5. Simulator de microprocesor (III)

5.1. Salturi în program – **thermometer.asm**

Aplicaţia foloseşte un termometru care trebuie menţinut la o temperatură cuprinsă între 60° şi 80° cu ajutorul unui radiator de căldură. Aplicaţia foloseşte 2 porturi de intrare: de date (adresa 125) şi de control (adresa 127). La început, temperatura este 0 (portul 125 iniţializat cu 0), utilizatorul putând interveni şi controla (porni/opri) radiatorul de căldură (portul 127) sau temperatura aerului. Temperatura creşte repede de la valoarea 0, până depăşeşte 80°, apoi radiatorul de căldură se opreşte din program (automat) şi până ce temperatura nu ajunge să fie mai mică de 60° nu mai porneşte.

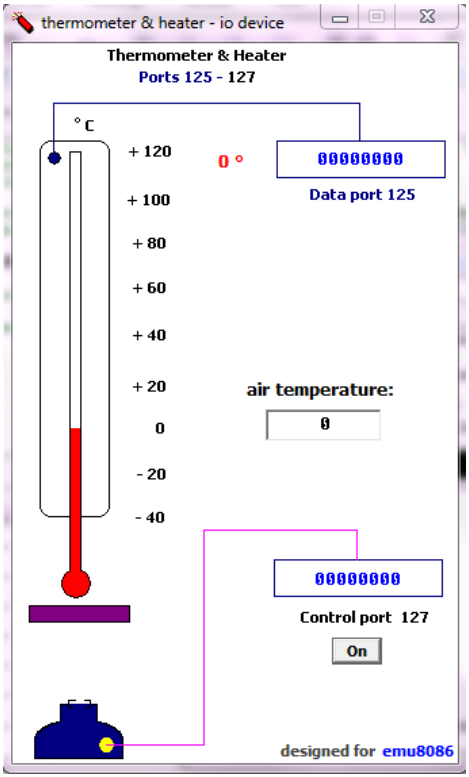

Fig.5.1. Fereastra aplicaţiei thermometer.asm

Programul **thermometer.asm** este prezentat în continuare:

```
; setează adresa segmentului de date înspre segmentul de cod:
mov ax, cs
mov ds, ax
start:                ;eticheta START

in al, 125            ;preia valoarea temperaturii de pe portul
                      ;de date cu adresa 125
cmp al, 60            ;compara valoarea temp. curente cu 60°
jl  low               ;daca temp. este < 60° sare la et. LOW

cmp al, 80      ; compară valoarea temp. curente cu 80°
jle  ok         ; daca temperatura este <= 80°,
                ; sare la eticheta OK
jg   high       ; daca temp. este > 80°, sare la eticheta HIGH
low:            ; eticheta LOW
mov al, 1       ; AL=1
out 127, al     ; pune radiatorul de caldura pe "on",
                ; trimitand 1 pe port
jmp ok          ; salt neconditionat la eticheta OK

high:
mov al, 0       ; AL=0
out 127, al     ; pune radiatorul de caldura pe "off",
                ; trimitand 0 pe port

ok:
jmp start       ; salt înapoi la START,
                ; deci bucla se reia la infinit.
```

Exerciţii şi teme (I)

1. Modificaţi limitele între care se încearcă păstrarea temperaturii la 70° şi 90°.
2. Modificaţi valoarea ce se încarcă în registrul AL la eticheta LOW să fie 2. Salvaţi şi rulaţi din nou. Ce observaţi ? Cum explicaţi ?

5.2. Operaţii de incrementare/decrementare: **examples**, **LED display test**

Aplicaţia este una foarte simplă, foloseşte operaţii de incrementare/decrementare asupra registrului AX şi salturi necondiţionate în program. Acest exemplu foloseşte portul 199 pentru a emula existenţa unui dispozitiv virtual precum cel din figura 5.2.

Fig.5.2. Fereastra aplicaţiei LED display

Programul **LED_display_test.asm** este prezentat în continuare:

```
mov ax, 1234    ;AX=1234
out 199, ax     ;continutul reg AX se trimite la portul 199,
                ;deci se va vizualiza pe display valoarea 1234
mov ax, -5678   ;AX=-5678
out 199, ax     ; pe display se va vizualiza valoarea -5678
; bucla infinita
mov ax, 0       ; AX=0
x1:             ; eticheta x1
  out 199, ax   ; se afiseaza pe display valoarea din AX
  inc ax        ; se incrementeaza continutul registrului AX
jmp x1          ; salt automat la eticheta x1
hlt
```

Exerciţii şi teme (II)

1. Modificaţi aplicaţia astfel încât să funcţioneze asemănător termometrului: valoarea să se incrementeze până ajunge la 50, apoi să descrească până ajunge la 40 şi tot aşa. Sugestie: folosiţi instrucţiuni de salt condiţionat.

5.3. Scrierea datelor în memoria video: **„Hello, world" (examples)**

Fişierul asm se deschide din meniul emulatorului, folosind opţiunea examples, şi se găseşte sub numele „Hello, world", efectul fiind cel ilustrat în figura urmatoare.

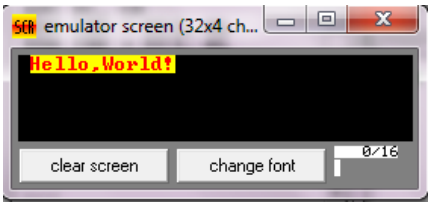

Fig.5.3. Fereastra de vizualizare a ecranului

Emulatorul foloseste 8 pagini de memorie video, cuprinse intre adresele 0B8000h-0C0000h. Ecranul emulatorului poate fi redimensionat, astfel incat sa fie necesara mai putina memorie pentru fiecare pagină.

Folosind acest program se afişează în ecranul emulatorului mesajul „hello world” prin scriere directa în memoria video. O caracteristică a memoriei VGA este că primul octet are semnificaţie de cod Ascii al caracterului de afişat, iar octetul următor (consecutiv) reprezintă atributul lui de culoare.

Atributul de culoare este scris pe octet, cei mai semnificativi 4 biţi specificând culoarea de fond (background), iar cei mai puţin semnificativi 4 biţi specificând culoarea de scriere (foreground), după regula: primul bit – dacă este intermitent sau nu, iar următorii 3 culoarea în format RGB. Dintre cele mai uzuale culori, se pot specifica: 0000-negru, 0001-albastru, 0010-verde, 0100-rosu, 0111-gri deschis, 1000-gri inchis, 1001-albastru deschis, ..., 1110-galben, 1111-alb.

Observatii

MOV [02h],'H' – va copia codul Ascii al literei H la adresa [02h] în memoria video.

MOV [BL],AL – copiază AL la adresa dată de BL. Registrul BL poate fi făcut pointer la prima poziţie în memoria video, apoi prin incrementare cu 2 se trece la următoarea locaţie specifică codului caracterului de afişat.

LOOP este specific buclelor: se execută instrucţiunile din cadrul buclei atât timp cât CX este diferit de 0. Instrucţiunea LOOP lucrează împreună cu registrul CX, pe care il decrementează la fiecare trecere prin buclă, dar şi cu o etichetă care specifică locul de salt.

Programul **hi-world.asm** este prezentat în continuare:

```
name "hi-world"
org 100h
;_______SCRIERE IN MEMORIA VIDEO__________

mov ax, 3    ;setare mod video
int 10h      ;mod text 80coloane x 25linii, 16 culori, 8 pagini
            ;ah=0, al=3)
mov ax, 1003h          ;invalidare intermitentă
                       ;permiterea celor 16 culori
mov bx, 0
int 10h

mov     ax, 0b800h     ;setare registru segment
mov     ds, ax

;pe masură ce se scrie în memoria video, datele se afişează pe
;ecran
;se afişează mesajul Hello, World
;primul octet (la adrese pare) e codul Ascii,
;al doilea octet e atributul de culoare (la adrese impare).
```

```
mov [02h], 'H'
mov [04h], 'e'
mov [06h], 'l'
mov [08h], 'l'
mov [0ah], 'o'
mov [0ch], ','
mov [0eh], 'W'
mov [10h], 'o'
mov [12h], 'r'
mov [14h], 'l'
mov [16h], 'd'
mov [18h], '!'

; colorează toate caracterele:
mov cx, 12            ;numărul de caractere
mov di, 03h           ;adresa culoare litera ,H'

c:  mov [di], 11101100b   ;roşu deschis(1100)/fond galben(1110)
    add di, 2             ;adresa culoare litera urmatoare
    loop c

; aşteaptă apăsarea unei taste
mov ah, 0
int 16h
ret
```

Exerciţii şi teme (III)

1. Executaţi aplicaţia pas cu pas, modificând atributul de culoare. Folosiţi pentru fond culoare roşu deschis, iar pentru scris culoarea verde.

2. Scrieţi un nou program, în care să înseraţi următoarele instrucţiuni:

```
MOV AX, 0B800h     ; AX= B800h.
MOV DS, AX         ; DS=B800h
MOV CL, 'A'        ; CL =codul Ascii al lui A, adică 41h.
MOV CH, 1101_1111b ; CH =11011111b.
MOV BX, 15Eh       ; BX=15Eh.
MOV [BX], CX       ; in memorie, la adr. B800:015E se depune CX
```

Rulaţi pas cu pas, explicaţi codul şi specificaţi efectul programului. Modificaţi astfel încât să se afiseze 3 caractere B de culori diferite: unul roşu, unul verde, unul albastru, toate pe fond gri.

3. Scrieţi un program care să reproducă imaginea din figura 5.3, adică să apară textul respectiv pe linii diferite; în plus, modificaţi atributele de culoare să fie diferite de pe un rând pe altul.