Universitatea Tehnică a Moldovei
Catedra Automatică și Tehnologii Informaționale

Raport

La Sisteme de Operare: Mecanisme interne și principii de proiectare
Lucrarea de laborator Nr.6

Tema : Integrarea sistemului de operare.

A efectuat:
st.gr. SI-131 Popov Eugen

A verificat:
lect. asist. Ostapenco S.
lect. asist. Petcov V.

Chișinău 2015

Scopul lucrării

Integrarea lucrărilor de laborator 2 - 5 într-un sistem de operare complet funcțional, urmînd condițiile.

Obiective

Luând nucleul dezvoltat la Laboratorul Nr.3, de implementat modificări conform condițiilor.

Condiții

Sistemul de operare obligator va fi format din Loader - preLoader - Nucleu, se va pastra forma de incarcare a nucleului din laboratorul 3 (citirea sector cu sector la realizarea condiției de încarcare), la fel va contine o interfață(ca punct de reper luați condiția Lab4) și desigur Laboratorul 5 cu comutarea contextului.

În final veți prezenta un sistem de operare cu minim 6 - 8 functionalitați care 3 din ele vor fi obligatoriu de dezvoltat pentru toți:

1. De implementat un joc

2. Help - informatii generale acesibile atît viziual cît și la tastarea butonului F1

3. Creare și salvarea unui fișier .DAT.

Mersul lucrării

Lucrarea de laborator constă în elaborarea kernel-ului dezvoltat, cu funcționalitățile respective.

În bootloader și în kernel am folosit următoarele proceduri:

- *WriteString* – procedură de afișare a unui șir de caractere pe eran , care folosește funcția 0x13 a întreruperii 0x10.
- *Delay* – procedură de realizeare a unei rețineri de 5 secunde , care folosește funcția 0x86 a întreruperii 0x15.
- *ClearScreen* – curățarea ecranului . Funcția 0x06 a întreruperii 0x10.
- *WriteMessages* – procedura ce afișează string-urile din toate 8 offset-uri la coordonata anumită a ecranului. Procedura data apelează procedura *WriteString*.
- *Beep* – procedura ce efectuează un sunet la accesarea tastei enter. (Nu funcționează în VirtualBox)
- *ReadSector* – procedura citește kernelul de pe sectorul dischetei și îl încarcă în memorie.
- *CheckSector* – procedura verifică dacă citirea este efectuată cu succes. Afișăm eroare dacă floppy disc-ul nu a fost găsit.
- *DrawButton* – procedura desenează un buton din meniul principal. (Colorat în culoarea verde)
- *DrawSelectedButton* – procedura desenează un buton care este selectat și urmează a fi accesat.(Colorat în culoarea roșie)
- *DrawWindow* – procedura desenează fereastra preview a funcției selectate din meniu.
- *CheckButton* – verifică butonul apăsat pe tastatură de către utilizator.
- *SelectFunction* – alege funcționalul în dependență de butonul selectat din meniu.
- *Triangle* – desenează un triunghi dreptunghic în meniul Draw.
- *Romb* – desenează un romb în meniul Draw.
- *Rectangle* - desenează un dreptunghi în meniul Draw.
- *DisplayHour* – convertează din hexazecimal în ASCII și afișează ora în meniul Time.
- *DisplayMinute* – convertează din hexazecimal în ASCII și afișează minutele în meniul Time.[1]
- *Morse* – procedura emite S.O.S. în codul Morse. (Funcția Beep din meniul principal)
- *Validate* – procedura verifică dacă spre calcul este propus un număr întreg . În caz contrar afișează eroarea "*Not an integer!*". (Funcția Calc din meniul principal)

[1] Pentru afișarea timpului de sistem am folosit funcția 02h a întreruperii 1ah , care nu funcționează în EMU8086.

- *CheckSignA* – stabilește semnul primului număr.
- *CheckSignB* – stabilește semnul celui de-al doilea număr.
- *Operation* – selecția operației în dependență de simbolul introdus de la tastatură. (+,-,*,/)
- *DisplayResult* – afișează rezultatul operației de adunare, scădere, înmulțire.
- *DisplayDivisionResult* - afișează rezultatul operației de împărțire.[2]
- *ResultSign* – determină semnul rezultatului.[3]

Pentru a înscrie fișierele pe dischetă am folosit programul Emu8086.

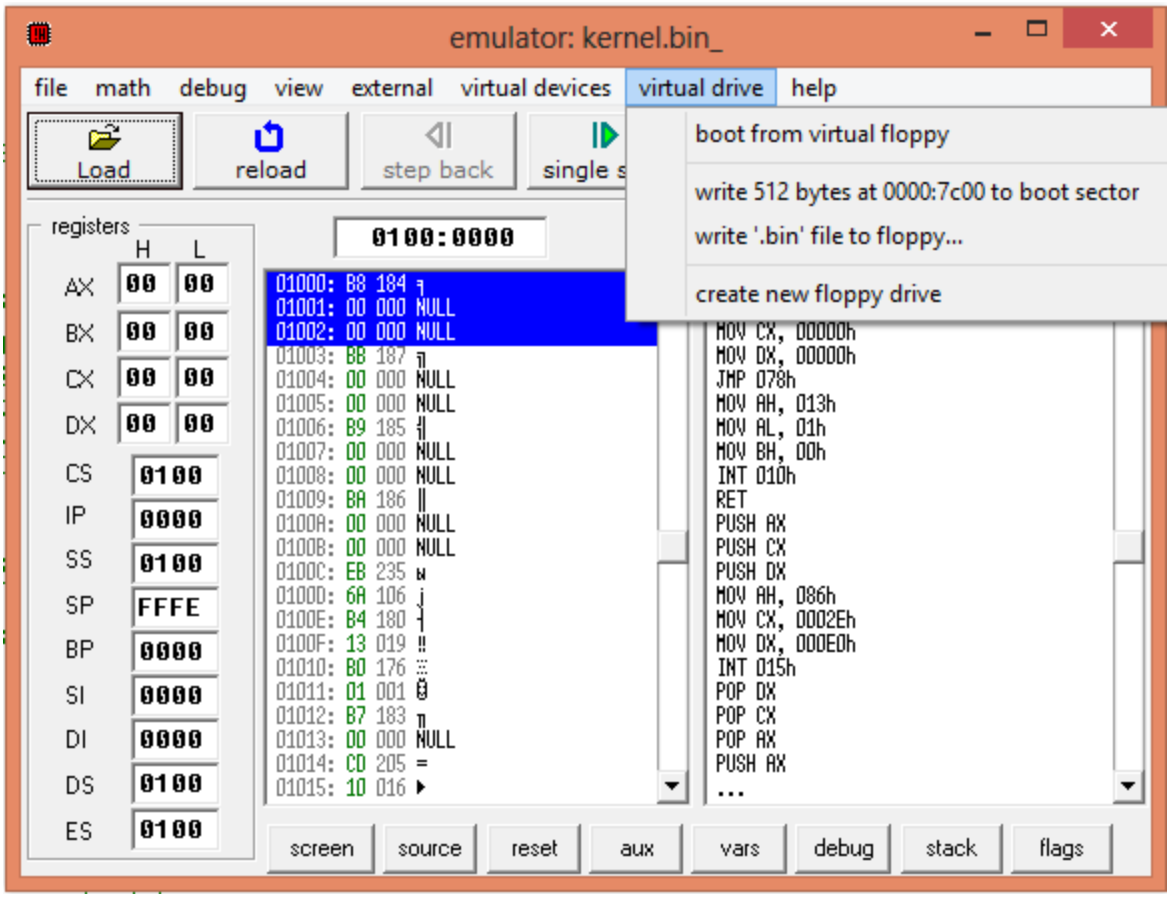

Fig. 1 Folosirea Emu8086 pentru înscrierea pe floppy virtual

Această imagine a dischetei o înscriem pe floppy-ul virtual cu ajutorul softului RawWrite.

[2] Rezultatul împărțirii este afișat în formă de parte întreagă și rest.

[3] Funcția Calc din meniul principal operează cu numere din maxim 3 cifre pentru operațiile de adunare, scădere și împărțire . La operația de înmuțire operanzii pot fi de maxim 2 cifre, în caz contrar se afișează eroarea "*Overflow!*".

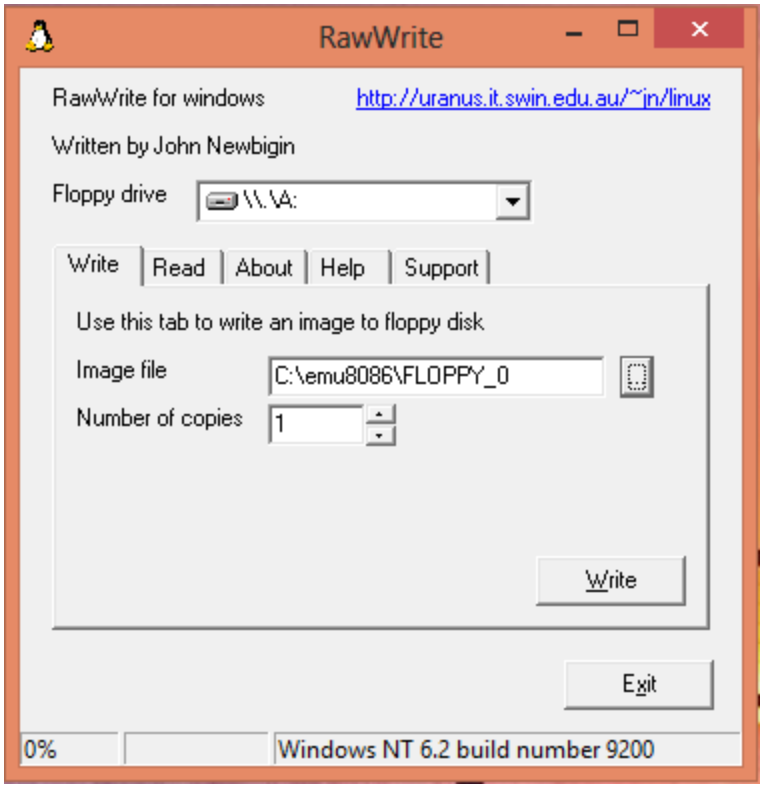

Fig. 2 Înscrierea imaginii dischetei pe floppy-ul virtual

La pornirea sistemului în VirtualBox vedem următoarele:

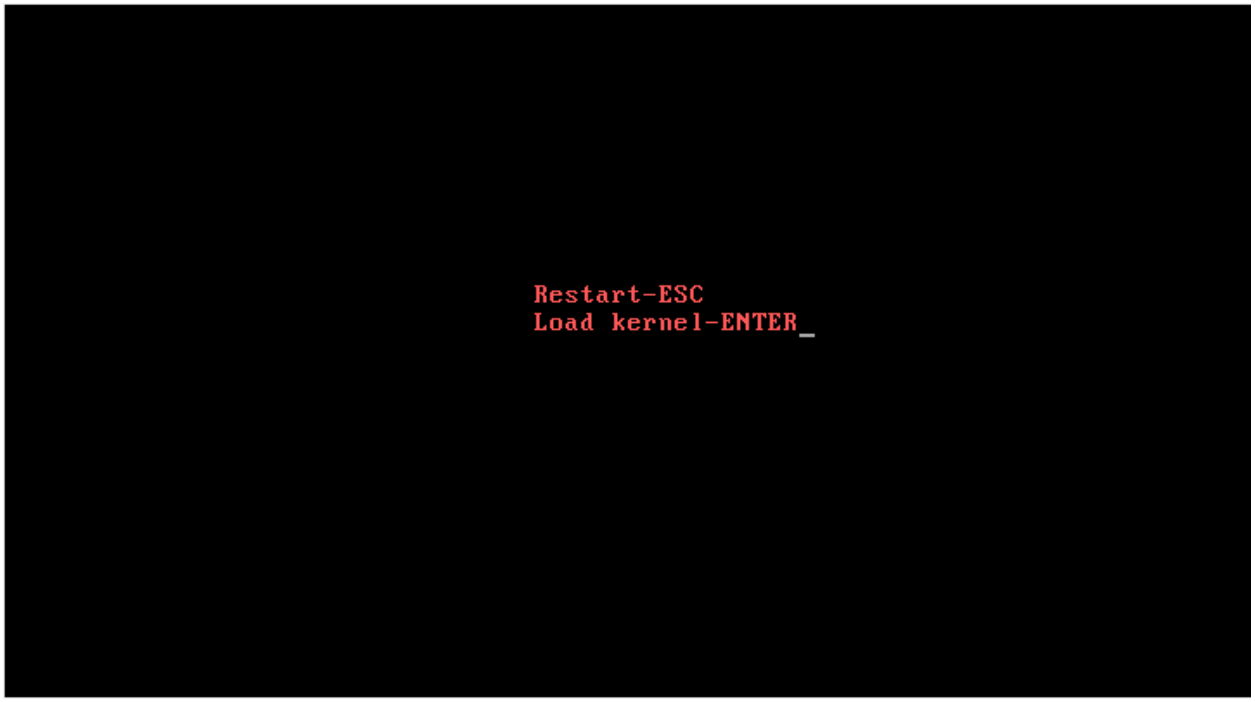

Fig. 3 Încărcarea bootloader-ului

În continuare la accesarea tastei ENTER se va încărca elemental GUI .

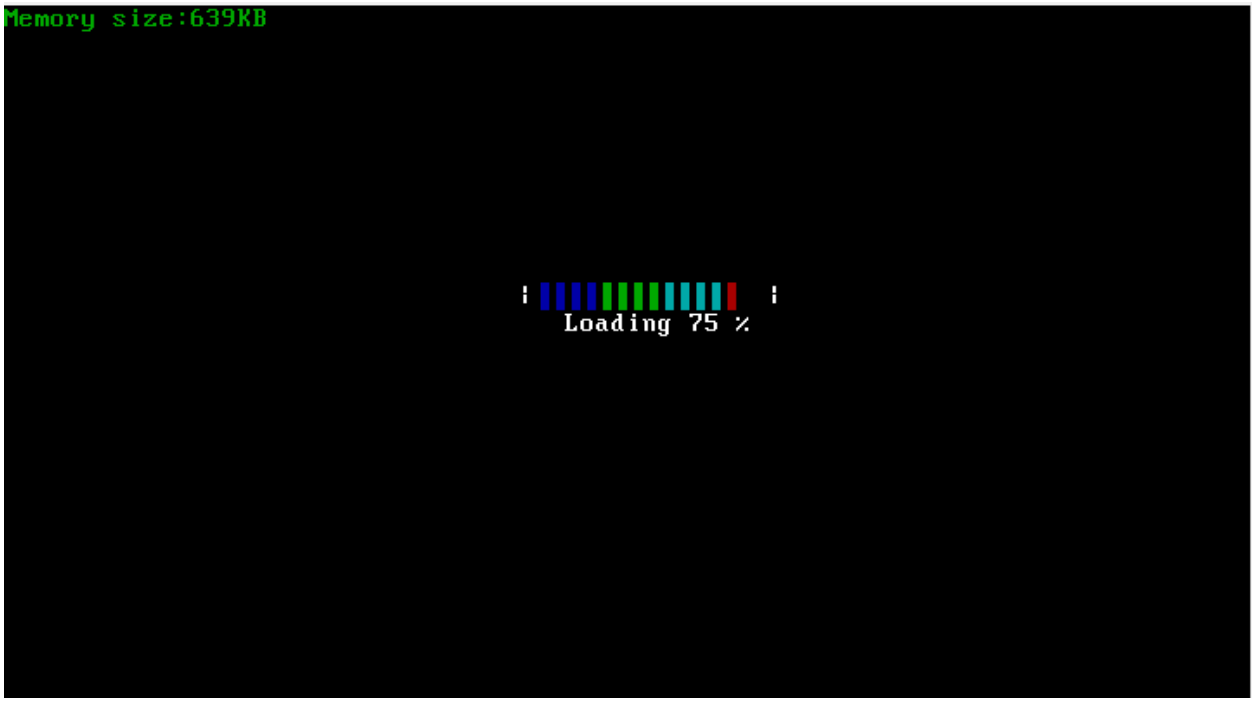

Fig. 4 Încărcarea kernel-ului

Dacă este depistată o eroare de disc, vedem următoarele.

Fig. 5 Eroare de citire

După încărcare controlul este transmis kernel-ului.

Fig. 5 Kernel-ul

Fig. 6 Funcția Calc

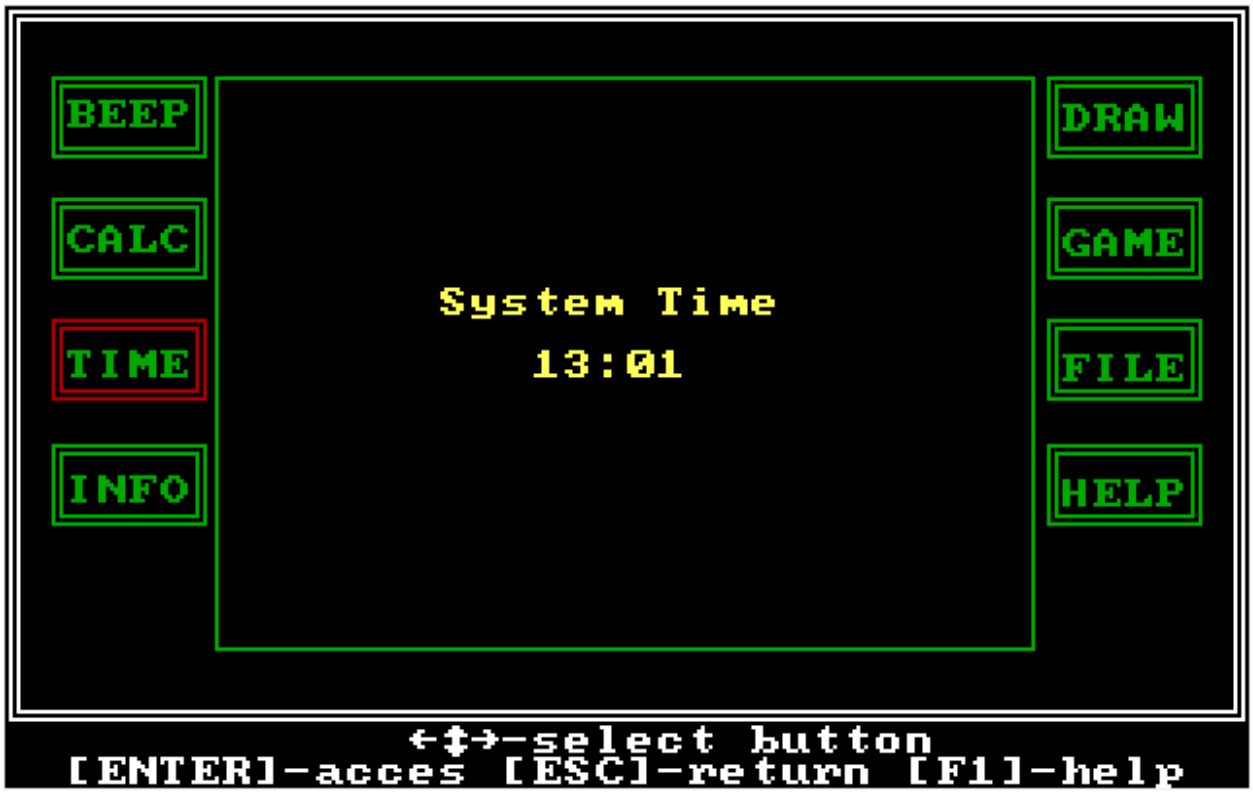

Fig. 7 Funcția Time

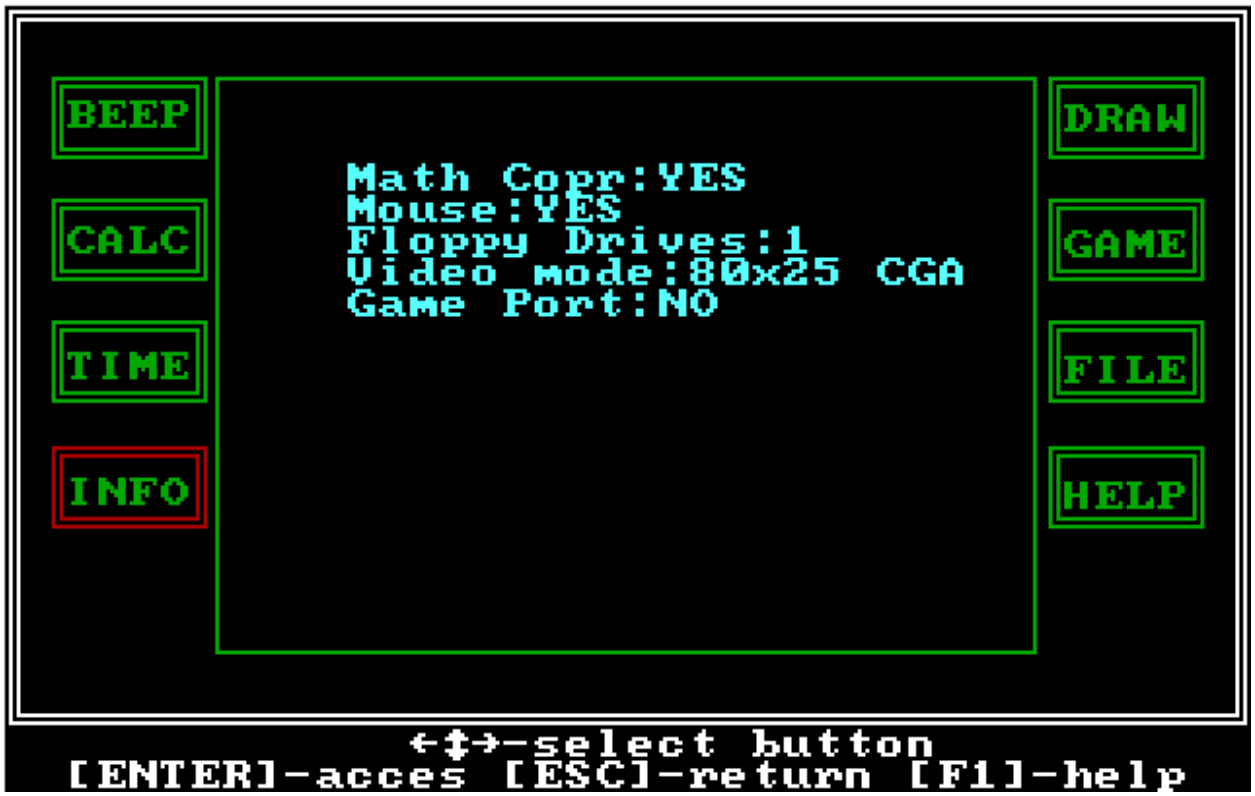

Fig. 8 Funcția Info

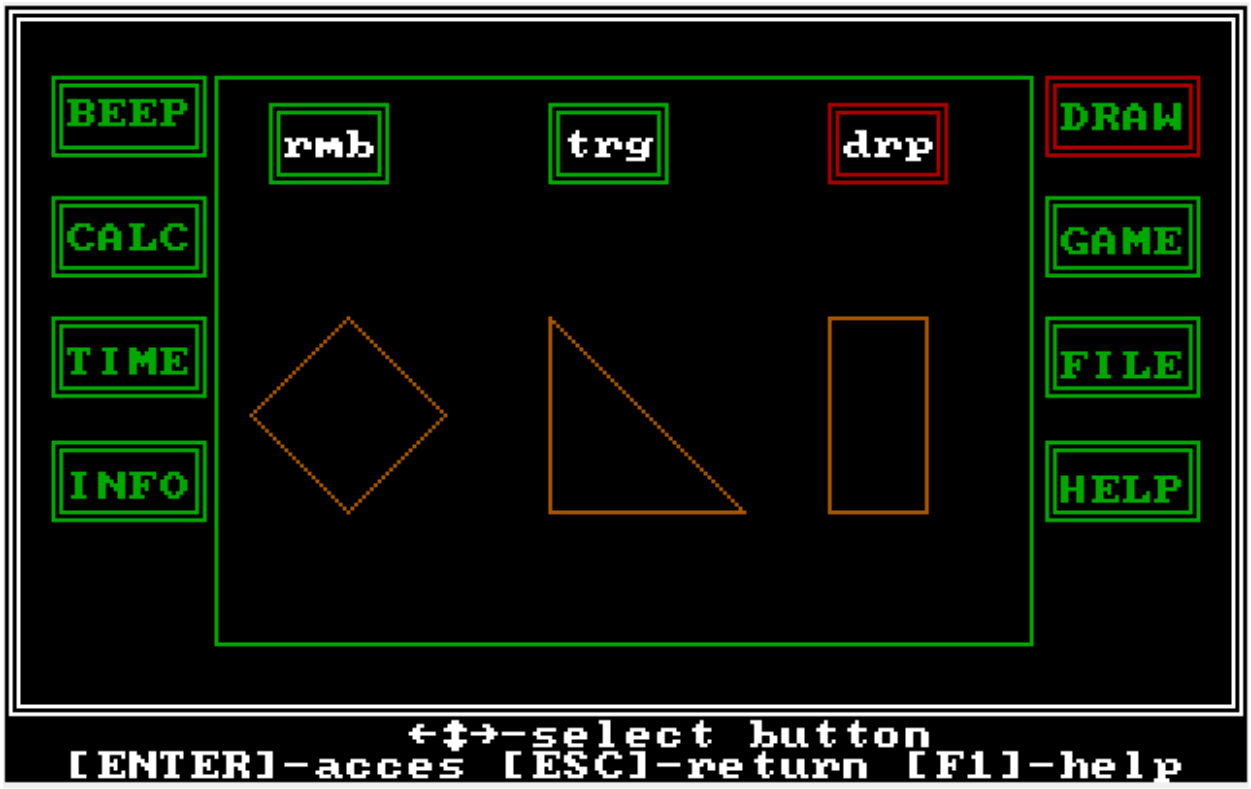

Fig. 9 Funcția Draw

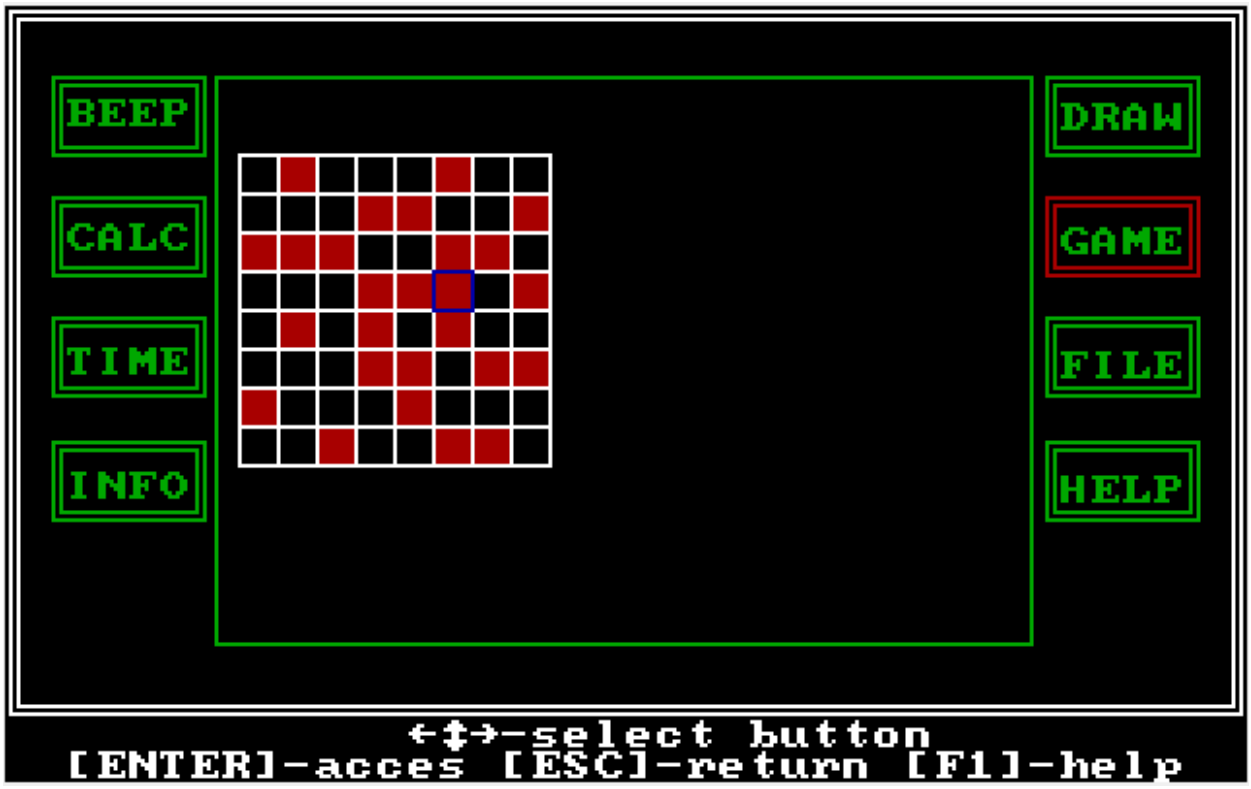

Fig. 10 Funcția Game

Concluzii

În urma efectuării lucrării de laborator am consolidat cunoștințele în lucrul cu softul EMU 8086 și cu particularitățile limbajului de asamblare pentru acest emulator. Am căpătat abilități practice în realizarea unui meniu grafic, am studiat întreruperile BIOS 10h, 11h, 12h, 13h, 15h, 16h , 1Ah și întreruperea DOS 21h și diverse funcții ale acestora, care le-am folosit pentru realizarea sarcinii de afișare dinamică a mesajelor, tratarea erorilor , citire de pe sectorul dischetei, conversiei datelor ș.a. . La fel am consolidat cunoștințele în lucrul cu programele în limbajul de asamblare, ce ocupă mai multe sectoare de disc și în gestionarea lor.

Bibliografie

1. dreamincode. [ASM] Load Kernel From Bootloader. [Resursă electronică].- Regim de acces : http://www.dreamincode.net/forums/topic/262898-asm-load-kernel-from-bootloader/
2. Asmcommunity. Looping in NASM. [Resursă electronică].- Regim de acces : http://www.asmcommunity.net/forums/topic/?id=30742
3. Nasm.us. The Netwide Assembler: NASM . [Resursă electronică].- Regim de acces: http://www.nasm.us/doc/nasmdoc4.html
4. Stackoverflow. Basic NASM bootstrap. [Resursă electronică].- Regim de acces : http://stackoverflow.com/questions/10853425/basic-nasm-bootstrap
5. Codenet. Функции BIOS - INT 10H: видео сервис. [Resursă electronică].- Regim de acces : http://www.codenet.ru/progr/dos/int_0009.php
6. Codenet. Функции BIOS - INT 13H: дисковый ввод-вывод. [Resursă electronică].- Regim de acces : http://www.codenet.ru/progr/dos/int_0012.php
7. Codenet. Функции BIOS - INT 12H: размер используемой памяти. [Resursă electronică].- Regim de acces : http://www.codenet.ru/progr/dos/int_0011.php
8. courses.engr.illinois.edu. 11.2 VGA Mode 13h Graphics. [Resursă electronică].- Regim de acces : https://courses.engr.illinois.edu/ece390/books/labmanual/graphics-mode13h.html
9. cyberforum.ru. Генерация звука – Assembler. [Resursă electronică].- Regim de acces : http://www.cyberforum.ru/assembler/thread54835.html
10. Codenet. Функции BIOS - INT 11H: проверка оборудования. [Resursă electronică].- Regim de acces : http://www.codenet.ru/progr/dos/int_0010.php
11. Codenet. Функции BIOS - INT 1aH: ввод-вывод для времени. [Resursă electronică].- Regim de acces : http://www.codenet.ru/progr/dos/int_0019.php

Anexă

Anexa A. Codul sursă a bootloader-ului

```
#make_boot#

ORG 0x7C00     ;Origin, tell the assembler that where the code will be in memory after it is been loaded

jmp main       ; Salt la punctul de intrare in program

WriteString:

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

;;   Procedura de afisare a unui sir de caractere      ;;

;;   Functia 0x13 a intreruperii BIOS 0x10             ;;

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

mov ah, 13h    ; Codul functiei de afisare a unui sir

mov al, 1      ; Modul de afisare

mov bh, 0x00  ; Numarul paginii

int 0x10          ; Call video interrupt

ret

WriteMessages:

push ax            ; ? Salvam continutul

push bx            ; | registrelor in starea

push cx            ; | in care se aflau inainte de

push dx            ; L apelarea functiei

mov bp, str1           ; salvam offsetul mesajului in bp (Hello World!)

mov bl, 0ch           ; setam codul culorii

mov cx, word[str1_len]   ; incarcam in cx numarul de caractere a sirului

mov dh, 10             ; rindul din care incepe afisarea

mov dl, [str1_x]         ; coloana din care incepe afisarea
```

```
call WriteString
mov bp, str2            ; salvam offsetul mesajului in bp (This is )
mov bl, 0ch             ; setam codul culorii
mov cx, word[str2_len]  ; incarcam in cx numarul de caractere a sirului
mov dh, 11              ; rindul din care incepe afisarea
mov dl, [str2_x]        ; coloana din care incepe afisarea
call WriteString
pop dx          ; ? Restabilim
pop cx          ; | continutul
pop bx          ; | registrelor
pop ax          ; L in starea inainte de apelare
ret
Delay:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;      Procedura de efectuare a unei retineri        ;;
;;      Functia 0x86 a intreruperii BIOS 0x15         ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
push ax         ;Salvam
push cx         ;continutul
push dx         ;registrelor
mov ah, 0x86    ; Codul functiei
mov cx, 0x3     ; Durata pauzei in microsecunde (in cx bitul superior)
mov dx, 0x20    ; in dx bitul inferior (800 milisecunde)
int 0x15        ; Apelarea intreruperii BIOS
```

```
pop dx          ; Restabilim
pop cx          ; continutul
pop ax          ; registrelor
ret
beep:
mov ah, 02h
mov dl, 07h
int 21h
ret
ReadSector:
push ax
push bx
push cx
push dx
call beep
call Delay
pop dx
pop cx
pop bx
pop ax
mov ah, 02h
mov al, 1
mov ch, 0
mov cl, 9
mov dh, 0
```

```
;mov dl, 0
mov bx, 0800h
mov es, bx
mov bx, 0
int 13h
; integrity check:
;cmp     es:[0000],0E9h   ; first byte of kernel must be 0E9 (jmp).
;je      integrity_check_ok
integrity_check_ok:
; pass control to kernel:
jmp     0800h:0000h
ret

;;;;;;;;;;;;;;;;;;;;;;;;FUNCTIA PRINCIPALA;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
main:
call WriteMessages
mov ah , 00h
int 16h
cmp al, 1bh
jnz here
 int 19h
here:
cmp al,13
je ReadSector
```

```
;Data

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

;;        Definima:  4 mesaje in 8 offset-uri          ;;

;;            Lungimea  fiecarui  string               ;;

;;        Coordonata pe x a fiecarui  string           ;;

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


str1 db 'Restart-ESC', 0

str1_len  dw 11

str1_x db 34

str2 db 'Load kernel-ENTER'  ,0

str2_len  dw 17

str2_x db 34

TIMES  510 - ($ - $$) db 0     ;Fill  the rest of sector with  0

DW 0xAA55                      ;Add boot signature  at the end of bootloader
```

Anexa B. Codul sursă al kernel-ului

```
#make_bin#

mov ax,0600h
mov ds, ax
mov xf, 1
mov yf, 1
mov xw, 80
mov yw, 18
jmp main

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; PROCEDURI ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;---------------------------------------------------------------------------
DrawButton PROC
;Deseneaza un buton la coordonata specificata cu ajutorul functiei OCh/INT 10h;
;coordonata indica coltul stinga-sus a butonului
;Primeste: coordonata X si Y
;Returneaza: Nimic
;---------------------------------------------------------------------------
   push ax      ;salvam continutul registrelor
   push bx
   push cx
   push dx

   mov ax, x
   mov x_int, ax
   add x_int, 2
   mov ax, y
```

```
mov y_int, ax
add y_int, 2

mov ah, draw_pixel          ;introducem codul functiei
mov al, default_color       ;introducem culoarea de desenare(verde)

mov cx, button_height   ;setam contorul egal cu inaltimea butonului
l:
push cx             ;salvam contorul
mov cx, x           ;plasam coordonata x
mov dx, y           ;plasam coordonata y
int 10h             ;apela de intrerupere
inc y               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop l

mov cx, button_width   ;setam contorul egal cu inaltimea butonului
l1:
push cx             ;salvam contorul
mov cx, x           ;plasam coordonata x
mov dx, y           ;plasam coordonata y
int 10h             ;apela de intrerupere
inc x               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop l1

mov cx, button_height   ;setam contorul egal cu inaltimea butonului
l2:
push cx             ;salvam contorul
mov cx, x           ;plasam coordonata x
mov dx, y           ;plasam coordonata y
int 10h             ;apela de intrerupere
```

```
dec y               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop l2

mov cx, button_width   ;setam contorul egal cu inaltimea butonului
l3:
push cx             ;salvam contorul
mov cx, x           ;plasam coordonata x
mov dx, y           ;plasam coordonata y
int 10h             ;apela de intrerupere
dec x               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop l3

mov cx, button_height_int   ;setam contorul egal cu inaltimea butonului
i:
push cx             ;salvam contorul
mov cx, x_int           ;plasam coordonata x
mov dx, y_int           ;plasam coordonata y
int 10h             ;apela de intrerupere
inc y_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop i

mov cx, button_width_int   ;setam contorul egal cu inaltimea butonului
i1:
push cx             ;salvam contorul
mov cx, x_int           ;plasam coordonata x
mov dx, y_int           ;plasam coordonata y
int 10h             ;apela de intrerupere
inc x_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
```

```
    loop i1

    mov cx, button_height_int  ;setam contorul egal cu inaltimea butonului
    i2:
    push cx             ;salvam contorul
    mov cx, x_int             ;plasam coordonata x
    mov dx, y_int             ;plasam coordonata y
    int 10h             ;apela de intrerupere
    dec y_int               ;ne deplasam cu un pixel in jos
    pop cx              ;restabilim contorul
    loop i2

    mov cx, button_width_int  ;setam contorul egal cu inaltimea butonului
    i3:
    push cx             ;salvam contorul
    mov cx, x_int             ;plasam coordonata x
    mov dx, y_int             ;plasam coordonata y
    int 10h             ;apela de intrerupere
    dec x_int               ;ne deplasam cu un pixel in jos
    pop cx              ;restabilim contorul
    loop i3

    pop dx           ;rstabilim continutul registrelor
    pop cx
    pop bx
    pop ax
    RET
DrawButton ENDP

DrawSelectedButton PROC
;Deseneaza un buton la coordonata specificata cu ajutorul functiei OCh/INT 10h;
;coordonata indica coltul stinga-sus a butonului
```

```
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
   push ax      ;salvam continutul registrelor
   push bx
   push cx
   push dx

   mov ax, x
   mov x_int, ax
   add x_int, 2
   mov ax, y
   mov y_int, ax
   add y_int, 2

   mov ah, draw_pixel          ;introducem codul functiei
   mov al, select_color        ;introducem culoarea de desenare(verde)

   mov cx, button_height   ;setam contorul egal cu inaltimea butonului
   ls:
   push cx              ;salvam contorul
   mov cx, x            ;plasam coordonata x
   mov dx, y            ;plasam coordonata y
   int 10h              ;apela de intrerupere
   inc y                ;ne deplasam cu un pixel in jos
   pop cx               ;restabilim contorul
   loop ls

   mov cx, button_width   ;setam contorul egal cu inaltimea butonului
   ls1:
   push cx              ;salvam contorul
   mov cx, x            ;plasam coordonata x
```

```
mov dx, y             ;plasam coordonata y
int 10h               ;apela de intrerupere
inc x                 ;ne deplasam cu un pixel in jos
pop cx                ;restabilim contorul
loop ls1

mov cx, button_height   ;setam contorul egal cu inaltimea butonului
ls2:
push cx               ;salvam contorul
mov cx, x             ;plasam coordonata x
mov dx, y             ;plasam coordonata y
int 10h               ;apela de intrerupere
dec y                 ;ne deplasam cu un pixel in jos
pop cx                ;restabilim contorul
loop ls2

mov cx, button_width   ;setam contorul egal cu inaltimea butonului
ls3:
push cx               ;salvam contorul
mov cx, x             ;plasam coordonata x
mov dx, y             ;plasam coordonata y
int 10h               ;apela de intrerupere
dec x                 ;ne deplasam cu un pixel in jos
pop cx                ;restabilim contorul
loop ls3

mov cx, button_height_int   ;setam contorul egal cu inaltimea butonului
is:
push cx               ;salvam contorul
mov cx, x_int             ;plasam coordonata x
mov dx, y_int             ;plasam coordonata y
int 10h               ;apela de intrerupere
```

```
inc y_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop is

mov cx, button_width_int   ;setam contorul egal cu inaltimea butonului
is1:
push cx             ;salvam contorul
mov cx, x_int           ;plasam coordonata x
mov dx, y_int           ;plasam coordonata y
int 10h             ;apela de intrerupere
inc x_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop is1

mov cx, button_height_int   ;setam contorul egal cu inaltimea butonului
is2:
push cx             ;salvam contorul
mov cx, x_int           ;plasam coordonata x
mov dx, y_int           ;plasam coordonata y
int 10h             ;apela de intrerupere
dec y_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop is2

mov cx, button_width_int   ;setam contorul egal cu inaltimea butonului
is3:
push cx             ;salvam contorul
mov cx, x_int           ;plasam coordonata x
mov dx, y_int           ;plasam coordonata y
int 10h             ;apela de intrerupere
dec x_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
```

```
    loop is3

    pop dx              ;rstabilim continutul registrelor
    pop cx
    pop bx
   pop ax
    RET
DrawSelectedButton ENDP

;--------------------------------------------------------------------------
DrawWindow PROC
;Deseneaza fereastra de preview cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;--------------------------------------------------------------------------
    push ax      ;salvam continutul registrelor
    push bx
    push cx
    push dx



    mov ah, draw_pixel          ;introducem codul functiei
    mov al, default_color       ;introducem culoarea de desenare(verde)

    mov cx, window_height   ;setam contorul egal cu inaltimea butonului
    w:
    push cx             ;salvam contorul
    mov cx, xw              ;plasam coordonata x
    mov dx, yw              ;plasam coordonata y
    int 10h             ;apela de intrerupere
    inc yw              ;ne deplasam cu un pixel in jos
    pop cx              ;restabilim contorul
```

```
loop w

mov cx, window_width  ;setam contorul egal cu inaltimea butonului
w1:
push cx          ;salvam contorul
mov cx, xw       ;plasam coordonata x
mov dx, yw       ;plasam coordonata y
int 10h          ;apela de intrerupere
inc xw           ;ne deplasam cu un pixel in jos
pop cx           ;restabilim contorul
loop w1

mov cx, window_height  ;setam contorul egal cu inaltimea butonului
w2:
push cx          ;salvam contorul
mov cx, xw       ;plasam coordonata x
mov dx, yw       ;plasam coordonata y
int 10h          ;apela de intrerupere
dec yw           ;ne deplasam cu un pixel in jos
pop cx           ;restabilim contorul
loop w2

mov cx, window_width  ;setam contorul egal cu inaltimea butonului
w3:
push cx          ;salvam contorul
mov cx, xw       ;plasam coordonata x
mov dx, yw       ;plasam coordonata y
int 10h          ;apela de intrerupere
dec xw           ;ne deplasam cu un pixel in jos
pop cx           ;restabilim contorul
loop w3
```

```
   pop dx            ;rstabilim continutul registrelor
   pop cx
   pop bx
   pop ax
   RET
DrawWindow ENDP


;--------------------------------------------------------------------------
DrawFrame PROC
;Deseneaza fereastra de preview cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;--------------------------------------------------------------------------
   push ax      ;salvam continutul registrelor
   push bx
   push cx
   push dx


   mov ah, draw_pixel          ;introducem codul functiei
   mov al, 15       ;introducem culoarea de desenare(verde)

   mov cx, frame_height   ;setam contorul egal cu inaltimea butonului
   f:
   push cx             ;salvam contorul
   mov cx, xf             ;plasam coordonata x
   mov dx, yf             ;plasam coordonata y
   int 10h             ;apela de intrerupere
   inc yf              ;ne deplasam cu un pixel in jos
   pop cx              ;restabilim contorul
```

```
loop f

mov cx, frame_width   ;setam contorul egal cu inaltimea butonului
f1:
push cx               ;salvam contorul
mov cx, xf            ;plasam coordonata x
mov dx, yf            ;plasam coordonata y
int 10h               ;apela de intrerupere
inc xf                ;ne deplasam cu un pixel in jos
pop cx                ;restabilim contorul
loop f1

mov xf, 159
mov yf, 1

mov cx, frame_width   ;setam contorul egal cu inaltimea butonului
f3:
push cx               ;salvam contorul
mov cx, xf            ;plasam coordonata x
mov dx, yf            ;plasam coordonata y
int 10h               ;apela de intrerupere
dec xf                ;ne deplasam cu un pixel in jos
pop cx                ;restabilim contorul
loop f3


mov xf, 159
mov yf, 182
mov cx, frame_width   ;setam contorul egal cu inaltimea butonului
f4:
push cx               ;salvam contorul
mov cx, xf            ;plasam coordonata x
```

```
mov dx, yf              ;plasam coordonata y
int 10h                 ;apela de intrerupere
inc xf                  ;ne deplasam cu un pixel in jos
pop cx                  ;restabilim contorul
loop f4


mov cx, frame_height    ;setam contorul egal cu inaltimea butonului
f5:
push cx                 ;salvam contorul
mov cx, xf              ;plasam coordonata x
mov dx, yf              ;plasam coordonata y
int 10h                 ;apela de intrerupere
dec yf                  ;ne deplasam cu un pixel in jos
pop cx                  ;restabilim contorul
loop f5


mov cx, frame_width     ;setam contorul egal cu inaltimea butonului
f6:
push cx                 ;salvam contorul
mov cx, xf              ;plasam coordonata x
mov dx, yf              ;plasam coordonata y
int 10h                 ;apela de intrerupere
dec xf                  ;ne deplasam cu un pixel in jos
pop cx                  ;restabilim contorul
loop f6

mov xf, 3
mov yf, 3
mov cx, frame_height-4  ;setam contorul egal cu inaltimea butonului
fi:
```

```
push cx              ;salvam contorul
mov cx, xf           ;plasam coordonata x
mov dx, yf           ;plasam coordonata y
int 10h              ;apela de intrerupere
inc yf               ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop fi

mov cx, frame_width-2   ;setam contorul egal cu inaltimea butonului
fi1:
push cx              ;salvam contorul
mov cx, xf           ;plasam coordonata x
mov dx, yf           ;plasam coordonata y
int 10h              ;apela de intrerupere
inc xf               ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop fi1

mov xf, 159
mov yf, 3

mov cx, frame_width-2   ;setam contorul egal cu inaltimea butonului
fi3:
push cx              ;salvam contorul
mov cx, xf           ;plasam coordonata x
mov dx, yf           ;plasam coordonata y
int 10h              ;apela de intrerupere
dec xf               ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop fi3
```

```
mov xf, 159
mov yf, 180
mov cx, frame_width-2   ;setam contorul egal cu inaltimea butonului
fi4:
push cx              ;salvam contorul
mov cx, xf           ;plasam coordonata x
mov dx, yf           ;plasam coordonata y
int 10h              ;apela de intrerupere
inc xf               ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop fi4

mov cx, frame_height-4   ;setam contorul egal cu inaltimea butonului
fi5:
push cx              ;salvam contorul
mov cx, xf           ;plasam coordonata x
mov dx, yf           ;plasam coordonata y
int 10h              ;apela de intrerupere
dec yf               ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop fi5

mov cx, frame_width-2   ;setam contorul egal cu inaltimea butonului
fi6:
push cx              ;salvam contorul
mov cx, xf           ;plasam coordonata x
mov dx, yf           ;plasam coordonata y
int 10h              ;apela de intrerupere
dec xf               ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
```

```
   loop fi6

   pop dx          ;rstabilim continutul registrelor
   pop cx
   pop bx
   pop ax
   RET
DrawFrame ENDP

WriteString:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   Procedura de afisare a unui sir de caractere     ;;
;;    Functia  0x13 a intreruperii BIOS 0x10          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1     ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10          ; Call video interrupt
ret


WriteMessages:

push ax           ; ? Salvam continutul
push bx           ; | registrelor in starea
push cx           ; | in care se aflau inainte de
push dx           ; L apelarea functiei

mov bl, 15
mov bp, offset message2        ; salvam offsetul mesajului in bp
mov cx, 26  ; incarcam in cx numarul de caractere a sirului
mov dh, 24          ; rindul din care incepe afisarea
```

```
mov dl, 6        ; coloana din care incepe afisarea
call WriteString

mov bl, 2
mov bp, offset button            ; salvam offsetul mesajului in bp
mov cx, 4  ; incarcam in cx numarul de caractere a sirului
mov dh, 3              ; rindul din care incepe afisarea
mov dl, 5        ; coloana din care incepe afisarea
call WriteString

mov bl, 2
mov bp, offset button1           ; salvam offsetul mesajului in bp
mov cx, 4  ; incarcam in cx numarul de caractere a sirului
mov dh, 7              ; rindul din care incepe afisarea
mov dl, 5        ; coloana din care incepe afisarea
call WriteString

mov bl, 2
mov bp, offset button2           ; salvam offsetul mesajului in bp
mov cx, 4  ; incarcam in cx numarul de caractere a sirului
mov dh, 11             ; rindul din care incepe afisarea
mov dl, 5        ; coloana din care incepe afisarea
call WriteString

mov bl, 2
mov bp, offset button3           ; salvam offsetul mesajului in bp
mov cx, 4  ; incarcam in cx numarul de caractere a sirului
mov dh, 15             ; rindul din care incepe afisarea
mov dl, 5        ; coloana din care incepe afisarea
call WriteString

mov bl, 2
```

```
mov bp, offset button4          ; salvam offsetul mesajului in bp
mov cx, 4 ; incarcam in cx numarul de caractere a sirului
mov dh, 19              ; rindul din care incepe afisarea
mov dl, 5       ; coloana din care incepe afisarea
call WriteString


mov bl, 15
mov bp, offset message1             ; salvam offsetul mesajului in bp
mov cx, 17 ; incarcam in cx numarul de caractere a sirului
mov dh, 23              ; rindul din care incepe afisarea
mov dl, 13        ; coloana din care incepe afisarea
call WriteString

mov ah, 01h           ;ascundem
mov cx, 2607h          ;cursorul
int 10h

pop dx              ; ? Restabilim
pop cx              ; | continutul
pop bx              ; | registrelor
pop ax              ; L in starea inainte de apelare
ret

RedrawButton PROC
   push ax              ; ? Salvam continutul
   push bx              ; | registrelor in starea
   push cx              ; | in care se aflau inainte de
   push dx              ; L apelarea functiei

   mov cx, 36
   mov dx, 18
```

```
mov ah, 0dh
int 10h
cmp al, 4
jne rb
mov x, 36
mov y, 18
call DrawButton
jmp end
rb:
mov cx, 36
mov dx, 49
mov ah, 0dh
int 10h
cmp al, 4
jne rb1
mov x, 36
mov y, 49
call DrawButton
jmp end
rb1:
mov cx, 36
mov dx, 80
mov ah, 0dh
int 10h
cmp al, 4
jne rb2
mov x, 36
mov y, 80
call DrawButton
jmp end
rb2:
mov cx, 36
```

```
    mov dx, 112
    mov ah, 0dh
    int 10h
    cmp al, 4
    jne rb3
    mov x, 36
    mov y, 112
    call DrawButton
    jmp end
    rb3:
    mov cx, 36
    mov dx, 144
    mov ah, 0dh
    int 10h
    cmp al, 4
    mov x, 36
    mov y, 144
    call DrawButton

    end:
    pop dx          ; ? Restabilim
    pop cx          ; | continutul
    pop bx          ; | registrelor
    pop ax          ; L in starea inainte  de apelare


    ret
RedrawButton  ENDP


CheckButton  PROC
mov ah , 00h
```

```
int 16h
cmp ah, 50h
jne a
inc location
jmp check
a:
cmp ah, 48h
jne o
dec location
jmp check
o:
cmp al, 13
jne b
call SelectFunction
ret
CheckButton ENDP

SelectFunction PROC
   push ax          ; ? Salvam continutul
   push bx          ; | registrelor in starea
   push cx          ; | in care se aflau inainte de
   push dx          ; L apelarea functiei


   cmp location, 0
   je b
   cmp location, 1
   jne sf
   call Morse
   jmp end1
   sf:
   cmp location, 2
```

```
    jne sf1
    call Calc
    jmp end1
    sf1:
    cmp location, 3
    jne sf2
    call Time
    jmp end1
    sf2:
    cmp location, 4
    jne sf3
    call Info
    jmp end1
    sf3:
    call Draw


    end1:
    pop dx          ; ? Restabilim
    pop cx          ; | continutul
    pop bx          ; | registrelor
    pop ax          ; L in starea inainte de apelare
    ret
SelectFunction ENDP


Draw PROC

mov ah, 02h
mov al, 3
mov ch, 0
mov cl, 8
```

```
mov dh, 0
mov dl, 0
mov bx, 0300h
mov es, bx
mov bx, 0
int 13h

jmp     0300h:0000h
ret
Draw ENDP

Time  PROC

mov ah, 02h
mov al, 1
mov ch, 0
mov cl, 11
mov dh, 0
mov dl, 0
mov bx, 0400h
mov es, bx
mov bx, 0
int 13h

jmp     0400h:0000h
ret
Time  ENDP

Calc PROC

mov ah, 02h
mov al, 3
```

```
mov ch, 0
mov cl, 12
mov dh, 0
mov dl, 0
mov bx, 0500h
mov es, bx
mov bx, 0
int 13h

jmp     0500h:0000h
ret
Calc ENDP

Morse PROC

mov ah, 02h
mov al, 1
mov ch, 0
mov cl, 15
mov dh, 0
mov dl, 0
mov bx, 0200h
mov es, bx
mov bx, 0
int 13h

jmp     0200h:0000h
ret
Morse ENDP

Info PROC
```

```
mov ah, 02h
mov al, 2
mov ch, 0
mov cl, 16
mov dh, 0
mov dl, 0
mov bx, 0900h
mov es, bx
mov bx, 0
int 13h

jmp     0900h:0000h
ret
Info  ENDP


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  FUNCTIA PRINCIPALA ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main:
xor ah, ah       ;initializam
mov al, 13h      ;modul
int 10h          ;grafic   320x200 pixeli

mov x, 36
mov y, 18
call DrawButton

mov x, 36
mov y, 49
call DrawButton

mov x, 36
```

```
mov y, 80
call DrawButton

mov x, 36
mov y, 112
call DrawButton

mov x, 36
mov y, 144
call DrawButton

call DrawWindow
call WriteMessages
call DrawFrame

;-----------------Verificam tasta accesata-----------------------
b:
call CheckButton


check:
cmp location,0
jnle q
mov location, 5
q:
cmp location,5
jna m
mov location, 1
m:
cmp location,1
jne s
call RedrawButton
```

```
mov x, 36
mov y, 18
call DrawSelectedButton
s:
cmp location,2
jne t
call RedrawButton
mov x, 36
mov y, 49
call DrawSelectedButton
t:
cmp location,3
jne u
call RedrawButton
mov x, 36
mov y, 80
call DrawSelectedButton
u:
cmp location,4
jne v
call RedrawButton
mov x, 36
mov y, 112
call DrawSelectedButton
v:
cmp location,5
jne b
call RedrawButton
mov x, 36
mov y, 144
call DrawSelectedButton
jmp b
```

```
jmp $           ;ciclu infinit
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; DATE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
default_color equ 2     ;culoarea verde
select_color equ 4      ;culoarea rosie
draw_pixel equ 0ch      ;codul functiei de desenare a unui pixel
button_width equ 39     ;latimea butonului
button_height equ 20    ;inaltimea butonului
window_width equ 210     ;latimea butonului
window_height equ 146   ;inaltimea butonului
frame_width equ 159     ;latimea butonului
frame_height equ 181        ;inaltimea butonului
button_width_int equ 35     ;latimea butonului
button_height_int equ 16    ;inaltimea butonului
frame_color equ 15
x dw 0
y dw 0
xf dw 1
yf dw 1
xw dw 80
yw dw 18
x_int dw 0
y_int dw 0
message1 db 27,18, 26,'-select button',0
message2 db '[ENTER]-acces [ESC]-return'
button db 'BEEP', 0
button1 db 'CALC', 0
button2 db 'TIME', 0
button3 db 'INFO', 0
button4 db 'DRAW', 0
```

location db 0

db 512 - ($-$$) DUP(0)

Anexa C. Codul sursă a loader-ului

```
#make_bin#
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
#BP=0000h#
mov ax,0800h
mov ds, ax
jmp main
Delay1:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;      Procedura de efectuare a unei retineri       ;;
;;      Functia 0x86 a intreruperii BIOS 0x15        ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
push ax         ;Salvam
push bx
push cx         ;continutul
push dx         ;registrelor

mov ah, 0x86     ; Codul functiei
mov cx, 0x13      ; Durata pauzei in microsecunde (in cx bitul superior)
mov dx, 0x88     ; in dx bitul inferior  (800 milisecunde)
```

```
int 0x15          ; Apelarea intreruperii BIOS


pop dx            ; Restabilim
pop cx            ; continutul
pop bx
pop ax            ; registrelor
ret


DrawBar:
push ax
push bx
push cx
push dx
mov bp, offset bar          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 15          ; setam codul culorii
mov cx, 1  ; incarcam in cx numarul de caractere a sirului
mov dh, 10              ; rindul din care incepe afisarea
mov dl, 33        ; coloana din care incepe afisarea
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1       ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10            ; Call video interrupt
mov cx, 1  ; incarcam in cx numarul de caractere a sirului
mov dh, 10              ; rindul din care incepe afisarea
mov dl, 49        ; coloana din care incepe afisarea
```

```
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1      ; Modul de afisare
mov bh, 0x00   ; Numarul paginii
int 0x10       ; Call video interrupt
pop dx
pop cx
pop bx
pop ax
ret

DrawBox:
push ax
push bx
push cx
push dx
mov bp, offset symbol      ; salvam offsetul mesajului in bp (Hello World!)
mov bl, ds:[color]         ; setam codul culorii
mov cx, 1  ; incarcam in cx numarul de caractere a sirului
mov dh, 10         ; rindul din care incepe afisarea
mov dl, [x]        ; coloana din care incepe afisarea
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1      ; Modul de afisare
mov bh, 0x00   ; Numarul paginii
int 0x10       ; Call video interrupt
mov ah, 01h        ;ascundem
```

```
mov cx, 2607h           ;cursorul
int 10h
pop dx
pop cx
pop bx
pop ax
ret

DrawPercent:
push ax
push bx
push cx
push dx
mov bp, offset percent          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 15              ; setam codul culorii
mov cx, 1  ; incarcam in cx numarul de caractere a sirului
mov dh, 11              ; rindul din care incepe afisarea
mov dl, 47         ; coloana din care incepe afisarea
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1       ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10              ; Call video interrupt
pop dx
pop cx
pop bx
```

```
pop ax
ret
ReadSector:
mov ah, 02h
mov al, 1
mov ch, 0
mov cl, 9
mov dh, 0
mov bx, 0600h
mov es, bx
mov bx, 0
int 13h
; pass control to kernel:
jmp     0600h:0000h
ret
CheckSector:
push ax
push bx
push cx
push dx
mov ah, 02h
mov al, 1
mov ch, 0
mov cl, 9
mov dh, 0
```

```
mov bx, 0400h
mov es, bx
mov bx, 0
int 13h
mov bx, 0800h
mov es, bx
cmp al, 0
jne c
call Error
b:
mov ah , 00h
int 16h
cmp al, 1bh
jne b
int 19h
c:
pop dx
pop cx
pop bx
pop ax
ret

Error:
push ax
push bx
```

```
push cx
push dx
mov bp, offset err_msg          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 4           ; setam codul culorii
mov cx, 29  ; incarcam in cx numarul de caractere a sirului
mov dh, 8           ; rindul din care incepe afisarea
mov dl, 28        ; coloana din care incepe afisarea
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1       ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10            ; Call video interrupt
mov bp, offset reb          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 4           ; setam codul culorii
mov cx, 36  ; incarcam in cx numarul de caractere a sirului
mov dh, 13          ; rindul din care incepe afisarea
mov dl, 24        ; coloana din care incepe afisarea
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1       ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10            ; Call video interrupt
pop dx
pop cx
pop bx
pop ax
ret
```

```
Message:
push ax
push bx
push cx
push dx
mov bp, offset msg          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 15          ; setam codul culorii
mov cx, 9  ; incarcam in cx numarul de caractere a sirului
mov dh, 11          ; rindul din care incepe afisarea
mov dl, 36       ; coloana din care incepe afisarea
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1       ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10          ; Call video interrupt
pop dx
pop cx
pop bx
pop ax
ret

LoadMess:
push ax
push bx
push cx
```

```
push dx
;mov bp, offset load25+si          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 15          ; setam codul culorii
mov cx,2  ; incarcam in cx numarul de caractere a sirului
mov dh, 11          ; rindul din care incepe afisarea
mov dl, 44        ; coloana din care incepe afisarea
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1      ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10          ; Call video interrupt
pop dx
pop cx
pop bx
pop ax
ret
LoadComplete:
push ax
push bx
push cx
push dx
mov bp, offset load100          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 15          ; setam codul culorii
mov cx, 3  ; incarcam in cx numarul de caractere a sirului
mov dh, 11          ; rindul din care incepe afisarea
mov dl, 44        ; coloana din care incepe afisarea
```

```
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1      ; Modul de afisare
mov bh, 0x00   ; Numarul paginii
int 0x10             ; Call video interrupt
mov ah, 01h          ;ascundem
mov cx, 2607h        ;cursorul
int 10h
pop dx
pop cx
pop bx
pop ax
ret
PrintRAM:
push ax
push bx
push cx
push dx
mov bp, offset ram_mess        ; salvam offsetul mesajului in bp
mov bl, 2                      ; setam codul culorii
mov cx, 17                     ; incarcam in cx numarul de caractere a sirului
mov dh, 0                      ; rindul din care incepe afisarea
mov dl, 0                      ; coloana din care incepe afisarea
mov ah, 13h                    ; Codul functiei de afisare a unui sir
mov al, 1                      ; Modul de afisare
mov bh, 0x00                   ; Numarul paginii
```

```
int 0x10                    ; Call video interrupt
mov ah, 01h                 ;ascundem
mov cx, 2607h               ;cursorul
int 10h
pop dx
pop cx
pop bx
pop ax
ret

DisplayRAM:
push ax
push bx
push cx
push dx
int 12h
mov bl,10
div bl
or ah, 30h
mov di,offset result
mov byte ptr[di+2],ah
xor ah, ah
div bl
or ah, 30h
mov di,offset result
```

```
mov  byte ptr[di+1],ah
xor ah, ah
div bl
or ah, 30h
mov  byte ptr[di],ah
xor ah, ah
mov bp, offset  result            ; salvam  offsetul  mesajului  in bp (Hello  World!)
mov bl, 2            ; setam  codul culorii
mov cx, 3     ; incarcam  in cx numarul  de caractere a sirului
mov dh, 0              ; rindul  din care incepe  afisarea
mov dl, 12         ; coloana  din care incepe  afisarea
mov ah, 13h    ; Codul functiei  de afisare  a unui  sir
mov al, 1       ; Modul de afisare
mov bh, 0x00  ; Numarul  paginii
int 0x10              ; Call video  interrupt
mov ah, 01h            ;ascundem
mov cx, 2607h           ;cursorul
int 10h
pop dx
pop cx
pop bx
pop ax
ret


main:
```

```
call PrintRAM
call DisplayRAM
call DrawBar          ;desanam cite o bara la inceput  si sfirsit
call Message          ;afisam  mesajul  "Loading"
call DrawPercent      ;afisam  mesajul  "%"
mov bp, offset  load25    ;incarcam  in  bp adresa ce contine  valoarea  25
mov cx, 15            ; setam contorul  ciclului(desenam  15 elemente)
l:
call CheckSector
push bp               ;salvam  continutul  bp pentru a incrementa  valoarea acestuia
ulterior(incrementam  offsetul  25, 50, 75)
call DrawBox          ;desenam  un element
call Delay1
pop bp                ;restabilim  continutul  bp
mov ax, cx            ;incarcam  contorul  in ax pentru impartirea  ulterioara
mov bl, 4
div bl                ;impartim  la 4
cmp ah, 0             ;verificam  daca restul inmpartirii  este 0
jnz et2               ;daca da incrementam  culoarea  si offsetul  pentru valoarea
procentelor,  daca nu- salt la et2
inc color
call LoadMess
add bp, 3
et2:
inc x                 ;incrementam  coordonata x pentru box
loop l
```

```
call LoadComplete               ;afisam mesajul "Loading 100%"
call Delay1
call ReadSector
jmp $
bar db 124
x db 34
y db 10
symbol db 222, 0
msg db 'Loading ',48,0
percent db 37, 0
load25 db 50,53,0
load50 db 53,48,0
load75 db 55,53,0
load100 db 49,48,48,0
ram_mess db 'Memory size:   KB',0
color db 1
result db 0,0,0,0,0
err_msg db 'ERROR: Floppy disk not found!',0
reb db 'Enter disk and press [ESC] to reboot',0


db 512 - ($ - $$) DUP(0)        ;Fill the rest of sector with 0
;DW 0xAA55                      ;Add boot signature at the end of bootloader
```

Anexa D. Codul sursă a funcției BEEP

```
#make_bin#
mov ax, 0200h
mov ds, ax

jmp main



Delay:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;        Procedura de efectuare a unei retineri           ;;
;;        Functia 0x86 a intreruperii BIOS 0x15            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
push ax          ;Salvam
push cx          ;continutul
push dx          ;registrelor

mov ah, 0x86       ; Codul functiei
mov cx, 0x3        ; Durata pauzei in microsecunde (in cx bitul superior)
mov dx, 0x20       ; in dx bitul inferior   (800 milisecunde)
int 0x15           ; Apelarea intreruperii BIOS

pop dx           ; Restabilim
pop cx           ; continutul
pop ax           ; registrelor
ret

main:
mov cx, 3

a:
push cx
mov cx, 3
l:
mov ah, 02h
```

```
mov dl, 07h
int 21h
loop l
call Delay
mov cx, 3
m:

mov ah, 02h
mov dl, 07h
int 21h
call Delay
loop m

mov cx, 3
n:
mov ah, 02h
mov dl, 07h
int 21h
loop n

pop cx
call Delay
loop a

p:
mov ah , 00h
int 16h
cmp al, 1bh
jne p
mov ax, 0600h
mov es, ax
jmp 0600h:0000h

jmp $
```

Anexa E. Codul sursă a funcției CALC

```
#make_bin#
mov ax, 0500h
mov ds, ax
mov ax, 0
mov bx, 0
mov cx, 0
mov dx, 0
jmp main

Validate PROC
   cmp al, 30h
   jnl v
   cmp al, 2dh
   je endv
   cmp al, 0dh
   je endv
   mov bp, offset error
   mov bl, 0ch          ; setam codul culorii
   mov cx, err_len  ; incarcam in cx numarul de caractere a sirului
   mov dh, 8            ; rindul din care incepe afisarea
   mov dl, 12       ; coloana din care incepe afisarea
   call WriteString
   mov ah, 0
   int 16h
   call ClearScreen
   jmp main

   v:
   cmp al, 39h
   jna endv
   mov bp, offset error
   mov bl, 0ch          ; setam codul culorii
   mov cx, err_len  ; incarcam in cx numarul de caractere a sirului
   mov dh, 8            ; rindul din care incepe afisarea
```

```
    mov dl, 12        ; coloana din care incepe afisarea
    call WriteString
    mov ah, 0
    int 16h
    call ClearScreen
    jmp main

    endv:
    ret
Validate ENDP

ClearScreen PROC
    push ax           ; ? Salvam continutul
    push bx           ; | registrelor in starea
    push cx           ; | in care se aflau inainte de
    push dx           ; L apelarea functiei

    mov bp, offset clear
    mov bl, 00h       ; setam codul culorii
    mov cx, 17  ; incarcam in cx numarul de caractere a sirului
    mov dh, 8         ; rindul din care incepe afisarea
    mov dl, 12        ; coloana din care incepe afisarea
    call WriteString

    mov cx, 17  ; incarcam in cx numarul de caractere a sirului
    mov dh, 9         ; rindul din care incepe afisarea
    mov dl, 12        ; coloana din care incepe afisarea
    call WriteString

    mov cx,17  ; incarcam in cx numarul de caractere a sirului
    mov dh, 10        ; rindul din care incepe afisarea
    mov dl, 12        ; coloana din care incepe afisarea
    call WriteString

    mov cx, 18  ; incarcam in cx numarul de caractere a sirului
```

```
   mov dh, 11           ; rindul din care incepe afisarea
   mov dl, 12       ; coloana din care incepe afisarea
   call WriteString


   mov cx, 18  ; incarcam in cx numarul de caractere a sirului
   mov dh, 12           ; rindul din care incepe afisarea
   mov dl, 12       ; coloana din care incepe afisarea
   call WriteString

   pop dx           ; ? Restabilim
   pop cx           ; | continutul
   pop bx           ; | registrelor
   pop ax           ; L in starea inainte de apelare
   ret

ClearScreen ENDP

WriteString PROC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;    Procedura de afisare a unui sir de caractere      ;;
;;     Functia  0x13 a intreruperii BIOS 0x10           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1     ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10          ; Call video interrupt
ret
WriteString ENDP

WriteMessages PROC

push ax           ; ? Salvam continutul
push bx           ; | registrelor in starea
push cx           ; | in care se aflau inainte de
```

```
push dx                 ; L apelarea functiei


mov bl, 0ch             ; setam codul culorii
mov cx, msg1_len   ; incarcam  in cx numarul  de caractere a sirului
mov dh, 10              ; rindul  din care incepe  afisarea
mov dl, 12         ; coloana  din care incepe  afisarea
call WriteString




pop dx              ; ? Restabilim
pop cx              ; | continutul
pop bx              ; | registrelor
pop ax              ; L in starea inainte  de apelare
ret
WriteMessages  ENDP

WriteMessageB  PROC

push ax                 ; ? Salvam  continutul
push bx                 ; | registrelor  in starea
push cx                 ; | in care se aflau  inainte  de
push dx                 ; L apelarea functiei


mov bl, 0ch             ; setam codul culorii
mov cx, msg3_len   ; incarcam  in cx numarul  de caractere a sirului
mov dh, 11              ; rindul  din care incepe afisarea
mov dl, 12         ; coloana  din care incepe  afisarea
call WriteString
```

```
pop dx          ; ? Restabilim
pop cx          ; | continutul
pop bx          ; | registrelor
pop ax          ; L in starea inainte de apelare
ret
WriteMessageB ENDP

ReadA PROC

mov bp, offset msg1          ; salvam offsetul mesajului in bp (Hello World!)
call WriteMessages
mov cx, 4

l:
push cx
mov ah , 00h
int 16h
call Validate
cmp al, 0dh
je l2
cmp al, 2dh
je l6
cmp cx,4
jne l7
pop cx
dec cx
push cx
l7:
mov digit, al
xor digit, 30h
cmp cx, 3
jne l1
mov bl, 100
mov al, digit
```

```
mul bl
mov a, ax
l1:
cmp cx, 2
jne l3
mov bl, 10
mov al, digit
mul bl
xor ah, ah
add a, ax
l3:
cmp cx, 1
jne l4
xor ah, ah
mov al, digit
xor ah, ah
add a, ax
l4:

mov al, digit
or al, 30h
l6:
cmp cx, 4
jne l8
mov signa, al
l8:
mov ah, 09h
mov bh, 0
mov bl, 2
mov cx, 1
int 10h
mov ah, 2
mov bh, 0
mov dh, 10
mov dl, cursor_pos
```

```
int 10h
inc cursor_pos

pop cx
loop l
jmp end1

l2:
cmp cx, 2
jne l5
mov ax, a
mov bl, 100
div bl
mov a, ax
pop cx
l5:
cmp cx, 1
jne end1
mov ax, a
mov bl, 10
div bl
mov a, ax
pop cx
end1:
ret
ReadA ENDP

ReadB PROC
mov cursor_pos, 22
mov bp, offset msg3          ; salvam offsetul mesajului in bp (Hello World!)
call WriteMessageB
mov cx, 4

lb:
push cx
```

```
mov ah , 00h
int 16h
call Validate
cmp al, 0dh
je lb2
cmp al, 2dh
je lb6
cmp cx,4
jne lb7
pop cx
dec cx
push cx
lb7:
mov digit,  al
xor digit,  30h
cmp cx, 3
jne lb1
mov bl, 100
mov al, digit
mul bl
mov b, ax
lb1:
cmp cx, 2
jne lb3
mov bl, 10
mov al, digit
mul bl
xor ah, ah
add b, ax
lb3:
cmp cx, 1
jne lb4
xor ah, ah
mov al, digit
xor ah, ah
```

```
add b, ax
lb4:

mov al, digit
or al, 30h
lb6:
cmp cx, 4
jne lb8
mov signb, al
lb8:
mov ah, 09h
mov bh, 0
mov bl, 2
mov cx, 1
int 10h
mov ah, 2
mov bh, 0
mov dh, 11
mov dl, cursor_pos
int 10h
inc cursor_pos

pop cx
loop lb
jmp end1b

lb2:
cmp cx, 2
jne lb5
mov ax, b
mov bl, 100
div bl
mov b, ax
pop cx
lb5:
```

```
cmp cx, 1
jne end1
mov ax, b
mov bl, 10
div bl
mov b, ax
pop cx
end1b:
ret
ReadB ENDP

CheckSignA PROC
   cmp signa, 2dh
   jne c
   mov ax, 0
   sub ax, a
   mov a, ax

   c:
   ret
CheckSignA ENDP

CheckSignB PROC
   cmp signb, 2dh
   jne d
   mov ax, 0
   sub ax, b
   mov b, ax

   d:
   ret
CheckSignB ENDP


Operation PROC
```

```
s:
mov bp, offset  msg2
mov bl, 0ch             ; setam codul culorii
mov cx, msg2_len   ; incarcam  in cx numarul  de caractere a sirului
mov dh, 12              ; rindul  din care incepe  afisarea
mov dl, 12         ; coloana  din care incepe  afisarea
call WriteString
mov ah , 00h
int 16h
mov operation_s,  al


cmp al, 2bh  ;+
jne o
mov ah, 09h          ;afisam
mov bh, 0            ;simbolul
mov bl, 2
mov cx, 1            ;la ecran
int 10h
mov ax,a
add ax, b
pushf
pop dx
xor dl, 80h
cmp dl, 80h
ja o4
mov signr,  1
o4:
mov result,  ax
call DisplayResult
jmp endo


o:
cmp al, 2dh    ;-
```

```
jne o1
mov ah, 09h
mov bh, 0
mov bl, 2
mov cx, 1
int 10h
mov ax, a
sub ax, b
pushf
pop dx
xor dl, 80h
cmp dl, 80h
ja o3
mov signr, 1
o3:
mov result, ax
call DisplayResult
jmp endo



o1:
call ResultSign
cmp al, 2ah    ;*
jne o2
mov ah, 09h
mov bh, 0
mov bl, 2
mov cx, 1
int 10h
cmp a, 99
jng op1
inc nmul
op1:
cmp b, 99
```

```
 jng op2
 inc nmul
 op2:
 cmp nmul , 2
 jne op3
mov bp, offset error1
mov bl, 0ch             ; setam codul culorii
mov cx, err1_len  ; incarcam in cx numarul de caractere a sirului
mov dh, 8               ; rindul din care incepe afisarea
mov dl, 12        ; coloana din care incepe afisarea
call WriteString
mov ah, 0
int 16h
call ClearScreen
jmp main
 op3:
 mov ax, a
 imul b
; pushf
; pop dx
; xor dl, 80h
 ;cmp dl, 80h
; ja o5
; mov signr, 1
; o5:
 mov result, ax
 call DisplayResult
 jmp endo



 o2:
 call ResultSign
 cmp al, 2fh    ;/
 jne s
```

```
    mov ah, 09h
    mov bh, 0
    mov bl, 2
    mov cx, 1
    int 10h


    mov ax, a
    cwd
    mov bx, b
    idiv bx
    mov integer, ax
    mov remainder, dx
    call DisplayDivisionResult

     endo:
   ret
Operation ENDP

DisplayResult PROC
push ax
push bx
push cx
push dx


cmp signr, 1
jne dr
neg result

dr:
mov ax, result
xor dx, dx
mov bx,10
idiv bx
```

```
or dl, 30h
mov di,offset resultd
mov byte ptr[di+3],dl
xor dx, dx
div bx
or dl, 30h
mov di,offset resultd
mov byte ptr[di+2],dl
xor dx, dx
div bx
or dl, 30h
mov byte ptr[di+1],dl
xor dx, dx
div bx
or dl, 30h
mov byte ptr[di],dl

cmp signr, 1              ;daca rezultatul e negativ afisam semnul
jne dr4
mov bp, offset sign_mess          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 2           ; setam codul culorii
mov cx, 1     ; incarcam in cx numarul de caractere a sirului
mov dh, 13              ; rindul din care incepe afisarea
mov dl, 12         ; coloana din care incepe afisarea
mov ah, 13h    ; Codul functiei de afisare a unui sir
mov al, 1       ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10
dr4:


mov bp, offset resultd            ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 2           ; setam codul culorii
mov cx, result_length      ; incarcam in cx numarul de caractere a sirului
mov dh, 13              ; rindul din care incepe afisarea
```

```
mov dl, 13        ; coloana din care incepe afisarea
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1      ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10            ; Call video interrupt
mov ah, 01h          ;ascundem
mov cx, 2607h          ;cursorul
int 10h
pop dx
pop cx
pop bx
pop ax
ret
DisplayResult ENDP

DisplayDivisionResult PROC
push ax
push bx
push cx
push dx

cmp signr, 1

jne ddr9
neg integer

cmp signa, 2dh
jne ddr9
neg remainder
ddr9:

cmp signsum, 5ah
jne ddr10
neg remainder
ddr10:
```

```
mov ax, integer
mov cx, 2
ddr:
ddr1:
push cx
xor dx, dx
mov bx,10
idiv bx
or dl, 30h
mov di,offset resultd
mov byte ptr[di+3],dl
xor dx, dx
div bx
or dl, 30h
mov di,offset resultd
mov byte ptr[di+2],dl
xor dx, dx
div bx
or dl, 30h
mov byte ptr[di+1],dl
xor dx, dx
div bx
or dl, 30h
mov byte ptr[di],dl

cmp signr, 1 ;daca rezultatul e negativ afisam semnul
jne ddr4
mov bp, offset sign_mess ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 2 ; setam codul culorii
mov cx, 1 ; incarcam in cx numarul de caractere a sirului
mov dh, 13 ; rindul din care incepe afisarea
mov dl, 12 ; coloana din care incepe afisarea
mov ah, 13h ; Codul functiei de afisare a unui sir
mov al, 1 ; Modul de afisare
mov bh, 0x00 ; Numarul paginii
```

```
int 0x10
ddr4:


mov bp, offset resultd          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 2           ; setam codul culorii
mov cx, 4    ; incarcam in cx numarul de caractere a sirului
mov dh, 13              ; rindul din care incepe afisarea
mov dl, x        ; coloana din care incepe afisarea
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1       ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10            ; Call video interrupt
mov ax, remainder
mov x, 18
pop cx
loop ddr
mov ah, 01h           ;ascundem
mov cx, 2607h          ;cursorul
int 10h
pop dx
pop cx
pop bx
pop ax
ret
DisplayDivisionResult ENDP



ResultSign PROC
    pusha
    mov al, signa
   add al, signb
   mov signsum, al
   cmp al, 2dh
```

```
    jne rs
    mov signr,  1
    jmp exit
    rs:
    mov signr,  0

     exit:

    popa
    ret
ResultSign  ENDP




;=================================== MAIN
FUNCTION=============================================================
=====

main:
mov cursor_pos,  22
mov nmul,  0
mov signsum,  0
mov x, 13
mov signa,  0
mov signb,  0
mov signr,  0
mov signsum,  0
mov operation_s,  0
call ReadA
call CheckSignA
call ReadB
call CheckSignB
call Operation
mov ah, 0
```

```
int 16h
mov ax, 0600h
mov es, ax
jmp 0600h:0000h


;=============================================== DATA
================================================================


end:
jmp $


digit db 0
a dw 0
b dw 0
result dw 0
cursor_pos db 0
msg1 db 'Input a: ',0
msg1_len = 9
msg2 db 'Input operation: ',0
msg2_len = 17
msg3 db 'Input b: ',0
msg3_len = 9
signa db 0
signb db 0
signr db 0
error db 'Not an integer!',0
error1 db 'Overflow!',0
err_len = 15
err1_len = 9
resultd db 0,0,0,0,0
result_length dw 4
nmul db 0
sign_mess db '-',0
```

```
operation_s  db 0
x db 13   ; coloana  de unde  se afiseaza  rezultatul  impartirii
integer  dw 0
remainder  dw 0
signsum  db 0
clear db '000000000000000000',0
```

Anexa F. Codul sursă a funcției TIME

```
#make_bin#


mov ax, 0400h
mov ds, ax
jmp main

WriteString PROC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   Procedura de afisare a unui sir de caractere   ;;
;;   Functia 0x13 a intreruperii BIOS 0x10   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1     ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10      ; Call video interrupt
ret
WriteString ENDP

DisplayHour PROC
xor ax, ax
mov al, buffer_hour
ror ax, 4
xor bx, bx
mov bl, ah
rol bx, 4
xor al, 30h
xor bh, 30h
mov tetrad1, al
mov tetrad2, bh
mov bp, offset tetrad1      ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 0eh      ; setam codul culorii
mov cx, 1    ; incarcam in cx numarul de caractere a sirului
```

```
mov dh, 11              ; rindul  din care incepe afisarea
mov dl, 21        ; coloana  din care incepe  afisarea
call WriteString
mov bp, offset  tetrad2           ; salvam  offsetul  mesajului  in bp (Hello  World!)
mov cx, 1     ; incarcam  in cx numarul  de caractere a sirului
mov dh, 11              ; rindul  din care incepe afisarea
mov dl, 22        ; coloana  din care incepe  afisarea
call WriteString              ; Call video  interrupt
mov bp, offset  message           ; salvam  offsetul  mesajului  in bp (Hello  World!)
mov cx, 11     ; incarcam  in cx numarul  de caractere a sirului
mov dh, 9             ; rindul  din care incepe afisarea
mov dl, 18        ; coloana  din care incepe  afisarea
call WriteString              ; Call video  interrupt
mov ah, 01h           ;ascundem
mov cx, 2607h           ;cursorul
int 10h
ret
DisplayHour  ENDP

DisplayMinute  PROC
xor ax, ax
mov al, buffer_minute
ror ax, 4
xor bx, bx
mov bl, ah
rol bx, 4
xor al, 30h
xor bh, 30h
mov tetrad1, al
mov tetrad2, bh
mov bp, offset  tetrad1           ; salvam  offsetul  mesajului  in bp (Hello  World!)
mov bl, 0eh             ; setam codul culorii
mov cx, 1     ; incarcam  in cx numarul  de caractere a sirului
mov dh, 11              ; rindul  din care incepe afisarea
mov dl, 24        ; coloana  din care incepe  afisarea
```

```
call WriteString
mov bp, offset tetrad2          ; salvam offsetul mesajului in bp (Hello World!)
mov cx, 1    ; incarcam in cx numarul de caractere a sirului
mov dh, 11          ; rindul din care incepe afisarea
mov dl, 25       ; coloana din care incepe afisarea
call WriteString          ; Call video interrupt
mov ah, 01h          ;ascundem
mov cx, 2607h          ;cursorul
int 10h
ret
DisplayMinute ENDP



DisplaySign PROC
   mov bp, offset sign          ; salvam offsetul mesajului in bp (Hello World!)
   mov bl, 0eh          ; setam codul culorii
   mov cx, 1    ; incarcam in cx numarul de caractere a sirului
   mov dh, 11          ; rindul din care incepe afisarea
   mov dl, 23       ; coloana din care incepe afisarea
   call WriteString
   ret
DisplaySign ENDP



main:
mov ah,02h
int 1ah
mov buffer_hour, ch
mov buffer_minute, cl
call DisplayHour
call DisplaySign
call DisplayMinute
```

```
p:
mov ah , 00h
int 16h
cmp al, 1bh
jne p
mov ax, 0600h
mov es, ax
jmp 0600h:0000h

jmp $

buffer_hour  db 0
buffer_minute  db 0
tetrad1 db 0
tetrad2 db 0
sign  db ':',0
message  db 'System Time',  0

db 512 - ($ - $$) DUP(0)        ;Fill  the rest of sector with  0
```

Anexa G. Codul sursă a funcției INFO

```
#make_bin#
mov ax,0900h
mov ds, ax
jmp main

WriteString PROC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   Procedura de afisare a unui sir de caractere        ;;
;;   Functia 0x13 a intreruperii BIOS 0x10               ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1     ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10          ; Call video interrupt
ret
WriteString ENDP



Math PROC
  pusha
  mov ax, hardware
  and ax, math_mask
  jnz m
  mov bp, offset n
  jmp display
  m:
  mov bp, offset y

  display:
  mov bl, color          ; setam codul culorii
  mov cx, 3  ; incarcam in cx numarul de caractere a sirului
  mov dh, 5            ; rindul din care incepe afisarea
  mov dl, 21       ; coloana din care incepe afisarea
```

```
    call WriteString

    mov bp , offset  mess1
    mov bl, color            ; setam codul culorii
    mov cx, mess1_len   ; incarcam in cx numarul de caractere a sirului
    mov dh, 5              ; rindul din care incepe afisarea
    mov dl, 11         ; coloana din care incepe afisarea
    call WriteString

    popa
    ret
Math ENDP

Mouse PROC
    pusha
    mov ax, hardware
    and ax, mouse_mask
    jnz m1
    mov bp, offset  n
    jmp display1
    m1:
    mov bp, offset  y

    display1:
    mov bl, color            ; setam codul culorii
    mov cx, 3  ; incarcam in cx numarul de caractere a sirului
    mov dh, 6              ; rindul din care incepe afisarea
    mov dl, 17         ; coloana din care incepe afisarea
    call WriteString

    mov bp , offset  mess2
    mov bl, color            ; setam codul culorii
    mov cx, mess2_len   ; incarcam in cx numarul de caractere a sirului
    mov dh, 6              ; rindul din care incepe afisarea
    mov dl, 11         ; coloana din care incepe afisarea
```

```
    call WriteString

    popa
    ret
Mouse ENDP

Video PROC
    pusha
    mov ax, hardware
    and ax, video_mask
    mov videores,ax
    cmp videores, 00h
    jne v
    mov bp, offset video1
    mov cx, video1_len   ; incarcam in cx numarul de caractere a sirului
    mov dl, 11+mess3_len        ; coloana din care incepe afisarea
    jmp display3
    v:
    cmp videores, 10h
    jne v1
    mov bp, offset video2
    mov cx, video2_len   ; incarcam in cx numarul de caractere a sirului
    mov dl, 11+mess3_len        ; coloana din care incepe afisarea
    jmp display3
    v1:
    cmp videores, 20h
    jne v2
    mov bp, offset video3
    mov cx, video3_len   ; incarcam in cx numarul de caractere a sirului
    mov dl, 11+mess3_len        ; coloana din care incepe afisarea
    jmp display3
    v2:
    cmp videores, 30h
    jne v3
    mov bp, offset video4
```

```
    mov cx, video4_len   ; incarcam in cx numarul de caractere a sirului
    mov dl, 11+mess3_len          ; coloana din care incepe afisarea
    jmp display3
    v3:



    display3:
    mov bl, color              ; setam codul culorii

    mov dh, 8               ; rindul din care incepe afisarea

    call WriteString

    mov bp , offset mess3
    mov bl, color              ; setam codul culorii
    mov cx, mess3_len   ; incarcam in cx numarul de caractere a sirului
    mov dh, 8               ; rindul din care incepe afisarea
    mov dl, 11         ; coloana din care incepe afisarea
    call WriteString

    popa
    ret
Video ENDP

Floppy PROC
    pusha
    mov ax, hardware
    and ax, floppy_mask
    mov floppyres,ax
    cmp floppyres, 00h
    jne f
    mov bp, offset floppy1
    jmp display4
    f:
    cmp floppyres, 40h
```

```
    jne f1
    mov bp, offset  floppy2
    jmp display4
    f1:
    cmp floppyres,  80h
    jne f2
    mov bp, offset  floppy3
    jmp display4
    f2:
    cmp floppyres,  0C0h
    jne f3
    mov bp, offset  floppy4
    jmp display4
    f3:


    display4:
    mov bl, color           ; setam codul culorii
    mov cx, 1  ; incarcam  in cx numarul  de caractere a sirului
    mov dh, 7               ; rindul  din care incepe  afisarea
    mov dl, 11+mess4_len        ; coloana  din care incepe  afisarea
    call WriteString

    mov bp , offset  mess4
    mov bl, color           ; setam codul culorii
    mov cx, mess4_len   ; incarcam  in cx numarul  de caractere a sirului
    mov dh, 7               ; rindul  din care incepe  afisarea
    mov dl, 11        ; coloana  din care incepe  afisarea
    call WriteString

    popa
    ret
Floppy ENDP

Game PROC
```

```
    pusha
    mov ax, hardware
    and ax, game_mask
    jnz m2
    mov bp, offset  n
    jmp display2
    m2:
    mov bp, offset  y

    display2:
    mov bl, color             ; setam codul culorii
    mov cx, 3  ; incarcam  in cx numarul  de caractere a sirului
    mov dh, 9               ; rindul  din care incepe  afisarea
    mov dl, 11+mess5_len          ; coloana  din care incepe  afisarea
    call WriteString

    mov bp , offset  mess5
    mov bl, color             ; setam codul culorii
    mov cx, mess5_len   ; incarcam  in cx numarul  de caractere a sirului
    mov dh, 9               ; rindul  din care incepe  afisarea
    mov dl, 11         ; coloana  din care incepe  afisarea
    call WriteString

    popa
    ret
Game ENDP




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



main:
mov al, 13h
```

```
int 10h
int 11h
mov hardware, ax
call Math
call Mouse
call Game
call Floppy
call Video
p:
mov ah , 00h
int 16h
cmp al, 1bh
jne p
mov ax, 0600h
mov es, ax
jmp 0600h:0000h
jmp $


;=================================================================
==================
hardware dw 0
math_mask dw 02h
mouse_mask dw 04h
video_mask dw 30h
floppy_mask dw 0c0h
game_mask dw 1000h
color = 11

y db 'YES',0
n db 'NO',0

video1 db 'EGA/VGA/PGA',0
video1_len = 11
video2 db '40x25 CGA',0
```

```
video2_len  = 9
video3  db '80x25 CGA',0
video3_len  = 9
video4  db '80x25 mono  text',0
video4_len  = 15
videores  dw 0

floppy1  db '1',0
floppy2  db '2',0
floppy3  db '3',0
floppy4  db '4',0
floppy_len  = 1
floppyres  dw 0


mess1  db 'Math Copr:',0
mess1_len  = 10
mess2  db 'Mouse:',0
mess2_len  = 6
mess3  db 'Video  mode:',0
mess3_len  = 11
mess4  db 'Floppy Drives:',0
mess4_len  = 14
mess5  db 'Game  Port:',0
mess5_len  = 10
```

Anexa H. Codul sursă a funcției DRAW

```
#make_bin#

mov ax,0300h
mov ds, ax
jmp main

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; PROCEDURI ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ClearScreen:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;      Procedura de curatare a ecranului           ;;
;;      Functia 0x06 a intreruperii BIOS 0x10         ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

push ax          ; Ă Salvam continutul
push bx          ; | registrelor in starea
push cx          ; | in care se aflau inainte de
push dx          ; L apelarea functiei

mov ah, 0x06       ; Codul functiei
mov al, 0          ; Numarul de rinduri care vor fi scroll-ate (al=0 -curata ecranul)
mov bh, 0x00       ; Setarile grafice (culoarea fundalului pentru liniile curatate)
mov ch, 7          ; Rindul din care incepe curatarea
mov cl, 12         ; Coloana din care incepe curatarea
mov dh, 24         ; Rindul in care sfirseste curatarea
mov dl, 79         ; Coloana in care sfirseste curatarea
int 0x10           ; Apelarea intreruperii BIOS

pop dx           ; Ă Restabilim
pop cx           ; | continutul
pop bx           ; | registrelor
pop ax           ; L in starea inainte de apelare
ret
```

```
;-------------------------------------------------------------------------
DrawButton PROC
;Deseneaza un buton la coordonata specificata cu ajutorul functiei OCh/INT 10h;
;coordonata indica coltul stinga-sus a butonului
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
   push ax      ;salvam continutul registrelor
   push bx
   push cx
   push dx

   mov ax, x
   mov x_int, ax
   add x_int, 2
   mov ax, y
   mov y_int, ax
   add y_int, 2

   mov ah, draw_pixel          ;introducem codul functiei
   mov al, default_color       ;introducem culoarea de desenare(verde)

   mov cx, button_height   ;setam contorul egal cu inaltimea butonului
   l:
   push cx              ;salvam contorul
   mov cx, x            ;plasam coordonata x
   mov dx, y            ;plasam coordonata y
   int 10h              ;apela de intrerupere
   inc y                ;ne deplasam cu un pixel in jos
   pop cx               ;restabilim contorul
   loop l

   mov cx, button_width   ;setam contorul egal cu inaltimea butonului
```

```
l1:
push cx              ;salvam contorul
mov cx, x            ;plasam coordonata x
mov dx, y            ;plasam coordonata y
int 10h              ;apela de intrerupere
inc x                ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop l1

mov cx, button_height   ;setam contorul egal cu inaltimea butonului
l2:
push cx              ;salvam contorul
mov cx, x            ;plasam coordonata x
mov dx, y            ;plasam coordonata y
int 10h              ;apela de intrerupere
dec y                ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop l2

mov cx, button_width   ;setam contorul egal cu inaltimea butonului
l3:
push cx              ;salvam contorul
mov cx, x            ;plasam coordonata x
mov dx, y            ;plasam coordonata y
int 10h              ;apela de intrerupere
dec x                ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop l3

mov cx, button_height_int   ;setam contorul egal cu inaltimea butonului
i:
push cx              ;salvam contorul
mov cx, x_int            ;plasam coordonata x
mov dx, y_int            ;plasam coordonata y
int 10h              ;apela de intrerupere
```

```
inc y_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop i

mov cx, button_width_int   ;setam contorul egal cu inaltimea butonului
i1:
push cx             ;salvam contorul
mov cx, x_int           ;plasam coordonata x
mov dx, y_int           ;plasam coordonata y
int 10h             ;apela de intrerupere
inc x_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop i1

mov cx, button_height_int   ;setam contorul egal cu inaltimea butonului
i2:
push cx             ;salvam contorul
mov cx, x_int           ;plasam coordonata x
mov dx, y_int           ;plasam coordonata y
int 10h             ;apela de intrerupere
dec y_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop i2

mov cx, button_width_int   ;setam contorul egal cu inaltimea butonului
i3:
push cx             ;salvam contorul
mov cx, x_int           ;plasam coordonata x
mov dx, y_int           ;plasam coordonata y
int 10h             ;apela de intrerupere
dec x_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop i3

pop dx              ;rstabilim continutul registrelor
```

```
    pop cx
    pop bx
   pop ax
    RET
DrawButton  ENDP

DrawSelectedButton  PROC
;Deseneaza  un buton la coordonata specificata  cu ajutorul functiei  OCh/INT 10h;
;coordonata  indica  coltul  stinga-sus  a butonului
;Primeste:  coordonata  X si Y
;Returneaza:  Nimic
;--------------------------------------------------------------------------
    push ax       ;salvam  continutul  registrelor
    push bx
    push cx
    push dx

    mov ax,  x
    mov x_int,  ax
    add x_int,  2
    mov ax,  y
    mov y_int,  ax
    add y_int,  2

    mov ah, draw_pixel           ;introducem  codul functiei
    mov al, select_color         ;introducem  culoarea  de desenare(verde)

    mov cx, button_height   ;setam contorul  egal cu inaltimea  butonului
    ls:
    push cx               ;salvam  contorul
    mov cx,  x             ;plasam  coordonata  x
    mov dx,  y             ;plasam  coordonata  y
    int  10h              ;apela  de intrerupere
    inc  y                ;ne deplasam  cu un pixel  in jos
    pop cx                ;restabilim  contorul
```

```
loop ls

mov cx, button_width  ;setam contorul egal cu inaltimea butonului
ls1:
push cx               ;salvam contorul
mov cx, x             ;plasam coordonata x
mov dx, y             ;plasam coordonata y
int 10h               ;apela de intrerupere
inc x                 ;ne deplasam cu un pixel in jos
pop cx                ;restabilim contorul
loop ls1

mov cx, button_height  ;setam contorul egal cu inaltimea butonului
ls2:
push cx               ;salvam contorul
mov cx, x             ;plasam coordonata x
mov dx, y             ;plasam coordonata y
int 10h               ;apela de intrerupere
dec y                 ;ne deplasam cu un pixel in jos
pop cx                ;restabilim contorul
loop ls2

mov cx, button_width  ;setam contorul egal cu inaltimea butonului
ls3:
push cx               ;salvam contorul
mov cx, x             ;plasam coordonata x
mov dx, y             ;plasam coordonata y
int 10h               ;apela de intrerupere
dec x                 ;ne deplasam cu un pixel in jos
pop cx                ;restabilim contorul
loop ls3

mov cx, button_height_int  ;setam contorul egal cu inaltimea butonului
is:
push cx               ;salvam contorul
```

```
mov cx, x_int            ;plasam coordonata x
mov dx, y_int            ;plasam coordonata y
int 10h             ;apela de intrerupere
inc y_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop is

mov cx, button_width_int  ;setam contorul egal cu inaltimea butonului
is1:
push cx             ;salvam contorul
mov cx, x_int            ;plasam coordonata x
mov dx, y_int            ;plasam coordonata y
int 10h             ;apela de intrerupere
inc x_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop is1

mov cx, button_height_int  ;setam contorul egal cu inaltimea butonului
is2:
push cx             ;salvam contorul
mov cx, x_int            ;plasam coordonata x
mov dx, y_int            ;plasam coordonata y
int 10h             ;apela de intrerupere
dec y_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop is2

mov cx, button_width_int  ;setam contorul egal cu inaltimea butonului
is3:
push cx             ;salvam contorul
mov cx, x_int            ;plasam coordonata x
mov dx, y_int            ;plasam coordonata y
int 10h             ;apela de intrerupere
dec x_int               ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
```

```
   loop is3

   pop dx          ;rstabilim continutul registrelor
   pop cx
   pop bx
  pop ax
   RET
DrawSelectedButton ENDP




WriteString:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   Procedura de afisare a unui sir de caractere       ;;
;;    Functia  0x13 a intreruperii BIOS 0x10            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1      ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10            ; Call video interrupt
ret


WriteMessages:

push ax              ; ? Salvam continutul
push bx              ; | registrelor in starea
push cx              ; | in care se aflau inainte de
push dx              ; L apelarea functiei

mov bl, 15
mov bp, offset message          ; salvam offsetul mesajului in bp
mov cx, 3         ; incarcam in cx numarul de caractere a sirului
mov dh, 4            ; rindul din care incepe afisarea
```

```
mov dl, 12        ; coloana din care incepe afisarea
call WriteString

mov bl, 15
mov bp, offset message1          ; salvam offsetul mesajului in bp
mov cx, 3         ; incarcam in cx numarul de caractere a sirului
mov dh, 4            ; rindul din care incepe afisarea
mov dl, 21        ; coloana din care incepe afisarea
call WriteString

mov bl, 15
mov bp, offset message2          ; salvam offsetul mesajului in bp
mov cx, 3         ; incarcam in cx numarul de caractere a sirului
mov dh, 4            ; rindul din care incepe afisarea
mov dl, 30        ; coloana din care incepe afisarea
call WriteString



pop dx            ; ? Restabilim
pop cx            ; | continutul
pop bx            ; | registrelor
pop ax            ; L in starea inainte de apelare
ret

RedrawButton PROC
   push ax              ; ? Salvam continutul
   push bx              ; | registrelor in starea
   push cx              ; | in care se aflau inainte de
   push dx              ; L apelarea functiei

   mov cx, 92
   mov dx, 25
   mov ah, 0dh
   int 10h
```

```
    cmp al, 4
    jne rb
    mov x, 92
    mov y, 25
    call DrawButton
    jmp end
    rb:
    mov cx, 164
    mov dx, 25
    mov ah, 0dh
    int 10h
    cmp al, 4
    jne rb1
    mov x, 164
    mov y, 25
    call DrawButton
    jmp end
    rb1:
    mov cx, 236
    mov dx, 25
    mov ah, 0dh
    int 10h
    cmp al, 4
    mov x, 236
    mov y, 25
    call DrawButton

    end:
    pop dx          ; ? Restabilim
    pop cx          ; | continutul
    pop bx          ; | registrelor
    pop ax          ; L in starea inainte  de apelare

    ret
RedrawButton  ENDP
```

```
CheckButton PROC
mov ah , 00h
int 16h
cmp ah, 4dh
jne a
inc location
jmp check
a:
cmp ah, 4bh
jne o
dec location
jmp check
o:
cmp al, 1bh
jne p
mov ax, 0600h
mov es, ax
jmp 0600h:0000h
p:
cmp al, 13
jne b
call SelectFunction
ret
CheckButton ENDP

SelectFunction PROC
   push ax          ; ? Salvam continutul
   push bx          ; | registrelor in starea
   push cx          ; | in care se aflau inainte de
   push dx          ; L apelarea functiei


   cmp location, 0
```

```
    je b
    cmp location, 1
    jne sf
    call Romb
    jmp end1
    sf:
    cmp location, 2
    jne sf1
    call Triangle
    jmp end1
    sf1:
    cmp location, 3
    jne sf2
    call Rectangle
    jmp end1
    sf2:
    cmp location, 4
    jne sf3

    jmp end1
    sf3:



    end1:
    pop dx          ; ? Restabilim
    pop cx          ; | continutul
    pop bx          ; | registrelor
    pop ax          ; L in starea inainte de apelare
    ret
SelectFunction ENDP


;-------------------------------------------------------------------------
Triangle PROC
```

```
;Deseneaza deseneaza un triunghi dreptunghic cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
   push ax      ;salvam continutul registrelor
   push bx
   push cx
   push dx

   mov xw, 170          ;plasam coordonata x
   mov yw, 80           ;plasam coordonata y
   mov ah, draw_pixel         ;introducem codul functiei
   mov al, 6            ;introducem culoarea de desenare(verde)

   mov cx, 50  ;setam contorul egal cu inaltimea butonului
   w:
   push cx           ;salvam contorul
   mov cx, xw           ;plasam coordonata x
   mov dx, yw           ;plasam coordonata y
   int 10h           ;apela de intrerupere
   inc yw            ;ne deplasam cu un pixel in jos
   pop cx            ;restabilim contorul
   loop w

   mov cx, 50  ;setam contorul egal cu inaltimea butonului
   w1:
   push cx           ;salvam contorul
   mov cx, xw           ;plasam coordonata x
   mov dx, yw           ;plasam coordonata y
   int 10h           ;apela de intrerupere
   inc xw            ;ne deplasam cu un pixel in jos
   pop cx            ;restabilim contorul
   loop w1

   mov cx, 50  ;setam contorul egal cu inaltimea butonului
```

```
   w2:
   push cx              ;salvam contorul
   mov cx, xw           ;plasam coordonata x
   mov dx, yw           ;plasam coordonata y
   int 10h              ;apela de intrerupere
   dec yw               ;ne deplasam cu un pixel in jos
   dec xw
   pop cx               ;restabilim contorul
   loop w2


   pop dx           ;rstabilim continutul registrelor
   pop cx
   pop bx
   pop ax
   RET
Triangle ENDP

;-------------------------------------------------------------------------
Romb PROC
;Deseneaza deseneaza un triunghi dreptunghic cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
   push ax      ;salvam continutul registrelor
   push bx
   push cx
   push dx

   mov xw, 84           ;plasam coordonata x
   mov yw, 110          ;plasam coordonata y
   mov ah, draw_pixel       ;introducem codul functiei
   mov al, 6            ;introducem culoarea de desenare(verde)

   mov cx, 25  ;setam contorul egal cu inaltimea butonului
```

```
wo:
push cx             ;salvam contorul
mov cx, xw          ;plasam coordonata x
mov dx, yw          ;plasam coordonata y
int 10h             ;apela de intrerupere
inc yw              ;ne deplasam cu un pixel in jos
inc xw
pop cx              ;restabilim contorul
loop wo


mov cx, 25  ;setam contorul egal cu inaltimea butonului
wo1:
push cx             ;salvam contorul
mov cx, xw          ;plasam coordonata x
mov dx, yw          ;plasam coordonata y
int 10h             ;apela de intrerupere
inc xw              ;ne deplasam cu un pixel in jos
dec yw
pop cx              ;restabilim contorul
loop wo1

mov cx, 25  ;setam contorul egal cu inaltimea butonului
wo2:
push cx             ;salvam contorul
mov cx, xw          ;plasam coordonata x
mov dx, yw          ;plasam coordonata y
int 10h             ;apela de intrerupere
dec xw
dec yw
pop cx              ;restabilim contorul
loop wo2

mov cx, 25  ;setam contorul egal cu inaltimea butonului
wo3:
```

```
    push cx              ;salvam contorul
    mov cx, xw           ;plasam coordonata x
    mov dx, yw           ;plasam coordonata y
    int 10h              ;apela de intrerupere
    dec xw
    inc yw
    pop cx               ;restabilim contorul
    loop wo3


    pop dx           ;rstabilim continutul registrelor
    pop cx
    pop bx
    pop ax
    RET
Romb ENDP

;--------------------------------------------------------------------------
Rectangle PROC
;Deseneaza deseneaza un triunghi dreptunghic cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;--------------------------------------------------------------------------
    push ax      ;salvam continutul registrelor
    push bx
    push cx
    push dx

    mov xw, 240          ;plasam coordonata x
    mov yw, 80           ;plasam coordonata y
    mov ah, draw_pixel          ;introducem codul functiei
    mov al, 6            ;introducem culoarea de desenare(verde)

    mov cx, 50  ;setam contorul egal cu inaltimea butonului
    wo4:
```

```
push cx             ;salvam contorul
mov cx, xw          ;plasam coordonata x
mov dx, yw          ;plasam coordonata y
int 10h             ;apela de intrerupere
inc yw              ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop wo4


mov cx, 25  ;setam contorul egal cu inaltimea butonului
wo5:
push cx             ;salvam contorul
mov cx, xw          ;plasam coordonata x
mov dx, yw          ;plasam coordonata y
int 10h             ;apela de intrerupere
inc xw              ;ne deplasam cu un pixel in jos
pop cx              ;restabilim contorul
loop wo5

mov cx, 50  ;setam contorul egal cu inaltimea butonului
wo6:
push cx             ;salvam contorul
mov cx, xw          ;plasam coordonata x
mov dx, yw          ;plasam coordonata y
int 10h             ;apela de intrerupere
dec yw
pop cx              ;restabilim contorul
loop wo6

mov cx, 25  ;setam contorul egal cu inaltimea butonului
wo7:
push cx             ;salvam contorul
mov cx, xw          ;plasam coordonata x
mov dx, yw          ;plasam coordonata y
int 10h             ;apela de intrerupere
```

```
   dec xw
   pop cx            ;restabilim  contorul
   loop wo7



   pop dx          ;rstabilim  continutul  registrelor
   pop cx
   pop bx
   pop ax
   RET
Rectangle  ENDP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  FUNCTIA PRINCIPALA ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main:


mov x, 92
mov y, 25
call  DrawButton

mov x, 164
mov y, 25
call  DrawButton

mov x, 236
mov y, 25
call  DrawButton
call  WriteMessages



;-----------------Verificam  tasta accesata-----------------------
b:
call  CheckButton



check:
```

```
cmp location,0
jnle q
mov location, 3
q:
cmp location,3
jna m
mov location, 1
m:
cmp location,1
jne s
call RedrawButton
mov x, 92
mov y, 25
call DrawSelectedButton
s:
cmp location,2
jne t
call RedrawButton
mov x, 164
mov y, 25
call DrawSelectedButton
t:
cmp location,3
jne b
call RedrawButton
mov x, 236
mov y, 25
call DrawSelectedButton
jmp b
```

```
jmp $            ;ciclu infinit

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; DATE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
default_color equ 2      ;culoarea verde
select_color equ 4      ;culoarea rosie
draw_pixel equ 0ch      ;codul functiei de desenare a unui pixel
button_width equ 30     ;latimea butonului
button_height equ 20    ;inaltimea butonului
window_width equ 210     ;latimea butonului
window_height equ 146    ;inaltimea butonului
frame_width equ 159     ;latimea butonului
frame_height equ 181       ;inaltimea butonului
button_width_int equ 26     ;latimea butonului
button_height_int equ 16    ;inaltimea butonului
frame_color equ 15
x dw 0
y dw 0
xf dw 1
yf dw 1
xw dw 80
yw dw 18
x_int dw 0
y_int dw 0
message db 'rmb',0
message1 db 'trg', 0
message2 db 'drp', 0
location db 0

 db 512 - ($-$$) DUP(0)
```

ANEXA I. Codul sursă a funcției Game

```
#make_bin#

mov ax,0450h
mov ds, ax
jmp main

WriteString PROC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;    Procedura de afisare a unui sir de caractere       ;;
;;    Functia 0x13 a intreruperii BIOS 0x10              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1     ; Modul de afisare
mov bh, 0x00 ; Numarul paginii
int 0x10          ; Call video interrupt
ret
WriteString ENDP

DisplayHour PROC
xor ax, ax
mov al, buffer_hour1
ror ax, 4
xor bx, bx
mov bl, ah
rol bx, 4
xor al, 30h
xor bh, 30h
mov tetrad1, al
mov tetrad2, bh
mov bp, offset tetrad1         ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 0eh          ; setam codul culorii
mov cx, 1    ; incarcam in cx numarul de caractere a sirului
mov dh, 11           ; rindul din care incepe afisarea
mov dl, 21      ; coloana din care incepe afisarea
```

```
call WriteString
mov bp, offset tetrad2          ; salvam offsetul mesajului in bp (Hello World!)
mov cx, 1    ; incarcam in cx numarul de caractere a sirului
mov dh, 11          ; rindul din care incepe afisarea
mov dl, 22       ; coloana din care incepe afisarea
call WriteString          ; Call video interrupt
mov bp, offset message          ; salvam offsetul mesajului in bp (Hello World!)
mov cx, 9    ; incarcam in cx numarul de caractere a sirului
mov dh, 9          ; rindul din care incepe afisarea
mov dl, 20       ; coloana din care incepe afisarea
call WriteString
ret
DisplayHour ENDP

DisplayMinute PROC
xor ax, ax
mov al, buffer_minute1
ror ax, 4
xor bx, bx
mov bl, ah
rol bx, 4
xor al, 30h
xor bh, 30h
mov tetrad1, al
mov tetrad2, bh
mov bp, offset tetrad1          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 0eh          ; setam codul culorii
mov cx, 1    ; incarcam in cx numarul de caractere a sirului
mov dh, 11          ; rindul din care incepe afisarea
mov dl, 24       ; coloana din care incepe afisarea
call WriteString
mov bp, offset tetrad2          ; salvam offsetul mesajului in bp (Hello World!)
mov cx, 1    ; incarcam in cx numarul de caractere a sirului
mov dh, 11          ; rindul din care incepe afisarea
mov dl, 25       ; coloana din care incepe afisarea
```

```
call WriteString
ret
DisplayMinute ENDP

DisplaySecond PROC
xor ax, ax
mov al, buffer_second1
ror ax, 4
xor bx, bx
mov bl, ah
rol bx, 4
xor al, 30h
xor bh, 30h
mov tetrad1, al
mov tetrad2, bh
mov bp, offset tetrad1          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 0eh             ; setam codul culorii
mov cx, 1    ; incarcam in cx numarul de caractere a sirului
mov dh, 11              ; rindul din care incepe afisarea
mov dl, 27       ; coloana din care incepe afisarea
call WriteString
mov bp, offset tetrad2          ; salvam offsetul mesajului in bp (Hello World!)
mov cx, 1    ; incarcam in cx numarul de caractere a sirului
mov dh, 11              ; rindul din care incepe afisarea
mov dl, 28       ; coloana din care incepe afisarea
call WriteString              ; Call video interrupt
mov ah, 01h             ;ascundem
mov cx, 2607h           ;cursorul
int 10h
ret
DisplaySecond ENDP



DisplaySign PROC
```

```
    mov bp, offset points          ; salvam offsetul mesajului in bp (Hello World!)
    mov bl, 0eh            ; setam codul culorii
    mov cx, 1     ; incarcam in cx numarul de caractere a sirului
    mov dh, 11             ; rindul din care incepe afisarea
    mov dl, 23        ; coloana din care incepe afisarea
    call WriteString
    ret
DisplaySign ENDP

DisplaySign1 PROC
    mov bp, offset points          ; salvam offsetul mesajului in bp (Hello World!)
    mov bl, 0eh            ; setam codul culorii
    mov cx, 1     ; incarcam in cx numarul de caractere a sirului
    mov dh, 11             ; rindul din care incepe afisarea
    mov dl, 26        ; coloana din care incepe afisarea
    call WriteString
    ret
DisplaySign1 ENDP

CalcSecond PROC

    push ax
    push bx
    push cx
    push dx

    mov al, buffer_second1
    rol ax, 4
    rol al, 4


    mov bl, buffer_second
    rol bx, 4
    rol bl, 4
```

```
sub al, bl
cmp al, 0
jnl c
neg al
c:
sub ah, bh
cmp ah, 0
jnl d
neg ah
mov sign, 1
d:

cmp sign, 1
jne e
mov bh, 06h
mov bl, 00h
sub bl, al
cmp bl, 0
jnl g
neg bl
sub bl, 10
neg bl
dec bh
g:
sub bh, ah
ror bl, 4
ror bx, 4
mov buffer_second1, bl
jmp f
e:
ror al, 4
ror ax, 4
mov buffer_second1, al
f:
pop dx
```

```
    pop cx
    pop bx
    pop ax
    ret
CalcSecond ENDP

CalcMinute  PROC

    push ax
    push bx
    push cx
    push dx

    mov al, buffer_minute1
    rol ax, 4
    rol al, 4



    mov bl, buffer_minute
    rol bx, 4
    rol bl, 4

    sub al, bl
    cmp al, 0
    jnl c1
    neg al
    c1:
    sub ah, bh
    cmp ah, 0
    jnl d1
    neg ah
    mov sign,  1
    d1:

    cmp sign,  1
```

```
    jne e1
    mov bh, 06h
    mov bl, 00h
    sub bl, al
    cmp bl, 0
    jnl g1
    neg bl
    sub bl, 10
    neg bl
    dec bh
    g1:
    sub bh, ah
    ror bl, 4
    ror bx, 4
    mov buffer_minute1,  bl
    jmp f1
    e1:
    ror al, 4
    ror ax, 4
    mov buffer_minute1,  al
    f1:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
CalcMinute  ENDP

CalcHour PROC

    push ax
    push bx
    push cx
    push dx
```

```
mov al, buffer_hour1
rol ax, 4
rol al, 4


mov bl, buffer_hour
rol bx, 4
rol bl, 4

sub al, bl
cmp al, 0
jnl c2
neg al
c2:
sub ah, bh
cmp ah, 0
jnl d2
neg ah
mov sign,  1
d2:

cmp sign,  1
jne e2
mov bh, 06h
mov bl, 00h
sub bl, al
cmp bl, 0
jnl g2
neg bl
sub bl, 10
neg bl
dec bh
g2:
sub bh, ah
ror bl, 4
```

```
    ror bx, 4
    mov buffer_hour1,  bl
    jmp f2
    e2:
    ror al, 4
    ror ax, 4
    mov buffer_hour1,  al
    f2:
    pop dx
    pop cx
    pop bx
    pop ax
    ret
CalcHour ENDP

;-------------------------------------------------------------------------
DrawField PROC
;Deseneaza fereastra de preview cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
    push ax      ;salvam continutul registrelor
    push bx
    push cx
    push dx

    mov x, 60
    mov y, 38
    mov xv, 70
    mov yv, 38
    mov xh, 60
    mov yh, 48

    mov ah, 0ch          ;introducem codul functiei
    mov al, 15       ;introducem culoarea de desenare(verde)
```

```
mov cx, dimension  ;setam contorul egal cu inaltimea butonului
w:
push cx            ;salvam contorul
mov cx, x          ;plasam coordonata x
mov dx, y          ;plasam coordonata y
int 10h            ;apela de intrerupere
inc y              ;ne deplasam cu un pixel in jos
pop cx             ;restabilim contorul
loop w

mov cx, dimension  ;setam contorul egal cu inaltimea butonului
w1:
push cx            ;salvam contorul
mov cx, x          ;plasam coordonata x
mov dx, y          ;plasam coordonata y
int 10h            ;apela de intrerupere
inc x              ;ne deplasam cu un pixel in jos
pop cx             ;restabilim contorul
loop w1

mov cx, dimension  ;setam contorul egal cu inaltimea butonului
w2:
push cx            ;salvam contorul
mov cx, x          ;plasam coordonata x
mov dx, y          ;plasam coordonata y
int 10h            ;apela de intrerupere
dec y              ;ne deplasam cu un pixel in jos
pop cx             ;restabilim contorul
loop w2

mov cx, dimension  ;setam contorul egal cu inaltimea butonului
w3:
push cx            ;salvam contorul
mov cx, x          ;plasam coordonata x
```

```
    mov dx, y           ;plasam coordonata y
    int 10h             ;apela de intrerupere
    dec x               ;ne deplasam cu un pixel in jos
    pop cx              ;restabilim contorul
    loop w3

    mov cx, 7
    a:
    call DrawVertical
    add xv, 10
    mov yv, 38
    loop a

    mov cx, 7
    b1:
    call DrawHorizontal
    add yh, 10
    mov xh, 60
    loop b1

    pop dx          ;rstabilim continutul registrelor
    pop cx
    pop bx
    pop ax
    RET
DrawField ENDP

;-------------------------------------------------------------------------
DrawVertical PROC
;Deseneaza fereastra de preview cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
    push ax     ;salvam continutul registrelor
    push bx
```

```
    push cx
    push dx



    mov ah, 0ch          ;introducem codul functiei
    mov al, 15       ;introducem culoarea de desenare(verde)

    mov cx, dimension   ;setam contorul egal cu inaltimea butonului
    v:
    push cx              ;salvam contorul
    mov cx, xv           ;plasam coordonata x
    mov dx, yv           ;plasam coordonata y
    int 10h              ;apela de intrerupere
    inc yv               ;ne deplasam cu un pixel in jos
    pop cx               ;restabilim contorul
    loop v               ;restabilim contorul



    pop dx           ;rstabilim continutul registrelor
    pop cx
    pop bx
    pop ax
    RET
DrawVertical ENDP

;-------------------------------------------------------------------------
DrawHorizontal PROC
;Deseneaza fereastra de preview cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
    push ax      ;salvam continutul registrelor
    push bx
    push cx
```

```
   push dx


   mov ah, 0ch          ;introducem codul functiei
   mov al, 15       ;introducem culoarea de desenare(verde)

   mov cx, dimension   ;setam contorul egal cu inaltimea butonului
   h:
   push cx              ;salvam contorul
   mov cx, xh           ;plasam coordonata x
   mov dx, yh           ;plasam coordonata y
   int 10h              ;apela de intrerupere
   inc xh               ;ne deplasam cu un pixel in jos
   pop cx               ;restabilim contorul
   loop h               ;restabilim contorul


   pop dx           ;rstabilim continutul registrelor
   pop cx
   pop bx
   pop ax
   RET
DrawHorizontal ENDP

;-------------------------------------------------------------------------
Fill PROC
;Deseneaza fereastra de preview cu ajutorul functiei OCh/INT 10h;
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
   push ax      ;salvam continutul registrelor
   push bx
   push cx
   push dx
```

```
mov bx, xsel
inc bx
mov xf, bx
mov bx, ysel
inc bx
mov yf, bx
mov ah, 0ch          ;introducem codul functiei
mov al, 4        ;introducem culoarea de desenare(verde)

mov cx, 9
fl:
push cx

mov cx, 9  ;setam contorul egal cu inaltimea butonului
fl1:
push cx              ;salvam contorul
mov cx, xf             ;plasam coordonata x
mov dx, yf             ;plasam coordonata y
int 10h              ;apela de intrerupere
inc xf               ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop fl1              ;restabilim contorul

mov bx, xsel
inc bx
mov xf, bx
inc yf
pop cx
loop fl


call CheckField

pop dx             ;rstabilim continutul registrelor
pop cx
```

```
    pop bx
    pop ax
    RET
Fill ENDP

CheckButton PROC
mov ah , 00h
int 16h
cmp ah, 4dh
jne a1
inc locationh
inc xg
add xsel, 10
;mov ysel, 38
jmp check
a1:
cmp ah, 4bh
jne ov
dec locationh
dec xg
sub xsel, 10
;mov ysel, 38
jmp check
ov:
cmp ah, 50h
jne as
inc locationv
inc yg
add ysel, 10
jmp check1
as:
cmp ah, 48h
jne o
dec locationv
dec yg
```

```
sub ysel, 10
jmp check1
o:
cmp al, 1bh
jne p
mov ax, 0600h
mov es, ax
jmp 0600h:0000h
p:
cmp al, 13
jne b
cmp count, 0
jne jk
call Fill
call InitialTime
call CheckCoordonate
inc count
jmp ui
jk:
;call CheckCoordonate
mov ax, x1
mov bx, y1
cmp xsel, ax
jne we
cmp ysel, bx
jne we
call Fill
call CheckCoordonate
jmp ui

we:
;call CheckCoordonate
mov ax, x2
mov bx, y2
cmp xsel, ax
```

```
jne we1
cmp ysel, bx
jne we1
call Fill
call CheckCoordonate
jmp ui

we1:
;call CheckCoordonate
mov ax, x3
mov bx, y3
cmp xsel, ax
jne we2
cmp ysel, bx
jne we2
call Fill
call CheckCoordonate
jmp ui

we2:
;call CheckCoordonate
mov ax, x4
mov bx, y4
cmp xsel, ax
jne we3
cmp ysel, bx
jne we3
call Fill
call CheckCoordonate
jmp ui

we3:
;call CheckCoordonate
mov ax, x5
mov bx, y5
```

```
cmp xsel, ax
jne we4
cmp ysel, bx
jne we4
call Fill
call CheckCoordonate
jmp ui

we4:
;call CheckCoordonate
mov ax, x6
mov bx, y6
cmp xsel, ax
jne we5
cmp ysel, bx
jne we5
call Fill
call CheckCoordonate
jmp ui

we5:
;call CheckCoordonate
mov ax, x7
mov bx, y7
cmp xsel, ax
jne we6
cmp ysel, bx
jne we6
call Fill
call CheckCoordonate
jmp ui

we6:
;call CheckCoordonate
mov ax, x8
```

```
mov bx, y8
cmp xsel, ax
jne ui
cmp ysel, bx
jne ui
call Fill
call CheckCoordonate
jmp ui

ui:
ret
CheckButton ENDP

CheckCoordonate PROC
   push ax          ; ? Salvam continutul
   push bx          ; | registrelor in starea
   push cx          ; | in care se aflau inainte de
   push dx          ; L apelarea functiei

   sub yf, 10
   dec xf
   mov ax, xf
   mov bx, yf

   add ax, 10
   sub bx, 20
   mov x1, ax
   mov y1, bx

   mov ax, xf
   mov bx, yf

   add ax, 20
   sub bx, 10
   mov x2, ax
```

```
mov y2, bx

mov ax, xf
mov bx, yf

add ax, 20
add bx, 10
mov x3, ax
mov y3, bx

mov ax, xf
mov bx, yf

add ax, 10
add bx, 20
mov x4, ax
mov y4, bx

mov ax, xf
mov bx, yf

sub ax, 10
add bx, 20
mov x5, ax
mov y5, bx

mov ax, xf
mov bx, yf

sub ax, 20
add bx, 10
mov x6, ax
mov y6, bx

mov ax, xf
```

```
    mov bx, yf

    sub ax, 20
    sub bx, 10
    mov x7, ax
    mov y7, bx

    mov ax, xf
    mov bx, yf

    sub ax, 10
    sub bx, 20
    mov x8, ax
    mov y8, bx



    pop dx          ; ? Restabilim
    pop cx          ; | continutul
    pop bx          ; | registrelor
    pop ax          ; L in starea inainte de apelare

    ret
CheckCoordonate ENDP

RedrawCell PROC
    push ax         ; ? Salvam continutul
    push bx         ; | registrelor in starea
    push cx         ; | in care se aflau inainte de
    push dx         ; L apelarea functiei

    mov cx, 60
    mov dx, 38
    mov ah, 0dh
    int 10h
    cmp al, 3
```

```
    jne rb
    mov x, 60
    mov y, 38
    call DrawCell
    jmp end
    rb:
    mov cx, 70
    mov dx, 38
    mov ah, 0dh
    int 10h
    cmp al, 3
    jne rb1
    mov x, 70
    mov y, 38
    call DrawCell
    jmp end
    rb1:
    mov cx, 80
    mov dx, 38
    mov ah, 0dh
    int 10h
    cmp al, 3
    mov x, 80
    mov y, 38
    call DrawCell

    end:
    pop dx          ; ? Restabilim
    pop cx          ; | continutul
    pop bx          ; | registrelor
    pop ax          ; L in starea inainte  de apelare

    ret
RedrawCell ENDP
```

```
DrawSelectedCell PROC
;Deseneaza un buton la coordonata specificata cu ajutorul functiei OCh/INT 10h;
;coordonata indica coltul stinga-sus a butonului
;Primeste: coordonata X si Y
;Returneaza: Nimic
;-------------------------------------------------------------------------
   push ax      ;salvam continutul registrelor
   push bx
   push cx
   push dx


   mov ah, 0ch          ;introducem codul functiei
   mov al, 1       ;introducem culoarea de desenare(verde)

   mov cx, 10  ;setam contorul egal cu inaltimea butonului
   dsc:
   push cx              ;salvam contorul
   mov cx, xsel            ;plasam coordonata x
   mov dx, ysel            ;plasam coordonata y
   int 10h              ;apela de intrerupere
   inc ysel               ;ne deplasam cu un pixel in jos
   pop cx               ;restabilim contorul
   loop dsc

   mov cx, 10  ;setam contorul egal cu inaltimea butonului
   dsc1:
   push cx              ;salvam contorul
   mov cx, xsel            ;plasam coordonata x
   mov dx, ysel            ;plasam coordonata y
   int 10h              ;apela de intrerupere
   inc xsel               ;ne deplasam cu un pixel in jos
   pop cx               ;restabilim contorul
   loop dsc1
```

```
   mov cx, 10  ;setam contorul  egal cu inaltimea  butonului
   dsc2:
   push cx              ;salvam  contorul
   mov cx, xsel             ;plasam  coordonata x
   mov dx, ysel             ;plasam  coordonata y
   int  10h             ;apela de intrerupere
   dec ysel               ;ne deplasam cu un pixel  in jos
   pop cx               ;restabilim  contorul
   loop dsc2

   mov cx, 10  ;setam contorul  egal cu inaltimea  butonului
   dsc3:
   push cx              ;salvam  contorul
   mov cx, xsel             ;plasam  coordonata x
   mov dx, ysel             ;plasam  coordonata y
   int  10h             ;apela de intrerupere
   dec xsel               ;ne deplasam cu un pixel  in jos
   pop cx               ;restabilim  contorul
   loop dsc3

   pop dx            ;rstabilim  continutul  registrelor
   pop cx
   pop bx
  pop ax
   RET
DrawSelectedCell  ENDP

DrawCell  PROC
;Deseneaza  un buton la coordonata specificata  cu ajutorul  functiei  OCh/INT 10h;
;coordonata  indica  coltul  stinga-sus  a butonului
;Primeste:  coordonata  X si Y
;Returneaza:  Nimic
;-------------------------------------------------------------------------
   push ax      ;salvam  continutul  registrelor
   push bx
```

```
push cx
push dx


mov ah, 0ch          ;introducem codul functiei
mov al, 15       ;introducem culoarea de desenare(verde)

mov cx, 10  ;setam contorul egal cu inaltimea butonului
dc:
push cx              ;salvam contorul
mov cx, x            ;plasam coordonata x
mov dx, y            ;plasam coordonata y
int 10h              ;apela de intrerupere
inc y                ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop dc

mov cx, 10  ;setam contorul egal cu inaltimea butonului
dc1:
push cx              ;salvam contorul
mov cx, x            ;plasam coordonata x
mov dx, y            ;plasam coordonata y
int 10h              ;apela de intrerupere
inc x                ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
loop dc1

mov cx, 10  ;setam contorul egal cu inaltimea butonului
dc2:
push cx              ;salvam contorul
mov cx, x            ;plasam coordonata x
mov dx, y            ;plasam coordonata y
int 10h              ;apela de intrerupere
dec y                ;ne deplasam cu un pixel in jos
pop cx               ;restabilim contorul
```

```
    loop dc2

    mov cx, 10  ;setam contorul  egal cu inaltimea  butonului
    dc3:
    push cx              ;salvam  contorul
    mov cx, x            ;plasam coordonata x
    mov dx, y            ;plasam coordonata y
    int 10h              ;apela de intrerupere
    dec x                ;ne deplasam cu un pixel in jos
    pop cx               ;restabilim  contorul
    loop dc3

    pop dx           ;rstabilim continutul registrelor
    pop cx
    pop bx
   pop ax
    RET
DrawCell ENDP


InitialTime  PROC
    push ax
    push bx
    push cx
    push dx

    mov ah,02h
    int 1ah
    mov buffer_hour,  ch
    mov buffer_minute,  cl
    mov buffer_second,  dh

    pop dx
    pop cx
    pop bx
```

```
    pop ax
    ret
InitialTime  ENDP

FinalTime  PROC
    push ax
    push bx
    push cx
    push dx

    mov ah,02h
    int 1ah
    mov buffer_hour1, ch
    mov buffer_minute1, cl
    mov buffer_second1, dh

    call CalcHour
    call CalcMinute
    call CalcSecond

    call DisplayHour
    call DisplaySign
    call DisplayMinute
    call DisplaySign1
    call DisplaySecond

    pop dx
    pop cx
    pop bx
    pop ax
    ret
FinalTime  ENDP

CheckField  PROC
    push ax
```

```
push bx
push cx
push dx

mov count_fill, 0
mov xcf, 65
mov ycf, 45

mov cx, 8
cf:
push cx

mov cx, 8
cf2:
push cx
mov ah, 0Dh
mov cx, xcf
mov dx, ycf
int 10h
cmp al, 4
jne cf1
inc count_fill
cmp count_fill, 64
jne cf1
call FinalTime
cf1:
add xcf, 10
pop cx
loop cf2

mov xcf, 65
add ycf, 10
pop cx
loop cf
```

```
    pop dx
    pop cx
    pop bx
    pop ax
    ret
CheckField  ENDP

;-------------------------------------------------MAIN-------------------------------------
-------------------------
;-------------------------------------------------------------------------------------------------------
----------------------
main:

call DrawField



b:
call CheckButton



check:
cmp locationh,0
jnle q
mov locationh,  1
q:
cmp locationh,8
jna m
mov locationh,  8
m:
cmp locationh,1
jne s
call DrawField
mov xsel,  60
call DrawSelectedCell
```

```
jmp b
s:
cmp locationh,2
jne t
call DrawField
call DrawSelectedCell
jmp b
t:
cmp locationh,3
jne tv
call DrawField
call DrawSelectedCell
jmp b
tv:
cmp locationh,4
jne tv1
call DrawField

call DrawSelectedCell
jmp b
tv1:
cmp locationh,5
jne tv2
call DrawField
call DrawSelectedCell
jmp b
tv2:
cmp locationh,6
jne tv3
call DrawField
call DrawSelectedCell
jmp b
tv3:
cmp locationh,7
jne tv4
```

```
call DrawField
call DrawSelectedCell
jmp b
tv4:
cmp locationh,8
jne qw
call DrawField
mov xsel, 130
call DrawSelectedCell
jmp b



check1:
qw:
cmp locationv,0
jnle qv
mov locationv, 1
qv:
cmp locationv,8
jna mv
mov locationv, 8
mv:
cmp locationv,1
jne sv
call DrawField
mov ysel, 38
call DrawSelectedCell
jmp b
sv:
cmp locationv,2
jne ta
call DrawField
call DrawSelectedCell
jmp b
```

```
ta:
cmp locationv,3
jne tva
call DrawField
call DrawSelectedCell
jmp b
tva:
cmp locationv,4
jne tva1
call DrawField

call DrawSelectedCell
jmp b
tva1:
cmp locationv,5
jne tva2
call DrawField
call DrawSelectedCell
jmp b
tva2:
cmp locationv,6
jne tva3
call DrawField
call DrawSelectedCell
jmp b
tva3:
cmp locationv,7
jne tva4
call DrawField
call DrawSelectedCell
jmp b
tva4:
cmp locationv,8
jne b
call DrawField
```

```
mov ysel, 108
call DrawSelectedCell
jmp b

jmp $
;-------------------------------------------------DATA-------------------------------------------
-------------------------
;-----------------------------------------------------------------------------------------------------------
----------------------
x dw 60
y dw 38
xv dw 70
yv dw 38
xh dw 60
yh dw 48
xf dw 61
yf dw 39
xsel dw 60
ysel dw 38
dimension  dw 80
locationh  db 0
locationv  db 0
count  db 0
xg db 0
yg db 0

x1 dw 0
y1 dw 0
x2 dw 0
y2 dw 0
x3 dw 0
y3 dw 0
x4 dw 0
y4 dw 0
x5 dw 0
```

```
y5 dw 0
x6 dw 0
y6 dw 0
x7 dw 0
y7 dw 0
x8 dw 0
y8 dw 0
xcf dw 0
ycf dw 0
count_fill db 0

buffer_hour db 0
buffer_minute db 0
buffer_second db 0
buffer_hour1 db 0
buffer_minute1 db 0
buffer_second1 db 0
tetrad1 db 0
tetrad2 db 0
points db ':',0
message db 'Your Time', 0
resultd db 0
sign db 0
db 512 - ($ - $$) DUP(0)        ;Fill the rest of sector with 0
```

Anexa J. Codul sursă al funcției File

```
org 100h
mov ah, 3ch
mov cx, 0
mov dx, offset filename
mov ah, 3ch
int 21h ; create file...
mov handle, ax

mov bx, handle
mov dx, offset data
mov cx, data_size
mov ah, 40h
int 21h ; write to file...

mov bx, handle
mov ah, 3eh
int 21h ; close file...
ret

filename db "myfile.dat", 0
handle dw ?
data db " some data "
data_size=$-offset data
```

Anexa K. Codul sursă al funcției Help

```
#make_bin#

mov ax,0950h
mov ds, ax

jmp main

WriteString:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;    Procedura de afisare a unui sir de caractere        ;;
;;     Functia  0x13 a intreruperii BIOS 0x10             ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mov ah, 13h   ; Codul functiei de afisare a unui sir
mov al, 1      ; Modul de afisare
mov bh, 0x00  ; Numarul paginii
int 0x10             ; Call video interrupt
ret

WriteMessages:

push ax              ; ? Salvam continutul
push bx              ; | registrelor in starea
push cx              ; | in care se aflau inainte de
push dx              ; L apelarea functiei

mov bp, offset str1          ; salvam offsetul mesajului in bp (Hello World!)
mov bl, 1           ; setam codul culorii
mov cx, str1_len  ; incarcam in cx numarul de caractere a sirului
mov dh, 7              ; rindul din care incepe afisarea
mov dl, 17        ; coloana din care incepe afisarea
call WriteString

mov bp,offset str2          ; salvam offsetul mesajului in bp (This is )
```

```
mov bl, 0fh             ; setam codul culorii
mov cx, str2_len    ; incarcam in cx numarul de caractere a sirului
mov dh, 8               ; rindul din care incepe afisarea
mov dl, 8       ; coloana din care incepe afisarea
call WriteString

mov bp,offset str3              ; salvam offsetul mesajului in bp (This is )
mov bl, 0fh             ; setam codul culorii
mov cx, str3_len    ; incarcam in cx numarul de caractere a sirului
mov dh, 9               ; rindul din care incepe afisarea
mov dl, 8       ; coloana din care incepe afisarea
call WriteString

mov bp,offset str4              ; salvam offsetul mesajului in bp (This is )
mov bl, 0fh             ; setam codul culorii
mov cx, str4_len    ; incarcam in cx numarul de caractere a sirului
mov dh, 10              ; rindul din care incepe afisarea
mov dl, 8       ; coloana din care incepe afisarea
call WriteString

mov ah, 01h             ;ascundem
mov cx, 2607h           ;cursorul
int 10h

j:
mov ah, 00h
int 16h
cmp al, 1bh
jne j
mov ax, 0600h
mov es, ax
jmp 0600h:0000h


pop dx              ; ? Restabilim
```

```
pop cx              ; | continutul
pop bx              ; | registrelor
pop ax              ; L in starea inainte  de apelare
ret

main:
call WriteMessages

jmp $

str1 db "Mini  OS",0
str1_len  = $-str1
str2 db "Kernel  Version  3.0",0
str2_len  = $-str2
str3 db "Release  date 29.11.2015",0
str3_len  = $-str3
str4 db "Author  : Popov Eugen",0
str4_len  = $-str4

db 512 - ($ - $$) DUP(0)
```