Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №4

«Программирование часов реального времени»

Проверил: Выполнил:  
к. т. н., доцент студент гр. 150501

Одинец Д. Н. Климович А. Н.

Минск, 2023

**1 ПОСТАНОВКА ЗАДАЧИ**

**Первая часть**: Написать программу, которая будет считывать и устанавливать время в часах реального времени. Считанное время должно выводиться на экран в удобочитаемой форме.

*Условия выполнения лабораторной работы для первой части:*

1. Перед началом установки новых значений времени необходимо   
   считывать и анализировать старший байт регистра состояния 1 на предмет доступности значений для чтения и записи. Начинать   
   операцию записи новых значений, можно только в случае, когда этот бит установлен в '0' – то есть, регистры часов доступны.
2. После проверки доступности регистров (пункт 1), необходимо   
   отключить внутренний цикл обновления часов реального времени, воспользовавшись старшим битом регистра состояния 2. После окончания операции установки значений этот бит должен быть сброшен, для возобновления внутреннего цикла обновления часов реального времени.
3. Новые значения для регистров часов реального времени должны вводится с клавиатуры в удобном для пользователя виде.

**Вторая часть**:

1. Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний реализовать функцию задержки с точностью в миллисекунды (с возможностью изменения частоты генерации).

*Условия выполнения данного варианта:*

* Задержка должна вводится с клавиатуры в миллисекундах в   
  удобной для пользователя форме.
* После окончания периода задержки программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается.
* Предусмотреть возможность изменения частоты генерации   
  периодических прерываний вводом с клавиатуры соответствующих значений

1. Используя аппаратное прерывания часов реального времени и режим будильника ЧРВ (4A использовать не желательно) реализовать функции программируемого будильника.
2. Время будильника вводится с клавиатуры в удобной для пользователя форме.
3. При срабатывании будильника программа должна сообщить об этом в любой форме. При этом зависание компьютера не допускается.

**Общие условия для всей лабораторной работы:**

1. В программе должно быть реализовано меню, позволяющее выбрать

тестируемый функционал (установка времени, считывание времени, задержка и т.д.).

1. Весь ввод/вывод данных с/на консоль должен выполняться в   
   удобной для пользователя форме.
2. Зависание компьютера не допускается ни в ходе работы программы, ни после ее завершения.

2 АЛГОРИТМ

Перед установкой значений времени вызывается функция, которая считывает и анализирует старший байт регистра состояния 1 на предмет доступности значений для чтения и записи. Когда этот бит установлен в '0', отключается внутренний цикл обновления часов реального времени: для этого старший бит регистра состояния 2 устанавливается в '1'.

Считывание или запись значений времени происходит следующим образом: в порт 70h отправляется индекс регистра CMOS, соответствующий значению времени (секунды, часы и т. д.), затем происходит чтение значения из порта 71h (или запись значения в порт).

После установки значений времени вызывается функция, которая возобновляет внутренний цикл обновления часов реального времени.

Для реализации функции задержки заменён обработчик прерывания 0x70, в котором происходит отсчёт миллисекунд. Для включения периодического прерывания, происходящего примерно каждую миллисекунду, 6-й бит регистра B устанавливается в '1'.

3 КОД ПРОГРАММЫ

Далее приведен листинг программы, которая выполняет поставленные задачи.

.model tiny

.code

org 100h

jmp start

;=======================================

;MACROS:

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

PUSH\_ALL MACRO

push ax

push bx

push dx

push cx

ENDM

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

POP\_ALL MACRO

pop cx

pop dx

pop bx

pop ax

ENDM

PUTS\_STRING MACRO str

push ax

push dx

mov dx, offset str

mov ah, 09h

int 21h

pop dx

pop ax

ENDM

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

GETS\_STRING MACRO str

push ax

push dx

mov ah, 0ah

mov dx, offset str

int 21h

pop dx

pop ax

ENDM

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

PUTCHAR MACRO symbol

push ax

push dx

mov ah, 02h

mov dl, symbol

int 21h

pop dx

pop ax

ENDM

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

GETCHAR MACRO symbol

push ax

mov ah, 08h

int 21h

mov symbol, al

pop ax

ENDM

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

WRITE\_PORT MACRO n\_func source

call CHECK\_DOMAIN

cli

mov al, n\_func

out 70h, al

mov al, source

jmp short $+2

out 71h, al ;;end stop updating

sti

ENDM

READ\_PORT MACRO number\_func

;al-result

call CHECK\_DOMAIN

cli

mov al, number\_func

out 70h, al

jmp short $+2

in al, 71h ;;end stop updating

sti

ENDM

PRINT\_DATE\_OR\_TIME MACRO value

push ax

xor ax, ax

mov al, value

mov number\_digit, ax

call PRINT\_ZERO\_IF\_NEED

call PRINT\_NUMBER

push ax

ENDM

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

;FUNCTIONS:

;=======================================

CHECK\_DOMAIN PROC

push ax

check\_time\_on\_domain:

mov al, 0Ah

out 70h, al

jmp short $+2

in al, 71h

mov ah, 10000000b

and ah, al

cmp ah, 10000000b

je check\_time\_on\_domain

pop ax

ret

ENDP

;=======================================

PRINT\_ZERO\_IF\_NEED PROC

push bx

mov bl, 9

cmp al, bl

jbe zero\_is\_need

jmp zero\_is\_not\_need

zero\_is\_need:

PUTCHAR 30h ;'0'

zero\_is\_not\_need:

pop bx

ret

ENDP

;=======================================

CLEAR\_SCREEN PROC ;This proc clears the user's screen

PUSH\_ALL

mov ax, 0600h

mov bh, 07h

xor cx, cx

mov dx, 184Fh

int 10h

xor dx, dx ;set cursor at (0;0)

xor bx, bx

xor ax, ax

mov ah, 2 ;Interrupt service 10h to change cursor position

int 10h

POP\_ALL

ret

ENDP

;=======================================

GET\_CURRENT\_DATE\_AND\_TIME PROC

push ax

READ\_PORT 32h ;get century

mov century\_value, al

READ\_PORT 09h ;get year

mov year\_value, al

READ\_PORT 08h ;get month

mov month\_value, al

READ\_PORT 07h ;get day

mov day\_value, al

READ\_PORT 04h ;get hours

mov hours\_value, al

READ\_PORT 02h ;get minutes

mov minutes\_value, al

READ\_PORT 00h ;get seconds

mov seconds\_value, al

pop ax

ret

ENDP

;=======================================

SET\_CURRENT\_TIME PROC

PUSH\_ALL

xor cx, cx

mov di, offset hours\_value+2

mov cl, hours\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov hours\_value, al

xor cx, cx

mov di, offset minutes\_value+2

mov cl, minutes\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov minutes\_value, al

xor cx, cx

mov di, offset seconds\_value+2

mov cl, seconds\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov seconds\_value, al

READ\_PORT 0Bh ;stop updating clock

mov ah, al

or ah, 10000000b

WRITE\_PORT 0Bh ah ;end stop updating

WRITE\_PORT 04h hours\_value

WRITE\_PORT 02h minutes\_value

WRITE\_PORT 00h seconds\_value

READ\_PORT 0Bh ;start clock

mov ah, al ;start updating clock

and ah, 01111111b

WRITE\_PORT 0Bh ah ;end start updating

POP\_ALL

ret

ENDP

SET\_CURRENT\_DATE PROC

PUSH\_ALL

xor cx, cx

mov di, offset century\_value+2

mov cl, century\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov century\_value, al

xor cx, cx

mov di, offset year\_value+2

mov cl, year\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov year\_value, al

xor cx, cx

mov di, offset month\_value+2

mov cl, month\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov month\_value, al

xor cx, cx

mov di, offset day\_value+2

mov cl, day\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov day\_value, al

READ\_PORT 0Bh ;stop updating clock

mov ah, al

or ah, 10000000b

WRITE\_PORT 0Bh ah ;end stop updating

WRITE\_PORT 32h century\_value

WRITE\_PORT 09h year\_value

WRITE\_PORT 08h month\_value

WRITE\_PORT 07h day\_value

READ\_PORT 0Bh ;start clock

mov ah, al ;start updating clock

and ah, 01111111b

WRITE\_PORT 0Bh ah ;end start updating

POP\_ALL

ret

ENDP

;=======================================

SET\_ALARM\_TIME PROC

PUSH\_ALL

mov ax,2570h ;set interrapt vector for alarm

mov dx, offset ALARM\_HANDLER

int 21h

xor cx, cx

mov di, offset hours\_value+2

mov cl, hours\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov hours\_value, al

xor cx, cx

mov di, offset minutes\_value+2

mov cl, minutes\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov minutes\_value, al

xor cx, cx

mov di, offset seconds\_value+2

mov cl, seconds\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov seconds\_value, al

READ\_PORT 0Bh ;stop updating clock

mov ah, al

or ah, 10000000b

WRITE\_PORT 0Bh ah ;end of stop update

WRITE\_PORT 05h hours\_value

WRITE\_PORT 03h minutes\_value

WRITE\_PORT 01h seconds\_value

READ\_PORT 0Bh ;start updating clock

mov ah, al

and ah, 01111111b

WRITE\_PORT 0Bh ah ;end start updating

READ\_PORT 0Bh ;enable interrupt of alarm

mov ah, al

or ah, 00100000b

and ah, 10101111b

WRITE\_PORT 0Bh ah

POP\_ALL

ret

ENDP

;=======================================

ALARM\_HANDLER PROC FAR ;IRQ8 - interrupt for alarm

mov ax,0B88Ch ;adress of video memory, 14x160, where 14 is line

mov es,ax

mov cx,20

xor di,di

output:

mov al, 0Eh ;music symbol

mov es:[di],al

inc di

mov al,11001001b ;7 - miganie, 6-4 - background color, 3 - miganie, 0-2 - symbol color

mov es:[di],al

inc di

loop output

READ\_PORT 0Ch ;end start updating

mov al, 20h ;clear interrupt conroller ISR(master)

out 0a0h, al

mov al, 20h ;clear interrupt conroller ISR(slave)

out 20h, al

iret

ENDP

GET\_ALARM\_TIME PROC

push ax

READ\_PORT 05h ;get hours

mov hours\_value, al

READ\_PORT 03h ;get minutes

mov minutes\_value, al

READ\_PORT 01h ;get seconds

mov seconds\_value, al

pop ax

ret

ENDP

;=======================================

SET\_DELAY PROC

PUSH\_ALL

mov ax, 2570h ;set interrupt vector for delay

mov dx, offset DELAY\_HANDLER

int 21h

xor cx, cx

mov di, offset delay\_value+2

mov cl, delay\_value+1

call CONVERT\_STR\_TO\_DECIMAL

mov bx, ax

READ\_PORT 0Bh

mov ah, al

or ah, 01000000b ;enable to period interruptions

and ah, 11001111b ;stop clock, enable to period interruptions, unenable alarm, unenable inter-  
 ;rupt about change of time

WRITE\_PORT 0Bh ah

delay\_loop:

cmp bx, 0

jbe stop\_delay

jmp delay\_loop

stop\_delay:

READ\_PORT 0Bh ;enable interrupt

mov ah, al

and ah, 10111111b

WRITE\_PORT 0Bh, ah

POP\_ALL

ret

ENDP

;=======================================

DELAY\_HANDLER proc ;IRQ8 - handler for delay

dec bx

READ\_PORT 0Ch

mov al, 20h ;clear interrapt conroller ISR(master)

out 0a0h, al

mov al, 20h ;clear interrapt conroller ISR(slave)

out 20h, al

iret

ENDP

CONVERT\_STR\_TO\_DECIMAL PROC

;di-string offset

;ax-result number

;cx-len of string

push bx

push dx

xor ax,ax

xor bx,bx

mov si,10

m1:

mul si

mov bl,[di]

sub bl,30h

add ax,bx

inc di

loop m1

pop dx

pop bx

ret

ENDP

;=======================================

PRINT\_NUMBER PROC

PUSH\_ALL

mov ax,number\_digit ;our number

mov bx,10 ;base of counting system

xor cx,cx ;amount of digits in hex-number

L1:

xor dx,dx

div bx ;div number on 16

push dx ;save ostatok in stack

inc cx ;increment amoount of digits

test ax,ax ;does stay zero after division?

jnz L1 ;if not, then continue

L2:

pop ax ;take digit from stack

cmp al, 9 ;if digit > 9 (without sign)

ja isLetter

jmp isDigit

isLetter:

add al, 55 ;to letter

jmp print\_digit

isDigit:

add al,'0' ;transfer digit to ASCII symbol

print\_digit:

PUTCHAR al ;print digit

loop L2 ;all digits

POP\_ALL

ret

ENDP

print\_error:

POP\_ALL

PUTS\_STRING error\_msg

jmp choice\_loop

EXIT proc

mov ax,4C00h

int 21h

ret

endp

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

; MAIN FUNCTION

;=======================================

start:

mov al, 00000000b ;set mask for first conroller of int(permit all interrupts)

out 21h, al

mov al, 00000000b ;set mask for second conroller of int (permit all interrupts)

out 0a1h, al

WRITE\_PORT 0Bh al ;set binary instead of bsd-code

mov ah, al

or ah, 00000100b

WRITE\_PORT 0Bh ah

choice\_loop:

PUTS\_STRING invite\_msg

PUTS\_STRING options

GETCHAR bl

cmp bl, '1'

je get\_date\_and\_time\_option

cmp bl, '2'

je set\_time\_option

cmp bl, '3'

je set\_date\_option

cmp bl, '4'

je set\_alarm\_option

cmp bl, '5'

je get\_alarm\_option

cmp bl, '6'

je set\_delay\_option

cmp bl, '7'

je get\_delay\_option

cmp bl, '8'

je set\_frequancy\_option

cmp bl, '9'

je get\_frequancy\_option

cmp bl, '0'

je exit\_option

PUTS\_STRING error\_msg

loopne choice\_loop

;options:

;=======================================

get\_date\_and\_time\_option:

call CLEAR\_SCREEN

PUSH\_ALL

call GET\_CURRENT\_DATE\_AND\_TIME

PUTS\_STRING date\_msg

PRINT\_DATE\_OR\_TIME day\_value ;day

PUTCHAR 2Eh ; '.'

PRINT\_DATE\_OR\_TIME month\_value ;month

PUTCHAR 2Eh ; '.'

PRINT\_DATE\_OR\_TIME century\_value ;century

PRINT\_DATE\_OR\_TIME year\_value ;year

PUTS\_STRING \n

PUTS\_STRING time\_msg

PRINT\_DATE\_OR\_TIME hours\_value ;hours

PUTCHAR 3Ah ; ':'

PRINT\_DATE\_OR\_TIME minutes\_value ;minutes

PUTCHAR 3Ah ; ':'

PRINT\_DATE\_OR\_TIME seconds\_value ;seconds

PUTS\_STRING \n

PUTS\_STRING \n

POP\_ALL

jmp choice\_loop

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

set\_time\_option:

call CLEAR\_SCREEN

PUTS\_STRING hours\_msg ;hours

GETS\_STRING hours\_value

PUTS\_STRING minutes\_msg ;minutes

GETS\_STRING minutes\_value

PUTS\_STRING seconds\_msg ;seconds

GETS\_STRING seconds\_value

call SET\_CURRENT\_TIME

call CLEAR\_SCREEN

jmp choice\_loop

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

set\_date\_option:

call CLEAR\_SCREEN

PUTS\_STRING century\_msg ;century

GETS\_STRING century\_value

PUTS\_STRING \n

PUTS\_STRING year\_msg ;year

GETS\_STRING year\_value

PUTS\_STRING month\_msg ;month

GETS\_STRING month\_value

PUTS\_STRING day\_msg ;day

GETS\_STRING day\_value

call SET\_CURRENT\_DATE

call CLEAR\_SCREEN

jmp choice\_loop

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

set\_alarm\_option:

call CLEAR\_SCREEN

PUTS\_STRING hours\_msg

GETS\_STRING hours\_value

PUTS\_STRING minutes\_msg

GETS\_STRING minutes\_value

PUTS\_STRING seconds\_msg

GETS\_STRING seconds\_value

call SET\_ALARM\_TIME

call CLEAR\_SCREEN

jmp choice\_loop

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

get\_alarm\_option:

call CLEAR\_SCREEN

call GET\_ALARM\_TIME

PUTS\_STRING time\_msg

PRINT\_DATE\_OR\_TIME hours\_value ;hours

PUTCHAR 3Ah ; ':'

PRINT\_DATE\_OR\_TIME minutes\_value ;minutes

PUTCHAR 3Ah ; ':'

PRINT\_DATE\_OR\_TIME seconds\_value ;seconds

PUTS\_STRING \n

PUTS\_STRING \n

jmp choice\_loop

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

set\_delay\_option:

call CLEAR\_SCREEN

PUSH\_ALL

PUTS\_STRING delay\_msg

GETS\_STRING delay\_value

call SET\_DELAY

POP\_ALL

call CLEAR\_SCREEN

jmp choice\_loop

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

get\_delay\_option:

call CLEAR\_SCREEN

PUSH\_ALL

READ\_PORT 0Bh

mov ah, 10000000b

mov cx, 8

jmp output\_register\_bits

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

set\_frequancy\_option:

call CLEAR\_SCREEN

PUSH\_ALL

PUTS\_STRING frequancy\_menu

GETS\_STRING choosing\_freq

xor cx, cx

xor ax, ax

mov di, offset choosing\_freq+2

mov cl, choosing\_freq+1

call CONVERT\_STR\_TO\_DECIMAL

mov di, offset possible\_freq

dec ax

add di, ax

mov bl, [di]

READ\_PORT 0Ah

mov ah, al

and ah, 11110000b ;last 4 bits - our setting frequancy

or ah, bl

WRITE\_PORT 0Ah ah

READ\_PORT 0Ah

POP\_ALL

call CLEAR\_SCREEN

jmp choice\_loop

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

get\_frequancy\_option:

call CLEAR\_SCREEN

PUSH\_ALL

READ\_PORT 0Ah

mov ah, 10000000b

mov cx, 8

output\_register\_bits:

mov bl, al

and bl, ah

cmp bl, 0

jne print\_one\_bit

je print\_zero\_bit

print\_one\_bit:

PUTCHAR 31h

jmp continue\_output\_register\_bits

print\_zero\_bit:

PUTCHAR 30h

continue\_output\_register\_bits:

shr ah, 1

loop output\_register\_bits

jmp end\_of\_operation

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

exit\_option:

call CLEAR\_SCREEN

call EXIT

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

end\_of\_operation:

PUTS\_STRING continue\_msg

GETCHAR bl

POP\_ALL

call CLEAR\_SCREEN

jmp choice\_loop

;=======================================

;DATA:

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

number\_digit dw 0

delay\_value db 10 dup('$')

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

century\_value db 10 dup('$')

year\_value db 10 dup('$')

month\_value db 10 dup('$')

day\_value db 1000 dup(20h)

hours\_value db 10 dup('$')

minutes\_value db 10 dup('$')

seconds\_value db 10 dup('$')

choosing\_freq db 5 dup('$')

possible\_freq db 03h, 04h, 05h, 06h, 07h, 08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh

;MENU:

invite\_msg db "CHOOSE AN OPTION:", 0dh, 0ah, '$'

options db "1. Get date&time;", 0dh, 0ah

db "2. Set time;", 0dh, 0ah

db "3. Set date;", 0dh, 0ah

db "4. Set alarm;", 0ah, 0dh

db "5. Get alarm;", 0ah, 0dh

db "6. Set delay;", 0ah, 0dh

db "7. Get register 0Bh;", 0ah, 0dh

db "8. Set frequancy;", 0ah, 0dh

db "9. Get register 0Ah;", 0ah, 0dh

db "0. Exit.", 0dh, 0ah, '$'

frequancy\_menu db "1. 8 kHz;", 0ah, 0dh

db "2. 4 kHz;", 0ah, 0dh

db "3. 2 kHz;", 0ah, 0dh

db "4. 1 kHz;", 0ah, 0dh

db "5. 512 Hz;", 0ah, 0dh

db "6. 256 Hz;", 0ah, 0dh

db "7. 128 Hz;", 0ah, 0dh

db "8. 64 Hz;", 0ah, 0dh

db "9. 32 Hz;", 0ah, 0dh

db "10. 16 Hz;", 0ah, 0dh

db "11. 8 Hz;", 0ah, 0dh

db "12. 4 Hz;", 0ah, 0dh

db "13. 2 Hz.", 0ah, 0dh, '$'

;INFORMATION STRINGS:

year\_msg db "Year: $"

century\_msg db 0ah, 0dh, "Century: $"

month\_msg db 0ah, 0dh, "Month: $"

day\_msg db 0ah, 0dh, "Day: $"

hours\_msg db "Hours: $"

minutes\_msg db 0ah, 0dh, "Minuts: $"

seconds\_msg db 0ah, 0dh, "Seconds: $"

date\_msg db "Date: $"

time\_msg db "Time: $"

delay\_msg db "ENTER A DELAY (ms, <65s): $"

trnd\_off\_msg db "ALARM IS TURNED OFF.", 0ah, 0dh, 0ah, 0dh, '$'

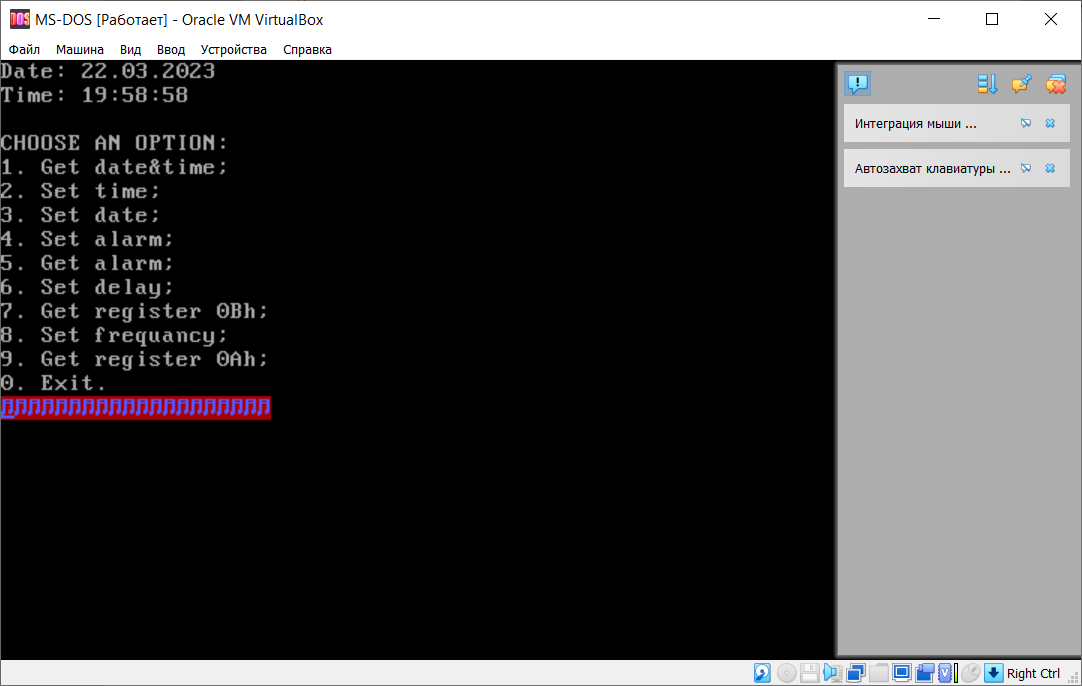
error\_msg db 0ah, 0dh, "ERROR!", 0dh, 0ah, '$'

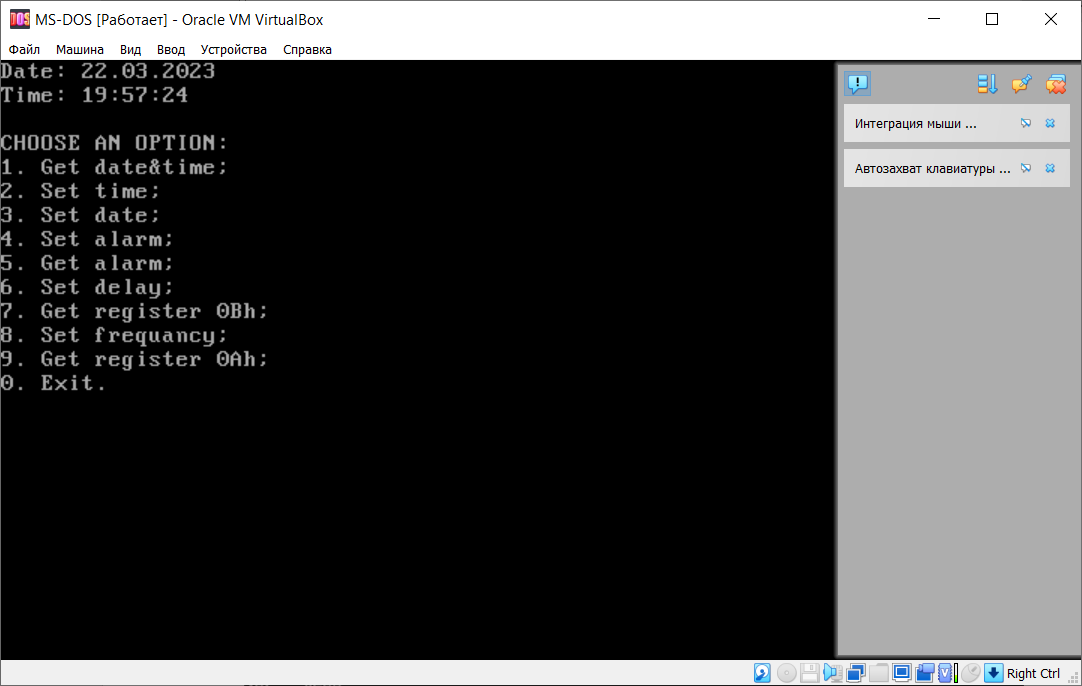
continue\_msg db 0ah, 0dh, "Press any key to continue...", 0ah, 0dh, '$'

end start

4 ТЕСТИРОВАНИЕ

На рисунке 4.1 а приведен скриншоты получения системного времени. На рисунке 4.1 б приведен скриншот срабатывания будильника.





а б

Рисунок 4.1 – Получение системного времени (а), срабатывание   
будильника (б)

5 ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были выполнены все поставленные задачи: написана программа, которая считывает и устанавливает время в часах реального времени, реализована функция задержки, используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний, а также была реализована функция программируемого будильника, используя аппаратное прерывания часов реального времени и режим будильника.

Программа компилировалась в Emu8086 и запускалась на MS-DOS, который эмулировался с помощью VirtualBox.