end_2
        or AL, 30h
        mov [SI], AL
end_2:
        pop DX
        pop CX
        ret
WRD_TO_DEC ENDP

PRINT PROC near
        mov AH, 09h
        int 21h
        ret
PRINT ENDP

_AM PROC near
        mov SI, offset AM
        add SI, 34
        mov AH, 4Ah
        mov BX, 0FFFFh
        int 21h
        mov AX, 16
        mul BX
        call WRD_TO_DEC
        mov DX, offset AM
        call PRINT
        ret
```

```
_AM ENDP

_EM PROC near
        mov SI, offset EM
        add SI, 27
        sub DX, DX

        mov AL, 30h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h      ; чтение младшего байта
        mov BL, AL    ; размер расширенной памяти
        mov AL, 31h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h     ; чтение старшего байта
                       ; размера расширенной памяти
        mov AH, AL
        mov Al, BL
        call WRD_TO_DEC
        mov DX, offset EM
        call PRINT
        ret
_EM ENDP

_MCB PROC  near
        mov AH, 52h
        int 21h
        sub BX, 2
        mov AX, ES:[BX]
        mov ES, AX
Chain:
;Type
        mov DI, offset MCB
        add DI, 10
        mov AX, ES:[00h]
        call BYTE_TO_HEX
        mov [DI], AL
        add DI, 1
        mov [DI], AH
;Adress
        mov DI, offset MCB
        add DI, 29
        mov AX, ES
        call WRD_TO_HEX
;Owner
        mov DI, offset MCB
        add DI, 42
        mov AX, ES:[01h]
        call WRD_TO_HEX
;Size
```

```
        mov SI, offset MCB
        add SI, 57
        mov AX, ES:[03h]
        mov BX, 16
        mul BX
        call WRD_TO_DEC
;Print
        mov DX, offset MCB
        call PRINT
;Name
        mov DI, offset MCB
        add DI, 58
        mov BX, 8
        mov CX, 7
        cycle:
        mov DL, ES:[BX]
                mov AH, 02h
                int 21h
                add BX, 1
                loop cycle

        mov AL, ES:[0h]
        cmp AL, 5ah
        je final

        mov BX, ES
        mov AX, ES:[03h]
        add AX, BX
        add AX, 1
        mov ES, AX
        mov DX, offset END_STR
        call PRINT
        jmp Chain
final:
      ret
_MCB ENDP

BEGIN:
        call _AM
        call _EM
;--------FREEING_MEMORY--------
        mov AH, 04Ah
        mov BX, offset ENDING
        int 21h
;-----------------------
        call _MCB
; Выход в DOS
        xor AL, AL
        mov AH, 4Ch
```

```
        int 21h
ENDING:
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

AM db 'Amount of available memory :       bytes;', 0DH, 0AH, '$'
EM db 'Extended memory size :      kbytes;', 0DH, 0AH, '$'
MCB db 'MCB Type:   , MCB Adress:     , Owner:     h, Size:       b, Name: $'
END_STR db 0DH, 0AH,'$'

; 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 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
        ret
BYTE_TO_DEC ENDP
;----------------------
; КОД
WRD_TO_DEC PROC near
        push CX
        push DX
        mov CX, 10
loop_wd:
        div CX
        or DL, 30h
        mov [SI], DL
        dec SI
        xor DX, DX
        cmp AX, 10
        jae loop_wd
        cmp AL, 00h
```

```
        je end_2
        or AL, 30h
        mov [SI], AL
end_2:
        pop DX
        pop CX
        ret
WRD_TO_DEC ENDP

PRINT PROC near
        mov AH, 09h
        int 21h
        ret
PRINT ENDP

_AM PROC near
        mov SI, offset AM
        add SI, 34
        mov AH, 4Ah
        mov BX, 0FFFFh
        int 21h
        mov AX, 16
        mul BX
        call WRD_TO_DEC
        mov DX, offset AM
        call PRINT
        ret
_AM ENDP

_EM PROC near
        mov SI, offset EM
        add SI, 27
        sub DX, DX

        mov AL, 30h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h     ; чтение младшего байта
        mov BL, AL   ; размер расширенной памяти
        mov AL, 31h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h     ; чтение старшего байта
                       ; размера расширенной памяти
        mov AH, AL
        mov Al, BL
        call WRD_TO_DEC
        mov DX, offset EM
        call PRINT
        ret
_EM ENDP
```

```
_MCB PROC  near
        mov AH, 52h
         int 21h
        sub BX, 2
        mov AX, ES:[BX]
        mov ES, AX
Chain:
;Type
        mov DI, offset MCB
        add DI, 10
        mov AX, ES:[00h]
        call BYTE_TO_HEX
         mov [DI], AL
        add DI, 1
        mov [DI], AH
;Adress
        mov DI, offset MCB
        add DI, 29
        mov AX, ES
        call WRD_TO_HEX
;Owner
        mov DI, offset MCB
        add DI, 42
        mov AX, ES:[01h]
        call WRD_TO_HEX
;Size
        mov SI, offset MCB
         add SI, 57
        mov AX, ES:[03h]
        mov BX, 16
        mul BX
        call WRD_TO_DEC
;Print
        mov DX, offset MCB
        call PRINT
;Name
        mov DI, offset MCB
        add DI, 58
        mov BX, 8
        mov CX, 7
        cycle:
        mov DL, ES:[BX]
                mov AH, 02h
                int 21h
                add BX, 1
                loop cycle

        mov AL, ES:[0h]
```

```
        cmp AL, 5ah
        je final

        mov BX, ES
        mov AX, ES:[03h]
        add AX, BX
        add AX, 1
        mov ES, AX
        mov DX, offset END_STR
        call PRINT
        jmp Chain
final:
      ret
_MCB ENDP

BEGIN:
        call _AM
        call _EM
;--------FREEING_MEMORY--------
        mov AH, 04Ah
        mov BX, offset ENDING
        int 21h
;-----------------------

;------- REQUESTING_MEMORY-------
        mov AH,48h
        mov BX,1000h
        int 21h
;-----------------------
        call _MCB
; Выход в DOS
        xor AL, AL
        mov AH, 4Ch
        int 21h
ENDING:
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

AM db 'Amount of available memory :       bytes;', 0DH, 0AH, '$'
EM db 'Extended memory size :      kbytes;', 0DH, 0AH, '$'
MCB db 'MCB Type:   , MCB Adress:     , Owner:     h, Size:       b, Name: $'
```

```
END_STR db 0DH, 0AH,'$'
ERR db 'Memory allocation error!', 0DH, 0AH, '$'

; 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 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
        ret
BYTE_TO_DEC ENDP
;----------------------
; КОД
WRD_TO_DEC PROC near
        push CX
        push DX
        mov CX, 10
loop_wd:
        div CX
        or DL, 30h
        mov [SI], DL
        dec SI
        xor DX, DX
        cmp AX, 10
        jae loop_wd
        cmp AL, 00h
        je end_2
        or AL, 30h
        mov [SI], AL
end_2:
        pop DX
        pop CX
        ret
WRD_TO_DEC ENDP

PRINT PROC near
        mov AH, 09h
        int 21h
        ret
PRINT ENDP
```

```
_AM PROC near
        mov SI, offset AM
        add SI, 34
        mov AH, 4Ah
        mov BX, 0FFFFh
        int 21h
        mov AX, 16
        mul BX
        call WRD_TO_DEC
        mov DX, offset AM
        call PRINT
        ret
_AM ENDP

_EM PROC near
        mov SI, offset EM
        add SI, 27
        sub DX, DX

        mov AL, 30h  ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h      ; чтение младшего байта
        mov BL, AL    ; размер расширенной памяти
        mov AL, 31h ; запись адреса ячейки CMOS
        out 70h, AL
        in AL, 71h      ; чтение старшего байта
                        ; размера расширенной памяти
        mov AH, AL
        mov Al, BL
        call WRD_TO_DEC
        mov DX, offset EM
        call PRINT
        ret
_EM ENDP

_MCB PROC  near
        mov AH, 52h
        int 21h
        sub BX, 2
        mov AX, ES:[BX]
        mov ES, AX
Chain:
;Type
        mov DI, offset MCB
        add DI, 10
        mov AX, ES:[00h]
        call BYTE_TO_HEX
        mov [DI], AL
        add DI, 1
```

```
        mov [DI], AH
;Adress
        mov DI, offset MCB
        add DI, 29
        mov AX, ES
        call WRD_TO_HEX
;Owner
        mov DI, offset MCB
        add DI, 42
         mov AX, ES:[01h]
        call WRD_TO_HEX
;Size
        mov SI, offset MCB
        add SI, 57
        mov AX, ES:[03h]
        mov BX, 16
        mul BX
        call WRD_TO_DEC
;Print
        mov DX, offset MCB
        call PRINT
;Name
        mov DI, offset MCB
        add DI, 58
        mov BX, 8
        mov CX, 7
        cycle:
        mov DL, ES:[BX]
                mov AH, 02h
                int 21h
                add BX, 1
                loop cycle

        mov AL, ES:[0h]
        cmp AL, 5ah
        je final

        mov BX, ES
        mov AX, ES:[03h]
        add AX, BX
        add AX, 1
        mov ES, AX
        mov DX, offset END_STR
        call PRINT
        jmp Chain
final:
      ret
_MCB ENDP
```

```
BEGIN:
        call _AM
        call _EM

;------- REQUESTING_MEMORY-------
        mov AH,48h
        mov BX,1000h
        int 21h
;-----------------------
        jc NOT_PASSING
        jmp PASSING

        NOT_PASSING:
        mov DX, offset ERR
        call PRINT

        PASSING:
;--------FREEING_MEMORY--------
        mov AH, 04Ah
        mov BX, offset ENDING
        int 21h
;-----------------------
        call _MCB
; Выход в DOS
        xor AL, AL
        mov AH, 4Ch
        int 21h
ENDING:
TESTPC  ENDS
        END START
```