Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №6

«Защищенный и реальный режим процессора.

Переход из одного режима в другой и обработка прерываний»

Выполнил: Проверил:

Студент группы 150503 Преподаватель

Федорович И.И. Одинец Д.Н.

Минск, 2023

1. Постановка задачи

Написать программу, которая выполняет следующие действия:

- Переход из реального режима в защищенный.

- Перехватывает аппаратное прерывание от клавиатуры, в обработчике которого считываются скан-коды клавиш и выводятся на экран. По нажатию клавиши «esc» осуществляется обратный переход в реальный режим.

- Перехватывает аппаратное прерывание от таймера, в обработчике которого отсчитывает секунды и выводит их на экран. По истечении времени, введенного при старте программы осуществляется обратный переход в реальный режим.

1. Алгоритм

1) Вводим время нахождения в защищенном режиме в секундах.

2) Настраиваем таймер на частоту 20 Гц.

3) Загружаем линейные адреса сегментов в дескрипторы.

4) Загружаем таблицу глобальных дескрипторов в регистр gdtr.

5) Запрещаем прерываний.

6) Сохраняем маски прерываний.

7) Инициализируем контроллеры.

8) Загружаем таблицу дескрипторов исключений в регистр idtr.

9) Переходим в защищенный режим.

10) Загружаем селекторы в регистры сегментов.

11) Разрешаем прерывания.

12) Производится обработка прерываний от клавиатуры и таймера.

13) Запрещаем прерываний.

14) Настраиваем теневые регистры сегментов для работы в реальном режиме.

15) Переходим в реальный режим.

16) Восстанавливаем значения сегментов.

17) Восстанавливаем таблицу векторов прерываний.

18) Инициализируем контроллеры.

19) Восстанавливаем маски прерываний.

20) Завершаем.

1. Листинг программы

Далее приведен листинг программы, реализующей все поставленные задачи.

.386P

.MODEL LARGE

CODE\_RM segment para use16

CODE\_RM\_BEGIN = $

assume cs:CODE\_RM,DS:DATA,ES:DATA

START:

mov ax,DATA

mov ds,ax

mov es,ax

lea dx,MSG\_ENTER

mov ah,9h

int 21h

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

push ax

push dx

lea dx,POINT

mov ah,9h

int 21h

pop ax

pop dx

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

call INPUT

mov ds:[TIME], al

call FILL\_CR\_0\_BUFFER\_RM

lea dx, BUFFER\_CR\_0\_RM

mov ah, 9h

int 21h

lea dx,MSG\_HELLO

mov ah,9h

int 21h

mov ah,7h

int 21h

PREPARE\_RTC:

mov al,0Bh

out 70h,al

in al,71h

or al,00000100b

out 71h,al

ENABLE\_A20:

in al,92h

or al,2

out 92h,al

SAVE\_MASK:

in al,21h

mov INT\_MASK\_M,al

in al,0A1h

mov INT\_MASK\_S,al

DISABLE\_INTERRUPTS:

cli

in al,70h

or al,10000000b

out 70h,al

nop

LOAD\_GDT:

mov ax,DATA

mov dl,ah

xor dh,dh

shl ax,4

shr dx,4

mov si,ax

mov di,dx

WRITE\_GDT:

lea bx,GDT\_GDT

mov ax,si

mov dx,di

add ax,offset GDT

adc dx,0

mov [bx][S\_DESC.BASE\_L],ax

mov [bx][S\_DESC.BASE\_M],dl

mov [bx][S\_DESC.BASE\_H],dh

WRITE\_CODE\_RM:

lea bx,GDT\_CODE\_RM

mov ax,cs

xor dh,dh

mov dl,ah

shl ax,4

shr dx,4

mov [bx][S\_DESC.BASE\_L],ax

mov [bx][S\_DESC.BASE\_M],dl

mov [bx][S\_DESC.BASE\_H],dh

WRITE\_DATA:

lea bx,GDT\_DATA

mov ax,si

mov dx,di

mov [bx][S\_DESC.BASE\_L],ax

mov [bx][S\_DESC.BASE\_M],dl

mov [bx][S\_DESC.BASE\_H],dh

WRITE\_STACK:

lea bx, GDT\_STACK

mov ax,ss

xor dh,dh

mov dl,ah

shl ax,4

shr dx,4

mov [bx][S\_DESC.BASE\_L],ax

mov [bx][S\_DESC.BASE\_M],dl

mov [bx][S\_DESC.BASE\_H],dh

WRITE\_CODE\_PM:

lea bx,GDT\_CODE\_PM

mov ax,CODE\_PM

xor dh,dh

mov dl,ah

shl ax,4

shr dx,4

mov [bx][S\_DESC.BASE\_L],ax

mov [bx][S\_DESC.BASE\_M],dl

mov [bx][S\_DESC.BASE\_H],dh

WRITE\_IDT:

lea bx,GDT\_IDT

mov ax,si

mov dx,di

add ax,OFFSET IDT

adc dx,0

mov [bx][S\_DESC.BASE\_L],ax

mov [bx][S\_DESC.BASE\_M],dl

mov [bx][S\_DESC.BASE\_H],dh

mov IDTR.IDT\_L,ax

mov IDTR.IDT\_H,dx

FILL\_IDT:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

mov al, 182

out 43h, al

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

irpc N, 0123456789ABCDEF

lea eax, EXC\_0&N

mov IDT\_0&N.OFFS\_L,ax

shr eax, 16

mov IDT\_0&N.OFFS\_H,ax

endm

irpc N, 0123456789ABCDEF

lea eax, EXC\_1&N

mov IDT\_1&N.OFFS\_L,ax

shr eax, 16

mov IDT\_1&N.OFFS\_H,ax

endm

lea eax, TIMER\_HANDLER

mov IDT\_TIMER.OFFS\_L,ax

shr eax, 16

mov IDT\_TIMER.OFFS\_H,ax

lea eax, KEYBOARD\_HANDLER

mov IDT\_KEYBOARD.OFFS\_L,ax

shr eax, 16

mov IDT\_KEYBOARD.OFFS\_H,ax

irpc N, 234567

lea eax,IDLE\_IRQ\_MASTER

mov IDT\_2&N.OFFS\_L, AX

shr eax,16

mov IDT\_2&N.OFFS\_H, AX

endm

irpc N, 89ABCDEF

lea eax,IDLE\_IRQ\_SLAVE

mov IDT\_2&N.OFFS\_L,ax

shr eax,16

mov IDT\_2&N.OFFS\_H,ax

endm

lgdt fword ptr GDT\_GDT

lidt fword ptr IDTR

mov eax,cr0

or al,00000001b

mov cr0,eax

OVERLOAD\_CS:

db 0eah

dw offset OVERLOAD\_SEGMENT\_REGISTERS

dw CODE\_RM\_DESC

OVERLOAD\_SEGMENT\_REGISTERS:

mov ax,DATA\_DESC

mov ds,ax

mov es,ax

mov ax,STACK\_DESC

mov ss,ax

xor ax,ax

mov fs,ax

mov gs,ax

lldt ax

PREPARE\_TO\_RETURN:

push cs

push offset BACK\_TO\_RM

lea edi,ENTER\_PM

mov eax,CODE\_PM\_DESC

push eax

push edi

REINITIALIAZE\_CONTROLLER\_FOR\_PM:

mov al,00010001b

out 20h,al

out 0A0h,al

mov al,20h

out 21h,al

mov al,28h

out 0A1h,al

mov al,04h

out 21h,al

mov al,02h

out 0A1h,al

mov al,11h

out 21h,al

mov al,01h

out 0A1h,al

mov al, 0

out 21h,al

out 0A1h,al

ENABLE\_INTERRUPTS\_0:

in al,70h

and al,01111111b

out 70h,al

nop

sti

GO\_TO\_CODE\_PM:

db 66h

retf

BACK\_TO\_RM:

cli

in al,70h

or AL,10000000b

out 70h,AL

nop

REINITIALISE\_CONTROLLER:

mov al,00010001b

out 20h,al

out 0A0h,al

mov al,8h

out 21h,al

mov al,70h

out 0A1h,al

mov al,04h

out 21h,al

mov al,02h

out 0A1h,al

mov al,11h

out 21h,al

mov al,01h

out 0A1h,al

PREPARE\_SEGMENTS:

mov GDT\_CODE\_RM.LIMIT,0FFFFh

mov GDT\_DATA.LIMIT,0FFFFh

mov GDT\_STACK.LIMIT,0FFFFh

db 0EAH

dw offset CONTINUE

dw CODE\_RM\_DESC

CONTINUE:

mov ax,DATA\_DESC

mov ds,ax

mov es,ax

mov fs,ax

mov gs,ax

mov ax,STACK\_DESC

mov ss,ax

ENABLE\_REAL\_MODE:

mov eax,cr0

and al,11111110b

mov cr0,eax

db 0EAH

dw offset CONTINUE2

dw CODE\_RM

CONTINUE2:

mov ax,STACK\_A

mov ss,ax

mov ax,DATA

mov ds,ax

mov es,ax

xor ax,ax

mov fs,ax

mov gs,ax

mov IDTR.LIMIT, 3FFH

mov dword ptr IDTR+2, 0

lidt fword ptr IDTR

REPEAIR\_MASK:

mov al,INT\_MASK\_M

out 21h,al

mov al,INT\_MASK\_S

out 0A1h,al

ENABLE\_INTERRUPTS:

in al,70h

and al,01111111b

out 70h,al

nop

sti

DISABLE\_A20:

in al,92h

and al,11111101b

out 92h, al

EXIT:

mov ax,3h

int 10H

lea dx,MSG\_EXIT

mov ah,9h

int 21h

call FILL\_CR\_0\_BUFFER\_RM

lea dx, BUFFER\_CR\_0\_RM

mov ah, 9h

int 21h

mov ax,4C00h

int 21H

INPUT proc near

mov ah,0ah

xor di,di

mov dx,offset ds:[INPUT\_TIME]

int 21h

mov dl,0ah

mov ah,02

int 21h

mov si,offset INPUT\_TIME+2

cmp byte ptr [si],"-"

jnz ii1

mov di,1

inc si

II1:

xor ax,ax

mov bx,10

II2:

mov cl,[si]

cmp cl,0dh

jz ii3

cmp cl,'0'

jl er

cmp cl,'9'

ja er

sub cl,'0'

mul bx

add ax,cx

inc si

jmp ii2

ER:

mov dx, offset MSG\_ERROR

mov ah,09

int 21h

int 20h

II3:

ret

INPUT endp

FILL\_CR\_0\_BUFFER\_RM proc near

push eax

push esi

push dx

mov eax, cr0

xor dx, dx

mov cx, 32

lea esi, BUFFER\_CR\_0\_RM

fill\_cr\_0\_loop\_rm:

mov dl, al

shl dl, 7

shr dl, 7

shr eax, 1

add dl, 48

mov [esi], dl

inc esi

xor dl, dl

loop fill\_cr\_0\_loop\_rm

pop dx

pop esi

pop eax

ret

FILL\_CR\_0\_BUFFER\_RM endp

SIZE\_CODE\_RM = ($ - CODE\_RM\_BEGIN)

CODE\_RM ends

CODE\_PM segment para use32

CODE\_PM\_BEGIN = $

assume cs:CODE\_PM,ds:DATA,es:DATA

ENTER\_PM:

call CLRSCR

xor edi,edi

lea esi,MSG\_HELLO\_PM

call BUFFER\_OUTPUT

add edi,160

lea esi,MSG\_KEYBOARD

call BUFFER\_OUTPUT

mov edi,320

lea esi,MSG\_TIME

call BUFFER\_OUTPUT

mov edi,480

lea esi,MSG\_COUNT

call BUFFER\_OUTPUT

call FILL\_CR\_0\_BUFFER

mov edi, 640

lea esi, BUFFER\_CR\_0

call BUFFER\_OUTPUT

mov DS:[COUNT],0

WAITING\_ESC:

jmp WAITING\_ESC

EXIT\_PM:

db 66H

retf

EXIT\_FROM\_INTERRUPT:

popad

pop es

pop ds

pop eax

pop eax

pop eax

sti

db 66H

retf

WORD\_TO\_DEC proc near

pushad

movzx eax,ax

xor cx,cx

mov bx,10

LOOP1:

xor dx,dx

div bx

add dl,'0'

push dx

inc cx

test ax,ax

jnz LOOP1

LOOP2:

pop dx

mov [di],dl

inc di

loop LOOP2

popad

ret

WORD\_TO\_DEC endp

FILL\_CR\_0\_BUFFER proc near

push eax

push esi

push dx

mov eax, cr0

xor dx, dx

mov cx, 32

lea esi, BUFFER\_CR\_0

fill\_cr\_0\_loop:

mov dl, al

shl dl, 7

shr dl, 7

shr eax, 1

add dl, 48

mov [esi], dl

inc esi

xor dl, dl

loop fill\_cr\_0\_loop

pop dx

pop esi

pop eax

ret

FILL\_CR\_0\_BUFFER endp

DIGIT\_TO\_HEX proc near

add al,'0'

cmp al,'9'

jle DTH\_END

add al,7

DTH\_END:

ret

DIGIT\_TO\_HEX endp

BYTE\_TO\_HEX proc near

push ax

mov ah,al

shr al,4

call DIGIT\_TO\_HEX

mov [di],al

inc di

mov al,ah

and al,0Fh

call DIGIT\_TO\_HEX

mov [di],al

inc di

pop ax

ret

BYTE\_TO\_HEX endp

M = 0

IRPC N, 0123456789ABCDEF

EXC\_0&N label word

cli

jmp EXC\_HANDLER

endm

M = 010H

IRPC N, 0123456789ABCDEF

EXC\_1&N label word

cli

jmp EXC\_HANDLER

endm

EXC\_HANDLER proc near

call CLRSCR

lea esi, MSG\_EXC

mov edi, 40\*2

call BUFFER\_OUTPUT

pop eax

pop eax

pop eax

sti

db 66H

retf

EXC\_HANDLER ENDP

IDLE\_IRQ\_MASTER proc near

push eax

mov al,20h

out 20h,al

pop eax

iretd

IDLE\_IRQ\_MASTER endp

IDLE\_IRQ\_SLAVE proc near

push eax

mov al,20h

out 20h,al

out 0A0h,al

pop eax

iretd

IDLE\_IRQ\_SLAVE endp

TIMER\_HANDLER proc near

push ds

push es

pushad

mov ax,DATA\_DESC

mov ds,ax

inc ds:[COUNT]

lea edi,ds:[BUFFER\_COUNT]

mov ax,ds:[COUNT]

call WORD\_TO\_DEC

mov edi,538

lea esi,BUFFER\_COUNT

call BUFFER\_OUTPUT

SHOW\_TIMER:

mov al,0h

out 70h,al

in al,71h

cmp al,ds:[SECOND]

je SKIP\_SECOND

mov ds:[SECOND],al

mov al,ds:[TIME]

cmp ds:[TIME],0

je DISABLE\_PM

xor ah,ah

lea edi,ds:[BUFFER\_TIME]

call WORD\_TO\_DEC

mov edi,356

lea esi,BUFFER\_TIME

call BUFFER\_OUTPUT

dec ds:[TIME]

lea esi,BUFFER\_TIME

call BUFFER\_CLEAR

jmp SKIP\_SECOND

DISABLE\_PM:

mov al,20h

out 20h,al

db 0eah

dd OFFSET EXIT\_FROM\_INTERRUPT

dw CODE\_PM\_DESC

SKIP\_SECOND:

mov al,20h

out 20h,al

popad

pop es

pop ds

iretd

TIMER\_HANDLER endp

KEYBOARD\_HANDLER proc near

push ds

push es

pushad

in ax,61h

and ax, 65532

out 61h, ax

in al,60h

cmp al,1

je KEYBOARD\_EXIT

mov ds:[KEY\_SCAN\_CODE],al

lea edi,ds:[BUFFER\_SCAN\_CODE]

mov al,ds:[KEY\_SCAN\_CODE]

xor ah,ah

call BYTE\_TO\_HEX

mov edi,200

lea esi,BUFFER\_SCAN\_CODE

;push dx

;call PIANO

;pop dx

call SET\_TONALiTY

call ENABLR\_SOUND

call BUFFER\_OUTPUT

jmp KEYBOARD\_RETURN

KEYBOARD\_EXIT:

mov al,20h

out 20h,al

db 0eah

dd OFFSET EXIT\_FROM\_INTERRUPT

dw CODE\_PM\_DESC

KEYBOARD\_RETURN:

mov al,20h

out 20h,al

popad

pop es

pop ds

iretd

KEYBOARD\_HANDLER endp

PIANO proc near

A\_4:

cmp al, 25

jnz Ad\_4

mov dx, 2712

call SET\_TONALiTY

call ENABLR\_SOUND

Ad\_4:

cmp al, 12

jnz B\_4

mov dx, 2560

call SET\_TONALiTY

call ENABLR\_SOUND

B\_4:

cmp al, 26

jnz C\_5

mov dx, 2415

call SET\_TONALiTY

call ENABLR\_SOUND

C\_5:

cmp al, 27

jnz Cd\_5

mov dx, 2280

call SET\_TONALiTY

call ENABLR\_SOUND

Cd\_5:

cmp al, 33

jnz D\_5

mov dx, 2152

call SET\_TONALiTY

call ENABLR\_SOUND

D\_5:

cmp al, 47

jnz Dd\_5

mov dx, 2031

call SET\_TONALiTY

call ENABLR\_SOUND

Dd\_5:

cmp al, 34

jnz E\_5

mov dx, 1917

call SET\_TONALiTY

call ENABLR\_SOUND

E\_5:

cmp al, 48

jnz F\_5

mov dx, 1810

call SET\_TONALiTY

call ENABLR\_SOUND

F\_5:

cmp al, 36

jnz Fd\_5

mov dx, 1708

call SET\_TONALiTY

call ENABLR\_SOUND

Fd\_5:

cmp al, 10

jnz G\_5

mov dx, 1612

call SET\_TONALiTY

call ENABLR\_SOUND

G\_5:

cmp al, 37

jnz Gd\_5

mov dx, 1522

call SET\_TONALiTY

call ENABLR\_SOUND

Gd\_5:

cmp al, 11

jnz A\_5

mov ax, 1436

call SET\_TONALiTY

call ENABLR\_SOUND

A\_5:

cmp al, 38

jnz END\_PIANO

mov dx, 1356

call SET\_TONALiTY

call ENABLR\_SOUND

END\_PIANO:

ret

PIANO endp

SET\_TONALiTY proc near

mov dx, ax

push ax

mov ax, dx

mov bl, 200

mul al

;add ax, 5000

out 42h, al

mov al, ah

out 42h, al

pop ax

ret

SET\_TONALiTY endp

ENABLR\_SOUND proc near

push ds

push es

pushad

in al,61h

or al, 3

out 61h, al

popad

pop es

pop ds

ret

ENABLR\_SOUND endp

CLRSCR proc near

push es

pushad

mov ax,TEXT\_DESC

mov es,ax

xor edi,edi

mov ecx,80\*25

mov ax,700h

rep stosw

popad

pop es

ret

CLRSCR endp

BUFFER\_CLEAR proc near

mov al,' '

mov [esi+0],al

mov [esi+1],al

mov [esi+2],al

mov [esi+3],al

mov [esi+4],al

mov [esi+5],al

mov [esi+6],al

mov [esi+7],al

ret

BUFFER\_CLEAR endp

BUFFER\_OUTPUT proc near

push es

PUSHAD

mov ax,TEXT\_DESC

mov es,ax

OUTPUT\_LOOP:

lodsb

or al,al

jz OUTPUT\_EXIT

stosb

inc edi

jmp OUTPUT\_LOOP

OUTPUT\_EXIT:

popad

pop es

ret

BUFFER\_OUTPUT ENDP

SIZE\_CODE\_PM = ($ - CODE\_PM\_BEGIN)

CODE\_PM ENDS

DATA segment para use16

DATA\_BEGIN = $

S\_DESC struc

LIMIT dw 0

BASE\_L dw 0

BASE\_M db 0

ACCESS db 0

ATTRIBS db 0

BASE\_H db 0

S\_DESC ends

I\_DESC struc

OFFS\_L dw 0

SEL dw 0

PARAM\_CNT db 0

ACCESS db 0

OFFS\_H dw 0

I\_DESC ends

R\_IDTR struc

LIMIT dw 0

IDT\_L dw 0

IDT\_H dw 0

R\_IDTR ends

GDT\_BEGIN = $

GDT label word

GDT\_0 S\_DESC <0,0,0,0,0,0>

GDT\_GDT S\_DESC <GDT\_SIZE-1,,,10010010b,0,>

GDT\_CODE\_RM S\_DESC <SIZE\_CODE\_RM-1,,,10011010b,0,>

GDT\_DATA S\_DESC <SIZE\_DATA-1,,,11110010b,0,>

GDT\_STACK S\_DESC <1000h-1,,,10010010b,0,>

GDT\_TEXT S\_DESC <2000h-1,8000h,0Bh,11110010b,0,0>

GDT\_CODE\_PM S\_DESC <SIZE\_CODE\_PM-1,,,10011010b,01000000b,>

GDT\_IDT S\_DESC <SIZE\_IDT-1,,,10010010b,0,>

GDT\_SIZE = ($ - GDT\_BEGIN)

CODE\_RM\_DESC = (GDT\_CODE\_RM - GDT\_0)

DATA\_DESC = (GDT\_DATA - GDT\_0)

STACK\_DESC = (GDT\_STACK - GDT\_0)

TEXT\_DESC = (GDT\_TEXT - GDT\_0)

CODE\_PM\_DESC = (GDT\_CODE\_PM - GDT\_0)

IDT\_DESC = (GDT\_IDT - GDT\_0)

;IDT

IDTR R\_IDTR <SIZE\_IDT,0,0>

IDT label word

IDT\_BEGIN = $

IRPC N, 0123456789ABCDEF

IDT\_0&N I\_DESC <0, CODE\_PM\_DESC,0,10001111b,0>

ENDM

IRPC N, 0123456789ABCDEF

IDT\_1&N I\_DESC <0, CODE\_PM\_DESC, 0, 10001111b, 0>

ENDM

IDT\_TIMER I\_DESC <0,CODE\_PM\_DESC,0,10001110b,0>

IDT\_KEYBOARD I\_DESC <0,CODE\_PM\_DESC,0,10001110b,0>

IRPC N, 23456789ABCDEF

IDT\_2&N I\_DESC <0, CODE\_PM\_DESC, 0, 10001110b, 0>

ENDM

SIZE\_IDT = ($ - IDT\_BEGIN)

MSG\_HELLO db "Press key to change mode to PM",13,10,"$"

MSG\_HELLO\_PM db "We are in PM. Press ESC or wait till timer ends to exit PM",0

MSG\_EXIT db "We are in RM",13,10,"$"

MSG\_KEYBOARD db "Scan code:",0

MSG\_TIME db "Go back to RM in XXXXXXX seconds",0

MSG\_COUNT db "Amount of interrupt calls:",0

MSG\_EXC db "Exception: XX",0

MSG\_ENTER db "Enter time in protected mode: $"

MSG\_ERROR db "incorrect error$"

HEX\_TAB db "0123456789ABCDEF"

POINT db ".$"

P\_0 db "=0=",0

P\_1 db "=1=",0

P\_2 db "=2=",0

P\_3 db "=3=",0

P\_4 db "=4=",0

P\_5 db "=5=",0

P\_6 db "=6=",0

P\_7 db "=7=",0

P\_8 db "=8=",0

P\_9 db "=9=",0

ESP32 dd 1 dup(?)

INT\_MASK\_M db 1 dup(?)

INT\_MASK\_S db 1 dup(?)

KEY\_SCAN\_CODE db 1 dup(?)

SECOND db 1 dup(?)

TIME db 1 dup(10)

COUNT dw 1 dup(0)

BUFFER\_COUNT db 8 dup(' ')

db 1 dup(0)

BUFFER\_SCAN\_CODE db 8 dup(' ')

db 1 dup(0)

BUFFER\_TIME db 8 dup(' ')

db 1 dup(0)

INPUT\_TIME db 6,7 dup(?)

BUFFER\_CR\_0 db 32 dup('?')

db 1 dup(0)

BUFFER\_CR\_0\_RM db 32 dup('?'), 13, 10, "$"

SIZE\_DATA = ($ - DATA\_BEGIN)

DATA ends

STACK\_A segment para stack

db 1000h dup(?)

STACK\_A ends

end START

1. Тестирование программы

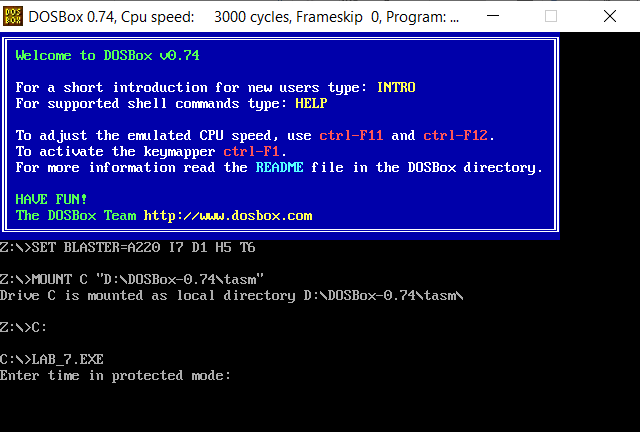


Рисунок 4.1. — Старт программы.

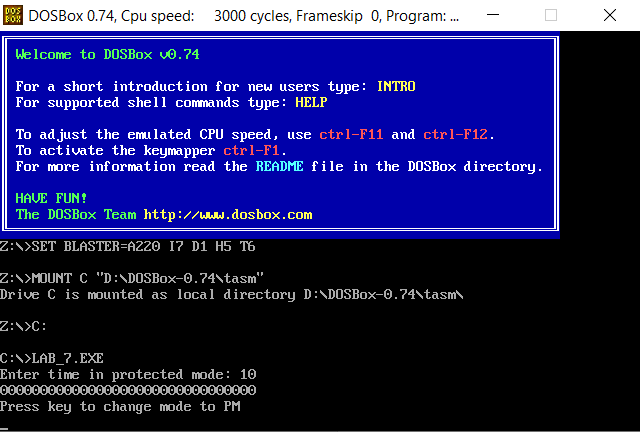


Рисунок 4.2. — Реальный режим.

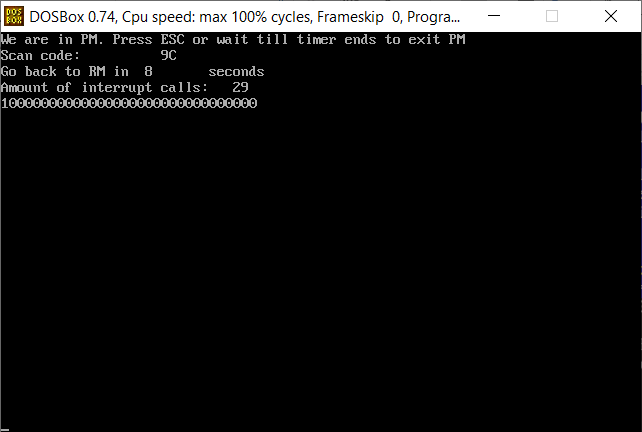


Рисунок 4.2. — Защищенный режим.

1. Заключение

В данной лабораторной работе были выполнены все поставленные задачи: был выполнен успешный переход в защищенный режим и возврат из него. Были написаны обработчики прерываний клавиатуры и таймера, выполняющие свою работу в защищенном режиме.

Программа запускалась в DOS, который эмулировался с помощью DOSBox 0.74-3.