**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. 6383 |  | Быков И. В. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

**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 исследуются структуры данных и работа функций управления памятью ядра операционной системы.

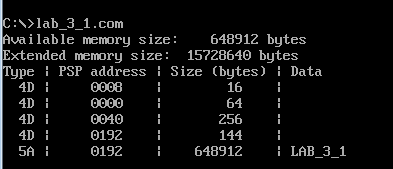
**Описание функций.**

|  |  |
| --- | --- |
| **Название функции** | **Назначение** |
| BYTE\_TO\_HEX | переводит число AL в коды символов 16-ой с/с, записывая получившееся в al и ah |
| TETR\_TO\_HEX | вспомогательная функция для работы функции BYTE\_TO\_HEX |
| WORD\_TO\_HEX | переводит 16-разрядное число AX в строку в 16-ой с/с, в AX – число, DI – адрес младшего символа |
| BYTE\_TO\_DEC | перевод в 10 с/с. В DI – адрес младшей цифры |
| TBYTE\_TO\_DEC | перевод числа, записанного в DX:AX в 10 с/с, DI – адрес младшего символа |
| PRINT\_NUM | выделяем цифру из числа в АХ и пишем ее в DI |
| CLEAR\_BYTES | очистка строки BYTE\_ |
| PRINTSTR | вывод строки |
| NEWLINE | перенос на новую строку |
| AV\_MEM | подсчет количества доступной памяти |
| EXT\_MEM | подсчет размера расширенной памяти |
| PRINT\_TABLE | вывод цепочки блоков управления памятью |

**Последовательность действий, выполняемых утилитой.**

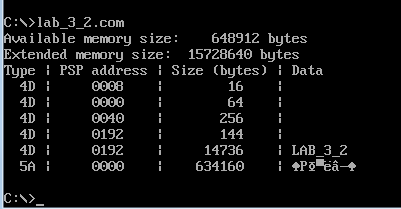
1. Вывод количества доступной памяти.
2. Вывод размера расширенной памяти.
3. Вывод цепочки блоков управления памятью.

**Шаг 1.** Запуск базовой программы продемонстрирован на рис. 1.



*Рисунок 1 – результат работы программы lab\_3\_1.com*

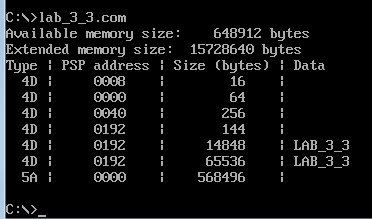
**Шаг 2.** Освобождена неиспользованная память. Результат показан на рис. 2.



*Рисунок 2 – результат работы программы lab3\_2.com*

Проанализировав результаты программы заметим, что размер блока памяти уменьшился до 14 736 байт и появился блок свободной памяти на 634160 байт.

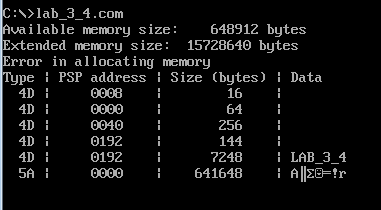
**Шаг 3.** Освобождена память и запрошено 64 Кб памяти. Результат работы программы представлен на рис. 3.



*Рисунок 3 – результат работы программы lab\_3\_3.com*

Проанализировав результаты программы заметим, что размер блока свободной памяти уменьшился на (634160 – 568496 = 65664 б) 64 Кб с точностью до десятых и образовался новый участок памяти в 64 Кб (6 строка)

**Шаг 4.** Запрошено 64 Кб памяти до ее освобождения. Результат показан на рис. 4.



*Рисунок 4 – результат работы программы lab\_3\_4.com*

Проанализировав результат работы программы заметим, что как и предполагалось, выделение памяти при отсутствии свободной памяти завершается неудачей, о чем и сообщает программа.

**Вывод.**

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»?*

Это свободный объем памяти, который загрузчик может выделить программе.

1. *Где MCB блок Вашей программы в списке?*

Блок первой программы расположен в конце списка (см. рис. 1).

Блок второй программы есть предпоследняя строка списка (см. рис. 2). В последней строке расположен блок освобожденной памяти.

Блок третьей программы расположен в пятой строке, после него идут блоки выделенной по запросу памяти и свободной памяти соответственно (см. рис. 3).

Блок четвертой программы есть предпоследняя строка списка (см. рис. 4)

1. *Какой размер памяти занимает программа в каждом случае?*

В первом случае программа занимает всю выделенную память: 648912 б.

Во втором случае программа занимает свой объем: 14736 б.

В третьем случае программа занимает свой размер и объем выделенной памяти: 14 848 + 65 536 = 80 834 б.

В четвертом случае: 7248 б.

1. **Ответы на контрольные вопросы.**
2. Что означает «доступный объем памяти?»

Доступный объем памяти – это тот объем памяти, в который можно загружать пользовательские программы.

1. Где MCB блок вашей программы в списке?

MCB блок программы находится в конце списка.

1. Какой размер памяти занимает программа в каждом случае?

В шаге 1 – 648912 Б, шаге 2 – 6432 Б, шаге 3 – 6432+65536 Б, шаге 4 – 6432 Б.

**Приложение А.**

Код программы lab\_3\_1.asm

.286

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: jmp BEGIN

; DATA

AVAILABLE\_MEM db 'Available memory size: ','$'

EXTENDEN\_MEM db 'Extended memory size: ','$'

BYTE\_ db ' bytes', '$'

STR\_HEXWRD db ' ','$'

TABLE db 'Type | PSP address | Size (bytes) | Data$'

TYPE\_ db ' |$'

PSP\_ db ' |$'

SIZE\_ db ' |$'

DATA\_ db ' $'

LINE\_ db 0DH,0AH,'$'

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

push AX

xor AH,AH

xor DX,DX

mov CX,10

LOOP\_BD: div CX

or DL,30H

mov [DI],DL

dec DI

xor DX,DX

cmp AX,10

jae LOOP\_BD

cmp AL,00H

je END\_L

or AL,30H

mov [DI],AL

END\_L: pop AX

pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

PRINT\_NUM PROC near

push CX

mov CX,0AH

div CX

or DX,30H

mov [DI],DL

dec DI

xor DX,DX

pop CX

ret

PRINT\_NUM ENDP

TBYTE\_TO\_DEC PROC near

push AX

push CX

push DX

mov CX,2710H

cmp DX,0H

jnz LOOP\_TBD

cmp AX,2710H

jb ENDLPTBD

LOOP\_TBD: div CX

push AX

mov AX,DX

xor DX,DX

call PRINT\_NUM

call PRINT\_NUM

call PRINT\_NUM

call PRINT\_NUM

pop AX

cmp AX,2710H

jnb LOOP\_TBD

ENDLPTBD: cmp AX,0H

JZ EXIT\_TBD

call PRINT\_NUM

jmp ENDLPTBD

EXIT\_TBD: pop DX

pop CX

pop AX

ret

TBYTE\_TO\_DEC ENDP

CLEAR\_BYTES PROC near

push DI

push CX

push DX

mov DI,OFFSET BYTE\_+8

mov CX,9

mov DL,' '

LOOP\_CLR: mov [DI],DL

dec DI

loop LOOP\_CLR

pop DX

pop CX

pop DI

ret

CLEAR\_BYTES ENDP

PRINT\_STRING PROC near

push AX

mov AH,09H

int 21H

pop AX

ret

PRINT\_STRING ENDP

NEWLINE PROC near

push AX

push DX

mov AH,02H

mov DL,0DH

int 21H

mov DL,0AH

int 21H

pop DX

pop AX

ret

NEWLINE ENDP

; AVAILABLE MEMORY

AV\_MEM PROC near

push AX

push BX

push DX

mov AH,4AH

mov BX,0FFFFH

int 21H

clc

mov DX,BX

mov AX,BX

shr DX,12

shl AX,4

mov DI,OFFSET BYTE\_+8

call TBYTE\_TO\_DEC

mov DX,OFFSET AVAILABLE\_MEM

call PRINT\_STRING

mov DX,OFFSET BYTE\_

call PRINT\_STRING

call CLEAR\_BYTES

mov DX,OFFSET LINE\_

call PRINT\_STRING

pop DX

pop BX

pop AX

ret

AV\_MEM ENDP

; EXTENDED MEMORY

EXT\_MEM PROC near

push DX

push AX

push BX

push DI

mov DX,OFFSET EXTENDEN\_MEM

call PRINT\_STRING

mov AL,30H

out 70H,AL

in AL,71H

mov BL,AL

mov AL,31H

out 70H,AL

in AL,71H

mov BH,AL

mov DX,BX

shr DX,6

mov AX,BX

shl AX,10

mov DI,OFFSET BYTE\_+8

call TBYTE\_TO\_DEC

mov DX,OFFSET BYTE\_

call PRINT\_STRING

call CLEAR\_BYTES

mov DX,OFFSET LINE\_

call PRINT\_STRING

pop DI

pop BX

pop AX

pop DX

ret

EXT\_MEM ENDP

PRINT\_TABLE PROC near

push AX

push BX

push CX

push DX

push ES

push DI

mov DX,OFFSET TABLE

call PRINT\_STRING

call NEWLINE

; GET "LIST OF LISTS"

mov AH,52H

int 21H

mov ES,ES:[BX-2]

PRINT\_LINE:

xor AX,AX

mov AL,ES:00H

mov DI,OFFSET TYPE\_+3

call BYTE\_TO\_HEX

mov [DI],AH

mov [DI-1],AL

mov DX,OFFSET TYPE\_

call PRINT\_STRING

mov AX,ES:01H

mov DI,OFFSET PSP\_+8

call WRD\_TO\_HEX

mov DX,OFFSET PSP\_

call PRINT\_STRING

mov AX,ES:03H

mov DI,OFFSET SIZE\_+9

mov DX,AX

shr DX,12

shl AX,4

call TBYTE\_TO\_DEC

mov DX,OFFSET SIZE\_

call PRINT\_STRING

mov DI,OFFSET DATA\_+1

mov AX,ES:08H

mov [DI],AX

mov AX,ES:0AH

mov [DI+2H],AX

mov AX,ES:0CH

mov [DI+4H],AX

mov AX,ES:0EH

mov [DI+6H],AX

mov DX,OFFSET DATA\_

call PRINT\_STRING

mov DX,OFFSET LINE\_

call PRINT\_STRING

mov AL,ES:00H

cmp AL,5AH

je EXIT

mov AX,ES

mov BX,ES:03H

add AX,BX

INC AX

mov ES,AX

jmp PRINT\_LINE

EXIT: pop DI

pop ES

pop DX

pop CX

pop BX

pop AX

ret

PRINT\_TABLE ENDP

; CODE

BEGIN:

call AV\_MEM

call EXT\_MEM

call PRINT\_TABLE

mov AH, 10H

int 16H

xor AL,AL

mov AH,4CH

int 21H

TESTPC ENDS

END START

**Приложение Б.**

Код программы lab\_3\_2.asm

.286

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: jmp BEGIN

; DATA

AVAILABLE\_MEM db 'Available memory size: ','$'

EXTENDEN\_MEM db 'Extended memory size: ','$'

BYTE\_ db ' bytes', '$'

STR\_HEXWRD db ' ','$'

TABLE db 'Type | PSP address | Size (bytes) | Data$'

TYPE\_ db ' |$'

PSP\_ db ' |$'

SIZE\_ db ' |$'

DATA\_ db ' $'

LINE\_ db 0DH,0AH,'$'

;PROGRAM\_LENGTH = ($ - START)/16

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

push AX

xor AH,AH

xor DX,DX

mov CX,10

LOOP\_BD: div CX

or DL,30H

mov [DI],DL

dec DI

xor DX,DX

cmp AX,10

jae LOOP\_BD

cmp AL,00H

je END\_L

or AL,30H

mov [DI],AL

END\_L: pop AX

pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

PRINT\_NUM PROC near

push CX

mov CX,0AH

div CX

or DX,30H

mov [DI],DL

dec DI

xor DX,DX

pop CX

ret

PRINT\_NUM ENDP

TBYTE\_TO\_DEC PROC near

push AX

push CX

push DX

mov CX,2710H

cmp DX,0H

jnz LOOP\_TBD

cmp AX,2710H

jb ENDLPTBD

LOOP\_TBD: div CX

push AX

mov AX,DX

xor DX,DX

call PRINT\_NUM

call PRINT\_NUM

call PRINT\_NUM

call PRINT\_NUM

pop AX

cmp AX,2710H

jnb LOOP\_TBD

ENDLPTBD: cmp AX,0H

JZ EXIT\_TBD

call PRINT\_NUM

jmp ENDLPTBD

EXIT\_TBD: pop DX

pop CX

pop AX

ret

TBYTE\_TO\_DEC ENDP

CLEAR\_BYTES PROC near

push DI

push CX

push DX

mov DI,OFFSET BYTE\_+8

mov CX,9

mov DL,' '

LOOP\_CLR: mov [DI],DL

dec DI

loop LOOP\_CLR

pop DX

pop CX

pop DI

ret

CLEAR\_BYTES ENDP

PRINT\_STRING PROC near

push AX

mov AH,09H

int 21H

pop AX

ret

PRINT\_STRING ENDP

NEWLINE PROC near

push AX

push DX

mov AH,02H

mov DL,0DH

int 21H

mov DL,0AH

int 21H

pop DX

pop AX

ret

NEWLINE ENDP

; AVAILABLE MEMORY

AV\_MEM PROC near

push AX

push BX

push DX

mov AH,4AH

mov BX,0FFFFH

int 21H

clc

mov DX,BX

mov AX,BX

shr DX,12

shl AX,4

mov DI,OFFSET BYTE\_+8

call TBYTE\_TO\_DEC

mov DX,OFFSET AVAILABLE\_MEM

call PRINT\_STRING

mov DX,OFFSET BYTE\_

call PRINT\_STRING

call CLEAR\_BYTES

mov DX,OFFSET LINE\_

call PRINT\_STRING

pop DX

pop BX

pop AX

ret

AV\_MEM ENDP

; EXTENDED MEMORY

EXT\_MEM PROC near

push DX

push AX

push BX

push DI

mov DX,OFFSET EXTENDEN\_MEM

call PRINT\_STRING

mov AL,30H

out 70H,AL

in AL,71H

mov BL,AL

mov AL,31H

out 70H,AL

in AL,71H

mov BH,AL

mov DX,BX

shr DX,6

mov AX,BX

shl AX,10

mov DI,OFFSET BYTE\_+8

call TBYTE\_TO\_DEC

mov DX,OFFSET BYTE\_

call PRINT\_STRING

call CLEAR\_BYTES

mov DX,OFFSET LINE\_

call PRINT\_STRING

pop DI

pop BX

pop AX

pop DX

ret

EXT\_MEM ENDP

PRINT\_TABLE PROC near

push AX

push BX

push CX

push DX

push ES

push DI

mov DX,OFFSET TABLE

call PRINT\_STRING

call NEWLINE

; GET "LIST OF LISTS"

mov AH,52H

int 21H

mov ES,ES:[BX-2]

PRINT\_LINE:

xor AX,AX

mov AL,ES:00H

mov DI,OFFSET TYPE\_+3

call BYTE\_TO\_HEX

mov [DI],AH

mov [DI-1],AL

mov DX,OFFSET TYPE\_

call PRINT\_STRING

mov AX,ES:01H

mov DI,OFFSET PSP\_+8

call WRD\_TO\_HEX

mov DX,OFFSET PSP\_

call PRINT\_STRING

mov AX,ES:03H

mov DI,OFFSET SIZE\_+9

mov DX,AX

shr DX,12

shl AX,4

call TBYTE\_TO\_DEC

mov DX,OFFSET SIZE\_

call PRINT\_STRING

mov DI,OFFSET DATA\_+1

mov AX,ES:08H

mov [DI],AX

mov AX,ES:0AH

mov [DI+2H],AX

mov AX,ES:0CH

mov [DI+4H],AX

mov AX,ES:0EH

mov [DI+6H],AX

mov DX,OFFSET DATA\_

call PRINT\_STRING

mov DX,OFFSET LINE\_

call PRINT\_STRING

mov AL,ES:00H

cmp AL,5AH

je EXIT

mov AX,ES

mov BX,ES:03H

add AX,BX

INC AX

mov ES,AX

jmp PRINT\_LINE

EXIT: pop DI

pop ES

pop DX

pop CX

pop BX

pop AX

ret

PRINT\_TABLE ENDP

; CODE

BEGIN:

call AV\_MEM

call EXT\_MEM

mov AH,4AH

mov BX,offset END\_

int 21H

call PRINT\_TABLE

xor AL,AL

mov AH,4CH

int 21H

END\_ db 0

TESTPC ENDS

END START

**Приложение В.**

Код программы lab\_3\_3.asm

.286

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: jmp BEGIN

; DATA

AVAILABLE\_MEM db 'Available memory size: ','$'

EXTENDEN\_MEM db 'Extended memory size: ','$'

BYTE\_ db ' bytes', '$'

STR\_HEXWRD db ' ','$'

TABLE db 'Type | PSP address | Size (bytes) | Data$'

TYPE\_ db ' |$'

PSP\_ db ' |$'

SIZE\_ db ' |$'

DATA\_ db ' $'

LINE\_ db 0DH,0AH,'$'

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

push AX

xor AH,AH

xor DX,DX

mov CX,10

LOOP\_BD: div CX

or DL,30H

mov [DI],DL

dec DI

xor DX,DX

cmp AX,10

jae LOOP\_BD

cmp AL,00H

je END\_L

or AL,30H

mov [DI],AL

END\_L: pop AX

pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

PRINT\_NUM PROC near

push CX

mov CX,0AH

div CX

or DX,30H

mov [DI],DL

dec DI

xor DX,DX

pop CX

ret

PRINT\_NUM ENDP

TBYTE\_TO\_DEC PROC near

push AX

push CX

push DX

mov CX,2710H

cmp DX,0H

jnz LOOP\_TBD

cmp AX,2710H

jb ENDLPTBD

LOOP\_TBD: div CX

push AX

mov AX,DX

xor DX,DX

call PRINT\_NUM

call PRINT\_NUM

call PRINT\_NUM

call PRINT\_NUM

pop AX

cmp AX,2710H

jnb LOOP\_TBD

ENDLPTBD: cmp AX,0H

JZ EXIT\_TBD

call PRINT\_NUM

jmp ENDLPTBD

EXIT\_TBD: pop DX

pop CX

pop AX

ret

TBYTE\_TO\_DEC ENDP

CLEAR\_BYTES PROC near

push DI

push CX

push DX

mov DI,OFFSET BYTE\_+8

mov CX,9

mov DL,' '

LOOP\_CLR: mov [DI],DL

dec DI

loop LOOP\_CLR

pop DX

pop CX

pop DI

ret

CLEAR\_BYTES ENDP

PRINT\_STRING PROC near

push AX

mov AH,09H

int 21H

pop AX

ret

PRINT\_STRING ENDP

NEWLINE PROC near

push AX

push DX

mov AH,02H

mov DL,0DH

int 21H

mov DL,0AH

int 21H

pop DX

pop AX

ret

NEWLINE ENDP

; AVAILABLE MEMORY

AV\_MEM PROC near

push AX

push BX

push DX

mov AH,4AH

mov BX,0FFFFH

int 21H

clc

mov DX,BX

mov AX,BX

shr DX,12

shl AX,4

mov DI,OFFSET BYTE\_+8

call TBYTE\_TO\_DEC

mov DX,OFFSET AVAILABLE\_MEM

call PRINT\_STRING

mov DX,OFFSET BYTE\_

call PRINT\_STRING

call CLEAR\_BYTES

mov DX,OFFSET LINE\_

call PRINT\_STRING

pop DX

pop BX

pop AX

ret

AV\_MEM ENDP

; EXTENDED MEMORY

EXT\_MEM PROC near

push DX

push AX

push BX

push DI

mov DX,OFFSET EXTENDEN\_MEM

call PRINT\_STRING

mov AL,30H

out 70H,AL

in AL,71H

mov BL,AL

mov AL,31H

out 70H,AL

in AL,71H

mov BH,AL

mov DX,BX

shr DX,6

mov AX,BX

shl AX,10

mov DI,OFFSET BYTE\_+8

call TBYTE\_TO\_DEC

mov DX,OFFSET BYTE\_

call PRINT\_STRING

call CLEAR\_BYTES

mov DX,OFFSET LINE\_

call PRINT\_STRING

pop DI

pop BX

pop AX

pop DX

ret

EXT\_MEM ENDP

PRINT\_TABLE PROC near

push AX

push BX

push CX

push DX

push ES

push DI

mov DX,OFFSET TABLE

call PRINT\_STRING

call NEWLINE

; GET "LIST OF LISTS"

mov AH,52H

int 21H

mov ES,ES:[BX-2]

PRINT\_LINE:

xor AX,AX

mov AL,ES:00H

mov DI,OFFSET TYPE\_+3

call BYTE\_TO\_HEX

mov [DI],AH

mov [DI-1],AL

mov DX,OFFSET TYPE\_

call PRINT\_STRING

mov AX,ES:01H

mov DI,OFFSET PSP\_+8

call WRD\_TO\_HEX

mov DX,OFFSET PSP\_

call PRINT\_STRING

mov AX,ES:03H

mov DI,OFFSET SIZE\_+9

mov DX,AX

shr DX,12

shl AX,4

call TBYTE\_TO\_DEC

mov DX,OFFSET SIZE\_

call PRINT\_STRING

mov DI,OFFSET DATA\_+1

mov AX,ES:08H

mov [DI],AX

mov AX,ES:0AH

mov [DI+2H],AX

mov AX,ES:0CH

mov [DI+4H],AX

mov AX,ES:0EH

mov [DI+6H],AX

mov DX,OFFSET DATA\_

call PRINT\_STRING

mov DX,OFFSET LINE\_

call PRINT\_STRING

mov AL,ES:00H

cmp AL,5AH

je EXIT

mov AX,ES

mov BX,ES:03H

add AX,BX

INC AX

mov ES,AX

jmp PRINT\_LINE

EXIT: pop DI

pop ES

pop DX

pop CX

pop BX

pop AX

ret

PRINT\_TABLE ENDP

; CODE

BEGIN:

call AV\_MEM

call EXT\_MEM

mov AH,4AH

mov BX,offset END\_

int 21H

mov AH,48H

mov BX,1000H

int 21H

call PRINT\_TABLE

xor AL,AL

mov AH,4CH

int 21H

END\_ db 0

TESTPC ENDS

END START

**Приложение Г.**

Код программы lab\_3\_4.asm

.286

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: jmp BEGIN

; DATA

AVAILABLE\_MEM db 'Available memory size: ','$'

EXTENDEN\_MEM db 'Extended memory size: ','$'

BYTE\_ db ' bytes', '$'

STR\_HEXWRD db ' ','$'

TABLE db 'Type | PSP address | Size (bytes) | Data$'

TYPE\_ db ' |$'

PSP\_ db ' |$'

SIZE\_ db ' |$'

DATA\_ db ' $'

LINE\_ db 0DH,0AH,'$'

ERROR\_ db 'Error in allocating memory','$'

END\_ db 0

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

push AX

xor AH,AH

xor DX,DX

mov CX,10

LOOP\_BD: div CX

or DL,30H

mov [DI],DL

dec DI

xor DX,DX

cmp AX,10

jae LOOP\_BD

cmp AL,00H

je END\_L

or AL,30H

mov [DI],AL

END\_L: pop AX

pop DX

pop CX

ret

BYTE\_TO\_DEC ENDP

PRINT\_NUM PROC near

push CX

mov CX,0AH

div CX

or DX,30H

mov [DI],DL

dec DI

xor DX,DX

pop CX

ret

PRINT\_NUM ENDP

TBYTE\_TO\_DEC PROC near

push AX

push CX

push DX

mov CX,2710H

cmp DX,0H

jnz LOOP\_TBD

cmp AX,2710H

jb ENDLPTBD

LOOP\_TBD: div CX

push AX

mov AX,DX

xor DX,DX

call PRINT\_NUM

call PRINT\_NUM

call PRINT\_NUM

call PRINT\_NUM

pop AX

cmp AX,2710H

jnb LOOP\_TBD

ENDLPTBD: cmp AX,0H

JZ EXIT\_TBD

call PRINT\_NUM

jmp ENDLPTBD

EXIT\_TBD: pop DX

pop CX

pop AX

ret

TBYTE\_TO\_DEC ENDP

CLEAR\_BYTES PROC near

push DI

push CX

push DX

mov DI,OFFSET BYTE\_+8

mov CX,9

mov DL,' '

LOOP\_CLR: mov [DI],DL

dec DI

loop LOOP\_CLR

pop DX

pop CX

pop DI

ret

CLEAR\_BYTES ENDP

PRINT\_STRING PROC near

push AX

mov AH,09H

int 21H

pop AX

ret

PRINT\_STRING ENDP

NEWLINE PROC near

push AX

push DX

mov AH,02H

mov DL,0DH

int 21H

mov DL,0AH

int 21H

pop DX

pop AX

ret

NEWLINE ENDP

; AVAILABLE MEMORY

AV\_MEM PROC near

push AX

push BX

push DX

mov AH,4AH

mov BX,0FFFFH

int 21H

clc

mov DX,BX

mov AX,BX

shr DX,12

shl AX,4

mov DI,OFFSET BYTE\_+8

call TBYTE\_TO\_DEC

mov DX,OFFSET AVAILABLE\_MEM

call PRINT\_STRING

mov DX,OFFSET BYTE\_

call PRINT\_STRING

call CLEAR\_BYTES

mov DX,OFFSET LINE\_

call PRINT\_STRING

pop DX

pop BX

pop AX

ret

AV\_MEM ENDP

; EXTENDED MEMORY

EXT\_MEM PROC near

push DX

push AX

push BX

push DI

mov DX,OFFSET EXTENDEN\_MEM

call PRINT\_STRING

mov AL,30H

out 70H,AL

in AL,71H

mov BL,AL

mov AL,31H

out 70H,AL

in AL,71H

mov BH,AL

mov DX,BX

shr DX,6

mov AX,BX

shl AX,10

mov DI,OFFSET BYTE\_+8

call TBYTE\_TO\_DEC

mov DX,OFFSET BYTE\_

call PRINT\_STRING

call CLEAR\_BYTES

mov DX,OFFSET LINE\_

call PRINT\_STRING

pop DI

pop BX

pop AX

pop DX

ret

EXT\_MEM ENDP

PRINT\_TABLE PROC near

push AX

push BX

push CX

push DX

push ES

push DI

mov DX,OFFSET TABLE

call PRINT\_STRING

call NEWLINE

; GET "LIST OF LISTS"

mov AH,52H

int 21H

mov ES,ES:[BX-2]

PRINT\_LINE:

xor AX,AX

mov AL,ES:00H

mov DI,OFFSET TYPE\_+3

call BYTE\_TO\_HEX

mov [DI],AH

mov [DI-1],AL

mov DX,OFFSET TYPE\_

call PRINT\_STRING

mov AX,ES:01H

mov DI,OFFSET PSP\_+8

call WRD\_TO\_HEX

mov DX,OFFSET PSP\_

call PRINT\_STRING

mov AX,ES:03H

mov DI,OFFSET SIZE\_+9

mov DX,AX

shr DX,12

shl AX,4

call TBYTE\_TO\_DEC

mov DX,OFFSET SIZE\_

call PRINT\_STRING

mov DI,OFFSET DATA\_+1

mov AX,ES:08H

mov [DI],AX

mov AX,ES:0AH

mov [DI+2H],AX

mov AX,ES:0CH

mov [DI+4H],AX

mov AX,ES:0EH

mov [DI+6H],AX

mov DX,OFFSET DATA\_

call PRINT\_STRING

mov DX,OFFSET LINE\_

call PRINT\_STRING

mov AL,ES:00H

cmp AL,5AH

je EXIT

mov AX,ES

mov BX,ES:03H

add AX,BX

INC AX

mov ES,AX

jmp PRINT\_LINE

EXIT: pop DI

pop ES

pop DX

pop CX

pop BX

pop AX

ret

PRINT\_TABLE ENDP

; CODE

BEGIN:

call AV\_MEM

call EXT\_MEM

mov AH,48H

mov BX,1000H

int 21H

jc ERR

jmp CONTINUE

ERR:

mov DX,offset ERROR\_

call PRINT\_STRING

call NEWLINE

CONTINUE:

mov AH,4AH

mov BX,offset END\_

int 21H

call PRINT\_TABLE

xor AL,AL

mov AH,4CH

int 21H

TESTPC ENDS

END START