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. 7381	Габов Е.С.
Преподаватель	Ефремов М.А

Санкт-Петербург 2019

Цель работы.

Исследование структур данных и работы функций управления памятью ядра операционной системы.

Описание функций и структур данных.

Название функции	Назначение
_AVAILABLEMEMORY	определяет размер доступной
	памяти
_EXTENDEDMEMORY	определяет размер расширенной
	памяти
_DATA	определяет цепочку блоков
	управления памятью
OUTPUT	выводит цепочку блоков управления
	памятью
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 ВҮТЕ_ТО_НЕХ
WRD_TO_HEX	переводит число АХ в строку в 16-
	ой c/c, 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
_TO_DEC	переводит два байта в 10-ую с/с
PRINT	вызывает функцию печати строки

Последовательность действий, выполняемых утилитой.

- 1) вывод количества доступной памяти;
- 2) вывод размера расширенной памяти;
- 3) вывод цепочки блоков управления памятью.

Результаты выполнения программ.

C:\>3_1.com				
Amount of available memory: 648912 b				
Extended memory size: 15360 kB				
MCB Adress	МСВ Туре	Owner	Size	Name
016F	4D	0008	16	
0171	4D	0000	64	
0176	4D	0040	256	
0187	4D	0192	144	
0191	5A	0192	6 4 8912	3_1

Рисунок 1 — Результат выполнения программы 3_1.com

	vailable me	_	2 Ъ	
Extended memory size: 15360 kB				
MCB Adress	MCB Type	Owner	Size	Name
016F	4D	0008	16	
0171	4D	0000	64	
0176	4D	0040	256	
0187	4D	0192	144	
0191	4D	0192	13280	3_2
04D0	5A	0000	635616	∨∎P⊈. _Γ â−

Рисунок 2 — Результат выполнения программы 3_2.com

C:\>3_3.com Amount of available memory: 648912 b					
	Extended memory size: 15360 kB				
MCB Adress	MCB Туре	Owner	Size	Name	
016F	4D	0008	16		
0171	4D	0000	64		
0176	4D	0040	256		
0187	4D	0192	144		
0191	4D	0192	13392	3_3	
04D7	4D	0192	65536	3_3	
14D8	5A	0000	569952	èl	

Рисунок 3 – Результат выполнения программы 3 3.com

C:\>3_4.com					
Amount of available memory: 648912 b					
Extended me	Extended memory size: 15360 kB				
Error!					
MCB Adress	МСВ Туре	Owner	Size	Name	
016F	4D	0008	16		
0171	4D	0000	64		
0176	4D	0040	256		
0187	4D	0192	144		
0191	4D	0192	13744	3_4	
04ED	5A	0000	635152	•Pìå× P⊽	
016F 0171 0176 0187 0191	4D 4D 4D 4D 4D	0008 0000 0040 0192 0192	16 64 256 144 13744	3_4	

Рисунок 4 – Результат выполнения программы 3 4.com

Выводы.

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

Ответы на контрольные вопросы.

1) Что означает «доступный объем памяти»?

Доступный объём памяти — такой объём памяти, который может быть использован для загрузки программ.

2) Где МСВ блок вашей программы в списке?

В программах 3 1.com, 3 2.com и 3 4.com МСВ блок имеет адрес 0191h, а в программе 3 3.com присутствует два МСВ блока, первый блок имеет адрес 0191h, второй — 04D7h. В каждой программе присутствует блок МСВ размером 1446, который имеет адрес 0187h. Данный блок служит для управления памятью для области среды программы.

3) Какой размер памяти занимает программа в каждом случае?

В первом случае программа занимает 648912 байт.

Во втором случае: 648912-635616-16=13280 байт.

В третьем случае: 648912-569952-65536-32=13392 байт.

В четвёртом случае: 648912-635152-16=13744 байт.

ПРИЛОЖЕНИЕ А

3_1.ASM

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;data

AVAILABLEMEMORY db ' Amount of available memory: b',0dh,0ah,\$'

EXTENDEDMEMORY db ' Extended memory size: kB',0dh,0ah,'\$'

HEAD db ' MCB Adress MCB Type Owner Size Name ', 0dh, 0ah, '\$'

DATA db ' ', 0dh, 0ah, '\$'

;procedures

TETR_TO_HEX PROC near

and al,0fh
cmp al,09
jbe NEXT
add al,07

udd

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 ca

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_T	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		d1,30h
	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	
;			
_TO_DEC		PROC	near
	push	cx	
	push	dx	
	push	ax	
	mov		cx,10
_loop_bd:			
	div		CX
	or	E 13. 11	d1,30h
	mov	[si],dl	
	dec	si	1 1
	xor		dx,dx
	cmp		ax,10
	jae		_loop_bd
	cmp		ax,00h
	jbe		_end_l

al,30h

or

```
mov
                               [si],al
_end_l:
                               ax
               pop
                               dx
               pop
               pop
                               cx
               ret
_TO_DEC
                      ENDP
;-----
PRINT PROC NEAR
               push
                       ah, 09h
               mov
               21h
         int
                       ax
         pop
         ret
PRINT ENDP
_AVAILABLEMEMORY PROC NEAR; Search for available memory
               push
                       ax
               push
                       bx
               push
                       dx
               push
                       si
                       ax, ax
               xor
                       ah, 04Ah
               mov
                       bx, 0FFFFh
               mov
               int
                       21h
                       ax, 10h
               mov
               mul
                       bx
                       si, offset AVAILABLEMEMORY
               mov
                       si, 23h
               add
                       _TO_DEC
               call
                       si
               pop
                       dx
               pop
               pop
                       bx
               pop
                       ax
               ret
_AVAILABLEMEMORY ENDP
;-----
_EXTENDEDMEMORY PROC near; Search for extended memory
               push
                       ax
               push
                       bx
               push
                       si
```

push

dx

```
al, 30h
                mov
                                  70h, al
                out
                                  al, 71h
                in
                                  bl, al
                mov
                                  al, 31h
                mov
                                  70h, al
                out
                                  al, 71h
                in
                         ah, al
                mov
                         al, bl
                mov
                         dx, dx
                sub
                         si, offset EXTENDEDMEMORY
                mov
                add
                         si, 28
                         _TO_DEC
                call
                                  dx
                pop
                pop
                                  \dot{si}
                pop
                                  bx
                pop
                                  ax
                ret
_EXTENDEDMEMORY ENDP
;-----
_DATA PROC near; Search for MCB
                         di, offset DATA; Address of MCB
                mov
                mov
                         ax, es
                add
                         di, 05h
                call
                         WRD_TO_HEX
                mov
                         di, offset DATA ; Type of MCB
                         di, 0Fh
                add
                xor
                         ah, ah
                         al, es:[00h]
                mov
                call
                         BYTE_TO_HEX
                         [di], al
                mov
                inc
                         di
                mov
                         [di], ah
                         di, offset DATA; Owner
                mov
                         ax, es:[01h]
                mov
                         di, 1Dh
                add
                call
                         WRD_TO_HEX
                         di, offset DATA; Size
                mov
                         ax, es:[03h]
                mov
                         bx, 10h
                mov
```

mul

bx

```
di, 2Eh
                add
                        si
                push
                mov
                        si, di
                        _TO_DEC
                call
                        si
                pop
                        di, offset DATA; Name
                mov
                        di, 35h
                add
                        bx, 0h
                mov
                print_:
                                mov dl, es:[bx + 8]
                                mov [di], dl
                                inc di
                                inc bx
                                cmp bx, 8h
                jne
                        print_
                mov
                        ax, es:[3h]
                mov
                        bl, es:[0h]
                ret
_DATA ENDP
;-----
OUTPUT PROC NEAR ; Search for a chain of memory management units
                        ah, 52h
                mov
                        21h
                int
                sub
                        bx, 2h
                        es, es:[bx]
                mov
                output_:
                        call
                                _DATA
                        mov
                                dx, offset DATA
                                PRINT
                        call
                        mov
                                cx, es
                        add
                                ax, cx
                        inc
                                ax
                                es, ax
                        mov
                                bl, 4Dh
                        cmp
                        je
                                        output_
                ret
OUTPUT ENDP
;-----
BEGIN:
                call
                        _AVAILABLEMEMORY
                                dx, offset AVAILABLEMEMORY
                mov
                        PRINT
                call
                        _EXTENDEDMEMORY
                call
                                dx, offset EXTENDEDMEMORY
                mov
```

call PRINT

lea dx, HEAD call PRINT call OUTPUT

xor al, al mov ah, 4ch int 21h

TESTPC ENDS

приложение Б

3_2.ASM

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;data

AVAILABLEMEMORY db ' Amount of available memory: b',0dh,0ah,'\$' EXTENDEDMEMORY db ' Extended memory size: kB',0dh,0ah,'\$'

HEAD db ' MCB Adress MCB Type Owner Size Name ', 0dh, 0ah, '\$'

DATA db ' ', 0dh, 0ah, '\$'

;procedurs

;-----

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 c

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

```
al,bh
                mov
                         BYTE_TO_HEX
                call
                         [di],ah
                mov
                         di
                dec
                mov
                         [di],al
                         bx
                pop
                ret
                        ENDP
WRD_TO_HEX
;-----
BYTE_TO_DEC
                        PROC
                                near
                push
                         cx
                         dx
                push
                xor
                         ah,ah
                         dx,dx
                xor
                         cx,10
                mov
loop_bd:div
                cx
                                 d1,30h
                or
                         [si],dl
                mov
                         si
                dec
                xor
                          dx,dx
                         ax,10
                cmp
                jae
                         loop_bd
                                 al,00h
                cmp
                                 end\_l
                je
                                 al,30h
                or
                mov
                         [si],al
end_l: pop
                dx
                                 cx
                pop
                ret
BYTE_TO_DEC
                        ENDP
_TO_DEC
                        PROC
                                near
                push
                         cx
                push
                         dx
                push
                         ax
                                 cx,10
                mov
_loop_bd:
                div
                                 cx
                                 d1,30h
                or
                         [si],dl
                mov
                dec
                         \dot{si}
                                 dx,dx
                xor
                                 ax,10
                cmp
                                 _loop_bd
                jae
                                 ax,00h
                cmp
```

jbe

_end_l

```
al,30h
               or
                               [si],al
               mov
_end_l:
                               ax
               pop
                               dx
               pop
                               cx
               pop
               ret
                      ENDP
_TO_DEC
;-----
PRINT PROC NEAR
               push
                       ax
                       ah, 09h
               mov
               21h
         int
         pop
                       ax
         ret
PRINT ENDP
;-----
_AVAILABLEMEMORY PROC NEAR; Search for available memory
               push
               push
                       bx
                       dx
               push
                       si
               push
               xor
                       ax, ax
                       ah, 04Ah
               mov
               mov
                       bx, 0FFFFh
                       21h
               int
                       ax, 10h
               mov
               mul
                       bx
               mov
                       si, offset AVAILABLEMEMORY
               add
                       si, 23h
               call
                       _TO_DEC
                       \dot{si}
               pop
               pop
                       dx
                       bx
               pop
               pop
               ret
_AVAILABLEMEMORY ENDP
;-----
_EXTENDEDMEMORY PROC near; Search for extended memory
               push
                       ax
               push
                       bx
               push
                       \dot{si}
               push
                       dx
```

```
al, 30h
                mov
                                  70h, al
                out
                                  al, 71h
                in
                                  bl, al
                mov
                                  al, 31h
                mov
                                  70h, al
                out
                                  al, 71h
                in
                         ah, al
                mov
                         al, bl
                mov
                         dx, dx
                sub
                         si, offset EXTENDEDMEMORY
                mov
                         si, 28
                add
                         _TO_DEC
                call
                pop
                                  dx
                pop
                                  si
                                  bx \\
                pop
                                  ax
                pop
                ret
_EXTENDEDMEMORY ENDP
;-----
_DATA PROC near ; Search for MCB
                mov
                         di, offset DATA; Address of MCB
                mov
                         ax, es
                add
                         di, 05h
                call
                         WRD_TO_HEX
                         di, offset DATA ; Type of MCB
                mov
                         di, 0Fh
                add
                         ah, ah
                xor
                         al, es:[00h]
                mov
                         BYTE_TO_HEX
                call
                         [di], al
                mov
                         di
                inc
                         [di], ah
                mov
                         di, offset DATA; Owner
                mov
                         ax, es:[01h]
                mov
                add
                         di, 1Dh
                         WRD_TO_HEX
                call
                         di, offset DATA; Size
                mov
                         ax, es:[03h]
                mov
```

bx, 10h

mov

```
mul
                         bx
                         di, 2Eh
                add
                push
                         si
                         si, di
                mov
                         _TO_DEC
                call
                         si
                pop
                         di, offset DATA; Name
                mov
                         di, 35h
                add
                         bx, 0h
                mov
                print_:
                                 mov dl, es:[bx + 8]
                                 mov [di], dl
                                 inc di
                                 inc bx
                                 cmp bx, 8h
                jne
                         print_
                mov
                         ax, es:[3h]
                         bl, es:[0h]
                mov
                ret
_DATA ENDP
;-----
OUTPUT PROC NEAR; Search for a chain of memory management units
                         ah, 52h
                mov
                         21h
                int
                sub
                         bx, 2h
                         es, es:[bx]
                mov
                output_:
                        call
                                 _DATA
                                 dx, offset DATA
                        mov
                                 PRINT
                        call
                        mov
                                 cx, es
                        add
                                 ax, cx
                        inc
                                 ax
                        mov
                                 es, ax
                                 bl, 4Dh
                        cmp
                                         output_
                        je
                ret
OUTPUT ENDP
BEGIN:
                         _AVAILABLEMEMORY
                call
                                 dx, offset AVAILABLEMEMORY
                mov
                         PRINT
                call
                         _EXTENDEDMEMORY
                call
```

mov dx, offset EXTENDEDMEMORY

call PRINT

mov ah, 4ah ; Freeing of memory

mov bx, offset END_PROG

int 21h

mov dx, offset HEAD

call PRINT call OUTPUT

xor al, al mov ah, 4ch int 21h

 $\mathsf{END}_\mathsf{PROG}\,\mathsf{db}\,0$

TESTPC ENDS

приложение в

3_3.ASM

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;data

AVAILABLEMEMORY db ' Amount of available memory: b',0dh,0ah,'\$'

EXTENDEDMEMORY db ' Extended memory size: kB',0dh,0ah,'\$'

HEAD db ' MCB Adress MCB Type Owner Size Name ', 0dh, 0ah, '\$'

DATA db ' ', 0dh, 0ah, '\$'

;procedurs

;-----

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

```
al,bh
                mov
                         BYTE_TO_HEX
                call
                         [di],ah
                mov
                         di
                dec
                mov
                         [di],al
                         bx
                pop
                ret
                        ENDP
WRD_TO_HEX
;-----
BYTE_TO_DEC
                        PROC
                                near
                push
                         cx
                         dx
                push
                xor
                         ah,ah
                         dx,dx
                xor
                         cx,10
                mov
loop_bd:div
                cx
                                 d1,30h
                or
                         [si],dl
                mov
                         si
                dec
                xor
                          dx,dx
                         ax,10
                cmp
                jae
                         loop_bd
                                 al,00h
                cmp
                                 end\_l
                je
                                 al,30h
                or
                mov
                         [si],al
end_l: pop
                dx
                                 cx
                pop
                ret
BYTE_TO_DEC
                        ENDP
_TO_DEC
                        PROC
                                near
                push
                         cx
                push
                         dx
                push
                         ax
                                 cx,10
                mov
_loop_bd:
                div
                                 cx
                                 d1,30h
                or
                         [si],dl
                mov
                dec
                         \dot{si}
                                 dx,dx
                xor
                                 ax,10
                cmp
                                 _loop_bd
                jae
                                 ax,00h
                cmp
                jbe
                                 _end_l
```

```
al,30h
               or
                               [si],al
               mov
_end_l:
                               ax
               pop
                               dx
               pop
                               cx
               pop
               ret
                      ENDP
_TO_DEC
;-----
PRINT PROC NEAR
               push
                       ax
                       ah, 09h
               mov
               21h
         int
         pop
                       ax
         ret
PRINT ENDP
;-----
_AVAILABLEMEMORY PROC NEAR; Search for available memory
               push
               push
                       bx
                       dx
               push
                       si
               push
               xor
                       ax, ax
                       ah, 04Ah
               mov
               mov
                       bx, 0FFFFh
                       21h
               int
                       ax, 10h
               mov
               mul
                       bx
               mov
                       si, offset AVAILABLEMEMORY
               add
                       si, 23h
               call
                       _TO_DEC
                       \dot{si}
               pop
               pop
                       dx
                       bx
               pop
               pop
               ret
_AVAILABLEMEMORY ENDP
;-----
_EXTENDEDMEMORY PROC near; Search for extended memory
               push
                       ax
               push
                       bx
               push
                       \dot{si}
               push
                       dx
```

```
al, 30h
                mov
                                  70h, al
                out
                                  al, 71h
                in
                                  bl, al
                mov
                                  al, 31h
                mov
                                  70h, al
                out
                                  al, 71h
                in
                         ah, al
                mov
                         al, bl
                mov
                         dx, dx
                sub
                         si, offset EXTENDEDMEMORY
                mov
                         si, 28
                add
                         _TO_DEC
                call
                pop
                                  dx
                pop
                                  si
                                  bx \\
                pop
                                  ax
                pop
                ret
_EXTENDEDMEMORY ENDP
;-----
_DATA PROC near ; Search for MCB
                mov
                         di, offset DATA; Address of MCB
                mov
                         ax, es
                add
                         di, 05h
                         WRD_TO_HEX
                call
                         di, offset DATA ; Type of MCB
                mov
                         di, 0Fh
                add
                         ah, ah
                xor
                         al, es:[00h]
                mov
                         BYTE_TO_HEX
                call
                         [di], al
                mov
                         di
                inc
                         [di], ah
                mov
                         di, offset DATA; Owner
                mov
                         ax, es:[01h]
                mov
                add
                         di, 1Dh
                         WRD_TO_HEX
                call
                         di, offset DATA; Size
                mov
                         ax, es:[03h]
                mov
```

bx, 10h

mov

```
mul
                         bx
                         di, 2Eh
                add
                push
                         si
                         si, di
                mov
                         _TO_DEC
                call
                         si
                pop
                         di, offset DATA; Name
                mov
                         di, 35h
                add
                         bx, 0h
                mov
                print_:
                                 mov dl, es:[bx + 8]
                                 mov [di], dl
                                 inc di
                                 inc bx
                                 cmp bx, 8h
                jne
                         print_
                mov
                         ax, es:[3h]
                         bl, es:[0h]
                mov
                ret
_DATA ENDP
;-----
OUTPUT PROC NEAR; Search for a chain of memory management units
                         ah, 52h
                mov
                         21h
                int
                sub
                         bx, 2h
                         es, es:[bx]
                mov
                output_:
                        call
                                 _DATA
                                 dx, offset DATA
                        mov
                                 PRINT
                        call
                        mov
                                 cx, es
                        add
                                 ax, cx
                        inc
                                 ax
                        mov
                                 es, ax
                                 bl, 4Dh
                        cmp
                                         output_
                        je
                ret
OUTPUT ENDP
BEGIN:
                        _AVAILABLEMEMORY
                call
                                 dx, offset AVAILABLEMEMORY
                mov
                        PRINT
                call
                        _EXTENDEDMEMORY
                call
```

mov dx, offset EXTENDEDMEMORY

call PRINT

mov ah, 4ah ; Freeing of memory

mov bx, offset END_PROG

int 21h

mov ah, 48h ; Request 64 KB of memory

mov bx, 1000h

int 21h

mov dx, offset HEAD

call PRINT call OUTPUT

xor al, al mov ah, 4ch int 21h

END_PROG db 0

TESTPC ENDS

приложение г

3_4.ASM

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

;data

AVAILABLEMEMORY db ' Amount of available memory: b',0dh,0ah,'\$'

EXTENDEDMEMORY db ' Extended memory size: kB',0dh,0ah,'\$'

HEAD db ' MCB Adress MCB Type Owner Size Name ', 0dh, 0ah, '\$'

DATA db ' ', 0dh, 0ah, '\$'

ERRORM db ' Error!', 0dh, 0ah, '\$'

;procedurs

;-----

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
                        al,bh
               mov
                        BYTE_TO_HEX
               call
                        [di],ah
               mov
               dec
                        di
                        [di],al
               mov
                        bx \\
               pop
               ret
WRD_TO_HEX
                       ENDP
;-----
BYTE_TO_DEC
                       PROC
                               near
               push
                        cx
               push
                        dx
                        ah,ah
               xor
                        dx,dx
               xor
                        cx,10
               mov
loop_bd:div
                cx
                                d1,30h
               or
                        [si],dl
               mov
               dec
                        si
                          dx,dx
               xor
                        ax,10
               cmp
                        loop_bd
               jae
                                al,00h
               cmp
                                end_l
               je
                                al,30h
               or
                        [si],al
               mov
end_l: pop
                dx
               pop
                                cx
               ret
BYTE\_TO\_DEC
                       ENDP
;-----
_TO_DEC
                       PROC
                               near
                        cx
               push
               push
                        dx
               push
               mov
                                cx,10
_loop_bd:
               div
                                cx
                                d1,30h
               or
                        [si],dl
               mov
               dec
                        si
                                dx,dx
               xor
                                ax,10
               cmp
                                _loop_bd
               jae
                                ax,00h
               cmp
```

```
_end_l
               jbe
                               al,30h
               or
               mov
                               [si],al
_end_l:
               pop
                               ax
                               dx
               pop
                               cx
               pop
               ret
_TO_DEC
                       ENDP
PRINT PROC NEAR
               push
                        ax
               mov
                        ah, 09h
         int
                21h
         pop
                        ax
         ret
PRINT ENDP
_AVAILABLEMEMORY PROC NEAR; Search for available memory
               push
                        ax
                        bx
               push
                        dx
               push
               push
                        si
               xor
                        ax, ax
               mov
                        ah, 04Ah
                        bx, 0FFFFh
               mov
                        21h
               int
               mov
                        ax, 10h
                        bx
               mul
                        si, offset AVAILABLEMEMORY
               mov
               add
                        si, 23h
                        _TO_DEC
               call
               pop
                        si
                        dx
               pop
               pop
                        bx
               pop
                        ax
               ret
_AVAILABLEMEMORY ENDP
;-----
_EXTENDEDMEMORY PROC near; Search for extended memory
               push
                        ax
               push
                        bx
```

push

si

```
push
                         dx
                                 al, 30h
                mov
                                 70h, al
                out
                                 al, 71h
                in
                                 bl, al
                mov
                                 al, 31h
                mov
                                 70h, al
                out
                                 al, 71h
                in
                         ah, al
                mov
                         al, bl
                mov
                         dx, dx
                sub
                         si, offset EXTENDEDMEMORY
                mov
                         si, 28
                add
                         _TO_DEC
                call
                pop
                                 dx
                                 si
                pop
                                 bx
                pop
                pop
                                 ax
                ret
_EXTENDEDMEMORY ENDP
;-----
_DATA PROC near ; Search for MCB
                mov
                         di, offset DATA; Address of MCB
                         ax, es
                mov
                add
                         di, 05h
                call
                         WRD_TO_HEX
                         di, offset DATA ; Type of MCB
                mov
                         di, 0Fh
                add
                         ah, ah
                xor
                         al, es:[00h]
                mov
                         BYTE_TO_HEX
                call
                         [di], al
                mov
                         di
                inc
                mov
                         [di], ah
                         di, offset DATA; Owner
                mov
                         ax, es:[01h]
                mov
                         di, 1Dh
                add
                         WRD_TO_HEX
                call
                         di, offset DATA; Size
                mov
                mov
                         ax, es:[03h]
```

```
bx, 10h
                 mov
                 mul
                          bx
                 add
                          di, 2Eh
                 push
                          \dot{si}
                          si, di
                 mov
                          _TO_DEC
                 call
                          \dot{si}
                 pop
                          di, offset DATA ; Name
                 mov
                          di, 35h
                 add
                          bx, 0h
                 mov
                 print_:
                                   mov dl, es:[bx + 8]
                                   mov [di], dl
                                   inc di
                                   inc bx
                                   cmp bx, 8h
                 jne
                          print_
                          ax, es:[3h]
                 mov
                          bl, es:[0h]
                 mov
                 ret
_DATA ENDP
OUTPUT PROC NEAR ; Search for a chain of memory management units
                          ah, 52h
                 mov
                 int
                          21h
                          bx, 2h
                 sub
                          es, es:[bx]
                 mov
                 output_:
                          call
                                   _DATA
                                   dx, offset DATA
                          mov
                                   PRINT
                          call
                          mov
                                   cx, es
                          add
                                   ax, cx
                          inc
                                   ax
                          mov
                                   es, ax
                                   bl, 4Dh
                          cmp
                          je
                                            output\_
                 ret
OUTPUT ENDP
BEGIN:
                          _AVAILABLEMEMORY
                 call
                                   dx, offset AVAILABLEMEMORY
                 mov
                          PRINT
                 call
```

call _EXTENDEDMEMORY

mov dx, offset EXTENDEDMEMORY

call PRINT

mov ah, 48h; Request 64 KB of memory

mov bx, 1000h int 21h

jc memoryErr; Error check

jmp next_

memoryErr:

mov dx, offset ErrorM

call PRINT

next_:; Freeing of memory

mov ah, 4ah

mov bx, offset PROGRAMM_ENDS

int 21h

mov dx, offset HEAD

call PRINT

call OUTPUT

xor al, al mov ah, 4ch int 21h

 ${\tt PROGRAMM_ENDS~db~0}$

TESTPC ENDS