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	____________________	Щемель Д.А.
Преподаватель	____________________	Ефремов М.А.

Санкт-Петербург

2020

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.

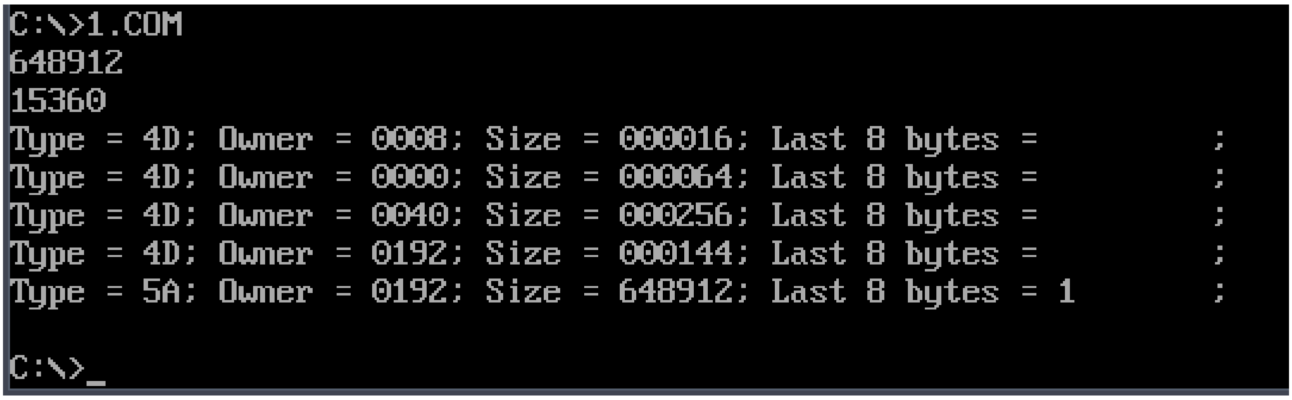
В лабораторной работе исследуется структуры данных и работа функций управления памятью ядра операционной системы.

Ход выполнения работы

Был написан исходный код для .COM-модуля, который выводит следующую информацию:

1. Количество доступной памяти
2. Размер расширенной памяти
3. Цепочку блоков управления памятью

Результат работы модуля приведён на скриншоте ниже.



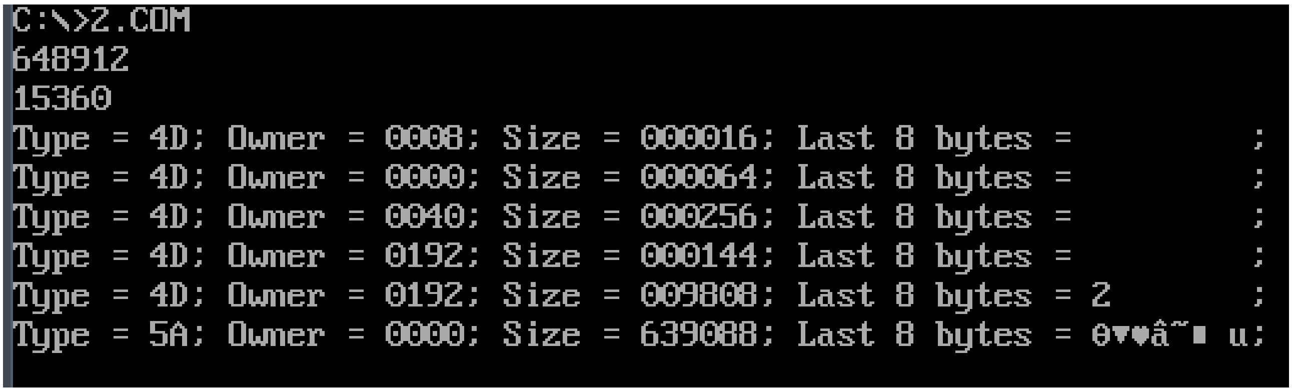
```
C:\>1.COM
648912
15360
Type = 4D; Owner = 0008; Size = 000016; Last 8 bytes =         ;
Type = 4D; Owner = 0000; Size = 000064; Last 8 bytes =         ;
Type = 4D; Owner = 0040; Size = 000256; Last 8 bytes =         ;
Type = 4D; Owner = 0192; Size = 000144; Last 8 bytes =         ;
Type = 5A; Owner = 0192; Size = 648912; Last 8 bytes = 1       ;

C:\>_
```

Figure 1: Результат работы Ш.1

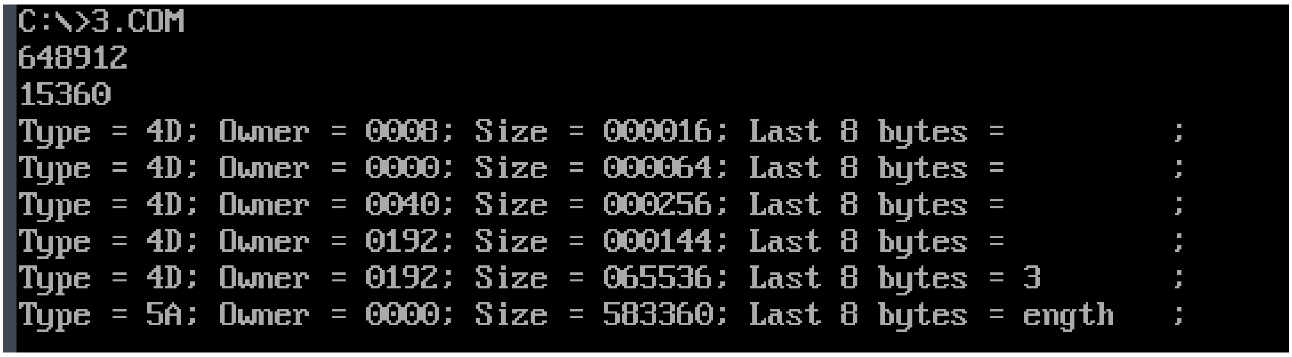
После чего программа была переделана, чтобы она освобождала память, которую не занимает. Результат работы программы представлен на скриншоте ниже.

После чего программа была изменена так, чтобы после освобождения памяти запрашивалось еще 64кб памяти. Результат работы на скриншоте ниже.



```
C:\>2.COM
648912
15360
Type = 4D; Owner = 0008; Size = 000016; Last 8 bytes =         ;
Type = 4D; Owner = 0000; Size = 000064; Last 8 bytes =         ;
Type = 4D; Owner = 0040; Size = 000256; Last 8 bytes =         ;
Type = 4D; Owner = 0192; Size = 000144; Last 8 bytes =         ;
Type = 4D; Owner = 0192; Size = 009808; Last 8 bytes = 2       ;
Type = 5A; Owner = 0000; Size = 639088; Last 8 bytes = θ▼♥â˜■ u;
```

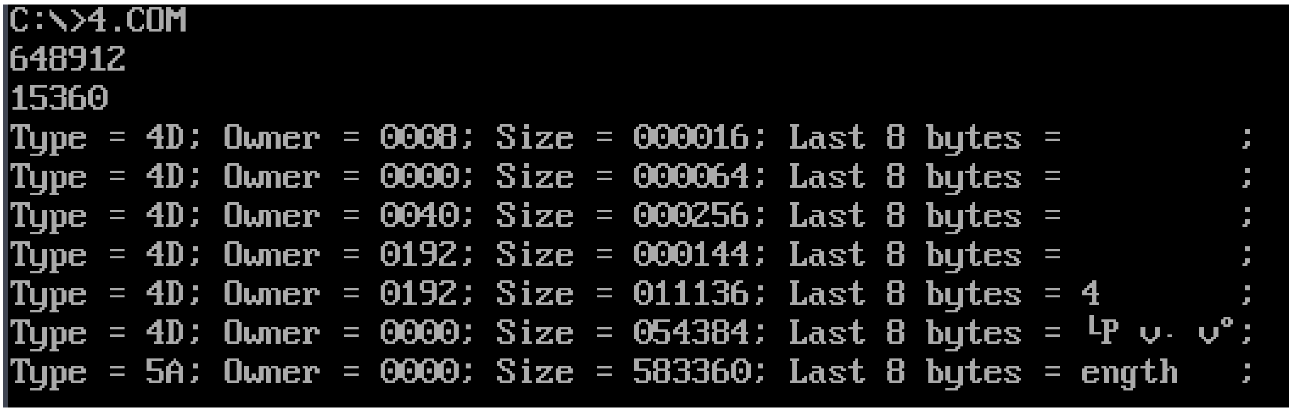
Figure 2: Результат работы Ш.2



```
C:\>3.COM
648912
15360
Type = 4D; Owner = 0008; Size = 000016; Last 8 bytes =         ;
Type = 4D; Owner = 0000; Size = 000064; Last 8 bytes =         ;
Type = 4D; Owner = 0040; Size = 000256; Last 8 bytes =         ;
Type = 4D; Owner = 0192; Size = 000144; Last 8 bytes =         ;
Type = 4D; Owner = 0192; Size = 065536; Last 8 bytes = 3       ;
Type = 5A; Owner = 0000; Size = 583360; Last 8 bytes = ength   ;
```

Figure 3: Результат работы Ш3

В конце программа была измена таким образом, чтобы запрашивать память до момента её освобождения. Результат работы на скриншоте ниже.



```
C:\>4.COM
648912
15360
Type = 4D; Owner = 0008; Size = 000016; Last 8 bytes =         ;
Type = 4D; Owner = 0000; Size = 000064; Last 8 bytes =         ;
Type = 4D; Owner = 0040; Size = 000256; Last 8 bytes =         ;
Type = 4D; Owner = 0192; Size = 000144; Last 8 bytes =         ;
Type = 4D; Owner = 0192; Size = 011136; Last 8 bytes = 4       ;
Type = 4D; Owner = 0000; Size = 054384; Last 8 bytes = └P υ·  υ°;
Type = 5A; Owner = 0000; Size = 583360; Last 8 bytes = ength   ;
```

Figure 4: Результат работы Ш.4

Контрольные вопросы

1. Что означает “Доступный объём памяти”?

   Размер памяти, доступный для использования программе.

2. Где MCB блок Вашей программы в списке?

Блоки, значение поля Owner который равняется 0192h.

3. Какой размер памяти занимает программа в каждом случае?

    1. 648912+ 144 = 649056б
    2. 9808+144 = 9952б
    3. 65536+144 = 65680б

Вывод

В ходе выполнения лабораторной работы была исследования организация управления основной памятью.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ ШАГА 1

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

AVAILABLE_MEMORY db '000000', 10, 13, '$'
EXPANDED_MEMORY db '00000', 10, 13, '$'
MCB_INFO db 'Type = 00; Owner = 0000; Size = 000000; Last 8 bytes = ', '$'
MCB_INFO_END db ';', 10, 13, '$'

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

CALC_AVAILABLE_MEMORY PROC near
```

```
    push ax
    push bx
    push dx
    push si
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, 16
    mul bx
    mov si, offset AVAILABLE_MEMORY
    add si, 5
    CALL WRD_TO_DEC

    mov dx, offset AVAILABLE_MEMORY
    call PRINT
    pop si
    pop dx
    pop bx
    pop ax
    ret
CALC_AVAILABLE_MEMORY ENDP

CALC_EXTENDED_MEMORY PROC near
    push ax
    push si
    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 dx, 0

    mov si, offset EXPANDED_MEMORY
    add si, 4

    CALL WRD_TO_DEC

    mov dx, offset EXPANDED_MEMORY
    call PRINT
    pop si
    pop ax
    ret
CALC_EXTENDED_MEMORY ENDP

PRINT_MCBs PROC near
    push ax
    push bx
    push dx

    mov ah, 52h
    int 21h
    mov es, es:[bx-2]

    PRINT_MCB:
        mov al, es:[0h]
        call BYTE_TO_HEX
        mov si, offset MCB_INFO
        add si, 7
        mov [si], ax

        add si, 14
        mov bx, es:[1h]
        mov al, bl
        call BYTE_TO_HEX
```

```
mov [si], ax
sub si, 2
mov al, bh
call BYTE_TO_HEX
mov [si], ax

add si, 18
mov ax, es:[3h]
mov bx, 16
mul bx
call WRD_TO_DEC

mov dx, offset MCB_INFO
call PRINT

mov si, 8h
mov cx, 8
mov ah, 2h
PRINT_SMB:
    mov dl, es:[si]
    int 21h
    inc si
    loop PRINT_SMB

mov dx, offset MCB_INFO_END
call PRINT

mov al, es:[0h]
cmp al, 5ah
je FINISH

mov ax, es
add ax, es:[3h]
inc ax
mov es, ax
```

```
        jmp PRINT_MCB

    FINISH:
    pop dx
    pop bx
    pop ax
    ret
PRINT_MCBs ENDP

EXIT PROC near
    xor AL, AL
    mov AH, 4ch
    int 21h
    ret
EXIT ENDP

PRINT PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT ENDP

WRD_TO_DEC PROC near
    push bx
    mov bx, 10
    DIVISION:
        div bx
        add dl, 30h
        mov [si], dl
        xor dx, dx
        dec si
        cmp ax, 0
        jne DIVISION
    pop bx
    ret
WRD_TO_DEC ENDP

MAIN:
    call CALC_AVAILABLE_MEMORY
    call CALC_EXTENDED_MEMORY
    call PRINT_MCBs

    call EXIT
MEMORYRESEARCH ENDS
END START
```

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПРОГРАММЫ ШАГА 2

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

AVAILABLE_MEMORY db '000000', 10, 13, '$'
EXPANDED_MEMORY db '00000', 10, 13, '$'
MCB_INFO db 'Type = 00; Owner = 0000; Size = 000000; Last 8 bytes = ', '$'
MCB_INFO_END db ';', 10, 13, '$'

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

CALC_AVAILABLE_MEMORY PROC near
```

```
    push ax
    push bx
    push dx
    push si
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, 16
    mul bx
    mov si, offset AVAILABLE_MEMORY
    add si, 5
    CALL WRD_TO_DEC

    mov dx, offset AVAILABLE_MEMORY
    call PRINT
    pop si
    pop dx
    pop bx
    pop ax
    ret
CALC_AVAILABLE_MEMORY ENDP

CALC_EXTENDED_MEMORY PROC near
    push ax
    push si
    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 dx, 0

    mov si, offset EXPANDED_MEMORY
    add si, 4

    CALL WRD_TO_DEC

    mov dx, offset EXPANDED_MEMORY
    call PRINT
    pop si
    pop ax
    ret
CALC_EXTENDED_MEMORY ENDP

PRINT_MCBs PROC near
    push ax
    push bx
    push dx

    mov ah, 52h
    int 21h
    mov es, es:[bx-2]

    PRINT_MCB:
        mov al, es:[0h]
        call BYTE_TO_HEX
        mov si, offset MCB_INFO
        add si, 7
        mov [si], ax

        add si, 14
        mov bx, es:[1h]
        mov al, bl
        call BYTE_TO_HEX
```

```
mov [si], ax
sub si, 2
mov al, bh
call BYTE_TO_HEX
mov [si], ax

add si, 18
mov ax, es:[3h]
mov bx, 16
mul bx
call WRD_TO_DEC

mov dx, offset MCB_INFO
call PRINT

mov si, 8h
mov cx, 8
mov ah, 2h
PRINT_SMB:
    mov dl, es:[si]
    int 21h
    inc si
    loop PRINT_SMB

mov dx, offset MCB_INFO_END
call PRINT

mov al, es:[0h]
cmp al, 5ah
je FINISH

mov ax, es
add ax, es:[3h]
inc ax
mov es, ax
```

```
        jmp PRINT_MCB

    FINISH:
    pop dx
    pop bx
    pop ax
    ret
PRINT_MCBs ENDP

EXIT PROC near
    xor AL, AL
    mov AH, 4ch
    int 21h
    ret
EXIT ENDP

PRINT PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT ENDP

WRD_TO_DEC PROC near
    push bx
    mov bx, 10
    DIVISION:
        div bx
        add dl, 30h
        mov [si], dl
        xor dx, dx
        dec si
        cmp ax, 0
        jne DIVISION
    pop bx
    ret
WRD_TO_DEC ENDP

MAIN:
    call CALC_AVAILABLE_MEMORY
    mov ah, 4ah
    mov bx, offset END_LABEL
    int 21h
    call CALC_EXTENDED_MEMORY
    call PRINT_MCBs

    call EXIT
    END_LABEL:
MEMORYRESEARCH ENDS
END START
```

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

AVAILABLE_MEMORY db '000000', 10, 13, '$'
EXPANDED_MEMORY db '00000', 10, 13, '$'
MCB_INFO db 'Type = 00; Owner = 0000; Size = 000000; Last 8 bytes = ', '$'
MCB_INFO_END db ';', 10, 13, '$'

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

CALC_AVAILABLE_MEMORY PROC near
    push ax
    push bx
    push dx
    push si
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, 16
    mul bx
    mov si, offset AVAILABLE_MEMORY
    add si, 5
    CALL WRD_TO_DEC

    mov dx, offset AVAILABLE_MEMORY
    call PRINT
    pop si
    pop dx
    pop bx
    pop ax
    ret
```

```
CALC_AVAILABLE_MEMORY ENDP

CALC_EXTENDED_MEMORY PROC near
    push ax
    push si
    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 dx, 0

    mov si, offset EXPANDED_MEMORY
    add si, 4

    CALL WRD_TO_DEC

    mov dx, offset EXPANDED_MEMORY
    call PRINT
    pop si
    pop ax
    ret
CALC_EXTENDED_MEMORY ENDP

PRINT_MCBs PROC near
    push ax
    push bx
    push dx

    mov ah, 52h
```

```
int 21h
mov es, es:[bx-2]

PRINT_MCB:
    mov al, es:[0h]
    call BYTE_TO_HEX
    mov si, offset MCB_INFO
    add si, 7
    mov [si], ax

    add si, 14
    mov bx, es:[1h]
    mov al, bl
    call BYTE_TO_HEX
    mov [si], ax
    sub si, 2
    mov al, bh
    call BYTE_TO_HEX
    mov [si], ax

    add si, 18
    mov ax, es:[3h]
    mov bx, 16
    mul bx
    call WRD_TO_DEC

    mov dx, offset MCB_INFO
    call PRINT

    mov si, 8h
    mov cx, 8
    mov ah, 2h
    PRINT_SMB:
        mov dl, es:[si]
        int 21h
```

```
            inc si
            loop PRINT_SMB

        mov dx, offset MCB_INFO_END
        call PRINT

        mov al, es:[0h]
        cmp al, 5ah
        je FINISH

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

    FINISH:
    pop dx
    pop bx
    pop ax
    ret
PRINT_MCBs ENDP

EXIT PROC near
    xor AL, AL
    mov AH, 4ch
    int 21h
    ret
EXIT ENDP

PRINT PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT ENDP

WRD_TO_DEC PROC near
    push bx
    mov bx, 10
    DIVISION:
        div bx
        add dl, 30h
        mov [si], dl
        xor dx, dx
        dec si
        cmp ax, 0
        jne DIVISION
    pop bx
    ret
WRD_TO_DEC ENDP

MAIN:
    call CALC_AVAILABLE_MEMORY
    call CALC_EXTENDED_MEMORY
    call PRINT_MCBs

    call EXIT
MEMORYRESEARCH ENDS
END START
```

ПРИЛОЖЕНИЕ В. ИСХОДНЫЙ КОД ПРОГРАММЫ ШАГА 3

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

AVAILABLE_MEMORY db '000000', 10, 13, '$'
EXPANDED_MEMORY db '00000', 10, 13, '$'
MCB_INFO db 'Type = 00; Owner = 0000; Size = 000000; Last 8 bytes = ', '$'
MCB_INFO_END db ';', 10, 13, '$'

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

CALC_AVAILABLE_MEMORY PROC near
```

```
    push ax
    push bx
    push dx
    push si
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, 16
    mul bx
    mov si, offset AVAILABLE_MEMORY
    add si, 5
    CALL WRD_TO_DEC

    mov dx, offset AVAILABLE_MEMORY
    call PRINT
    pop si
    pop dx
    pop bx
    pop ax
    ret
CALC_AVAILABLE_MEMORY ENDP

CALC_EXTENDED_MEMORY PROC near
    push ax
    push si
    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 dx, 0

    mov si, offset EXPANDED_MEMORY
    add si, 4

    CALL WRD_TO_DEC

    mov dx, offset EXPANDED_MEMORY
    call PRINT
    pop si
    pop ax
    ret
CALC_EXTENDED_MEMORY ENDP

PRINT_MCBs PROC near
    push ax
    push bx
    push dx

    mov ah, 52h
    int 21h
    mov es, es:[bx-2]

    PRINT_MCB:
        mov al, es:[0h]
        call BYTE_TO_HEX
        mov si, offset MCB_INFO
        add si, 7
        mov [si], ax

        add si, 14
        mov bx, es:[1h]
        mov al, bl
        call BYTE_TO_HEX
```

```
mov [si], ax
sub si, 2
mov al, bh
call BYTE_TO_HEX
mov [si], ax

add si, 18
mov ax, es:[3h]
mov bx, 16
mul bx
call WRD_TO_DEC

mov dx, offset MCB_INFO
call PRINT

mov si, 8h
mov cx, 8
mov ah, 2h
PRINT_SMB:
    mov dl, es:[si]
    int 21h
    inc si
    loop PRINT_SMB

mov dx, offset MCB_INFO_END
call PRINT

mov al, es:[0h]
cmp al, 5ah
je FINISH

mov ax, es
add ax, es:[3h]
inc ax
mov es, ax
```

```
        jmp PRINT_MCB

    FINISH:
    pop dx
    pop bx
    pop ax
    ret
PRINT_MCBs ENDP

EXIT PROC near
    xor AL, AL
    mov AH, 4ch
    int 21h
    ret
EXIT ENDP

PRINT PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT ENDP

WRD_TO_DEC PROC near
    push bx
    mov bx, 10
    DIVISION:
        div bx
        add dl, 30h
        mov [si], dl
        xor dx, dx
        dec si
        cmp ax, 0
        jne DIVISION
    pop bx
    ret
WRD_TO_DEC ENDP

MAIN:
    call CALC_AVAILABLE_MEMORY
    mov ah, 4ah
    mov bx, offset END_LABEL
    int 21h

    mov ah, 4ah
    mov bx, 4096
    int 21h
    call CALC_EXTENDED_MEMORY
    call PRINT_MCBs

    call EXIT
    END_LABEL:
MEMORYRESEARCH ENDS
END START
```

ПРИЛОЖЕНИЕ Г. ИСХОДНЫЙ КОД ПРОГРАММЫ ШАГА 4

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

AVAILABLE_MEMORY db '000000', 10, 13, '$'
EXPANDED_MEMORY db '00000', 10, 13, '$'
MCB_INFO db 'Type = 00; Owner = 0000; Size = 000000; Last 8 bytes = ', '$'
MCB_INFO_END db ';', 10, 13, '$'
ALLOC_ERROR_STRING db 'Error while allocating: 0000h', 10, 13, '$'

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

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

CALC_AVAILABLE_MEMORY PROC near
    push ax
    push bx
    push dx
    push si
    mov ah, 4ah
    mov bx, 0ffffh
    int 21h

    mov ax, 16
    mul bx
    mov si, offset AVAILABLE_MEMORY
    add si, 5
    CALL WRD_TO_DEC

    mov dx, offset AVAILABLE_MEMORY
    call PRINT
    pop si
    pop dx
    pop bx
    pop ax
    ret
CALC_AVAILABLE_MEMORY ENDP

CALC_EXTENDED_MEMORY PROC near
    push ax
    push si
    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 dx, 0

    mov si, offset EXPANDED_MEMORY
    add si, 4

    CALL WRD_TO_DEC

    mov dx, offset EXPANDED_MEMORY
    call PRINT
    pop si
    pop ax
    ret
CALC_EXTENDED_MEMORY ENDP

PRINT_MCBs PROC near
    push ax
    push bx
    push dx

    mov ah, 52h
    int 21h
    mov es, es:[bx-2]

    PRINT_MCB:
        mov al, es:[0h]
        call BYTE_TO_HEX
        mov si, offset MCB_INFO
        add si, 7
        mov [si], ax

        add si, 14
        mov bx, es:[1h]
        mov al, bl
        call BYTE_TO_HEX
        mov [si], ax
        sub si, 2
        mov al, bh
        call BYTE_TO_HEX
        mov [si], ax

        add si, 18
        mov ax, es:[3h]
        mov bx, 16
        mul bx
        call WRD_TO_DEC

        mov dx, offset MCB_INFO
        call PRINT

        mov si, 8h
        mov cx, 8
```

```
        mov ah, 2h
        PRINT_SMB:
            mov dl, es:[si]
            int 21h
            inc si
            loop PRINT_SMB

        mov dx, offset MCB_INFO_END
        call PRINT

        mov al, es:[0h]
        cmp al, 5ah
        je FINISH

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

    FINISH:
    pop dx
    pop bx
    pop ax
    ret
PRINT_MCBs ENDP

EXIT PROC near
    xor AL, AL
    mov AH, 4ch
    int 21h
    ret
EXIT ENDP

PRINT PROC near
```

```
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
PRINT ENDP

WRD_TO_DEC PROC near
    push bx
    mov bx, 10
    DIVISION:
        div bx
        add dl, 30h
        mov [si], dl
        xor dx, dx
        dec si
        cmp ax, 0
        jne DIVISION
    pop bx
    ret
WRD_TO_DEC ENDP

MAIN:
    call CALC_AVAILABLE_MEMORY
    mov ah, 4ah
    mov bx, 4096
    int 21h

    jc ALLOC_ERROR
    jmp ALLOC_OK

    ALLOC_ERROR:
        mov di, offset ALLOC_ERROR_STRING
        add di, 27
        call WRD_TO_HEX
        mov dx, offset ALLOC_ERROR_STRING
        call PRINT

    ALLOC_OK:

    mov ah, 4ah
    mov bx, offset END_LABEL
    int 21h

    call CALC_EXTENDED_MEMORY
    call PRINT_MCBs

    call EXIT
    END_LABEL:
MEMORYRESEARCH ENDS
END START
```