

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. 8383

Ларин А.

Преподаватель

Ефремов М.А,

Санкт-Петербург

2020

Цель работы.

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

Выполнение

Написан код .COM модуля, который при помощи прерывания 4Ah достает информацию о количестве доступной памяти, далее достает размер расширенной памяти из CMOS, информацию о блоках управления памятью MSB, и выводит эту информацию на экран. Результат работы представлен в приложении А, L3_1.

Далее программа модифицируется, и теперь освобождает всю неиспользуемую ей память. Результат работы представлен в приложении А, L3_2.

Далее программа сразу после освобождения памяти запрашивает дополнительно 64Кб. Результат работы представлен в приложении А, L3_3.

Далее изначальная программа модифицируется таким образом, чтобы запрашивать память перед ее очищением. Результат работы представлен в приложении А, L3_4.

Код программ представлен в приложении Б

Контрольные вопросы

1. Что означает «доступный объем памяти»?

Память, которая может быть отведена исполняемой программе.

2. Где MSB блок вашей программы в списке?

Под номером 5 для первых двух программ, 5,6, для третьей, и только 5 для четвертой, т. к. память была освобождена

3. Какой размер памяти занимает программа в каждом случае?

4. Программа 1 занимает всю доступную память — 59088 байт.

5. Программа 2 освобождает неиспользуемую память и занимает 1296 байт.

6. Программа 3 занимает 1360 байт. + 64Кб(57696 б), выделенных для программы

7. Программа 4 занимает 1360 байт.

Выводы.

В результате работы были разобраны некоторые концепции языка ассемблера и работы операционной системы DOS. Были исследованы структуры данных и функции управления памятью.

ПРИЛОЖЕНИЕ А

L3_1

E6D0

3C00

0001

MS DOS

0010

0002

free

0040

DPMILOAD

0003

0040

0100

0004

0192

0090

0005

0192

E6D0

L3_1

L3_2

E6D0

Mem freed

3C00

0001

MS DOS

0010

0002
free

0040

DPMILOAD

0003
0040
0100

0004
0192
0090

0005
0192
0510

L3_2

0006
free

E1B0

s kernel

L3_3

E6D0
Mem freed
Mem allocated

3C00

0001
MS DOS

0010

0002
free

0040

DPMILOAD

0003
0040
0100

0004
0192
0090

0005
0192
0550

L3_3

0006
0192
0000

L3_3

0007
free

E160

©□#

L3_4

E6D0
Mem not allocated
Mem freed

3C00

0001
MS DOS

0010

0002
free

0040

DPMILOAD

0003
0040
0100

0004
0192
0090

0005
0192
0550

L3_4

0006
free

E170

L3_3

ПРИЛОЖЕНИЕ Б

L3_1.ASM

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
```

```
START:  JMP BEGIN
;data
```

```
STR_01 db " free$";
STR_02 db " OS XMS UMB$";
STR_03 db " driver's top memory$";
STR_04 db " MS DOS$";
STR_05 db " control block 386MAX UMB$";
STR_06 db " blocked 386MAX$";
STR_07 db " 386MAX UMB$";
```

```
NEW_LINE db 0DH,0AH,'$'
```

```
STRING DB 'Some text          ',0DH,0AH,'$'
;procedures
```

```
DIGIT_TO_CHAR PROC near
;AL
```

```
    and al,0Fh
    cmp al,09h
    jle BLW
    add al,'A'
    sub al, 0Ah
    jmp DTC_CONT
```

```
BLW:
```

```
    add al,'0'
```

```
DTC_CONT:
```

```
    ret
```

```
DIGIT_TO_CHAR ENDP
```

```
;-----
```

```
PRINT_AS_HEX proc near
```

```
;AL - number
```

```
; ;breaks AX,CX,BX
```

```
    push ax
```

```
    push bx
```

```
    push cx
```



```

push dx
;mov bx,dx
mov ch,al
mov cl,4
shr al,cl
call DIGIT_TO_CHAR
mov dl,al
mov ah,02h
int 21h
mov al,ch
call DIGIT_TO_CHAR
mov dl,al
mov ah,02h
int 21h
;mov dx,bx
pop dx
pop cx
pop bx
pop ax

```

```

ret
PRINT_AS_HEX ENDP

```

```

PRINT_WORD proc near
;AX - word
xchg AL,AH
call PRINT_AS_HEX
xchg AL,AH
call PRINT_AS_HEX
ret

```

```

PRINT_WORD ENDP

```

```

LN PROC
push AX
push DX
mov DX, offset NEW_LINE
mov AH, 9h
int 21h
pop DX
pop AX
ret
LN ENDP

```

```

;-----
BEGIN:

```

```

;Avalible mem
    mov AH, 4Ah
    mov BX, 0FFFFh
    int 21h;
    shl BX, 4
    mov AX, BX
    call PRINT_WORD
;Extended mem
    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 AX, BX
    call LN
    call PRINT_WORD
    call LN
    call LN
;MCB's
    mov AH, 52h
    int 21h
    mov AX, ES:[BX-2]
    mov ES, AX
    xor CX, CX
NEXT:
    inc CX
    mov AX, CX
    call PRINT_WORD
    call LN
    push CX

    xor AX, AX
    mov AL, ES:[0h]
    push AX
    mov AX, ES:[1h]

    cmp AX, 0h
    je AREA_FREE
    cmp AX, 6h
    je AREA_DRIVER
    cmp AX, 7h
    je AREA_TOP
    cmp AX, 8h
    je AREA_DOS
    cmp AX, 0FFFAh

```

```

je AREA_BLOCK
cmp AX, 0FFFDh
je AREA_BLOCKED
cmp AX, 0FFFEh
je AREA_LAST
xor DX, DX
call PRINT_WORD
call LN
jmp AFTER_SWITCH

```

```

AREA_FREE:
    mov DX, offset STR_01
    jmp END_OF_SWITCH
AREA_DRIVER:
    mov DX, offset STR_02
    jmp END_OF_SWITCH
AREA_TOP:
    mov DX, offset STR_03
    jmp END_OF_SWITCH
AREA_DOS:
    mov DX, offset STR_04
    jmp END_OF_SWITCH
AREA_BLOCK:
    mov DX, offset STR_05
    jmp END_OF_SWITCH
AREA_BLOCKED:
    mov DX, offset STR_06
    jmp END_OF_SWITCH
AREA_LAST:
    mov DX, offset STR_07
END_OF_SWITCH:
    push AX
    mov AH, 9h
    int 21h
    pop AX
    call LN
    call LN

```

```

AFTER_SWITCH:
;size
    mov AX, ES:[3h]
    mov BX, 10h
    mul BX
    call PRINT_WORD
    call LN

```

```

mov CX, 8

```

```

    xor SI, SI
    call LN
PRINT_LAST_BYTES:
    mov DL, ES:[SI + 8h]
    mov AH, 02h
    int 21h
    inc SI
    loop PRINT_LAST_BYTES
    call LN

```

```

    mov AX, ES:[3h]
    mov BX, ES
    add BX, AX
    inc BX
    mov ES, BX
    pop AX
    pop CX
    cmp AL, 5Ah
    je END_PROC
    call LN
    jmp NEXT
END_PROC:

```

```

EXIT:
    xor AL, AL
    mov AH, 4Ch
    int 21h
TESTPC ENDS
END START

```

L3_2.ASM

```

TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H

```

```

START:  JMP BEGIN
;data

```

```

STR_01 db " free$";
STR_02 db " OS XMS UMB$";
STR_03 db " driver's top memory$";
STR_04 db " MS DOS$";
STR_05 db " control block 386MAX UMB$";
STR_06 db " blocked 386MAX$";
STR_07 db " 386MAX UMB$";

```

```
FREE_SUCCESS db "Mem freed$";
FREE_FAIL db "Mem not freed$";
```

```
NEW_LINE db 0DH,0AH,'$'
```

```
STRING DB 'Some text          ',0DH,0AH,'$'
;procedures
```

```
PRINT PROC
    push AX
    mov AH, 9h
    int 21h
    pop AX
    ret
```

```
PRINT ENDP
```

```
DIGIT_TO_CHAR PROC near
;AL
```

```
    and al,0Fh
    cmp al,09h
    jle BLW
    add al,'A'
    sub al, 0Ah
    jmp DTC_CONT
```

```
BLW:
```

```
    add al,'0'
```

```
DTC_CONT:
```

```
    ret
```

```
DIGIT_TO_CHAR ENDP
```

```
;-----
```

```
PRINT_AS_HEX proc near
```

```
;AL - number
```

```
;;breaks AX,CX,BX
```

```
    push ax
```

```
    push bx
```

```
    push cx
```

```
    push dx
```

```
    ;mov bx,dx
```

```
    mov ch,al
```

```
    mov cl,4
```

```
    shr al,cl
```

```
    call DIGIT_TO_CHAR
```

```
    mov dl,al
```

```
    mov ah,02h
```

```
    int 21h
```

```

    mov al,ch
    call DIGIT_TO_CHAR
    mov dl,al
    mov ah,02h
    int 21h
    ;mov dx,bx
    pop dx
    pop cx
    pop bx
    pop ax

```

```

    ret
PRINT_AS_HEX ENDP

```

```

PRINT_WORD proc near
;AX - word
    xchg AL,AH
    call PRINT_AS_HEX
    xchg AL,AH
    call PRINT_AS_HEX
    ret

```

```

PRINT_WORD ENDP

```

```

LN PROC
    push AX
    push DX
    mov DX, offset NEW_LINE
    mov AH, 9h
    int 21h
    pop DX
    pop AX
    ret
LN ENDP

```

```

;-----
BEGIN:

```

```

;Avalible mem
    mov AH, 4Ah
    mov BX, 0FFFFh
    int 21h;
    shl BX, 4
    mov AX, BX
    call PRINT_WORD

```

```

;Free mem
xor AX,AX
mov BX, offset STACK_BUF_END
add BX, 10Fh
shr BX, 4
;sub BX,500
mov AH, 4Ah
int 21h
jnc SUCCESS
mov DX, offset FREE_FAIL
call PRINT
call LN
jmp FREE_END

```

```

SUCCESS:
mov DX, offset FREE_SUCCESS
call PRINT
call LN
FREE_END:
xor AX,AX

```

```

;Extended mem
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 AX, BX
call LN
call PRINT_WORD
call LN
call LN

```

```

;MCB's
mov AH, 52h
int 21h
mov AX, ES:[BX-2]
mov ES, AX
xor CX, CX

```

```

NEXT:
inc CX
mov AX,CX
call PRINT_WORD
call LN

```

```

push CX

xor AX, AX
mov AL, ES:[0h]
push AX
mov AX, ES:[1h]

cmp AX, 0h
je AREA_FREE
cmp AX, 6h
je AREA_DRIVER
cmp AX, 7h
je AREA_TOP
cmp AX, 8h
je AREA_DOS
cmp AX, 0FFFAh
je AREA_BLOCK
cmp AX, 0FFFDh
je AREA_BLOCKED
cmp AX, 0FFFEh
je AREA_LAST
xor DX, DX
call PRINT_WORD
call LN
jmp AFTER_SWITCH

```

```

AREA_FREE:
    mov DX, offset STR_01
    jmp END_OF_SWITCH
AREA_DRIVER:
    mov DX, offset STR_02
    jmp END_OF_SWITCH
AREA_TOP:
    mov DX, offset STR_03
    jmp END_OF_SWITCH
AREA_DOS:
    mov DX, offset STR_04
    jmp END_OF_SWITCH
AREA_BLOCK:
    mov DX, offset STR_05
    jmp END_OF_SWITCH
AREA_BLOCKED:
    mov DX, offset STR_06
    jmp END_OF_SWITCH
AREA_LAST:
    mov DX, offset STR_07
END_OF_SWITCH:

```



```

    call PRINT
    call LN
    call LN

AFTER_SWITCH:
;size
    mov AX, ES:[3h]
    mov BX, 10h
    mul BX
    call PRINT_WORD
    call LN

    mov CX, 8
    xor SI, SI
    call LN
PRINT_LAST_BYTES:
    mov DL, ES:[SI + 8h]
    mov AH, 02h
    int 21h
    inc SI
    loop PRINT_LAST_BYTES
    call LN

    mov AX, ES:[3h]
    mov BX, ES
    add BX, AX
    inc BX
    mov ES, BX
    pop AX
    pop CX
    cmp AL, 5Ah
    je END_PROC
    call LN
    jmp NEXT
END_PROC:

EXIT:
    xor AL, AL
    mov AH, 4Ch
    int 21h

STACK_BUF:
    DW 128 dup(0)
STACK_BUF_END:

TESTPC ENDS
END START

```

L3_3.ASM

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H

START:  JMP BEGIN
;data

STR_01 db " free$";
STR_02 db " OS XMS UMB$";
STR_03 db " driver's top memory$";
STR_04 db " MS DOS$";
STR_05 db " control block 386MAX UMB$";
STR_06 db " blocked 386MAX$";
STR_07 db " 386MAX UMB$";

FREE_SUCCESS db "Mem freed$";
FREE_FAIL db "Mem not freed$";

ALLOC_SUCCESS db "Mem allocated$";
ALLOC_FAIL db "Mem not allocated$";

NEW_LINE db 0DH,0AH,'$'

STRING DB 'Some text          ',0DH,0AH,'$'
;procedures

PRINT PROC
    push AX
    mov AH, 9h
    int 21h
    pop AX
    ret
PRINT ENDP

DIGIT_TO_CHAR PROC near
;AL
    and al,0Fh
    cmp al,09h
    jle BLW
    add al,'A'
    sub al, 0Ah
    jmp DTC_CONT
```

```

BLW:
    add al, '0'
DTC_CONT:
    ret
DIGIT_TO_CHAR ENDP
;-----
PRINT_AS_HEX proc near
;AL - number
;;breaks AX,CX,BX
    push ax
    push bx
    push cx
    push dx
    ;mov bx,dx
    mov ch,al
    mov cl,4
    shr al,cl
    call DIGIT_TO_CHAR
    mov dl,al
    mov ah,02h
    int 21h
    mov al,ch
    call DIGIT_TO_CHAR
    mov dl,al
    mov ah,02h
    int 21h
    ;mov dx,bx
    pop dx
    pop cx
    pop bx
    pop ax

    ret
PRINT_AS_HEX ENDP

PRINT_WORD proc near
;AX - word
    xchg AL,AH
    call PRINT_AS_HEX
    xchg AL,AH
    call PRINT_AS_HEX
    ret

PRINT_WORD ENDP

LN PROC

```

```

    push AX
    push DX
    mov DX, offset NEW_LINE
    mov AH, 9h
    int 21h
    pop DX
    pop AX
    ret
LN ENDP

;-----
BEGIN:

;Avalible mem
    mov AH, 4Ah
    mov BX, 0FFFFh
    int 21h;
    shl BX, 4
    mov AX, BX
    call PRINT_WORD
    call LN

;Free mem
    xor AX,AX
    mov BX, offset STACK_BUF_END
    add BX, 10Fh
    shr BX, 4
    ;sub BX,500
    mov AH, 4Ah
    int 21h
    jnc SUCCESS
    mov DX, offset FREE_FAIL
    call PRINT
    call LN
    jmp FREE_END

SUCCESS:
    mov DX, offset FREE_SUCCESS
    call PRINT
    call LN
FREE_END:
    xor AX,AX
;Alloc mem
    mov BX, 1000h
    mov AH, 48h
    int 21h
    jnc ALLOCATE_SUCCESS
    mov DX, offset ALLOC_FAIL

```

```

    call PRINT
    call LN
    jmp ALLOCATE_DONE

ALLOCATE_SUCCESS:
    mov DX, offset ALLOC_SUCCESS
    call PRINT
    call LN

ALLOCATE_DONE:

;Extended mem
    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 AX, BX
    call LN
    call PRINT_WORD
    call LN
    call LN
;MCB's
    mov AH, 52h
    int 21h
    mov AX, ES:[BX-2]
    mov ES, AX
    xor CX, CX
NEXT:
    inc CX
    mov AX, CX
    call PRINT_WORD
    call LN
    push CX

    xor AX, AX
    mov AL, ES:[0h]
    push AX
    mov AX, ES:[1h]

    cmp AX, 0h
    je AREA_FREE
    cmp AX, 6h
    je AREA_DRIVER
    cmp AX, 7h

```

```

je AREA_TOP
cmp AX, 8h
je AREA_DOS
cmp AX, 0FFFAh
je AREA_BLOCK
cmp AX, 0FFFDh
je AREA_BLOCKED
cmp AX, 0FFFEh
je AREA_LAST
xor DX, DX
call PRINT_WORD
call LN
jmp AFTER_SWITCH

```

```

AREA_FREE:
    mov DX, offset STR_01
    jmp END_OF_SWITCH
AREA_DRIVER:
    mov DX, offset STR_02
    jmp END_OF_SWITCH
AREA_TOP:
    mov DX, offset STR_03
    jmp END_OF_SWITCH
AREA_DOS:
    mov DX, offset STR_04
    jmp END_OF_SWITCH
AREA_BLOCK:
    mov DX, offset STR_05
    jmp END_OF_SWITCH
AREA_BLOCKED:
    mov DX, offset STR_06
    jmp END_OF_SWITCH
AREA_LAST:
    mov DX, offset STR_07
END_OF_SWITCH:
    call PRINT
    call LN
    call LN

```

```

AFTER_SWITCH:
;size
    mov AX, ES:[3h]
    mov BX, 10h
    mul BX
    call PRINT_WORD
    call LN

```

```

    mov CX, 8
    xor SI, SI
    call LN
PRINT_LAST_BYTES:
    mov DL, ES:[SI + 8h]
    mov AH, 02h
    int 21h
    inc SI
    loop PRINT_LAST_BYTES
    call LN

```

```

    mov AX, ES:[3h]
    mov BX, ES
    add BX, AX
    inc BX
    mov ES, BX
    pop AX
    pop CX
    cmp AL, 5Ah
    je END_PROC
    call LN
    jmp NEXT
END_PROC:

```

```

EXIT:
    xor AL, AL
    mov AH, 4Ch
    int 21h

```

```

STACK_BUF:
    DW 128 dup(0)
STACK_BUF_END:

```

```

TESTPC ENDS
END START

```

L3_4.ASM

```

TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H

```

```

START:  JMP BEGIN
;data

```

```

STR_01 db " free$";
STR_02 db " OS XMS UMB$";
STR_03 db " driver's top memory$";

```

```
STR_04 db " MS DOS$";
STR_05 db " control block 386MAX UMB$";
STR_06 db " blocked 386MAX$";
STR_07 db " 386MAX UMB$";
```

```
FREE_SUCCESS db "Mem freed$";
FREE_FAIL db "Mem not freed$";
```

```
ALLOC_SUCCESS db "Mem allocated$";
ALLOC_FAIL db "Mem not allocated$";
```

```
NEW_LINE db 0DH,0AH,'$'
```

```
STRING DB 'Some text          ',0DH,0AH,'$'
;procedures
```

```
PRINT PROC
    push AX
    mov AH, 9h
    int 21h
    pop AX
    ret
```

```
PRINT ENDP
```

```
DIGIT_TO_CHAR PROC near
;AL
```

```
    and al,0Fh
    cmp al,09h
    jle BLW
    add al,'A'
    sub al, 0Ah
    jmp DTC_CONT
```

```
BLW:
```

```
    add al,'0'
```

```
DTC_CONT:
```

```
    ret
```

```
DIGIT_TO_CHAR ENDP
```

```
;-----
```

```
PRINT_AS_HEX proc near
```

```
;AL - number
```

```
; ;breaks AX,CX,BX
```

```
    push ax
    push bx
    push cx
    push dx
```



```

;mov bx,dx
mov ch,al
mov cl,4
shr al,cl
call DIGIT_TO_CHAR
mov dl,al
mov ah,02h
int 21h
mov al,ch
call DIGIT_TO_CHAR
mov dl,al
mov ah,02h
int 21h
;mov dx,bx
pop dx
pop cx
pop bx
pop ax

```

```

ret
PRINT_AS_HEX ENDP

```

```

PRINT_WORD proc near
;AX - word
xchg AL,AH
call PRINT_AS_HEX
xchg AL,AH
call PRINT_AS_HEX
ret

```

```

PRINT_WORD ENDP

```

```

LN PROC
push AX
push DX
mov DX, offset NEW_LINE
mov AH, 9h
int 21h
pop DX
pop AX
ret
LN ENDP

```

```

;-----
BEGIN:

```

```

;Avalible mem
    mov AH, 4Ah
    mov BX, 0FFFFh
    int 21h;
    shl BX, 4
    mov AX, BX
    call PRINT_WORD

;Alloc mem
    mov BX, 1000h
    mov AH, 48h
    int 21h
    jnc ALLOCATE_SUCCESS
    mov DX, offset ALLOC_FAIL
    call PRINT
    call LN
    jmp ALLOCATE_DONE

ALLOCATE_SUCCESS:
    mov DX, offset ALLOC_SUCCESS
    call PRINT
    call LN

ALLOCATE_DONE:

;Free mem
    xor AX,AX
    mov BX, offset STACK_BUF_END
    add BX, 10Fh
    shr BX, 4
    ;sub BX,500
    mov AH, 4Ah
    int 21h
    jnc SUCCESS
    mov DX, offset FREE_FAIL
    call PRINT
    call LN
    jmp FREE_END

SUCCESS:
    mov DX, offset FREE_SUCCESS
    call PRINT
    call LN
FREE_END:
    xor AX,AX

```

```

;Extended mem
    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 AX, BX
    call LN
    call PRINT_WORD
    call LN
    call LN
;MCB's
    mov AH, 52h
    int 21h
    mov AX, ES:[BX-2]
    mov ES, AX
    xor CX, CX
NEXT:
    inc CX
    mov AX, CX
    call PRINT_WORD
    call LN
    push CX

    xor AX, AX
    mov AL, ES:[0h]
    push AX
    mov AX, ES:[1h]

    cmp AX, 0h
    je AREA_FREE
    cmp AX, 6h
    je AREA_DRIVER
    cmp AX, 7h
    je AREA_TOP
    cmp AX, 8h
    je AREA_DOS
    cmp AX, 0FFFAh
    je AREA_BLOCK
    cmp AX, 0FFFDh
    je AREA_BLOCKED
    cmp AX, 0FFFEh
    je AREA_LAST
    xor DX, DX
    call PRINT_WORD

```

```

    call LN
    jmp AFTER_SWITCH

AREA_FREE:
    mov DX, offset STR_01
    jmp END_OF_SWITCH
AREA_DRIVER:
    mov DX, offset STR_02
    jmp END_OF_SWITCH
AREA_TOP:
    mov DX, offset STR_03
    jmp END_OF_SWITCH
AREA_DOS:
    mov DX, offset STR_04
    jmp END_OF_SWITCH
AREA_BLOCK:
    mov DX, offset STR_05
    jmp END_OF_SWITCH
AREA_BLOCKED:
    mov DX, offset STR_06
    jmp END_OF_SWITCH
AREA_LAST:
    mov DX, offset STR_07
END_OF_SWITCH:
    call PRINT
    call LN
    call LN

AFTER_SWITCH:
;size
    mov AX, ES:[3h]
    mov BX, 10h
    mul BX
    call PRINT_WORD
    call LN

    mov CX, 8
    xor SI, SI
    call LN
PRINT_LAST_BYTES:
    mov DL, ES:[SI + 8h]
    mov AH, 02h
    int 21h
    inc SI
    loop PRINT_LAST_BYTES
    call LN

```

```
    mov AX, ES:[3h]
    mov BX, ES
    add BX, AX
    inc BX
    mov ES, BX
    pop AX
    pop CX
    cmp AL, 5Ah
    je END_PROC
    call LN
    jmp NEXT
END_PROC:
```

```
EXIT:
    xor AL, AL
    mov AH, 4Ch
    int 21h
```

```
STACK_BUF:
    DW 128 dup(0)
STACK_BUF_END:
```

```
TESTPC ENDS
END START
```

