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. 8382 Мирончик П.Д.

Преподаватель Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование структур данных и работы функций управления памятью ядра операционной системы

Ход работы.

1. Была написана программа, выводящая список MCB, доступную память и расширенную память. На рисунке видно, что программа занимает всю доступную память.

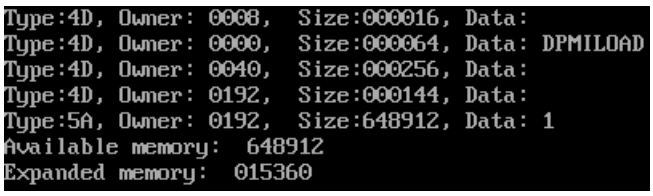

Рис.1. Результат работы программы 1

2. Добавлено освобождение памяти, неиспользуемой программой. Видно, что память, занятая программой, уменьшилась до 784 байт, а за ней появился свободный блок, занимающий оставшуюся часть памяти.

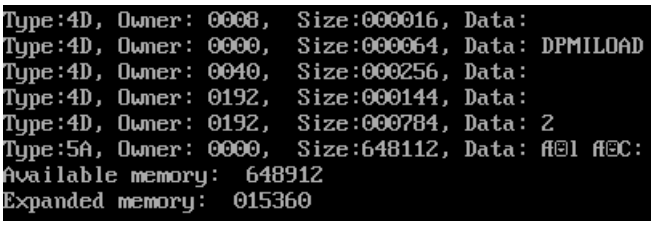

Рис.2. Результат работы программы 2

3. Добавлен запрос 64 Кб памяти при помощи функции 48h прерывания 21h. Видно, что сразу за блоком памяти программы появился блок, размером 64Кб, а размер памяти, доступный для расширения изначального блока программы, составил 784 байта.

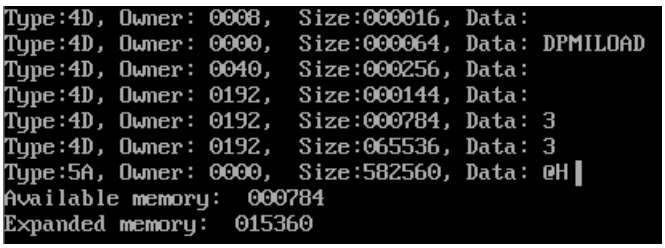

Рис.3. Результат работы программы 3

4. Далее, функции освобождения памяти и выделения нового блока памяти поменяли местами. В результате при попытке выделения 64Кб памяти была получена ошибка с кодом 8, и память не была выделена. При освобождении памяти блок программы был успешно уменьшен до 784 байт.

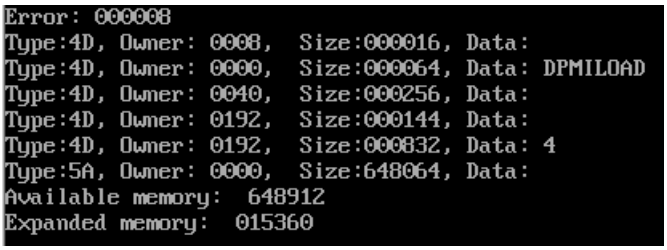

Рис.4. Результат работы программы 4

Ответы на вопросы

1. Что означает доступный объем памяти?

Объем памяти, доступный программе. Из задания не ясно, что именно подразумевалось под этим определением. Если говорить о перераспределении памяти, то это разница между перераспределяемым блоком и началом следующего занятого блока. Если же говорится о выделении памяти в новый блок, то это максимальный доступный свободный блок памяти.

2. Где MCB блок Вашей программы в списке?

В первой, второй и пятой программе это 4 и 5 блоки, в 3 программе – 4, 5, 6 блоки.

Это видно из адреса PSP владельца а также из содержимого последних 8 байт MCB

3. Какой размер памяти занимает программа в каждом случае?

Первая 649 052 байт

Вторая 928 байт

Третья 66 464 байт (с учетом выбеленного блока в 64Кб)

Четвертая 976 байт

Выводы

В ходе работы была исследована работа с функциями управления памятью ядра операционной системы на языке ассемблер.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПЕРВОЙ ПРОГРАММЫ

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

START:  JMP BEGIN
; DATA
ENDL db 13,10,'$'
MCB_DATA_TEXT db 'Type:  , Owner:     ,  Size:      , Data: $'
AVAILABLE_MEMORY_TEXT db 'Available memory:       ',13,10,'$'
EXPANDED_MEMORY_TEXT db 'Expanded memory:       ',13,10,'$'

; PROCEDURES
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
; input:        AL=F8h (число)
; output:       AL={f}, AH={8} (в фигурных скобках символы)
;
; переводит AL в два символа в 16-й сс в AX
; в AL находится старшая, в AH младшая цифры
        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
; input:        AX=FH7Ah (число)
;               DI={адрес} (указатель на последний символ в памяти, куда
будет записан результат)
; output:       начиная с [DI-3] лежат символы числа в 16-й сс
;               AX не сохраняет начальное значение
;
; перевод AX в 16-ю сс
        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
; input:        AL=0Fh (число)
;               SI={адрес} (адрес поля младшей цифры)
;
; перевод AL в 10-ю сс
        push    cx
        push    dx
        push    ax
        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     ax,10
        je      end_l
        or      al,30h
        mov     [si],al
end_l:
        pop     ax
        pop     dx
        pop     cx
        ret
BYTE_TO_DEC     ENDP

WRITE_AL_HEX PROC NEAR
        push ax
        push dx
        call BYTE_TO_HEX

        mov dl, al
        mov al, ah
        mov ah, 02h
        int 21h

        mov dl, al
        int 21h
        pop dx
        pop ax
        ret
WRITE_AL_HEX ENDP

; custom functions
```

```
WRD_TO_DEC PROC near
; input ax - value
;       di - lower num address
;       si - address of highest available num position (DI-max), or 0 if
;            prefix isn't need
;
; converts AX to DEC and writes to di address (to DI, DI-1, DI-2, ...)
        push bx
        push dx
        push di
        push si
        push ax

        mov bx, 10
        WRD_TO_DEC_loop:
                div bx
                add dl, '0'
                mov [di], dl
                xor dx, dx
                dec di
                cmp ax, 0
                jne WRD_TO_DEC_loop

        cmp si, 0
        je WRD_TO_DEC_no_prefix
        cmp si, di
        jge WRD_TO_DEC_no_prefix
        WRD_TO_DEC_prefix_loop:
                mov dl, '0'
                mov [di], dl
                dec di
                cmp di, si
                jg WRD_TO_DEC_prefix_loop

WRD_TO_DEC_no_prefix:
        pop ax
        pop si
        pop di
        pop dx
        pop bx
        ret
WRD_TO_DEC ENDP

PRINT_MCB PROC NEAR
        push ax
        push di
        push es
        push bx

        mov ah, 52h
        int 21h

        mov es, es:[bx-2]

        PRINT_MCB_loop:
                ; set type
                mov al, es:[0h]
```

```
        call BYTE_TO_HEX
        mov di, offset MCB_DATA_TEXT
        add di, 5
        mov [di], al
        inc di
        mov [di], ah

        ; set owner
        mov ax, es:[01h]
        mov di, offset MCB_DATA_TEXT
        add di, 19
        call WRD_TO_HEX

        ; set size
        mov ax, es:[03h]
        mov bx, 16
        mul bx
        mov di, offset MCB_DATA_TEXT
        mov si, offset MCB_DATA_TEXT
        add di, 33
        add si, 27
        call WRD_TO_DEC

        ; print MCB
        mov dx, offset MCB_DATA_TEXT
        mov ah, 09h
        int 21h

        ; print MCB's data
        mov cx, 8
        mov si, 08h
        mov di, offset MCB_DATA_TEXT
        add di, 42
        PRINT_MCB_data_loop:
                mov dl, es:[si]
                mov ah, 02h
                int 21h
                inc si
                loop PRINT_MCB_data_loop
        mov dx, offset ENDL
        mov ah, 09h
        int 21h

        mov al, es:[0h]
        cmp al, 5Ah
        je PRINT_MCB_end

        mov ax, es
        add ax, es:[3h]
        inc ax
        mov es, ax
        jmp PRINT_MCB_loop

PRINT_MCB_end:
        pop bx
        pop es
        pop di
        pop ax
```

```
                ret
PRINT_MCB ENDP

PRINT_AVAILABLE_MEMORY PROC NEAR
        push ax
        push bx
        push di
        push dx
        push es

        mov ax, cs
        mov es, ax
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h

        mov ax, 16
        mul bx
        mov di, offset AVAILABLE_MEMORY_TEXT
        mov si, offset AVAILABLE_MEMORY_TEXT
        add di, 24
        add si, 18
        CALL WRD_TO_DEC

        mov dx, offset AVAILABLE_MEMORY_TEXT
        mov ah, 09h
        int 21h

        pop es
        pop dx
        pop di
        pop bx
        pop ax
        ret
PRINT_AVAILABLE_MEMORY ENDP

PRINT_EXPANDED_MEMORY PROC near
        push ax
        push di
        push bx

        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 di, offset EXPANDED_MEMORY_TEXT
        mov si, offset EXPANDED_MEMORY_TEXT
        add di, 23
        add si, 17
        mov dx, 0
        CALL WRD_TO_DEC
```

```
        mov dx, offset EXPANDED_MEMORY_TEXT
        mov ah, 09h
        int 21h

        pop bx
        pop di
        pop ax
        ret
PRINT_EXPANDED_MEMORY ENDP

;---------------------------
; CODE
BEGIN:
        call PRINT_MCB
        call PRINT_AVAILABLE_MEMORY
        call PRINT_EXPANDED_MEMORY

; Выход в DOS
        xor     al,al
        mov     ah,4Ch
        int     21h
MAIN    ENDS
        END START
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ВТОРОЙ ПРОГРАММЫ

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

START:  JMP BEGIN
; DATA
ENDL db 13,10,'$'
MCB_DATA_TEXT db 'Type:  , Owner:     ,  Size:      , Data: $'
AVAILABLE_MEMORY_TEXT db 'Available memory:       ',13,10,'$'
EXPANDED_MEMORY_TEXT db 'Expanded memory:       ',13,10,'$'

; PROCEDURES
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
; input:        AL=F8h (число)
; output:       AL={f}, AH={8} (в фигурных скобках символы)
;
; переводит AL в два символа в 16-й сс в AX
; в AL находится старшая, в AH младшая цифры
        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
; input:        AX=FH7Ah (число)
;               DI={адрес} (указатель на последний символ в памяти, куда
будет записан результат)
; output:       начиная с [DI-3] лежат символы числа в 16-й сс
;               AX не сохраняет начальное значение
;
; перевод AX в 16-ю сс
        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
; input:        AL=0Fh (число)
;               SI={адрес} (адрес поля младшей цифры)
;
; перевод AL в 10-ю сс
        push    cx
        push    dx
        push    ax
        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     ax,10
        je      end_l
        or      al,30h
        mov     [si],al
end_l:
        pop     ax
        pop     dx
        pop     cx
        ret
BYTE_TO_DEC     ENDP

WRITE_AL_HEX PROC NEAR
        push ax
        push dx
        call BYTE_TO_HEX

        mov dl, al
        mov al, ah
        mov ah, 02h
        int 21h

        mov dl, al
        int 21h
        pop dx
        pop ax
        ret
WRITE_AL_HEX ENDP
```

```
; custom functions

WRD_TO_DEC PROC near
; input ax - value
;       di - lower num address
;       si - address of highest available num position (DI-max), or 0 if
;            prefix isn't need
;
; converts AX to DEC and writes to di address (to DI, DI-1, DI-2, ...)
        push bx
        push dx
        push di
        push si
        push ax

        mov bx, 10
        WRD_TO_DEC_loop:
                div bx
                add dl, '0'
                mov [di], dl
                xor dx, dx
                dec di
                cmp ax, 0
                jne WRD_TO_DEC_loop

        cmp si, 0
        je WRD_TO_DEC_no_prefix
        cmp si, di
        jge WRD_TO_DEC_no_prefix
        WRD_TO_DEC_prefix_loop:
                mov dl, '0'
                mov [di], dl
                dec di
                cmp di, si
                jg WRD_TO_DEC_prefix_loop

WRD_TO_DEC_no_prefix:
        pop ax
        pop si
        pop di
        pop dx
        pop bx
        ret
WRD_TO_DEC ENDP

PRINT_MCB PROC NEAR
        push ax
        push di
        push es
        push bx

        mov ah, 52h
        int 21h

        mov es, es:[bx-2]

        PRINT_MCB_loop:
                ; set type
```

```
        mov al, es:[0h]
        call BYTE_TO_HEX
        mov di, offset MCB_DATA_TEXT
        add di, 5
        mov [di], al
        inc di
        mov [di], ah

        ; set owner
        mov ax, es:[01h]
        mov di, offset MCB_DATA_TEXT
        add di, 19
        call WRD_TO_HEX

        ; set size
        mov ax, es:[03h]
        mov bx, 16
        mul bx
        mov di, offset MCB_DATA_TEXT
        mov si, offset MCB_DATA_TEXT
        add di, 33
        add si, 27
        call WRD_TO_DEC

        ; print MCB
        mov dx, offset MCB_DATA_TEXT
        mov ah, 09h
        int 21h

        ; print MCB's data
        mov cx, 8
        mov si, 08h
        mov di, offset MCB_DATA_TEXT
        add di, 42
        PRINT_MCB_data_loop:
                mov dl, es:[si]
                mov ah, 02h
                int 21h
                inc si
                loop PRINT_MCB_data_loop
        mov dx, offset ENDL
        mov ah, 09h
        int 21h

        mov al, es:[0h]
        cmp al, 5Ah
        je PRINT_MCB_end

        mov ax, es
        add ax, es:[3h]
        inc ax
        mov es, ax
        jmp PRINT_MCB_loop

PRINT_MCB_end:
        pop bx
        pop es
        pop di
```

```
                pop ax
                ret
PRINT_MCB ENDP

PRINT_AVAILABLE_MEMORY PROC NEAR
        push ax
        push bx
        push di
        push dx

        mov ax, cs
        mov es, ax
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h

        mov ax, 16
        mul bx
        mov di, offset AVAILABLE_MEMORY_TEXT
        mov si, offset AVAILABLE_MEMORY_TEXT
        add di, 24
        add si, 18
        CALL WRD_TO_DEC

        mov dx, offset AVAILABLE_MEMORY_TEXT
        mov ah, 09h
        int 21h

        pop dx
        pop di
        pop bx
        pop ax
        ret
PRINT_AVAILABLE_MEMORY ENDP

PRINT_EXPANDED_MEMORY PROC near
        push ax
        push di
        push bx

        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 di, offset EXPANDED_MEMORY_TEXT
        mov si, offset EXPANDED_MEMORY_TEXT
        add di, 23
        add si, 17
        mov dx, 0
        CALL WRD_TO_DEC

        mov dx, offset EXPANDED_MEMORY_TEXT
```

```
        mov ah, 09h
        int 21h

        pop bx
        pop di
        pop ax
        ret
PRINT_EXPANDED_MEMORY ENDP

;--------------------------
; CODE
BEGIN:
        ; compute minimal size we need
        mov dx, 0
        mov ax, offset END_POINTER
        mov bx, 16
        div bx

        inc ax
        mov bx, ax
        mov ah, 04Ah
        int 21h

        call PRINT_MCB
        call PRINT_AVAILABLE_MEMORY
        call PRINT_EXPANDED_MEMORY

; Выход в DOS
        xor     al,al
        mov     ah,4Ch
        int     21h
END_POINTER:
MAIN    ENDS
        END START
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ТРЕТЬЕЙ ПРОГРАММЫ

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

START:  JMP BEGIN
; DATA
ENDL db 13,10,'$'
MCB_DATA_TEXT db 'Type:  , Owner:     ,  Size:      , Data: $'
AVAILABLE_MEMORY_TEXT db 'Available memory:       ',13,10,'$'
EXPANDED_MEMORY_TEXT db 'Expanded memory:       ',13,10,'$'

; PROCEDURES
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
; input:        AL=F8h (число)
; output:       AL={f}, AH={8} (в фигурных скобках символы)
;
; переводит AL в два символа в 16-й сс в AX
; в AL находится старшая, в AH младшая цифры
        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
; input:        AX=FH7Ah (число)
;               DI={адрес} (указатель на последний символ в памяти, куда
будет записан результат)
; output:       начиная с [DI-3] лежат символы числа в 16-й сс
;               AX не сохраняет начальное значение
;
; перевод AX в 16-ю сс
        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
; input:        AL=0Fh (число)
;               SI={адрес} (адрес поля младшей цифры)
;
; перевод AL в 10-ю сс
        push    cx
        push    dx
        push    ax
        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     ax,10
        je      end_l
        or      al,30h
        mov     [si],al
end_l:
        pop     ax
        pop     dx
        pop     cx
        ret
BYTE_TO_DEC     ENDP

WRITE_AL_HEX PROC NEAR
        push ax
        push dx
        call BYTE_TO_HEX

        mov dl, al
        mov al, ah
        mov ah, 02h
        int 21h

        mov dl, al
        int 21h
        pop dx
        pop ax
        ret
WRITE_AL_HEX ENDP

; custom functions
```

```
WRD_TO_DEC PROC near
; input ax - value
;       di - lower num address
;       si - address of highest available num position (DI-max), or 0 if
;            prefix isn't need
;
; converts AX to DEC and writes to di address (to DI, DI-1, DI-2, ...)
        push bx
        push dx
        push di
        push si
        push ax

        mov bx, 10
        WRD_TO_DEC_loop:
                div bx
                add dl, '0'
                mov [di], dl
                xor dx, dx
                dec di
                cmp ax, 0
                jne WRD_TO_DEC_loop

        cmp si, 0
        je WRD_TO_DEC_no_prefix
        cmp si, di
        jge WRD_TO_DEC_no_prefix
        WRD_TO_DEC_prefix_loop:
                mov dl, '0'
                mov [di], dl
                dec di
                cmp di, si
                jg WRD_TO_DEC_prefix_loop

WRD_TO_DEC_no_prefix:
        pop ax
        pop si
        pop di
        pop dx
        pop bx
        ret
WRD_TO_DEC ENDP

PRINT_MCB PROC NEAR
        push ax
        push di
        push es
        push bx

        mov ah, 52h
        int 21h

        mov es, es:[bx-2]

        PRINT_MCB_loop:
                ; set type
                mov al, es:[0h]
```

```
        call BYTE_TO_HEX
        mov di, offset MCB_DATA_TEXT
        add di, 5
        mov [di], al
        inc di
        mov [di], ah

        ; set owner
        mov ax, es:[01h]
        mov di, offset MCB_DATA_TEXT
        add di, 19
        call WRD_TO_HEX

        ; set size
        mov ax, es:[03h]
        mov bx, 16
        mul bx
        mov di, offset MCB_DATA_TEXT
        mov si, offset MCB_DATA_TEXT
        add di, 33
        add si, 27
        call WRD_TO_DEC

        ; print MCB
        mov dx, offset MCB_DATA_TEXT
        mov ah, 09h
        int 21h

        ; print MCB's data
        mov cx, 8
        mov si, 08h
        mov di, offset MCB_DATA_TEXT
        add di, 42
        PRINT_MCB_data_loop:
                mov dl, es:[si]
                mov ah, 02h
                int 21h
                inc si
                loop PRINT_MCB_data_loop
        mov dx, offset ENDL
        mov ah, 09h
        int 21h

        mov al, es:[0h]
        cmp al, 5Ah
        je PRINT_MCB_end

        mov ax, es
        add ax, es:[3h]
        inc ax
        mov es, ax
        jmp PRINT_MCB_loop

PRINT_MCB_end:
        pop bx
        pop es
        pop di
        pop ax
```

```
                ret
PRINT_MCB ENDP

PRINT_AVAILABLE_MEMORY PROC NEAR
        push ax
        push bx
        push di
        push dx

        mov ax, cs
        mov es, ax
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h

        mov ax, 16
        mul bx
        mov di, offset AVAILABLE_MEMORY_TEXT
        mov si, offset AVAILABLE_MEMORY_TEXT
        add di, 24
        add si, 18
        CALL WRD_TO_DEC

        mov dx, offset AVAILABLE_MEMORY_TEXT
        mov ah, 09h
        int 21h

        pop dx
        pop di
        pop bx
        pop ax
        ret
PRINT_AVAILABLE_MEMORY ENDP

PRINT_EXPANDED_MEMORY PROC near
        push ax
        push di
        push bx

        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 di, offset EXPANDED_MEMORY_TEXT
        mov si, offset EXPANDED_MEMORY_TEXT
        add di, 23
        add si, 17
        mov dx, 0
        CALL WRD_TO_DEC

        mov dx, offset EXPANDED_MEMORY_TEXT
        mov ah, 09h
```

```
        int 21h

        pop bx
        pop di
        pop ax
        ret
PRINT_EXPANDED_MEMORY ENDP

;--------------------------
; CODE
BEGIN:
        ; compute minimal size we need
        mov dx, 0
        mov ax, offset END_POINTER
        mov bx, 16
        div bx

        inc ax
        mov bx, ax
        mov ah, 04Ah
        int 21h

        mov bx, 1000h
        mov ah, 48h
        int 21h

        call PRINT_MCB
        call PRINT_AVAILABLE_MEMORY
        call PRINT_EXPANDED_MEMORY

; Выход в DOS
        xor     al,al
        mov     ah,4Ch
        int     21h
END_POINTER:
MAIN    ENDS
        END START
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ЧЕТВЕРТОЙ ПРОГРАММЫ

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

START:  JMP BEGIN
; DATA
ENDL db 13,10,'$'
MCB_DATA_TEXT db 'Type:  , Owner:     ,  Size:      , Data: $'
AVAILABLE_MEMORY_TEXT db 'Available memory:       ',13,10,'$'
EXPANDED_MEMORY_TEXT db 'Expanded memory:       ',13,10,'$'
ERROR_TEXT db 'Error: 000000'

; PROCEDURES
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
; input:        AL=F8h (число)
; output:       AL={f}, AH={8} (в фигурных скобках символы)
;
; переводит AL в два символа в 16-й сс в AX
; в AL находится старшая, в AH младшая цифры
        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
; input:        AX=FH7Ah (число)
;               DI={адрес} (указатель на последний символ в памяти, куда
будет записан результат)
; output:       начиная с [DI-3] лежат символы числа в 16-й сс
;               AX не сохраняет начальное значение
;
; перевод AX в 16-ю сс
        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
; input:        AL=0Fh (число)
;               SI={адрес} (адрес поля младшей цифры)
;
; перевод AL в 10-ю сс
        push    cx
        push    dx
        push    ax
        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     ax,10
        je      end_l
        or      al,30h
        mov     [si],al
end_l:
        pop     ax
        pop     dx
        pop     cx
        ret
BYTE_TO_DEC     ENDP

WRITE_AL_HEX PROC NEAR
        push ax
        push dx
        call BYTE_TO_HEX

        mov dl, al
        mov al, ah
        mov ah, 02h
        int 21h

        mov dl, al
        int 21h
        pop dx
        pop ax
        ret
WRITE_AL_HEX ENDP
```

```
; custom functions

WRD_TO_DEC PROC near
; input ax - value
;       di - lower num address
;       si - address of highest available num position (DI-max), or 0 if
;            prefix isn't need
;
; converts AX to DEC and writes to di address (to DI, DI-1, DI-2, ...)
        push bx
        push dx
        push di
        push si
        push ax

        mov bx, 10
        WRD_TO_DEC_loop:
                div bx
                add dl, '0'
                mov [di], dl
                xor dx, dx
                dec di
                cmp ax, 0
                jne WRD_TO_DEC_loop

        cmp si, 0
        je WRD_TO_DEC_no_prefix
        cmp si, di
        jge WRD_TO_DEC_no_prefix
        WRD_TO_DEC_prefix_loop:
                mov dl, '0'
                mov [di], dl
                dec di
                cmp di, si
                jg WRD_TO_DEC_prefix_loop

WRD_TO_DEC_no_prefix:
        pop ax
        pop si
        pop di
        pop dx
        pop bx
        ret
WRD_TO_DEC ENDP

PRINT_MCB PROC NEAR
        push ax
        push di
        push es
        push bx

        mov ah, 52h
        int 21h

        mov es, es:[bx-2]

        PRINT_MCB_loop:
                ; set type
```

```
        mov al, es:[0h]
        call BYTE_TO_HEX
        mov di, offset MCB_DATA_TEXT
        add di, 5
        mov [di], al
        inc di
        mov [di], ah

        ; set owner
        mov ax, es:[01h]
        mov di, offset MCB_DATA_TEXT
        add di, 19
        call WRD_TO_HEX

        ; set size
        mov ax, es:[03h]
        mov bx, 16
        mul bx
        mov di, offset MCB_DATA_TEXT
        mov si, offset MCB_DATA_TEXT
        add di, 33
        add si, 27
        call WRD_TO_DEC

        ; print MCB
        mov dx, offset MCB_DATA_TEXT
        mov ah, 09h
        int 21h

        ; print MCB's data
        mov cx, 8
        mov si, 08h
        mov di, offset MCB_DATA_TEXT
        add di, 42
        PRINT_MCB_data_loop:
                mov dl, es:[si]
                mov ah, 02h
                int 21h
                inc si
                loop PRINT_MCB_data_loop
        mov dx, offset ENDL
        mov ah, 09h
        int 21h

        mov al, es:[0h]
        cmp al, 5Ah
        je PRINT_MCB_end

        mov ax, es
        add ax, es:[3h]
        inc ax
        mov es, ax
        jmp PRINT_MCB_loop

PRINT_MCB_end:
        pop bx
        pop es
        pop di
```

```
                pop ax
                ret
PRINT_MCB ENDP

PRINT_AVAILABLE_MEMORY PROC NEAR
        push ax
        push bx
        push di
        push dx

        mov ax, cs
        mov es, ax
        mov ah, 4ah
        mov bx, 0ffffh
        int 21h

        mov ax, 16
        mul bx
        mov di, offset AVAILABLE_MEMORY_TEXT
        mov si, offset AVAILABLE_MEMORY_TEXT
        add di, 24
        add si, 18
        CALL WRD_TO_DEC

        mov dx, offset AVAILABLE_MEMORY_TEXT
        mov ah, 09h
        int 21h

        pop dx
        pop di
        pop bx
        pop ax
        ret
PRINT_AVAILABLE_MEMORY ENDP

PRINT_EXPANDED_MEMORY PROC near
        push ax
        push di
        push bx

        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 di, offset EXPANDED_MEMORY_TEXT
        mov si, offset EXPANDED_MEMORY_TEXT
        add di, 23
        add si, 17
        mov dx, 0
        CALL WRD_TO_DEC

        mov dx, offset EXPANDED_MEMORY_TEXT
```

```
        mov ah, 09h
        int 21h

        pop bx
        pop di
        pop ax
        ret
PRINT_EXPANDED_MEMORY ENDP

;--------------------------
; CODE
BEGIN:
        mov bx, 1000h
        mov ah, 48h
        int 21h

        jc ON_ERROR
        jmp ON_SUCCESS

        ON_ERROR:
                mov di, offset ERROR_TEXT
                mov si, offset ERROR_TEXT
                add di, 12
                add si, 6
                mov dx, 0
                call WRD_TO_DEC
                mov dx, offset ERROR_TEXT
                mov ah, 09h
                int 21h
                mov dx, offset ENDL
                int 21h

ON_SUCCESS:
        ; compute minimal size we need
        mov dx, 0
        mov ax, offset END_POINTER
        mov bx, 16
        div bx

        inc ax
        mov bx, ax
        mov ah, 04Ah
        int 21h

        call PRINT_MCB
        call PRINT_AVAILABLE_MEMORY
        call PRINT_EXPANDED_MEMORY

; Выход в DOS
        xor     al,al
        mov     ah,4Ch
        int     21h
END_POINTER:
MAIN    ENDS
        END START
```