**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. 6381 |  | Поляков Н.С. |
| Преподаватель |  | Губкин А.Ф. |

Санкт-Петербург

2018

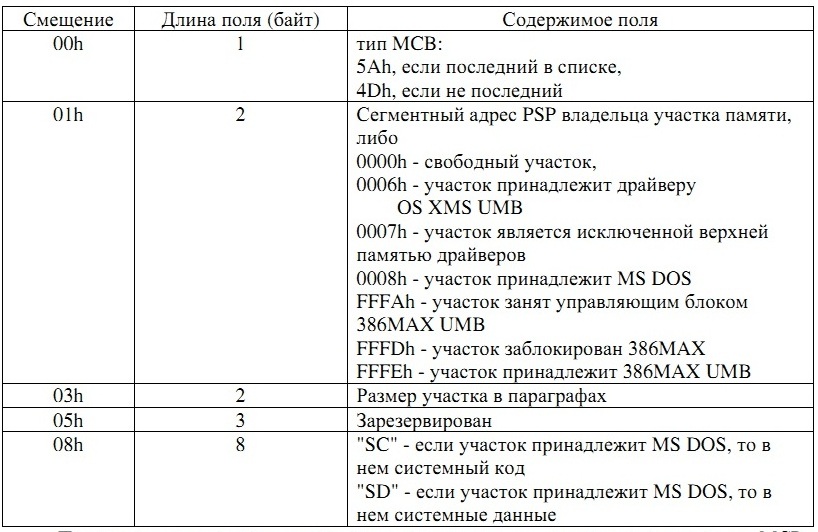
**Цель работы.**

Исследование структур данных и работы функций управления памятью.

**Необходимые сведения для составления программы.**

Учет занятой и свободной памяти ведётся при помощи списка блоков управления памятью МСВ (Memory Control Block). МСВ занимает 16 байт (параграф) и располагается всегда с адреса кратного 16 (адрес сегмента ОП) и находится в адресном пространстве непосредственно перед тем участком памяти, которым он управляет.

МСВ имеет следующую структуру:



По сегментному адресу и размеру участка памяти, контролируемого этим МСВ можно определить местоположение следующего МСВ в списке.

Адрес первого МСВ хранится во внутренней структуре MS DOS, называемой "List of Lists" (список списков). Доступ к указателю на эту структуру можно получить используя функцию 52h "Get List of Lists" int 21h. В результате выполнения этой функции ES:BX будет указывать на список списков. Слово по адресу ES:[BX-2] и есть адрес самого первого МСВ.

Размер расширенной памяти находится в ячейках 30h, 31h CMOS. CMOS это энергонезависимая память, в которой хранится информация о конфигурации ПЭВМ. Объем памяти составляет 64 байта. Размер расширенной памяти в Кбайтах можно определить обращаясь к ячейкам CMOS следующим образом:

mov al, 30h ;запись адреса ячейки CMOS

out 70h, al

in al, 71h ;чтение младшего байта

mov bl, al ;размера расширенной памяти

mov al, 31h ;запись адреса ячейки CMOS

out 70h, al

in al, 71h ;чтение старшего байта

;размера расширенной памяти

**Описание функций.**

|  |  |
| --- | --- |
| Имя | Описание |
| TETR\_TO\_HEX | Осуществляет перевод половины байта в AL в символ в 16-ричной с/c и помещает результат в AL |
| BYTE\_TO\_HEX | Осуществляет перевод байта в AL, в 16-тичную c/c и помещает результат по адресу DI |
| WRD\_TO\_HEX | Осуществляет перевод слова в AX, в 16-тичную c/c и помещает результат по адресу DI |
| BYTE\_TO\_DEC | Осуществляет перевод байта в AL, в 10-тичную c/c и помещает результат по адресу SI |
| WRD\_TO\_DEC | Осуществляет перевод слова в AX, в 10-тичную c/c и помещает результат по адресу SI |
| EX\_MEMORY | Определяет размер расширенной памяти и выводит их на экран |
| AVAILABLE\_MEM | Определяет количество доступной памяти и выводит их на экран |
| MCB | Выводит на экран цепочку блоков управления памятью |
| FREE\_MEM | Производит манипуляции с памятью |

**Описание структур данных.**

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| HEADER | db | Строка “# ADDR SIZE LAST8B ” |
| MCD\_BLOCK | db | Строка “0 0000 0000000 ” |
| AV\_MEM | db | Строка “Available memory: bytes” |
| EX\_MEM | db | Строка “Expanded memory: KB ” |
| ERROR | db | Строка “ITERRUPTION ERROER!” |

**Последовательность действий, выполняемых программой.**

* Вывод размера доступной памяти.
* Вывод размера расширенной памяти.
* Вывод цепочки блоков управления памятью.

1. **Программа lab3\_1.com**

Нет манипуляций с памятью.

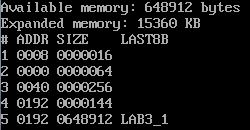
****

Рис.1 Результат работы программы lab3\_1.com.

1. **Программа lab3\_2.com**

Программа освобождает неиспользованную память и имеет меньший размер.

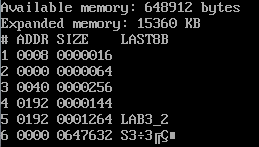
****

Рис.2 Результат работы программы lab3\_2.com.

1. **Программа lab3\_3.com**

Программа освобождает неиспользованную память и выделяет для программы дополнительные 64кб.

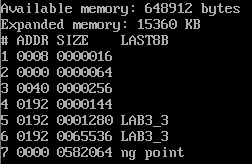
****

Рис.3 Результат работы программы lab3\_3.com.

1. **Программа lab3\_4.com**

Программа выделяет дополнительные 64кб и освобождает неиспользованную память. При попытке выделения дополнительных 64кб происходит ошибка и программа завершается.

../../../../../../../../../Users/nestor/Desktop/Screen%20Sho

Рис.4 Результат работы программы lab3\_4.com.

**Ответы на контрольные вопросы.**

1. Что означает «доступный объём памяти»?

Ответ: Максимально возможный размер программы.

1. Где MCB блок вашей программы в списке?

Ответ: В 1-2 пунктах под номером 5. В 3 пункте под номерами 5-6.

1. Какой размер памяти занимает программа в каждом случае?
2. Ответ: 1) 648 912 байта

2) 1 264 байта

3) 66 816 байт

4) 648 912 байт

**Вывод.**

В ходе выполнения лабораторной работы были изучены функции ядра, обеспечивающие работу с памятью, а также структура Memory Control Block и место нахождение данных блоков в памяти.

**Приложение 1. Код программы.**

**Lab3\_1.asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

HEADER db '# ADDR SIZE LAST8B ',13,10,36

MCB\_BLOCK db '0 0000 0000000 ',13,10,36

AV\_MEM db 'Available memory: bytes ',13,10,'$'

EX\_MEM db 'Expanded memory: KB ',13,10,'$'

;----------------------------

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

;----------------------------

WRD\_TO\_DEC PROC near

push CX

push DX

mov CX,10

loop\_b:

div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_b

cmp AL,00h

je endl

or AL,30h

mov [SI],AL

endl:

pop DX

pop CX

ret

WRD\_TO\_DEC ENDP

;----------------------------

EX\_MEMORY PROC near

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

mov si,offset EX\_MEM

add si,21

mov dx,0

call WRD\_TO\_DEC

mov ah,9

mov dx, offset EX\_MEM

int 21h

ret

EX\_MEMORY ENDP

;----------------------------

AVAILABLE\_MEM PROC near

push es

mov ah,4ah

mov bx,0ffffh

int 21h

mov ax,bx

mov si,offset AV\_MEM

add si,23

mov dx,0

mov bx,16

mul bx

call WRD\_TO\_DEC

mov ah,9

mov dx, offset AV\_MEM

int 21h

pop es

ret

AVAILABLE\_MEM ENDP

;----------------------------

MCB PROC near

push es

mov dx,offset HEADER

mov ah,9

int 21h

mov cx,48

mov ah,52h

int 21h

mov ax,es:[bx-2]

mov es,ax

mov bx,es

start1:

inc cl

cmp cl,60

je end1

mov ax,es:[1]

mov di,offset MCB\_BLOCK

mov [di],cl

add di,5

call WRD\_TO\_HEX

mov ax,es:[3]

mov dx,16

mul dx

mov si,offset MCB\_BLOCK

add si,13

call WRD\_TO\_DEC

mov di,offset MCB\_BLOCK

add di,15

mov ax,es:[8]

mov [di],ax

mov ax,es:[10]

mov [di+2],ax

mov ax,es:[12]

mov [di+4],ax

mov ax,es:[14]

mov [di+6],ax

mov dx,offset MCB\_BLOCK

mov ah,9

int 21h

mov ah, 5ah

cmp es:[0],ah

je end1

inc bx

add bx,es:[3]

mov es,bx

jmp start1

end1:

pop es

ret

MCB ENDP

;----------------------------

BEGIN:

call AVAILABLE\_MEM

call EX\_MEMORY

call MCB

xor al,al

mov ah,4ch

int 21h

TESTPC ENDS

END START

**Lab3\_2.asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

HEADER db '# ADDR SIZE LAST8B ',13,10,36

MCB\_BLOCK db '0 0000 0000000 ',13,10,36

AV\_MEM db 'Available memory: bytes ',13,10,'$'

EX\_MEM db 'Expanded memory: KB ',13,10,'$'

;----------------------------

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

;----------------------------

WRD\_TO\_DEC PROC near

push CX

push DX

mov CX,10

loop\_b:

div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_b

cmp AL,00h

je endl

or AL,30h

mov [SI],AL

endl:

pop DX

pop CX

ret

WRD\_TO\_DEC ENDP

;----------------------------

EX\_MEMORY PROC near

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

mov si,offset EX\_MEM

add si,21

mov dx,0

call WRD\_TO\_DEC

mov ah,9

mov dx, offset EX\_MEM

int 21h

ret

EX\_MEMORY ENDP

;----------------------------

AVAILABLE\_MEM PROC near

push es

mov ah,4ah

mov bx,0ffffh

int 21h

mov ax,bx

mov si,offset AV\_MEM

add si,23

mov dx,0

mov bx,16

mul bx

call WRD\_TO\_DEC

mov ah,9

mov dx, offset AV\_MEM

int 21h

pop es

ret

AVAILABLE\_MEM ENDP

;----------------------------

MCB PROC near

push es

call FREE\_MEM

mov dx,offset HEADER

mov ah,9

int 21h

mov cx,48

mov ah,52h

int 21h

mov ax,es:[bx-2]

mov es,ax

mov bx,es

start1:

inc cl

cmp cl,60

je end1

mov ax,es:[1]

mov di,offset MCB\_BLOCK

mov [di],cl

add di,5

call WRD\_TO\_HEX

mov dx,0

mov ax,es:[3]

mov si,offset MCB\_BLOCK

add si,13

mov dx,16

mul dx

call WRD\_TO\_DEC

mov di,offset MCB\_BLOCK

add di,15

mov ax,es:[8]

mov [di],ax

mov ax,es:[10]

mov [di+2],ax

mov ax,es:[12]

mov [di+4],ax

mov ax,es:[14]

mov [di+6],ax

mov dx,offset MCB\_BLOCK

mov ah,9

int 21h

mov ah, 5ah

cmp es:[0],ah

je end1

inc bx

add bx,es:[3]

mov es,bx

jmp start1

end1:

pop es

ret

MCB ENDP

;----------------------------

FREE\_MEM PROC near

push bx

push ax

mov bx,offset edofprog

sar bx,1

sar bx,1

sar bx,1

sar bx,1

add bx,20h

mov ah,4ah

int 21h

pop ax

pop bx

ret

FREE\_MEM ENDP

;----------------------------

BEGIN:

call AVAILABLE\_MEM

call EX\_MEMORY

call MCB

xor al,al

mov ah,4ch

int 21h

edofprog:

TESTPC ENDS

END START

**Lab3\_3.asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

HEADER db '# ADDR SIZE LAST8B ',13,10,36

MCB\_BLOCK db '0 0000 0000000 ',13,10,36

AV\_MEM db 'Available memory: bytes ',13,10,'$'

EX\_MEM db 'Expanded memory: KB ',13,10,'$'

;----------------------------

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

;----------------------------

WRD\_TO\_DEC PROC near

push CX

push DX

mov CX,10

loop\_b:

div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_b

cmp AL,00h

je endl

or AL,30h

mov [SI],AL

endl:

pop DX

pop CX

ret

WRD\_TO\_DEC ENDP

;----------------------------

EX\_MEMORY PROC near

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

mov si,offset EX\_MEM

add si,21

mov dx,0

call WRD\_TO\_DEC

mov ah,9

mov dx, offset EX\_MEM

int 21h

ret

EX\_MEMORY ENDP

;----------------------------

AVAILABLE\_MEM PROC near

push es

mov ah,4ah

mov bx,0ffffh

int 21h

mov ax,bx

mov si,offset AV\_MEM

add si,23

mov dx,0

mov bx,16

mul bx

call WRD\_TO\_DEC

mov ah,9

mov dx, offset AV\_MEM

int 21h

pop es

ret

AVAILABLE\_MEM ENDP

;----------------------------

MCB PROC near

push es

call FREE\_MEM

mov dx,offset HEADER

mov ah,9

int 21h

mov cx,48

mov ah,52h

int 21h

mov ax,es:[bx-2]

mov es,ax

mov bx,es

start1:

inc cl

cmp cl,60

je end1

mov ax,es:[1]

mov di,offset MCB\_BLOCK

mov [di],cl

add di,5

call WRD\_TO\_HEX

mov dx,0

mov ax,es:[3]

mov si,offset MCB\_BLOCK

add si,13

mov dx,16

mul dx

call WRD\_TO\_DEC

mov di,offset MCB\_BLOCK

add di,15

mov ax,es:[8]

mov [di],ax

mov ax,es:[10]

mov [di+2],ax

mov ax,es:[12]

mov [di+4],ax

mov ax,es:[14]

mov [di+6],ax

mov dx,offset MCB\_BLOCK

mov ah,9

int 21h

mov ah, 5ah

cmp es:[0],ah

je end1

inc bx

add bx,es:[3]

mov es,bx

jmp start1

end1:

pop es

ret

MCB ENDP

;----------------------------

FREE\_MEM PROC near

push bx

push ax

mov bx,offset edofprog

sar bx,1

sar bx,1

sar bx,1

sar bx,1

add bx,20h

mov ah,4ah

int 21h

mov bx,1000h

mov ah,48h

int 21h

pop ax

pop bx

ret

FREE\_MEM ENDP

;----------------------------

BEGIN:

call AVAILABLE\_MEM

call EX\_MEMORY

call MCB

xor al,al

mov ah,4ch

int 21h

edofprog:

TESTPC ENDS

END START

**Lab3\_4.asm**

TESTPC SEGMENT

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

ORG 100H

START: jmp BEGIN

HEADER db '# ADDR SIZE LAST8B ',13,10,36

MCB\_BLOCK db '0 0000 0000000 ',13,10,36

AV\_MEM db 'Available memory: bytes ',13,10,'$'

EX\_MEM db 'Expanded memory: KB ',13,10,'$'

ERROR db 'ITERRUPTION ERROER!',13,10,36

;----------------------------

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

;----------------------------

WRD\_TO\_DEC PROC near

push CX

push DX

mov CX,10

loop\_b:

div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_b

cmp AL,00h

je endl

or AL,30h

mov [SI],AL

endl:

pop DX

pop CX

ret

WRD\_TO\_DEC ENDP

;----------------------------

EX\_MEMORY PROC near

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

mov si,offset EX\_MEM

add si,21

mov dx,0

call WRD\_TO\_DEC

mov ah,9

mov dx, offset EX\_MEM

int 21h

ret

EX\_MEMORY ENDP

;----------------------------

AVAILABLE\_MEM PROC near

push es

mov ah,4ah

mov bx,0ffffh

int 21h

mov ax,bx

mov si,offset AV\_MEM

add si,23

mov dx,0

mov bx,16

mul bx

call WRD\_TO\_DEC

mov ah,9

mov dx, offset AV\_MEM

int 21h

pop es

ret

AVAILABLE\_MEM ENDP

;----------------------------

MCB PROC near

push es

call FREE\_MEM

mov dx,offset HEADER

mov ah,9

int 21h

mov cx,48

mov ah,52h

int 21h

mov ax,es:[bx-2]

mov es,ax

mov bx,es

start1:

inc cl

cmp cl,60

je end1

mov ax,es:[1]

mov di,offset MCB\_BLOCK

mov [di],cl

add di,5

call WRD\_TO\_HEX

mov ax,es:[3]

mov si,offset MCB\_BLOCK

add si,13

mov dx,16

mul dx

call WRD\_TO\_DEC

mov di,offset MCB\_BLOCK

add di,15

mov ax,es:[8]

mov [di],ax

mov ax,es:[10]

mov [di+2],ax

mov ax,es:[12]

mov [di+4],ax

mov ax,es:[14]

mov [di+6],ax

mov dx,offset MCB\_BLOCK

mov ah,9

int 21h

mov ah, 5ah

cmp es:[0],ah

je end1

inc bx

add bx,es:[3]

mov es,bx

jmp start1

end1:

pop es

ret

MCB ENDP

;----------------------------

FREE\_MEM PROC near

push bx

push ax

mov ah,48h

mov bx,1000h

int 21h

jnc no\_error1

mov dx,offset ERROR

mov ah,9

int 21h

pop ax

pop bx

pop ax

jmp exit\_prog

no\_error1:

mov bx,offset edofprog

sar bx,1

sar bx,1

sar bx,1

sar bx,1

add bx,20h

mov ah,4ah

int 21h

pop ax

pop bx

ret

FREE\_MEM ENDP

;----------------------------

BEGIN:

call AVAILABLE\_MEM

call EX\_MEMORY

call MCB

exit\_prog:

xor al,al

mov ah,4ch

int 21h

edofprog:

TESTPC ENDS

END START