МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Операционные системы»

Тема: Исследование организации управления основной памятью

Студент гр. 7381 _____ Трушников А.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Для исследования организации управления памятью необходимо ориентироваться на тип основной памяти, реализованной в компьютере и способ организации, принятый в ОС. В лабораторной работе рассматривается нестраничная память и способ управления динамическими раздела. 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.

Основные теоретические положения.

Учет занятой и свободной памяти ведется при помощи списка блоков управления памятью MCB. MCB занимает 16 байт и располагается всегда с адреса кратного 16 и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

MCB имеет следующую структуру:

Смещение	Длина поля (байт)	Содержание 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 исключительной 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”. Доступ к указателю на эту структуру можно получить, используя функцию 52h “Get Lists 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 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. 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, 71; чтение младшего байта
mov  bl, al; размера расширенной памяти
mov  al, 31h; запись адреса ячейки CMOS
out  70h, al
in   al, 71h; чтение старшего байта
           ;размера расширенной памяти
```

Выполнение работы.

Шаг 1.

```
C:\>LAB3_1.COM
Amount of available memory: 648912 bytes
Extended memory size: 15360  kilobytes
Chain of memory control units:
MCB type: 4Dh, Segment's adress: 0008h, MCB size:     16 b,
MCB type: 4Dh, Segment's adress: 0000h, MCB size:     64 b,
MCB type: 4Dh, Segment's adress: 0040h, MCB size:    256 b,
MCB type: 4Dh, Segment's adress: 0192h, MCB size:    144 b,
MCB type: 5Ah, Segment's adress: 0192h, MCB size: 648912 b, LAB3_1
```

Шаг 2.

```
C:\>LAB3_2.COM
Amount of available memory: 648912 bytes
Extended memory size: 15360  kilobytes
Chain of memory control units:
MCB type: 4Dh, Segment's adress: 0008h, MCB size:     16 b,
MCB type: 4Dh, Segment's adress: 0000h, MCB size:     64 b,
MCB type: 4Dh, Segment's adress: 0040h, MCB size:    256 b,
MCB type: 4Dh, Segment's adress: 0192h, MCB size:    144 b,
MCB type: 4Dh, Segment's adress: 0192h, MCB size:    800 b, LAB3_2
MCB type: 5Ah, Segment's adress: 0000h, MCB size: 648096 b,
```

Шаг 3.

```
C:\>LAB3_3.COM
Amount of available memory: 648912 bytes
Extended memory size: 15360  kilobytes
Chain of memory control units:
MCB type: 4Dh, Segment's adress: 0008h, MCB size:     16 b,
MCB type: 4Dh, Segment's adress: 0000h, MCB size:     64 b,
MCB type: 4Dh, Segment's adress: 0040h, MCB size:    256 b,
MCB type: 4Dh, Segment's adress: 0192h, MCB size:    144 b,
MCB type: 4Dh, Segment's adress: 0192h, MCB size:    816 b, LAB3_3
MCB type: 4Dh, Segment's adress: 0192h, MCB size: 264544 b, LAB3_3
MCB type: 5Ah, Segment's adress: 0000h, MCB size: 383520 b,
```

Шаг 4.

```
C:\>LAB3_4.COM
Amount of available memory: 648912 bytes
Error
```

Ответы на контрольные вопросы.

1. Что означает «доступный объём памяти?»

Доступный объём памяти - часть оперативной памяти, выделенная программе для работы.

2. Где MCB блок вашей программы в списке?

В первом случае блок программы последний и занимает всю доступную память. Во втором случае блок программы второй снизу. Это связано с тем, что программа освобождает неиспользуемую память, и блок с неиспользуемой ей памятью оказывается последним. В третьем случае блок программы второй третий снизу. Программа сначала освобождает неиспользуемую память, а затем запрашивает 64 Кб памяти. В четвёртом случае программа запрашивает память до того, как освобождает неиспользуемую – и при обработке завершения функций ядра (проверке флага CF) мы узнаём, что возникает ошибка.

3. Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает всю свободную память (648912 б). Во втором случае программа занимает 800б. В третьем случае программа занимает необходимый программе объем памяти и запрошенные 64 кб (265360+265544 б)

Выводы.

В ходе данной лабораторной работы были исследованы структуры данных и работу функций управления памятью ядра операционной системы.

ПРИЛОЖЕНИЕ А

LAB3_1.ASM

```
; ЛАБОРАТОРНАЯ РАБОТА 3
; ШАГ 1
; ПРОГРАММА ВЫВОДИТ
;          1)КОЛИЧЕСТВО ДОСТУПНОЙ ПАМЯТИ
;          2)РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ
;          3)ВЫВОДИТ ЦЕПОЧКУ БЛОКОВ УПРАВЛЕНИЯ ПАМЯТЬЮ
TESTPC  SEGMENT
     ASSUME  CS:TESTPC,  DS:TESTPC,  ES:NOTHING, SS:NOTHING
     ORG     100H
START:  JMP     BEGIN

; ДАННЫЕ
AVAILABEL_M     DB      'AMOUNT OF AVAILABLE MEMORY:       BYTES',0DH,0AH,'$'
EXTENDED_M      DB      'EXTENDED MEMORY SIZE:       KILOBYTES',0DH,0AH,'$'
MCB             DB      'CHAIN OF MEMORY CONTROL UNITS:',0DH,0AH,'$'
MCB_TYPE        DB      'MCB TYPE:   H, $'
MCB_SEG         DB      'SEGMENT`S ADRESS:     H, $'
MCB_SIZE        DB      'MCB SIZE:        B, $'
MCB_TAIL        DB      '               ',0DH,0AH,'$'
; ПРОЦЕДУРЫ

WRITE_MSG       PROC    NEAR
     MOV     AH,09H
     INT     21H
     RET
WRITE_MSG       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
; БАЙТ В AL ПЕРЕВОДИТСЯ В ДВА СИМВОЛА ШЕСТН. 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
; ПЕРВОД В 16 С/С 16-ТИ РАЗРЯДНОГО ЧИСЛА
; В AX - ЧИСЛО, 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
        XOR     AH,AH
        CALL    BYTE_TO_HEX
        MOV     [DI],AH
        DEC     DI
        MOV     [DI],AL
        POP     BX
        RET
    WRD_TO_HEX      ENDP

    WRD_TO_DEC      PROC    NEAR
    ; ПЕРЕВОД 2 БАЙТОВ В 10 С/С, SI - АДРЕС ПОЛЯ МЛАДШЕЙ ЦИФРЫ
        PUSH    CX
        PUSH    DX
        PUSH    AX
        MOV     CX,10
      WRD_LOOP_BD:
        DIV     CX
        OR      DL,30H
        MOV     [SI],DL
        DEC     SI
        XOR     DX,DX
        CMP     AX,10
        JAE     WRD_LOOP_BD
        CMP     AX,00H
        JBE     WRD_END_L
        OR      AL,30H
        MOV     [SI],AL
      WRD_END_L:
        POP     AX
        POP     DX
        POP     CX
        RET
    WRD_TO_DEC      ENDP

    DEFINE_AVAIL_M      PROC    NEAR
        PUSH    CX
        PUSH    BX
        PUSH    DX
        PUSH    AX
                PUSH    SI
        MOV         AH,  4AH     ; ПЫТАЕМСЯ ОСВОБОДИТЬ ЗАВЕДОМО БОЛЬШОЙ РАЗМЕР
ПАМЯТИ
        MOV     BX, 0FFFFH ; НЕИСПОЛЬЗУЕМЫЙ ПРОГРАММОЙ
        INT     21H         ;В BX ВЕРНЕТСЯ РАЗМЕР ДОСТУПНОЙ ПАМЯТИ В ПАРАГРАФАХ
        MOV     AX, BX
        MOV     CX, 10H  ;ПЕРЕВОДИМ ПАРАГРАФЫ В БАЙТЫ
        MUL     CX
        MOV     SI, OFFSET AVAILABEL_M + 33
        CALL    WRD_TO_DEC
        MOV     DX, OFFSET AVAILABEL_M
        CALL    WRITE_MSG
        POP     SI
        POP     AX
        POP     DX
        POP     BX
                POP             CX
        RET
    DEFINE_AVAIL_M  ENDP

    DEFINE_EXTENDED_M       PROC    NEAR
```

```
    PUSH   AX
    PUSH   BX
    PUSH   SI
    PUSH   DX
    MOV    AL, 30H ; ЗАПИСЬ АДРЕСА ЯЧЕЙКИ CMOS
    OUT    70H, AL
    IN     AL, 71H  ; ЧТЕНИЕ МЛАДШЕГО БАЙТЫ
    MOV    BL, AH  ; РАЗМЕРА РАСШИРЕННОЙ ПАМЯТИ
    MOV    AL, 31H ; ЗАПИСЬ АДРЕСА ЯЧЕЙКИ CMOS
    OUT    70H, AL
    IN     AL, 71H ; ЧТЕНИЕ СТАРШЕГО БАЙТА
    MOV    AH, AL  ; РАЗМЕРА РАСШИРЕННОЙ ПАМЯТИ
    MOV    AL, BL ; В AX РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ
    SUB    DX, DX
    MOV    SI, OFFSET EXTENDED_M + 26
    CALL   WRD_TO_DEC
    MOV    DX, OFFSET EXTENDED_M
    CALL   WRITE_MSG
    POP    DX
    POP    SI
    POP    BX
    POP    AX
    RET
DEFINE_EXTENDED_M   ENDP


PRINT_MCB   PROC  NEAR
    MOV    DX, OFFSET MCB
    CALL   WRITE_MSG
    MOV    AH, 52H
    INT    21H
    SUB    AX, AX
    SUB    CX, CX
    MOV    ES, ES:[BX-2] ;СОХРАНЯЕМ АДРЕС ПЕРВОГО MCB
    MOV    BX, 1

    CYCLE:
    SUB    AX, AX
    MOV    AL, ES:[00H]
    CALL   BYTE_TO_HEX
    CMP    AX, 4135H ;ПРОВЕРЯЕМ ЯВЛЯЕТСЯ ЛИ MCB ПОСЛЕДНИМ
    JNE    CONTINUE
    MOV    BX, 0
    CONTINUE:
    MOV    DI, OFFSET MCB_TYPE + 10
    MOV    [DI], AX
    LEA    DI,MCB_SEG+21
    MOV    AX,ES:[0001H]
    CALL   WRD_TO_HEX
    MOV    AX, ES:[03H]
    MOV    CX, 10H
    MUL    CX
    MOV    SI, OFFSET MCB_SIZE + 15
    CALL   WRD_TO_DEC
              MOV          SI, OFFSET MCB_TAIL
              MOV          DI,0008H
              MOV          CX,4
              CYCLE1:
              MOV          AX,ES:[DI]; СОХРАНЯЕМ XBOCT
              MOV          [SI],AX
              ADD    DI,2H
              ADD          SI,2H
```

```
                LOOP   CYCLE1
        MOV     DX, OFFSET MCB_TYPE
        CALL    WRITE_MSG
        MOV     DX, OFFSET MCB_SEG
        CALL    WRITE_MSG
        MOV     DX, OFFSET MCB_SIZE
        CALL    WRITE_MSG
                MOV     DX, OFFSET MCB_TAIL
        CALL    WRITE_MSG
        CMP     BX, 0
        JE      _END
        MOV     AX, ES
        ADD     AX, ES:[0003H]
        INC     AX
        MOV     ES, AX
        JMP     CYCLE
        _END:
        RET
    PRINT_MCB   ENDP

    BEGIN:
        CALL DEFINE_AVAIL_M
        CALL DEFINE_EXTENDED_M
        CALL PRINT_MCB

    ; ВЫХОД В DOS
        XOR     AL,AL
        MOV     AH,3CH
        INT     21H
        RET
    TESTPC  ENDS
END     START
```

LAB3_2.ASM

```
; ЛАБОРАТОРНАЯ РАБОТА 3
; ШАГ 2
; ПРОГРАММА ОСВОБОЖДАЕТ ПАМЯТЬ, КОТОРУЮ НЕ ЗАНИМАЕТ
; ПРОГРАММА ВЫВОДИТ
;          1)КОЛИЧЕСТВО ДОСТУПНОЙ ПАМЯТИ
;          2)РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ
;          3)ВЫВОДИТ ЦЕПОЧКУ БЛОКОВ УПРАВЛЕНИЯ ПАМЯТЬЮ
TESTPC  SEGMENT
     ASSUME  CS:TESTPC,  DS:TESTPC,  ES:NOTHING, SS:NOTHING
     ORG     100H
START:  JMP     BEGIN

; ДАННЫЕ
AVAILABEL_M     DB      'AMOUNT OF AVAILABLE MEMORY:       BYTES',0DH,0AH,'$'
EXTENDED_M      DB      'EXTENDED MEMORY SIZE:       KILOBYTES',0DH,0AH,'$'
MCB             DB      'CHAIN OF MEMORY CONTROL UNITS:',0DH,0AH,'$'
MCB_TYPE        DB      'MCB TYPE:   H, $'
MCB_SEG         DB      'SEGMENT`S ADRESS:     H, $'
MCB_SIZE        DB      'MCB SIZE:       B, $'
MCB_TAIL        DB      '               ',0DH,0AH,'$'
; ПРОЦЕДУРЫ

WRITE_MSG       PROC    NEAR
     MOV     AH,09H
     INT     21H
     RET
WRITE_MSG       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
; БАЙТ В AL ПЕРЕВОДИТСЯ В ДВА СИМВОЛА ШЕСТН. 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
; ПЕРВОД В 16 С/С 16-ТИ РАЗРЯДНОГО ЧИСЛА
; В AX - ЧИСЛО, 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
        XOR     AH,AH
        CALL    BYTE_TO_HEX
        MOV     [DI],AH
        DEC     DI
        MOV     [DI],AL
        POP     BX
        RET
WRD_TO_HEX      ENDP

WRD_TO_DEC      PROC    NEAR
; ПЕРЕВОД 2 БАЙТОВ В 10 С/С, SI - АДРЕС ПОЛЯ МЛАДШЕЙ ЦИФРЫ
        PUSH    CX
        PUSH    DX
        PUSH    AX
        MOV     CX,10
    WRD_LOOP_BD:
        DIV     CX
        OR      DL,30H
        MOV     [SI],DL
        DEC     SI
        XOR     DX,DX
        CMP     AX,10
        JAE     WRD_LOOP_BD
        CMP     AX,00H
        JBE     WRD_END_L
        OR      AL,30H
        MOV     [SI],AL
    WRD_END_L:
        POP     AX
        POP     DX
        POP     CX
        RET
WRD_TO_DEC      ENDP

DEFINE_AVAIL_M      PROC    NEAR
        PUSH    CX
        PUSH    BX
        PUSH    DX
        PUSH    AX
        MOV         AH, 4AH     ; ПЫТАЕМСЯ ОСВОБОДИТЬ ЗАВЕДОМО БОЛЬШОЙ РАЗМЕР
ПАМЯТИ
        MOV     BX, 0FFFFH ; НЕИСПОЛЬЗУЕМЫЙ ПРОГРАММОЙ
        INT     21H         ;В BX ВЕРНЕТСЯ РАЗМЕР ДОСТУПНОЙ ПАМЯТИ В ПАРАГРАФАХ
        MOV     AX, BX
        MOV     CX, 10H  ;ПЕРЕВОДИМ ПАРАГРАФЫ В БАЙТЫ
        MUL     CX
        MOV     SI, OFFSET AVAILABEL_M + 33
        CALL    WRD_TO_DEC
        MOV     DX, OFFSET AVAILABEL_M
        CALL    WRITE_MSG

                MOV     AX, OFFSET THE_END
                MOV     BX, 10H ; ПЕРЕВОДИМ БАЙТЫ В AX В ПАРАГРАФЫ
                SUB     DX, DX
                DIV     BX
                INC     AX
                MOV             BX, AX
                MOV     AL, 0
                MOV     AH, 4AH; ОСВОБОЖДАЕМ ПАМЯТЬ
```

```
                INT     21H
                POP     AX
        POP     DX
        POP     BX
        POP     CX
        RET
DEFINE_AVAIL_M  ENDP

DEFINE_EXTENDED_M       PROC    NEAR
        PUSH    AX
        PUSH    BX
        PUSH    SI
        PUSH    DX
        MOV     AL, 30H ; ЗАПИСЬ АДРЕСА ЯЧЕЙКИ CMOS
        OUT     70H, AL
        IN      AL, 71H  ; ЧТЕНИЕ МЛАДШЕГО БАЙТЫ
        MOV     BL, AH  ; РАЗМЕРА РАСШИРЕННОЙ ПАМЯТИ
        MOV     AL, 31H ; ЗАПИСЬ АДРЕСА ЯЧЕЙКИ CMOS
        OUT     70H, AL
        IN      AL, 71H ; ЧТЕНИЕ СТАРШЕГО БАЙТА
        MOV     AH, AL  ; РАЗМЕРА РАСШИРЕННОЙ ПАМЯТИ
        MOV     AL, BL ; В AX РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ
        SUB     DX, DX
        MOV     SI, OFFSET EXTENDED_M + 26
        CALL    WRD_TO_DEC
        MOV     DX, OFFSET EXTENDED_M
        CALL    WRITE_MSG
        POP     DX
        POP     SI
        POP     BX
        POP     AX
        RET
DEFINE_EXTENDED_M   ENDP


PRINT_MCB   PROC  NEAR
        MOV     DX, OFFSET MCB
        CALL    WRITE_MSG
        MOV     AH, 52H
        INT     21H
        SUB     AX, AX
        SUB     CX, CX
        MOV     ES, ES:[BX-2] ;СОХРАНЯЕМ АДРЕС ПЕРВОГО MCB
        MOV     BX, 1

        CYCLE:
        SUB     AX, AX
        MOV     AL, ES:[00H]
        CALL    BYTE_TO_HEX
        CMP     AX, 4135H ;ПРОВЕРЯЕМ ЯВЛЯЕТСЯ ЛИ ОН ПОСЛЕДНИМ
        JNE     CONTINUE
        MOV     BX, 0
        CONTINUE:
        MOV     DI, OFFSET MCB_TYPE + 10
        MOV     [DI], AX
        LEA     DI,MCB_SEG+21
        MOV     AX,ES:[0001H]
        CALL    WRD_TO_HEX
        MOV     AX, ES:[03H]
        MOV     CX, 10H
        MUL     CX
        MOV     SI, OFFSET MCB_SIZE + 15
```

```
    CALL    WRD_TO_DEC
            MOV         SI, OFFSET MCB_TAIL
            MOV         DI,0008H
            MOV         CX,4
            CYCLE1:
            MOV         AX,ES:[DI]; СОХРАНЯЕМ ХВОСТ
            MOV         [SI],AX
            ADD     DI,2H
            ADD         SI,2H
            LOOP    CYCLE1
    MOV     DX, OFFSET MCB_TYPE
    CALL    WRITE_MSG
    MOV     DX, OFFSET MCB_SEG
    CALL    WRITE_MSG
    MOV     DX, OFFSET MCB_SIZE
    CALL    WRITE_MSG
            MOV     DX, OFFSET MCB_TAIL
    CALL    WRITE_MSG
    CMP     BX, 0
    JE      _END
    MOV     AX, ES
    ADD     AX, ES:[0003H]
    INC     AX
    MOV     ES, AX
    JMP     CYCLE
    _END:
    RET
PRINT_MCB   ENDP

BEGIN:
    CALL DEFINE_AVAIL_M
    CALL DEFINE_EXTENDED_M
    CALL PRINT_MCB

; ВЫХОД В DOS
    XOR     AL,AL
    MOV     AH,3CH
    INT     21H
    RET
            THE_END:
TESTPC  ENDS
    END     START
```

LAB3_3.ASM

```
; ЛАБОРАТОРНАЯ РАБОТА 3
; ШАГ 3
; ПРОГРАММА ПОСЛЕ ОСВОБОЖДЕНИЯ ПАМЯТИ, КОТОРУЮ НЕ ЗАНИМАЕТ, ЗАПРАШИВАЕТ 64КБ
; ПРОГРАММА ВЫВОДИТ
;       1)КОЛИЧЕСТВО ДОСТУПНОЙ ПАМЯТИ
;       2)РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ
;       3)ВЫВОДИТ ЦЕПОЧКУ БЛОКОВ УПРАВЛЕНИЯ ПАМЯТЬЮ
TESTPC  SEGMENT
    ASSUME  CS:TESTPC,  DS:TESTPC,  ES:NOTHING, SS:NOTHING
    ORG     100H
START:  JMP     BEGIN

; ДАННЫЕ
AVAILABEL_M     DB      'AMOUNT OF AVAILABLE MEMORY:        BYTES',0DH,0AH,'$'
EXTENDED_M      DB      'EXTENDED MEMORY SIZE:        KILOBYTES',0DH,0AH,'$'
MCB             DB      'CHAIN OF MEMORY CONTROL UNITS:',0DH,0AH,'$'
MCB_TYPE        DB      'MCB TYPE:   H, $'
MCB_SEG         DB      'SEGMENT`S ADRESS:     H, $'
MCB_SIZE        DB      'MCB SIZE:        B, $'
MCB_TAIL        DB      '               ',0DH,0AH,'$'
; ПРОЦЕДУРЫ

WRITE_MSG       PROC    NEAR
    MOV     AH,09H
    INT     21H
    RET
WRITE_MSG       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
; БАЙТ В AL ПЕРЕВОДИТСЯ В ДВА СИМВОЛА ШЕСТН. 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
; ПЕРВОД В 16 С/С 16-ТИ РАЗРЯДНОГО ЧИСЛА
; В AX - ЧИСЛО, 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
      XOR     AH,AH
      CALL    BYTE_TO_HEX
      MOV     [DI],AH
      DEC     DI
      MOV     [DI],AL
      POP     BX
      RET
WRD_TO_HEX      ENDP

WRD_TO_DEC      PROC    NEAR
; ПЕРЕВОД 2 БАЙТОВ В 10 С/С, SI - АДРЕС ПОЛЯ МЛАДШЕЙ ЦИФРЫ
      PUSH    CX
      PUSH    DX
      PUSH    AX
      MOV     CX,10
   WRD_LOOP_BD:
      DIV     CX
      OR      DL,30H
      MOV     [SI],DL
      DEC     SI
      XOR     DX,DX
      CMP     AX,10
      JAE     WRD_LOOP_BD
      CMP     AX,00H
      JBE     WRD_END_L
      OR      AL,30H
      MOV     [SI],AL
   WRD_END_L:
      POP     AX
      POP     DX
      POP     CX
      RET
```

```
WRD_TO_DEC     ENDP

DEFINE_AVAIL_M     PROC    NEAR
    PUSH    CX
    PUSH    BX
    PUSH    DX
    PUSH    AX
    MOV         AH,  4AH     ; ПЫТАЕМСЯ ОСВОБОДИТЬ ЗАВЕДОМО БОЛЬШОЙ РАЗМЕР
ПАМЯТИ
    MOV     BX, 0FFFFH ; НЕИСПОЛЬЗУЕМЫЙ ПРОГРАММОЙ
    INT     21H         ;В BX ВЕРНЕТСЯ РАЗМЕР ДОСТУПНОЙ ПАМЯТИ В ПАРАГРАФАХ
    MOV     AX, BX
    MOV     CX, 10H  ;ПЕРЕВОДИМ ПАРАГРАФЫ В БАЙТЫ
    MUL     CX
    MOV     SI, OFFSET AVAILABEL_M + 33
    CALL    WRD_TO_DEC
    MOV     DX, OFFSET AVAILABEL_M
    CALL    WRITE_MSG
            MOV     AX, OFFSET THE_END
            MOV     BX, 10H ; ПЕРЕВОДИМ БАЙТЫ В AX В ПАРАГРАФЫ
            SUB     DX, DX
            DIV     BX
            INC     AX
            MOV             BX, AX
            MOV     AL, 0
            MOV     AH, 4AH; ОСВОБОЖДАЕМ ПАМЯТЬ
            INT     21H
            MOV             AH, 48H; ЗАПРАШИВАЕМ 64 КБ ПАМЯТИ
            MOV     BX, 4096H
            INT     21H
            POP     AX
    POP         DX
    POP         BX
    POP         CX
    RET
DEFINE_AVAIL_M  ENDP

DEFINE_EXTENDED_M       PROC    NEAR
    PUSH    AX
    PUSH    BX
    PUSH    SI
    PUSH    DX
    MOV     AL, 30H ; ЗАПИСЬ АДРЕСА ЯЧЕЙКИ CMOS
    OUT     70H, AL
    IN      AL, 71H  ; ЧТЕНИЕ МЛАДШЕГО БАЙТЫ
    MOV     BL, AH  ; РАЗМЕРА РАСШИРЕННОЙ ПАМЯТИ
    MOV     AL, 31H ; ЗАПИСЬ АДРЕСА ЯЧЕЙКИ CMOS
    OUT     70H, AL
```

```
    IN      AL, 71H ; ЧТЕНИЕ СТАРШЕГО БАЙТА
    MOV     AH, AL  ; РАЗМЕРА РАСШИРЕННОЙ ПАМЯТИ
    MOV     AL, BL ; В AX РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ
    SUB     DX, DX
    MOV     SI, OFFSET EXTENDED_M + 26
    CALL    WRD_TO_DEC
    MOV     DX, OFFSET EXTENDED_M
    CALL    WRITE_MSG
    POP     DX
    POP     SI
    POP     BX
    POP     AX
    RET
DEFINE_EXTENDED_M   ENDP


PRINT_MCB   PROC  NEAR
    MOV     DX, OFFSET MCB
    CALL    WRITE_MSG
    MOV     AH, 52H
    INT     21H
    SUB     AX, AX
    SUB     CX, CX
    MOV     ES, ES:[BX-2] ;СОХРАНЯЕМ АДРЕС ПЕРВОГО MCB
    MOV     BX, 1

    CYCLE:
    SUB     AX, AX
    MOV     AL, ES:[00H]
    CALL    BYTE_TO_HEX
    CMP     AX, 4135H ;ПРОВЕРЯЕМ ЯВЛЯЕТСЯ ЛИ ОН ПОСЛЕДНИМ
    JNE     CONTINUE
    MOV     BX, 0
    CONTINUE:
    MOV     DI, OFFSET MCB_TYPE + 10
    MOV     [DI], AX
    LEA     DI,MCB_SEG+21
    MOV     AX,ES:[0001H]
    CALL    WRD_TO_HEX
    MOV     AX, ES:[03H]
    MOV     CX, 10H
    MUL     CX
    MOV     SI, OFFSET MCB_SIZE + 15
    CALL    WRD_TO_DEC
                MOV             SI, OFFSET MCB_TAIL
                MOV             DI,0008H
                MOV             CX,4
                CYCLE1:
```

```
                MOV             AX,ES:[DI]; СОХРАНЯЕМ ХВОСТ
                MOV             [SI],AX
                ADD     DI,2H
                ADD             SI,2H
                LOOP    CYCLE1
        MOV     DX, OFFSET MCB_TYPE
        CALL    WRITE_MSG
        MOV     DX, OFFSET MCB_SEG
        CALL    WRITE_MSG
        MOV     DX, OFFSET MCB_SIZE
        CALL    WRITE_MSG
                MOV     DX, OFFSET MCB_TAIL
        CALL    WRITE_MSG
        CMP     BX, 0
        JE      _END
        MOV     AX, ES
        ADD     AX, ES:[0003H]
        INC     AX
        MOV     ES, AX
        JMP     CYCLE
        _END:
        RET
PRINT_MCB   ENDP

BEGIN:
        CALL DEFINE_AVAIL_M
        CALL DEFINE_EXTENDED_M
        CALL PRINT_MCB

; ВЫХОД В DOS
        XOR     AL,AL
        MOV     AH,3CH
        INT     21H
        RET
                THE_END:
TESTPC  ENDS
        END     START
```

LAB3_3.ASM

```
; ЛАБОРАТОРНАЯ РАБОТА 3
; ШАГ 3
; ПРОГРАММА ДО ОСВОБОЖДЕНИЯ ПАМЯТИ ЗАПРАШИВАЕТ 64КБ
; ПРОГРАММА ВЫВОДИТ
;          1)КОЛИЧЕСТВО ДОСТУПНОЙ ПАМЯТИ
;          2)РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ
;          3)ВЫВОДИТ ЦЕПОЧКУ БЛОКОВ УПРАВЛЕНИЯ ПАМЯТЬЮ
TESTPC  SEGMENT
     ASSUME  CS:TESTPC,  DS:TESTPC,  ES:NOTHING, SS:NOTHING
     ORG     100H
START:  JMP     BEGIN

; ДАННЫЕ
_ERROR                          DB              'ERROR',0DH,0AH,'$'
AVAILABEL_M     DB      'AMOUNT OF AVAILABLE MEMORY:       BYTES',0DH,0AH,'$'
EXTENDED_M      DB      'EXTENDED MEMORY SIZE:       KILOBYTES',0DH,0AH,'$'
MCB             DB      'CHAIN OF MEMORY CONTROL UNITS:',0DH,0AH,'$'
MCB_TYPE        DB      'MCB TYPE:   H, $'
MCB_SEG         DB      'SEGMENT`S ADRESS:     H, $'
MCB_SIZE        DB      'MCB SIZE:        B, $'
MCB_TAIL        DB      '               ',0DH,0AH,'$'
; ПРОЦЕДУРЫ

WRITE_MSG       PROC    NEAR
     MOV     AH,09H
     INT     21H
     RET
WRITE_MSG       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
; БАЙТ В AL ПЕРЕВОДИТСЯ В ДВА СИМВОЛА ШЕСТН. 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
; ПЕРВОД В 16 С/С 16-ТИ РАЗРЯДНОГО ЧИСЛА
; В AX - ЧИСЛО, 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
      XOR     AH,AH
      CALL    BYTE_TO_HEX
      MOV     [DI],AH
      DEC     DI
      MOV     [DI],AL
      POP     BX
      RET
WRD_TO_HEX      ENDP

WRD_TO_DEC      PROC    NEAR
; ПЕРЕВОД 2 БАЙТОВ В 10 С/С, SI - АДРЕС ПОЛЯ МЛАДШЕЙ ЦИФРЫ
      PUSH    CX
      PUSH    DX
      PUSH    AX
      MOV     CX,10
   WRD_LOOP_BD:
      DIV     CX
      OR      DL,30H
      MOV     [SI],DL
      DEC     SI
      XOR     DX,DX
      CMP     AX,10
      JAE     WRD_LOOP_BD
      CMP     AX,00H
      JBE     WRD_END_L
      OR      AL,30H
      MOV     [SI],AL
   WRD_END_L:
      POP     AX
      POP     DX
      POP     CX
      RET
```

```
WRD_TO_DEC      ENDP

DEFINE_EXTENDED_M       PROC    NEAR
        PUSH    AX
        PUSH    BX
        PUSH    SI
        PUSH    DX
        MOV     AL, 30H ; ЗАПИСЬ АДРЕСА ЯЧЕЙКИ CMOS
        OUT     70H, AL
        IN      AL, 71H  ; ЧТЕНИЕ МЛАДШЕГО БАЙТЫ
        MOV     BL, AH  ; РАЗМЕРА РАСШИРЕННОЙ ПАМЯТИ
        MOV     AL, 31H ; ЗАПИСЬ АДРЕСА ЯЧЕЙКИ CMOS
        OUT     70H, AL
        IN      AL, 71H ; ЧТЕНИЕ СТАРШЕГО БАЙТА
        MOV     AH, AL  ; РАЗМЕРА РАСШИРЕННОЙ ПАМЯТИ
        MOV     AL, BL ; В AX РАЗМЕР РАСШИРЕННОЙ ПАМЯТИ
        SUB     DX, DX
        MOV     SI, OFFSET EXTENDED_M + 26
        CALL    WRD_TO_DEC
        MOV     DX, OFFSET EXTENDED_M
        CALL    WRITE_MSG
        POP     DX
        POP     SI
        POP     BX
        POP     AX
        RET
DEFINE_EXTENDED_M   ENDP


PRINT_MCB   PROC  NEAR
        MOV     DX, OFFSET MCB
        CALL    WRITE_MSG
        MOV     AH, 52H
        INT     21H
        SUB     AX, AX
        SUB     CX, CX
        MOV     ES, ES:[BX-2] ;СОХРАНЯЕМ АДРЕС ПЕРВОГО MCB
        MOV     BX, 1

        CYCLE:
        SUB     AX, AX
        MOV     AL, ES:[00H]
        CALL    BYTE_TO_HEX
        CMP     AX, 4135H ;ПРОВЕРЯЕМ ЯВЛЯЕТСЯ ЛИ ОН ПОСЛЕДНИМ
        JNE     CONTINUE
        MOV     BX, 0
        CONTINUE:
        MOV     DI, OFFSET MCB_TYPE + 10
```

```
        MOV     [DI], AX
        LEA     DI,MCB_SEG+21
        MOV     AX,ES:[0001H]
        CALL    WRD_TO_HEX
        MOV     AX, ES:[03H]
        MOV     CX, 10H
        MUL     CX
        MOV     SI, OFFSET MCB_SIZE + 15
        CALL    WRD_TO_DEC
                MOV             SI, OFFSET MCB_TAIL
                MOV             DI,0008H
                MOV             CX,4
                CYCLE1:
                MOV             AX,ES:[DI]; СОХРАНЯЕМ ХВОСТ
                MOV             [SI],AX
                ADD     DI,2H
                ADD             SI,2H
                LOOP    CYCLE1
        MOV     DX, OFFSET MCB_TYPE
        CALL    WRITE_MSG
        MOV     DX, OFFSET MCB_SEG
        CALL    WRITE_MSG
        MOV     DX, OFFSET MCB_SIZE
        CALL    WRITE_MSG
                MOV     DX, OFFSET MCB_TAIL
        CALL    WRITE_MSG
        CMP     BX, 0
        JE      _END
        MOV     AX, ES
        ADD     AX, ES:[0003H]
        INC     AX
        MOV     ES, AX
        JMP     CYCLE
        _END:
        RET
    PRINT_MCB   ENDP

    BEGIN:
        MOV         AH,  4AH     ; ПЫТАЕМСЯ ОСВОБОДИТЬ ЗАВЕДОМО БОЛЬШОЙ РАЗМЕР
ПАМЯТИ
        MOV     BX, 0FFFFH ; НЕИСПОЛЬЗУЕМЫЙ ПРОГРАММОЙ
        INT     21H         ;В BX ВЕРНЕТСЯ РАЗМЕР ДОСТУПНОЙ ПАМЯТИ В ПАРАГРАФАХ
        MOV     AX, BX
        MOV     CX, 10H  ;ПЕРЕВОДИМ ПАРАГРАФЫ В БАЙТЫ
        MUL     CX
        MOV     SI, OFFSET AVAILABEL_M + 33
        CALL    WRD_TO_DEC
        MOV     DX, OFFSET AVAILABEL_M
```

```
    CALL    WRITE_MSG
                MOV             AH, 48H; ЗАПРАШИВАЕМ 64 КБ ПАМЯТИ
                MOV     BX, 4096H
                INT     21H
                JNC     CF_0
                MOV             DX, OFFSET _ERROR
                CALL    WRITE_MSG
                JMP             CF_1

                CF_0:
    CALL DEFINE_EXTENDED_M
    CALL PRINT_MCB

; ВЫХОД В DOS
                CF_1:
    XOR     AL,AL
    MOV     AH,3CH
    INT     21H
    RET
TESTPC  ENDS
    END     START
```