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. 9383 _____ Лысова А.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Рассмотреть нестраничную память и способ управления динамическими разделами. Исследовать структуры данных и работу функций управления памятью ядра и 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.

Основные теоретические положения.

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: 5 Ah, если последний в списке, 4Dh, если не последний
01h	2	Сегментный алрес 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 эгим МСВ можно определить местоположение следующего МСВ В списке.

Адрес первого МСВ хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эгу 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 МСВ.

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,7lh ; чтение младшего байта
mov BL,AL ; размера расширенной памяти
mov AL,3lh ; запись адреса ячейки CMOS
out 70h,AL
```

in AL,7lh ; чтение старшего байта

; размера расширенной памяти

Выполнение работы.

Были написаны тексты исходных загрузочных .COM модулей, выполняющих поставленные задачи. После чего были получены .COM файлы, благодаря компиляции и использованию exe2bin.exe.

Для вывода количества доступной памяти вызывается функция выделения блока памяти 4Ah прерывания 21h с заведомо большим размером памяти, чем может предоставить операционная система. Тогда в регистр BX возвращается размер доступной памяти в параграфах (16 байт). Это число перемещается в регистр AX, умножается его на 16, чтобы получить размер в байтах, и выводится в десятичной системе счисления.

Для вывода размера расширенной памяти происходит обращение к энергонезависимой памяти CMOS, в ячейках 30h и 31h которой хранится необходимая информация. Число кладётся в регистр AX, умножается на 16 и выводится на экран.

Для вывода цепочки блоков управления памятью MCB вызывается функция 52h прерывания 21h, которая возвращает адрес списка списков, слово перед которым является адресом первого MCB. Затем для каждого MCB выводится сегментный адрес PSP владельца участка памяти, размер участка и SC/SD.

Используемые функции.

- HEX_TO_WRD — шаблонная функция, которая преобразует число в регистре AL в код ASCII символа.
- PRINT_HEX — шаблонная функция, которая печатает на экран шестнадцатеричное представление числа на экран.

- PRINT_DEC — шаблонная функция, которая печатает десятичное представление числа.
- ENTR — функция для переноса строки или каретки.
- PRINT_STRING — функция, которая печатает строку на экран.
- PRINT_CHAR — функция, которая печатает символ на экран.
- TABLE— шаблонная функция для вывода таблиц.
- FREE — освобождает память, неиспользуемую программой.
- ALLOC — выделяет программе 64 Кб памяти.

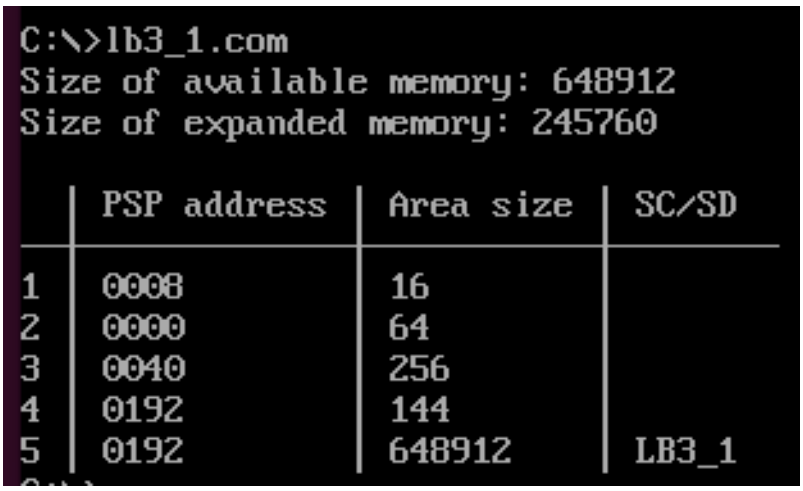

Рисунок 1: Вывод lb3_1

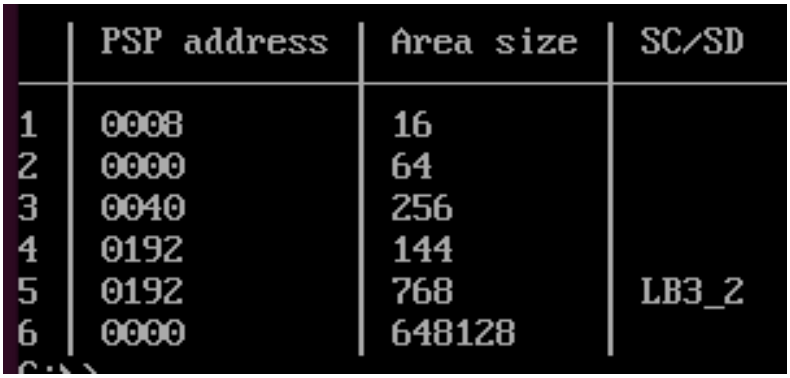

Рисунок 2: Вывод lb3_2

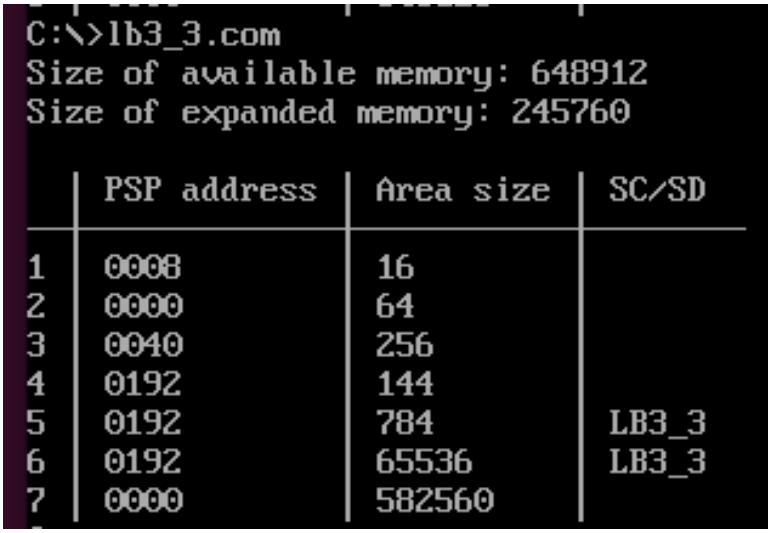

Рисунок 3: Вывод lb3_3

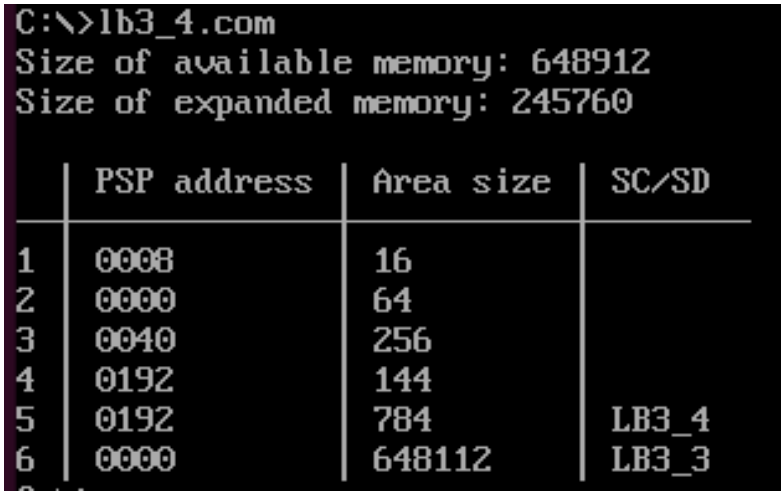

Рисунок 4: B\ывод lb3_4

Контрольные вопросы.

1) Что означает «доступный объем памяти»?

Доступный объём памяти – это часть оперативно памяти, занимаема и используемая программой.

2) Где MCB блок Вашей программы в списке?

MCB блок первой программы находится на последнем месте списка (вывод lb3_1), поскольку программа не освобождает память, которую она не занимает.

MCB блок второй программы находится на предпоследнем месте списка (вывод lb3_2), поскольку программа освободила память, которую она не занимает, соответственно, на последнем месте блок с освобождённой памятью.

MCB блок третьей программы располагается сразу в двух предпоследних местах списка (вывод lb3_3), поскольку программа сперва освободила память, которую она не занимает, а затем выделила новый блок памяти.

MCB блок четвёртой программы находится на предпоследнем месте списка (вывод lb3_4), поскольку программа не смогла выделить 64Кб ввиду того, что весь доступный объём памяти занимает сама программа.

3) Какой размер памяти занимает программа в каждом случае?

Первая программа занимает весь доступный объём памяти, то есть 648912 байт.

Вторая программа занимает только необходимую ей память – 768 байт.

Третья программа занимает необходимую ей память 784 байт и те 64Кб, которые выделяются программой, т.е. 66320 байт.

Четвёртая программа занимает только выделенные ей 784 байта, поскольку выделить 64Кб ей не удалось.

Выводы.

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

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД.

Название файла: lb3_1.asm

```
CODE  SEGMENT
      ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
      ORG 100H
START:

      ENTER    db 0Dh, 0Ah, '$'
      AVAILABLE_MEM   db 'Size of available memory: $'
      EXPANDED_MEM   db 'Size of expanded memory: $'
      HEAD_TABLE   db '  ', 179, ' PSP address ', 179, ' Area size ',
179, ' SC/SD$'
      LINE_TABLE   db 2 dup (196), 197, 13 dup (196), 197, 11 dup (196),
197, 8 dup (196), '$'

PRINT_STRING    PROC near

        push    AX
        mov     AH, 09h
        int     21h
        pop     AX
        ret

PRINT_STRING    ENDP


PRINT_CHAR PROC near

        push    AX
        mov     AH, 02h
        int     21h
        pop     AX
        ret

PRINT_CHAR ENDP
```

```
ENTR  PROC near

        push    AX
        push    DX
        mov     DX, OFFSET ENTER
        mov     AH, 09h
        int     21h
        pop     DX
        pop     AX
        ret

ENTR  ENDP


TABLE PROC near

        push    AX
        push    CX
        push    DX
        xor     CH, CH
        mov     CL, AL
table_loop:
     mov     AH, 02h
        mov     DL, 20h
        int     21h
        loop    table_loop

        mov     AH, 02h
        mov     DL, 179
        int     21h
        mov     AH, 02h
        mov     DL, 20h
        int     21h
        pop     DX
        pop     CX
        pop     AX
        ret
```

```
TABLE ENDP


HEX_TO_WRD PROC near

        cmp     AL, 9
        ja      lttr
        add     AL, 30h
        jmp     ok
lttr:
     add     AL, 37h
ok:
     ret

HEX_TO_WRD ENDP


PRINT_HEX  PROC near

        push    AX
        push    BX
        push    CX
        push    DX
        mov     BX, 0F000h
        mov     DL, 12
        mov     CX, 4

loop_hex:
     push    CX
        push    AX
        and     AX, BX
        mov     CL, DL
        shr     AX, CL
        call    HEX_TO_WRD
        push    DX
        mov     DL, AL
        call    PRINT_CHAR
        pop     DX
        inc     SI
```

```
        pop     AX
        mov     CL, 4
        shr     BX, CL
        sub     DL, 4
        pop     CX
        loop    loop_hex
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        ret

PRINT_HEX  ENDP


PRINT_DEC  PROC near

        push    AX
        push    BX
        push    CX
        push    DX
        mov     BX, 10
        xor     CX, CX
div_dec:
     div     BX
     push    DX
        inc     CX
        xor     DX, DX
        cmp     AX, 0
        jne     div_dec
        mov     DI, CX
loop_dec:
     pop     DX
        xor     DH, DH
        add     DL, 30h
        call    PRINT_CHAR
        loop    loop_dec
        pop     DX
        pop     CX
```

```
        pop     BX
        pop     AX
        ret

PRINT_DEC  ENDP


MAIN  PROC far
; available memory
        mov     DX, OFFSET AVAILABLE_MEM
        call    PRINT_STRING
        mov     AH, 4Ah
        mov     BX, 0FFFFh
        int     21h
        mov     AX, BX
        mov     BX, 16
        mul     BX
        call    PRINT_DEC
        call    ENTR
; expanded memory
        mov     DX, OFFSET EXPANDED_MEM
        call    PRINT_STRING
        mov     AL, 30h
        out     70h, AL
        in      AL, 71h
        mov     AH, AL
        mov     AL, 31h
        out     70h, AL
        in      Al, 71h
        xchg    AH, AL
        mov     BX, 16
        mul     BX
        call    PRINT_DEC
        call    ENTR
; list of MCB
        call    ENTR
        mov     DX, OFFSET HEAD_TABLE
        call    PRINT_STRING
        call    ENTR
```

```
        mov     DX, OFFSET LINE_TABLE
        call    PRINT_STRING
        call    ENTR
        mov     AH, 52h
        int     21h
        mov     ES, ES:[BX-2]
        mov     CX, 1
list:
     mov     AX, CX
        xor     DX, DX
        call    PRINT_DEC
        mov     AL, 1
        call    TABLE
        mov     AX, ES:[01h]
        call    PRINT_HEX
        mov     AL, 8
        call    TABLE
        mov     AX, ES:[03h]
        mov     DX, 16
        mul     DX
        call    PRINT_DEC
        mov     DX, 10
        sub     DX, DI
        mov     AX, DX
        call    TABLE
        push    CX
        push    DS
        mov     AX, ES
        mov     DS, AX
        mov     AH, 40h
        mov     BX, 1
        mov     CX, 8
        mov     DX, OFFSET ES:[08h]
        int     21h
        pop     DS
        pop     CX
        inc     CX
        mov     AL, ES:[00h]
        cmp     AL, 5Ah
```

```
        je      end_main
        cmp     AL, 4Dh
        jne     end_main
        mov     AX, ES:[03h]
        mov     DX, ES
        add     AX, DX
        inc     AX
        mov     ES, AX
        call    ENTR
        jmp     list

end_main:
     mov     AX, 4C00h
        int     21h

MAIN  ENDP

CODE  ENDS
     END   MAIN
```

Название файла: lb3_2.asm

```
CODE  SEGMENT
     ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
     begin = $
     ORG 100H
START:

     ENTER    db 0Dh, 0Ah, '$'
     AVAILABLE_MEM   db 'Size of available memory: $'
     EXPANDED_MEM   db 'Size of expanded memory: $'
     HEAD_TABLE   db '  ', 179, ' PSP address ', 179, ' Area size ',
179, ' SC/SD$'
     LINE_TABLE   db 2 dup (196), 197, 13 dup (196), 197, 11 dup (196),
197, 8 dup (196), '$'

PRINT_STRING    PROC near

        push    AX
```

```
        mov     AH, 09h
        int     21h
        pop     AX
        ret

PRINT_STRING    ENDP


PRINT_CHAR PROC near

        push    AX
        mov     AH, 02h
        int     21h
        pop     AX
        ret

PRINT_CHAR ENDP


ENTR  PROC near

        push    AX
        push    DX
        mov     DX, OFFSET ENTER
        mov     AH, 09h
        int     21h
        pop     DX
        pop     AX
        ret

ENTR  ENDP


TABLE PROC near

        push    AX
        push    CX
        push    DX
        xor     CH, CH
```

```
        mov     CL, AL
table_loop:
     mov     AH, 02h
        mov     DL, 20h
        int     21h
        loop    table_loop

        mov     AH, 02h
        mov     DL, 179
        int     21h
        mov     AH, 02h
        mov     DL, 20h
        int     21h
        pop     DX
        pop     CX
        pop     AX
        ret

TABLE ENDP


FREE  PROC near

        push    AX
        push    BX
        push    DX
        xor     DX, DX
        mov     AX, endproc - begin
        mov     BX, 16
        div     BX
        mov     BX, AX
        inc     BX
        mov     AH, 4Ah
        int     21h
        pop     DX
        pop     BX
        pop     AX
        ret
```

```
FREE  ENDP


HEX_TO_WRD PROC near

        cmp     AL, 9
        ja      lttr
        add     AL, 30h
        jmp     ok
lttr:
     add     AL, 37h
ok:
     ret

HEX_TO_WRD ENDP


PRINT_HEX  PROC near

        push    AX
        push    BX
        push    CX
        push    DX
        mov     BX, 0F000h
        mov     DL, 12
        mov     CX, 4

loop_hex:
     push    CX
        push    AX
        and     AX, BX
        mov     CL, DL
        shr     AX, CL
        call    HEX_TO_WRD
        push    DX
        mov     DL, AL
        call    PRINT_CHAR
        pop     DX
        inc     SI
```

```
        pop     AX
        mov     CL, 4
        shr     BX, CL
        sub     DL, 4
        pop     CX
        loop    loop_hex
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        ret

PRINT_HEX  ENDP


PRINT_DEC  PROC near

        push    AX
        push    BX
        push    CX
        push    DX
        mov     BX, 10
        xor     CX, CX
div_dec:
     div     BX
     push    DX
        inc     CX
        xor     DX, DX
        cmp     AX, 0
        jne     div_dec
        mov     DI, CX
loop_dec:
     pop     DX
        xor     DH, DH
        add     DL, 30h
        call    PRINT_CHAR
        loop    loop_dec
        pop     DX
        pop     CX
```

```
        pop     BX
        pop     AX
        ret

PRINT_DEC  ENDP


MAIN  PROC far
; available memory
        mov     DX, OFFSET AVAILABLE_MEM
        call    PRINT_STRING
        mov     AH, 4Ah
        mov     BX, 0FFFFh
        int     21h
        mov     AX, BX
        mov     BX, 16
        mul     BX
        call    PRINT_DEC
        call    ENTR
; expanded memory
        mov     DX, OFFSET EXPANDED_MEM
        call    PRINT_STRING
        mov     AL, 30h
        out     70h, AL
        in      AL, 71h
        mov     AH, AL
        mov     AL, 31h
        out     70h, AL
        in      Al, 71h
        xchg    AH, AL
        mov     BX, 16
        mul     BX
        call    PRINT_DEC
        call    ENTR

        call    FREE

; list of MCB
        call    ENTR
```

```
        mov     DX, OFFSET HEAD_TABLE
        call    PRINT_STRING
        call    ENTR
        mov     DX, OFFSET LINE_TABLE
        call    PRINT_STRING
        call    ENTR
        mov     AH, 52h
        int     21h
        mov     ES, ES:[BX-2]
        mov     CX, 1
list:
     mov     AX, CX
        xor     DX, DX
        call    PRINT_DEC
        mov     AL, 1
        call    TABLE
        mov     AX, ES:[01h]
        call    PRINT_HEX
        mov     AL, 8
        call    TABLE
        mov     AX, ES:[03h]
        mov     DX, 16
        mul     DX
        call    PRINT_DEC
        mov     DX, 10
        sub     DX, DI
        mov     AX, DX
        call    TABLE
        push    CX
        push    DS
        mov     AX, ES
        mov     DS, AX
        mov     AH, 40h
        mov     BX, 1
        mov     CX, 8
        mov     DX, OFFSET ES:[08h]
        int     21h
        pop     DS
        pop     CX
```

```
        inc     CX
        mov     AL, ES:[00h]
        cmp     AL, 5Ah
        je      end_main
        cmp     AL, 4Dh
        jne     end_main
        mov     AX, ES:[03h]
        mov     DX, ES
        add     AX, DX
        inc     AX
        mov     ES, AX
        call    ENTR
        jmp     list

end_main:
     mov     AX, 4C00h
        int     21h

MAIN  ENDP

endproc = $

CODE  ENDS
     END   MAIN
```

Название файла: lb3_3.asm

```
CODE  SEGMENT
     ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
     begin = $
     ORG 100H
START:

     ENTER    db 0Dh, 0Ah, '$'
     AVAILABLE_MEM   db 'Size of available memory: $'
     EXPANDED_MEM   db 'Size of expanded memory: $'
     HEAD_TABLE   db '  ', 179, ' PSP address ', 179, ' Area size ',
179, ' SC/SD$'
```

```
     LINE_TABLE   db 2 dup (196), 197, 13 dup (196), 197, 11 dup (196),
197, 8 dup (196), '$'

PRINT_STRING    PROC near

        push    AX
        mov     AH, 09h
        int     21h
        pop     AX
        ret

PRINT_STRING    ENDP


PRINT_CHAR PROC near

        push    AX
        mov     AH, 02h
        int     21h
        pop     AX
        ret

PRINT_CHAR ENDP


ENTR  PROC near

        push    AX
        push    DX
        mov     DX, OFFSET ENTER
        mov     AH, 09h
        int     21h
        pop     DX
        pop     AX
        ret

ENTR  ENDP
```

```
TABLE PROC near

        push    AX
        push    CX
        push    DX
        xor     CH, CH
        mov     CL, AL
table_loop:
     mov     AH, 02h
        mov     DL, 20h
        int     21h
        loop    table_loop

        mov     AH, 02h
        mov     DL, 179
        int     21h
        mov     AH, 02h
        mov     DL, 20h
        int     21h
        pop     DX
        pop     CX
        pop     AX
        ret

TABLE ENDP


FREE  PROC near

        push    AX
        push    BX
        push    DX
        xor     DX, DX
        mov     AX, endproc - begin
        mov     BX, 16
        div     BX
        mov     BX, AX
        inc     BX
        mov     AH, 4Ah
```

```
        int     21h
        pop     DX
        pop     BX
        pop     AX
        ret

FREE  ENDP


ALLOC PROC near

        push    AX
        push    BX
        mov     AH, 48h
        mov     BX, 4096
        int     21h
        pop     BX
        pop     AX
        ret

ALLOC ENDP


HEX_TO_WRD PROC near

        cmp     AL, 9
        ja      lttr
        add     AL, 30h
        jmp     ok
lttr:
     add     AL, 37h
ok:
     ret

HEX_TO_WRD ENDP


PRINT_HEX  PROC near
```

```
        push    AX
        push    BX
        push    CX
        push    DX
        mov     BX, 0F000h
        mov     DL, 12
        mov     CX, 4

loop_hex:
     push    CX
        push    AX
        and     AX, BX
        mov     CL, DL
        shr     AX, CL
        call    HEX_TO_WRD
        push    DX
        mov     DL, AL
        call    PRINT_CHAR
        pop     DX
        inc     SI
        pop     AX
        mov     CL, 4
        shr     BX, CL
        sub     DL, 4
        pop     CX
        loop    loop_hex
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        ret

PRINT_HEX  ENDP


PRINT_DEC  PROC near

        push    AX
        push    BX
```

```
        push    CX
        push    DX
        mov     BX, 10
        xor     CX, CX
div_dec:
     div     BX
     push    DX
        inc     CX
        xor     DX, DX
        cmp     AX, 0
        jne     div_dec
        mov     DI, CX
loop_dec:
     pop     DX
        xor     DH, DH
        add     DL, 30h
        call    PRINT_CHAR
        loop    loop_dec
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        ret

PRINT_DEC  ENDP


MAIN  PROC far
; available memory
        mov     DX, OFFSET AVAILABLE_MEM
        call    PRINT_STRING
        mov     AH, 4Ah
        mov     BX, 0FFFFh
        int     21h
        mov     AX, BX
        mov     BX, 16
        mul     BX
        call    PRINT_DEC
        call    ENTR
```

```
; expanded memory
        mov     DX, OFFSET EXPANDED_MEM
        call    PRINT_STRING
        mov     AL, 30h
        out     70h, AL
        in      AL, 71h
        mov     AH, AL
        mov     AL, 31h
        out     70h, AL
        in      Al, 71h
        xchg    AH, AL
        mov     BX, 16
        mul     BX
        call    PRINT_DEC
        call    ENTR

        call    FREE
        call    ALLOC

; list of MCB
        call    ENTR
        mov     DX, OFFSET HEAD_TABLE
        call    PRINT_STRING
        call    ENTR
        mov     DX, OFFSET LINE_TABLE
        call    PRINT_STRING
        call    ENTR
        mov     AH, 52h
        int     21h
        mov     ES, ES:[BX-2]
        mov     CX, 1
list:
     mov     AX, CX
        xor     DX, DX
        call    PRINT_DEC
        mov     AL, 1
        call    TABLE
        mov     AX, ES:[01h]
        call    PRINT_HEX
```

```
        mov     AL, 8
        call    TABLE
        mov     AX, ES:[03h]
        mov     DX, 16
        mul     DX
        call    PRINT_DEC
        mov     DX, 10
        sub     DX, DI
        mov     AX, DX
        call    TABLE
        push    CX
        push    DS
        mov     AX, ES
        mov     DS, AX
        mov     AH, 40h
        mov     BX, 1
        mov     CX, 8
        mov     DX, OFFSET ES:[08h]
        int     21h
        pop     DS
        pop     CX
        inc     CX
        mov     AL, ES:[00h]
        cmp     AL, 5Ah
        je      end_main
        cmp     AL, 4Dh
        jne     end_main
        mov     AX, ES:[03h]
        mov     DX, ES
        add     AX, DX
        inc     AX
        mov     ES, AX
        call    ENTR
        jmp     list

end_main:
     mov     AX, 4C00h
        int     21h
```

```
MAIN  ENDP

endproc = $

CODE  ENDS
      END   MAIN
```

Название файла: lb3_4.asm

```
CODE  SEGMENT
      ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
      begin = $
      ORG 100H
START:

      ENTER    db 0Dh, 0Ah, '$'
      AVAILABLE_MEM   db 'Size of available memory: $'
      EXPANDED_MEM   db 'Size of expanded memory: $'
      HEAD_TABLE   db '  ', 179, ' PSP address ', 179, ' Area size ',
179, ' SC/SD$'
      LINE_TABLE   db 2 dup (196), 197, 13 dup (196), 197, 11 dup (196),
197, 8 dup (196), '$'

PRINT_STRING    PROC near

        push    AX
        mov     AH, 09h
        int     21h
        pop     AX
        ret

PRINT_STRING    ENDP


PRINT_CHAR PROC near

        push    AX
        mov     AH, 02h
```

```
        int     21h
        pop     AX
        ret

PRINT_CHAR ENDP


ENTR  PROC near

        push    AX
        push    DX
        mov     DX, OFFSET ENTER
        mov     AH, 09h
        int     21h
        pop     DX
        pop     AX
        ret

ENTR  ENDP


TABLE PROC near

        push    AX
        push    CX
        push    DX
        xor     CH, CH
        mov     CL, AL
table_loop:
     mov     AH, 02h
        mov     DL, 20h
        int     21h
        loop    table_loop

        mov     AH, 02h
        mov     DL, 179
        int     21h
        mov     AH, 02h
        mov     DL, 20h
```

```
        int     21h
        pop     DX
        pop     CX
        pop     AX
        ret

TABLE ENDP


FREE  PROC near

        push    AX
        push    BX
        push    DX
        xor     DX, DX
        mov     AX, endproc - begin
        mov     BX, 16
        div     BX
        mov     BX, AX
        inc     BX
        mov     AH, 4Ah
        int     21h
        pop     DX
        pop     BX
        pop     AX
        ret

FREE  ENDP


ALLOC PROC near

        push    AX
        push    BX
        mov     AH, 48h
        mov     BX, 4096
        int     21h
        pop     BX
        pop     AX
```

```
        ret

ALLOC ENDP


HEX_TO_WRD PROC near

        cmp     AL, 9
        ja      lttr
        add     AL, 30h
        jmp     ok
lttr:
     add     AL, 37h
ok:
     ret

HEX_TO_WRD ENDP


PRINT_HEX  PROC near

        push    AX
        push    BX
        push    CX
        push    DX
        mov     BX, 0F000h
        mov     DL, 12
        mov     CX, 4

loop_hex:
     push    CX
        push    AX
        and     AX, BX
        mov     CL, DL
        shr     AX, CL
        call    HEX_TO_WRD
        push    DX
        mov     DL, AL
        call    PRINT_CHAR
```

```
        pop     DX
        inc     SI
        pop     AX
        mov     CL, 4
        shr     BX, CL
        sub     DL, 4
        pop     CX
        loop    loop_hex
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        ret

PRINT_HEX  ENDP


PRINT_DEC  PROC near

        push    AX
        push    BX
        push    CX
        push    DX
        mov     BX, 10
        xor     CX, CX
div_dec:
     div     BX
     push    DX
        inc     CX
        xor     DX, DX
        cmp     AX, 0
        jne     div_dec
        mov     DI, CX
loop_dec:
     pop     DX
        xor     DH, DH
        add     DL, 30h
        call    PRINT_CHAR
        loop    loop_dec
```

```
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        ret

PRINT_DEC  ENDP


MAIN  PROC far
; available memory
        mov     DX, OFFSET AVAILABLE_MEM
        call    PRINT_STRING
        mov     AH, 4Ah
        mov     BX, 0FFFFh
        int     21h
        mov     AX, BX
        mov     BX, 16
        mul     BX
        call    PRINT_DEC
        call    ENTR
; expanded memory
        mov     DX, OFFSET EXPANDED_MEM
        call    PRINT_STRING
        mov     AL, 30h
        out     70h, AL
        in      AL, 71h
        mov     AH, AL
        mov     AL, 31h
        out     70h, AL
        in      Al, 71h
        xchg    AH, AL
        mov     BX, 16
        mul     BX
        call    PRINT_DEC
        call    ENTR

        call    ALLOC
        call    FREE
```

```
; list of MCB
        call    ENTR
        mov     DX, OFFSET HEAD_TABLE
        call    PRINT_STRING
        call    ENTR
        mov     DX, OFFSET LINE_TABLE
        call    PRINT_STRING
        call    ENTR
        mov     AH, 52h
        int     21h
        mov     ES, ES:[BX-2]
        mov     CX, 1
list:
     mov     AX, CX
        xor     DX, DX
        call    PRINT_DEC
        mov     AL, 1
        call    TABLE
        mov     AX, ES:[01h]
        call    PRINT_HEX
        mov     AL, 8
        call    TABLE
        mov     AX, ES:[03h]
        mov     DX, 16
        mul     DX
        call    PRINT_DEC
        mov     DX, 10
        sub     DX, DI
        mov     AX, DX
        call    TABLE
        push    CX
        push    DS
        mov     AX, ES
        mov     DS, AX
        mov     AH, 40h
        mov     BX, 1
        mov     CX, 8
        mov     DX, OFFSET ES:[08h]
```

```
        int     21h
        pop     DS
        pop     CX
        inc     CX
        mov     AL, ES:[00h]
        cmp     AL, 5Ah
        je      end_main
        cmp     AL, 4Dh
        jne     end_main
        mov     AX, ES:[03h]
        mov     DX, ES
        add     AX, DX
        inc     AX
        mov     ES, AX
        call    ENTR
        jmp     list

end_main:
     mov     AX, 4C00h
        int     21h

MAIN  ENDP

endproc = $

CODE  ENDS
     END   MAIN
```