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 _____ Вологдин М.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

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 не страничная память и способ управления динамическими разделами. Для реализации управления памятью в этом случае строится список занятых и свободных участков памяти. Функции ядра, обеспечивающие управление основной памятью, просматривают и преобразуют этот список.

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

Описание функций:

TETR_TO_HEX	вспомогательная функция для работы функции BYTE_TO_HEX.
BYTE_TO_HEX	переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah
WRD_TO_HEX	переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры
BYTE_TO_DEC	переводит байт из AL в десятичную с/с и записывает получившееся число по адресу si, начиная с младшей цифры
PRINT	вызывает прерывание печати строки
DWORD_TO_DEC	переводит двойное слово из DX:AX в десятичную с/с и записывает получившееся число по адресу si, начиная с младшей цифры
CHECK_MEMORY	выводит информацию о памяти: количество доступной памяти, размер расширенной памяти, информацию о всех MCB
PRINT_MCB_INFO	выводит информацию о MCB по адресу ES

Описание структур данных:

STRAVLMEMINF	Строка, информирующая, что далее следует размер доступной памяти
STRAVLMEM	Строка для хранения размера доступной памяти
STREXPMEMINF	Строка, информирующая, что далее следует размер расширенной памяти
STREXPMEM	Строка для хранения размера расширенной памяти
STRMCBINFO	Строка, хранящая названия столбцов таблицы, в которую будут выводиться данные о MCB
STRMCBINFO2	Строка для хранения данных о MCB
STROVERFLOWERR	Строка, информирующая об ошибке переполнения
STRERROR	Строка, информирующая об ошибке
STRENDL	Строка, переводящая каретку на начало новой строки

Выполнение работы.

Был написан программный модуль .COM, который выбирает и распечатывает информацию о количестве доступной памяти, размере расширенной памяти и о блоках управления памятью

1) После выполнения первого шага запускаем полученный загрузочный модуль (результаты на рис. 1).

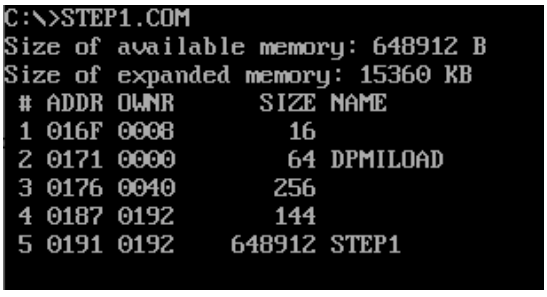

Рисунок 1 – Шаг 1

2) После выполнения второго шага снова запускаем загрузочный модуль (результаты на рис. 2).

Рисунок 2 – Шаг 2

3) После выполнения шага 3 запускаем загрузочный модуль (результаты на рис. 3). Появился ещё один узел списка блоков, равный выделенному блоку в нашем измененном модуле.

Рисунок 3 – Шаг 3

4) После выполнения шага 4 запускаем загрузочный модуль (результаты на рис. 4). ОС не смогла выделить память требуемого размера, поэтому мы видим сообщение об случившемся событии. Память модулю не была выделена.

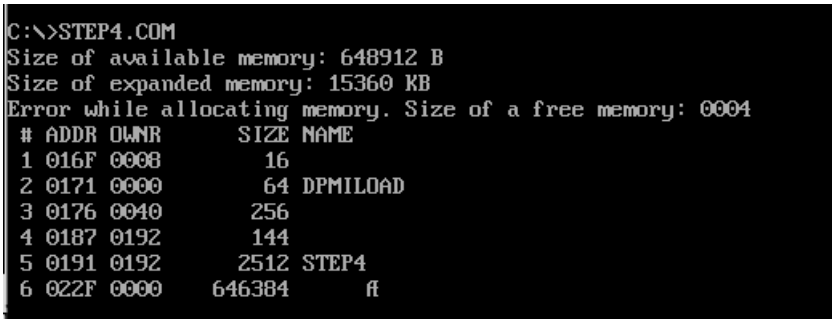

Рисунок 4 – Шаг 4

Выводы.

В процессе выполнения данной лабораторной работы были исследованы структуры данных и работа функций управления памятью ядра операционной системы.

Ответы на контрольные вопросы

1. Что означает «доступный объем памяти»?

Ответ: DOS является однопрограммной ОС, поэтому под доступным объемом памяти понимают весь объём памяти, доступный ОС, за исключением её ядра.

2. Где МСВ блок Вашей программы в списке?

Ответ: МСВ-блок программы находится перед каждым блоком памяти, который выделен для программы. Для каждого из шагов программе принадлежат следующие МСВ:

- Шаг 1: 4 и 5;
- Шаг 2: 4 и 5, 6 – свободная память;
- Шаг 3: 4, 5 и 6;
- Шаг 4: 4 и 5.

  4-ый блок МСВ для каждого из шагов владеет блоком памяти, содержащим область среды программы.

3. Какой размер памяти занимает программа в каждом случае?

Ответ:

В первом случае программа занимала всю доступную память (648912 байт).

Во втором случае необходимый программе объем памяти: 648912 - 646480 = 2432 байт — 16 байтов под МСВ, 2416 байт— под сегмент программы.

В третьем случае 2416+16+ 65536 +16 = 66384 байт — 16 байтов под МСВ программы, 16 байтов под МСВ выделенного блока, 65536 — размер выделенного блока, 2416 — сегмент программы.

В последнем случае 2512+16 байт = 2528 байта, поскольку не удалось выделить блок 65536 КБ.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД. ШАГ 1.

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
START: JMP BEGIN
STRAVLMEMINF DB 'SIZE OF AVAILABLE MEMORY: $'
STRAVLMEM DB '       B$'
STREXPMEMINF DB 'SIZE OF EXPANDED MEMORY: $'
STREXPMEM DB '      KB$'
STRMCBINFO  DB ' # ADDR OWNR      SIZE NAME$'
STRMCBINFO2 DB '                                $'
STROVERFLOWERR DB 'OVERFLOW ERROR.$'
STRERROR DB 'ERROR.$'
STRENDL DB 0DH,0AH,'$'
;--------------------------------------
PRINT PROC NEAR
     PUSH AX
     MOV AH,09H
     INT 21H
     POP AX
     RET
PRINT 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
;---------------------------------------
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
;---------------------------------------
BYTE_TO_DEC PROC NEAR
      PUSH CX
      PUSH DX
      XOR AH,AH
      XOR DX,DX
```

```
      MOV CX,10
LOOP_BD: DIV CX
      OR DL,30H
      MOV [SI],DL
      DEC SI
      XOR DX,DX
      CMP AX,10
      JAE LOOP_BD
      CMP AL,00H
      JE END_L
      OR AL,30H
      MOV [SI],AL
END_L: POP DX
      POP CX
      RET
BYTE_TO_DEC ENDP
;---------------------------------------
DWORD_TO_DEC PROC NEAR
      PUSH AX
      PUSH CX
      PUSH DX
      MOV CX,10
LOOP_DWD:
      DIV CX
      OR DL,30H
      MOV [SI],DL
      DEC SI
      XOR DX,DX
      CMP AX,10
      JAE LOOP_DWD
      CMP AL,00H
      JE DWD_END
      OR AL,30H
      MOV [SI],AL
```

```
DWD_END:
      POP DX
      POP CX
      POP AX
      RET
DWORD_TO_DEC ENDP
;---------------------------------------
CHECK_MEMORY PROC NEAR
      MOV DX,OFFSET STRAVLMEMINF
      CALL PRINT
      MOV SI,OFFSET STRAVLMEM+5
      MOV AH,4AH
      MOV BX,0FFFFH
      INT 21H
      MOV AX,BX
      XOR DX,DX
      MOV BX,10H
      MUL BX
      CALL DWORD_TO_DEC
      MOV DX,OFFSET STRAVLMEM
      CALL PRINT
      MOV DX,OFFSET STRENDL
      CALL PRINT

      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 STREXPMEM+4
CALL DWORD_TO_DEC

MOV DX,OFFSET STREXPMEMINF
CALL PRINT
MOV DX,OFFSET STREXPMEM
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

MOV DX, OFFSET STRMCBINFO
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

MOV AH,52H
INT 21H
MOV AX,ES:[BX-2]
MOV ES,AX
MOV DX,ES

MOV CX,01H
XOR BX,BX
CM_CYCLE:
     CALL PRINT_MCB_INFO

     CMP BYTE PTR ES:[00H],5AH
     JE CM_EXIT
     INC CX
     INC DX
     ADD DX,ES:[03H]
     MOV ES,DX
JMP CM_CYCLE
CM_EXIT:
```

```
      RET
CHECK_MEMORY ENDP
;---------------------------------------
PRINT_MCB_INFO PROC NEAR
      PUSH AX
      PUSH DX
      PUSH BX
      PUSH SI
      PUSH ES
      PUSH DI

      MOV SI,OFFSET STRMCBINFO2+1
      MOV AX,CX
      CALL BYTE_TO_DEC

      MOV DI,OFFSET STRMCBINFO2+6
      MOV AX,ES
      CALL WRD_TO_HEX

      MOV DI,OFFSET STRMCBINFO2+11
      MOV AX,ES:[01H]
      CALL WRD_TO_HEX

      MOV SI,OFFSET STRMCBINFO2+21
      MOV AX,ES:[03H]
      CMP AX,0A000H
      JB PCI_OVERFLOWCHECK
            MOV DX,OFFSET STROVERFLOWERR
            CALL PRINT
            XOR AL,AL
            MOV AH,4CH
            INT 21H
      PCI_OVERFLOWCHECK:
```

```
MOV BX,10H
MUL BX
CALL DWORD_TO_DEC

MOV BX,OFFSET STRMCBINFO2+30
MOV DX,ES:[0FH-1]
MOV [BX-1],DX
MOV DX,ES:[0FH-3]
MOV [BX-3],DX
MOV DX,ES:[0FH-5]
MOV [BX-5],DX
MOV DX,ES:[0FH-7]
MOV [BX-7],DX

MOV DX,OFFSET STRMCBINFO2
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

MOV AL,' '
MOV AH,' '
MOV SI,OFFSET STRMCBINFO2
MOV [SI+20],AX
MOV [SI+18],AX
MOV [SI+16],AX
MOV [SI+14],AX
MOV [SI+12],AX

POP DI
POP ES
POP SI
POP BX
POP DX
POP AX
```

```
     RET
PRINT_MCB_INFO ENDP
;---------------------------------------
BEGIN:
     CALL CHECK_MEMORY
     XOR AL,AL
     MOV AH,4CH
     INT 21H
TESTPC ENDS
 END START
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД. ШАГ 2.

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
START: JMP BEGIN
STRAVLMEMINF DB 'SIZE OF AVAILABLE MEMORY: $'
STRAVLMEM DB '       B$'
STREXPMEMINF DB 'SIZE OF EXPANDED MEMORY: $'
STREXPMEM DB '      KB$'
STRMCBINFO  DB ' # ADDR OWNR      SIZE NAME$'
STRMCBINFO2 DB '                                $'
STROVERFLOWERR DB 'OVERFLOW ERROR.$'
STRERROR DB 'ERROR.$'
STRENDL DB 0DH,0AH,'$'
;--------------------------------------
PRINT PROC NEAR
     PUSH AX
     MOV AH,09H
     INT 21H
     POP AX
     RET
PRINT 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
;---------------------------------------
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
;---------------------------------------
BYTE_TO_DEC PROC NEAR
      PUSH CX
      PUSH DX
      XOR AH,AH
      XOR DX,DX
```

```
      MOV CX,10
LOOP_BD: DIV CX
      OR DL,30H
      MOV [SI],DL
      DEC SI
      XOR DX,DX
      CMP AX,10
      JAE LOOP_BD
      CMP AL,00H
      JE END_L
      OR AL,30H
      MOV [SI],AL
END_L: POP DX
      POP CX
      RET
BYTE_TO_DEC ENDP
;---------------------------------------
DWORD_TO_DEC PROC NEAR
      PUSH AX
      PUSH CX
      PUSH DX
      MOV CX,10
LOOP_DWD:
      DIV CX
      OR DL,30H
      MOV [SI],DL
      DEC SI
      XOR DX,DX
      CMP AX,10
      JAE LOOP_DWD
      CMP AL,00H
      JE DWD_END
      OR AL,30H
      MOV [SI],AL
```

```
DWD_END:
      POP DX
      POP CX
      POP AX
      RET
DWORD_TO_DEC ENDP
;---------------------------------------
CHECK_MEMORY PROC NEAR
      MOV DX,OFFSET STRAVLMEMINF
      CALL PRINT
      MOV SI,OFFSET STRAVLMEM+5
      MOV AH,4AH
      MOV BX,0FFFFH
      INT 21H
      MOV AX,BX
      XOR DX,DX
      MOV BX,10H
      MUL BX
      CALL DWORD_TO_DEC
      MOV DX,OFFSET STRAVLMEM
      CALL PRINT
      MOV DX,OFFSET STRENDL
      CALL PRINT

      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 STREXPMEM+4
CALL DWORD_TO_DEC

MOV DX,OFFSET STREXPMEMINF
CALL PRINT
MOV DX,OFFSET STREXPMEM
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

; ШАГ 2: ОСВОБОЖДЕНИЕ ЛИШНЕЙ ПАМЯТИ
MOV AX,OFFSET END_LABEL
MOV BX,10H
XOR DX,DX
DIV BX
INC AX
ADD AX,040H
ADD AX,020H
MOV BX,AX
MOV AH,4AH
INT 21H
JNC CM_FREE_MEM_NO_ERR
     MOV DX,OFFSET STRERROR
     CALL PRINT
     MOV AH,4AH
     INT 21H
CM_FREE_MEM_NO_ERR:

MOV DX, OFFSET STRMCBINFO
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

MOV AH,52H
```

```
      INT 21H
      MOV AX,ES:[BX-2]
      MOV ES,AX
      MOV DX,ES

      MOV CX,01H
      XOR BX,BX
      CM_CYCLE:
                CALL PRINT_MCB_INFO

            CMP BYTE PTR ES:[00H],5AH
            JE CM_EXIT
            INC CX
            INC DX
            ADD DX,ES:[03H]
            MOV ES,DX
      JMP CM_CYCLE
      CM_EXIT:

      RET
CHECK_MEMORY ENDP
;--------------------------------------
PRINT_MCB_INFO PROC NEAR
      PUSH AX
      PUSH DX
      PUSH BX
      PUSH SI
      PUSH ES
      PUSH DI

      MOV SI,OFFSET STRMCBINFO2+1
      MOV AX,CX
      CALL BYTE_TO_DEC
```

```
MOV DI,OFFSET STRMCBINFO2+6
MOV AX,ES
CALL WRD_TO_HEX

MOV DI,OFFSET STRMCBINFO2+11
MOV AX,ES:[01H]
CALL WRD_TO_HEX

MOV SI,OFFSET STRMCBINFO2+21
MOV AX,ES:[03H]
CMP AX,0A000H
JB PCI_OVERFLOWCHECK
     MOV DX,OFFSET STROVERFLOWERR
     CALL PRINT
     XOR AL,AL
     MOV AH,4CH
     INT 21H
PCI_OVERFLOWCHECK:
MOV BX,10H
MUL BX
CALL DWORD_TO_DEC

MOV BX,OFFSET STRMCBINFO2+30
MOV DX,ES:[0FH-1]
MOV [BX-1],DX
MOV DX,ES:[0FH-3]
MOV [BX-3],DX
MOV DX,ES:[0FH-5]
MOV [BX-5],DX
MOV DX,ES:[0FH-7]
MOV [BX-7],DX

MOV DX,OFFSET STRMCBINFO2
CALL PRINT
```

```
        MOV DX,OFFSET STRENDL
        CALL PRINT

        MOV AL,' '
        MOV AH,' '
        MOV SI,OFFSET STRMCBINFO2
        MOV [SI+20],AX
        MOV [SI+18],AX
        MOV [SI+16],AX
        MOV [SI+14],AX
        MOV [SI+12],AX

        POP DI
        POP ES
        POP SI
        POP BX
        POP DX
        POP AX

        RET
PRINT_MCB_INFO ENDP
;---------------------------------------
BEGIN:
        CALL CHECK_MEMORY
        XOR AL,AL
        MOV AH,4CH
        INT 21H
        END_LABEL:
TESTPC ENDS
 END START
```

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД. ШАГ 3.

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
START: JMP BEGIN
STRAVLMEMINF DB 'SIZE OF AVAILABLE MEMORY: $'
STRAVLMEM DB '       B$'
STREXPMEMINF DB 'SIZE OF EXPANDED MEMORY: $'
STREXPMEM DB '      KB$'
STRMCBINFO  DB ' # ADDR OWNR      SIZE NAME$'
STRMCBINFO2 DB '                                $'
STROVERFLOWERR DB 'OVERFLOW ERROR.$'
STRERROR DB 'ERROR.$'
STRENDL DB 0DH,0AH,'$'
;--------------------------------------
PRINT PROC NEAR
     PUSH AX
     MOV AH,09H
     INT 21H
     POP AX
     RET
PRINT 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
;---------------------------------------
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
;---------------------------------------
BYTE_TO_DEC PROC NEAR
      PUSH CX
      PUSH DX
      XOR AH,AH
      XOR DX,DX
```

```
      MOV CX,10
LOOP_BD: DIV CX
      OR DL,30H
      MOV [SI],DL
      DEC SI
      XOR DX,DX
      CMP AX,10
      JAE LOOP_BD
      CMP AL,00H
      JE END_L
      OR AL,30H
      MOV [SI],AL
END_L: POP DX
      POP CX
      RET
BYTE_TO_DEC ENDP
;---------------------------------------
DWORD_TO_DEC PROC NEAR
      PUSH AX
      PUSH CX
      PUSH DX
      ;XOR AH,AH
      ;XOR DX,DX
      MOV CX,10
LOOP_DWD:
      DIV CX
      OR DL,30H
      MOV [SI],DL
      DEC SI
      XOR DX,DX
      CMP AX,10
      JAE LOOP_DWD
      CMP AL,00H
      JE DWD_END
```

```
      OR AL,30H
      MOV [SI],AL
DWD_END:
      POP DX
      POP CX
      POP AX
      RET
DWORD_TO_DEC ENDP
;---------------------------------------
CHECK_MEMORY PROC NEAR
      MOV DX,OFFSET STRAVLMEMINF
      CALL PRINT
      MOV SI,OFFSET STRAVLMEM+5
      MOV AH,4AH
      MOV BX,0FFFFH
      INT 21H
      MOV AX,BX
      XOR DX,DX
      MOV BX,10H
      MUL BX
      CALL DWORD_TO_DEC
      MOV DX,OFFSET STRAVLMEM
      CALL PRINT
      MOV DX,OFFSET STRENDL
      CALL PRINT

      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 STREXPMEM+4
CALL DWORD_TO_DEC

MOV DX,OFFSET STREXPMEMINF
CALL PRINT
MOV DX,OFFSET STREXPMEM
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

MOV AX,OFFSET END_LABEL
MOV BX,10H
XOR DX,DX
DIV BX
INC AX
ADD AX,040H
ADD AX,020H
MOV BX,AX
MOV AH,4AH
INT 21H
JNC CM_FREE_MEM_NO_ERR
     MOV DX,OFFSET STRERROR
     CALL PRINT
     MOV AH,4AH
     INT 21H
CM_FREE_MEM_NO_ERR:

; ШАГ 3: ЗАПРОС 64КБ ПАМЯТИ
MOV BX,1000H
MOV AH,48H
INT 21H
```

```
	MOV DX, OFFSET STRMCBINFO
	CALL PRINT
	MOV DX,OFFSET STRENDL
	CALL PRINT

	MOV AH,52H
	INT 21H
	MOV AX,ES:[BX-2]
	MOV ES,AX
	MOV DX,ES

	MOV CX,01H
	XOR BX,BX
	CM_CYCLE:
		CALL PRINT_MCB_INFO

		CMP BYTE PTR ES:[00H],5AH
		JE CM_EXIT
		INC CX
		INC DX
		ADD DX,ES:[03H]
		MOV ES,DX
	JMP CM_CYCLE
	CM_EXIT:

	RET
CHECK_MEMORY ENDP
;--------------------------------------
PRINT_MCB_INFO PROC NEAR
	PUSH AX
	PUSH DX
	PUSH BX
	PUSH SI
	PUSH ES
```

```
PUSH DI

MOV SI,OFFSET STRMCBINFO2+1
MOV AX,CX
CALL BYTE_TO_DEC

MOV DI,OFFSET STRMCBINFO2+6
MOV AX,ES
CALL WRD_TO_HEX

MOV DI,OFFSET STRMCBINFO2+11
MOV AX,ES:[01H]
CALL WRD_TO_HEX

MOV SI,OFFSET STRMCBINFO2+21
MOV AX,ES:[03H]
CMP AX,0A000H
JB PCI_OVERFLOWCHECK
     MOV DX,OFFSET STROVERFLOWERR
     CALL PRINT
     XOR AL,AL
     MOV AH,4CH
     INT 21H
PCI_OVERFLOWCHECK:
MOV BX,10H
MUL BX
CALL DWORD_TO_DEC

MOV BX,OFFSET STRMCBINFO2+30
MOV DX,ES:[0FH-1]
MOV [BX-1],DX
MOV DX,ES:[0FH-3]
MOV [BX-3],DX
MOV DX,ES:[0FH-5]
```

```
      MOV [BX-5],DX
      MOV DX,ES:[0FH-7]
      MOV [BX-7],DX

      MOV DX,OFFSET STRMCBINFO2
      CALL PRINT
      MOV DX,OFFSET STRENDL
      CALL PRINT

      MOV AL,' '
      MOV AH,' '
      MOV SI,OFFSET STRMCBINFO2
      MOV [SI+18],AX
      MOV [SI+16],AX
      MOV [SI+14],AX
      MOV [SI+12],AX

      POP DI
      POP ES
      POP SI
      POP BX
      POP DX
      POP AX
      RET
PRINT_MCB_INFO ENDP
;---------------------------------------
BEGIN:
      CALL CHECK_MEMORY
      XOR AL,AL
      MOV AH,4CH
      INT 21H
      END_LABEL:
TESTPC ENDS
 END START
```

ПРИЛОЖЕНИЕ Г
ИСХОДНЫЙ КОД. ШАГ 4.

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
START: JMP BEGIN
STRAVLMEMINF DB 'SIZE OF AVAILABLE MEMORY: $'
STRAVLMEM DB '       B$'
STREXPMEMINF DB 'SIZE OF EXPANDED MEMORY: $'
STREXPMEM DB '      KB$'
STRMCBINFO  DB ' # ADDR OWNR      SIZE NAME$'
STRMCBINFO2 DB '                                $'
STROVERFLOWERR DB 'OVERFLOW ERROR.$'
STRERROR DB 'ERROR.$'
STRMEMERROR DB 'ERROR WHILE ALLOCATING MEMORY. SIZE OF A FREE
MEMORY: $'
STRMEMERRORFREEMEM DB '     $'
STRENDL DB 0DH,0AH,'$'
;---------------------------------------
PRINT PROC NEAR
     PUSH AX
     MOV AH,09H
     INT 21H
     POP AX
     RET
PRINT 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
;---------------------------------------
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
;---------------------------------------
BYTE_TO_DEC PROC NEAR
      PUSH CX
```

```
      PUSH DX
      XOR AH,AH
      XOR DX,DX
      MOV CX,10
LOOP_BD: DIV CX
      OR DL,30H
      MOV [SI],DL
      DEC SI
      XOR DX,DX
      CMP AX,10
      JAE LOOP_BD
      CMP AL,00H
      JE END_L
      OR AL,30H
      MOV [SI],AL
END_L: POP DX
      POP CX
      RET
BYTE_TO_DEC ENDP
;---------------------------------------
DWORD_TO_DEC PROC NEAR
      PUSH AX
      PUSH CX
      PUSH DX
      ;XOR AH,AH
      ;XOR DX,DX
      MOV CX,10
LOOP_DWD:
      DIV CX
      OR DL,30H
      MOV [SI],DL
      DEC SI
      XOR DX,DX
      CMP AX,10
```

```
      JAE LOOP_DWD
      CMP AL,00H
      JE DWD_END
      OR AL,30H
      MOV [SI],AL
DWD_END:
      POP DX
      POP CX
      POP AX
      RET
DWORD_TO_DEC ENDP
;---------------------------------------
CHECK_MEMORY PROC NEAR
      MOV DX,OFFSET STRAVLMEMINF
      CALL PRINT
      MOV SI,OFFSET STRAVLMEM+5
      MOV AH,4AH
      MOV BX,0FFFFH
      INT 21H
      MOV AX,BX
      XOR DX,DX
      MOV BX,10H
      MUL BX
      CALL DWORD_TO_DEC
      MOV DX,OFFSET STRAVLMEM
      CALL PRINT
      MOV DX,OFFSET STRENDL
      CALL PRINT

      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 STREXPMEM+4
 CALL DWORD_TO_DEC

 MOV DX,OFFSET STREXPMEMINF
 CALL PRINT
 MOV DX,OFFSET STREXPMEM
 CALL PRINT
 MOV DX,OFFSET STRENDL
 CALL PRINT

 ; ШАГ 4: ЗАПРОС 64 КБ ПАМЯТИ ПЕРЕД ОСВОБОЖДЕНИЕМ ЛИШНЕЙ
ПАМЯТИ
 MOV BX,1000H
 MOV AH,48H
 INT 21H
 JNC CM_MEM_NO_ERR1
      MOV DX,OFFSET STRMEMERROR
      CALL PRINT
      MOV DI,OFFSET STRMEMERRORFREEMEM+3
      MOV AX,BX
      CALL WRD_TO_HEX
      MOV DX,OFFSET STRMEMERRORFREEMEM
      CALL PRINT
      MOV DX,OFFSET STRENDL
      CALL PRINT
      MOV AH,4AH
      INT 21H
 CM_MEM_NO_ERR1:
```

```
MOV AX,OFFSET END_LABEL
MOV BX,10H
XOR DX,DX
DIV BX
INC AX
ADD AX,040H
ADD AX,020H
MOV BX,AX
MOV AH,4AH
INT 21H
JNC CM_FREE_MEM_NO_ERR2
     MOV DX,OFFSET STRERROR
     CALL PRINT
     MOV DX,OFFSET STRENDL
     CALL PRINT
     MOV AH,4AH
     INT 21H
CM_FREE_MEM_NO_ERR2:

MOV DX, OFFSET STRMCBINFO
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

MOV AH,52H
INT 21H
MOV AX,ES:[BX-2]
MOV ES,AX
MOV DX,ES

MOV CX,01H
XOR BX,BX
CM_CYCLE:
          CALL PRINT_MCB_INFO
```

```
            CMP BYTE PTR ES:[00H],5AH
            JE CM_EXIT
            INC CX
            INC DX
            ADD DX,ES:[03H]
            MOV ES,DX
      JMP CM_CYCLE
      CM_EXIT:

      RET
CHECK_MEMORY ENDP
;---------------------------------------
PRINT_MCB_INFO PROC NEAR
      PUSH AX
      PUSH DX
      PUSH BX
      PUSH SI
      PUSH ES
      PUSH DI

      MOV SI,OFFSET STRMCBINFO2+1
      MOV AX,CX
      CALL BYTE_TO_DEC

      MOV DI,OFFSET STRMCBINFO2+6
      MOV AX,ES
      CALL WRD_TO_HEX

      MOV DI,OFFSET STRMCBINFO2+11
      MOV AX,ES:[01H]
      CALL WRD_TO_HEX

      MOV SI,OFFSET STRMCBINFO2+21
```

```
MOV AX,ES:[03H]
CMP AX,0A000H
JB PCI_OVERFLOWCHECK
     MOV DX,OFFSET STROVERFLOWERR
     CALL PRINT
     XOR AL,AL
     MOV AH,4CH
     INT 21H
PCI_OVERFLOWCHECK:
MOV BX,10H
MUL BX
CALL DWORD_TO_DEC

MOV BX,OFFSET STRMCBINFO2+30
MOV DX,ES:[0FH-1]
MOV [BX-1],DX
MOV DX,ES:[0FH-3]
MOV [BX-3],DX
MOV DX,ES:[0FH-5]
MOV [BX-5],DX
MOV DX,ES:[0FH-7]
MOV [BX-7],DX

MOV DX,OFFSET STRMCBINFO2
CALL PRINT
MOV DX,OFFSET STRENDL
CALL PRINT

MOV AL,' '
MOV AH,' '
MOV SI,OFFSET STRMCBINFO2
MOV [SI+20],AX
MOV [SI+18],AX
MOV [SI+16],AX
```

```
        MOV [SI+14],AX
        MOV [SI+12],AX

        POP DI
        POP ES
        POP SI
        POP BX
        POP DX
        POP AX

        RET
PRINT_MCB_INFO ENDP
;---------------------------------------
BEGIN:
        CALL CHECK_MEMORY
        XOR AL,AL
        MOV AH,4CH
        INT 21H
        END_LABEL:
TESTPC ENDS
 END START
```