TASKuri LAB 7:

Task 1. Rulati programul isr1c din Arhiva L6 de pe site şi analizaţi următoarele:

Intrebare1: **Ce observati, cum rulează programul, care este efectul lui?**

Răspuns: programul afiseaza ‘a’, asteapta 5 secunde, dupa care afiseaza ‘b’

Intrebare2: **De unde apar cele 5 secunde, timp de intarziere între afişarea lui ‘a’ şi afişarea lui ‘b’?**

Răspuns: se observă instrucţiunea **mov Si, 5000/55** împreună cu instrucţiunea **dec Si** din rutina de tratare a întreruperii Valoarea 5000 înseamnă câtă întârziere vrem noi să asigurăm în programul nostru, între pasul 2 şi pasul 3 al redirectării de întrerupere; Valoarea 55 reprezintă (în msecunde) baza de timp asigurată de timer (în BIOS este programată astfel). Astfel, noi vom încărca în registrul Si valoarea 90(=5000/55), valoare care va fi decrementată la fiecare 55 msec în RTI (Rutina de Tratare a Intreruperii).

Intrebare3: **Cum arată secvenţa de programare a timerului a.î. să obţinem baza de timp de 55 msec?**

Răspuns: Initial, din BIOS, ch0 al timerului a fost programat cu n0=0 => $F_{outCh0} = f_{Clk0} / 0 = f_{Clk0} / 2^{16} = = f_{Clk0} /65\,536$, unde n ar fi trebuit să fie în domeniul $[0; 2^{16}-1] = [0; 65\,535]$, dar pentru **0** se foloseşte valoarea **65 536**

Intrebare4: **Care este secvenţa de program pentru a programa baza de timp în BIOS cu n0=0 ?**

Răspuns:

; încărcăm RC (registrul de comandă al timerului 8253) cu informaţiile: vom programa Ch 0 (b7b6=00), îi vom trimite constanta de divizare pe rând, prima dată partea Low, apoi partea High (b5b4=11), îi spunem să folosească modul de lucru 3, adică generator de semnal dreptunghiular (b3b2b1=011) şi îi spunem să numere în binar (b0=0)

```
Mov AL, 00110110b
Out 43h, AL
; îi vom trimite constanta de divizare după cum am menţionat anterior
Mov AX, 0
Out 40h, AL   ; îi trimitem partea LOW a constantei de divizare
Mov AL, AH
Out 40h, AL  ; îi trimitem partea HIGH a constantei de divizare
```

Task 2. In continuare, vom vrea sa modificam aceasta bază de timp, la jumătate.

Astfel, pas 1: vom asigura o intarziere de 10 secunde folosind programul original.

DECI : n0=0 (este deja programat în BIOS)=> T=55 msec, apoi **mov SI, 10000/55**

!!!! Dacă vom reprograma baza de timp a timerului cu o altă valoare, avem obligaţia ca înainte de a părăsi programul, să restaurăm baza de timp aşa cum a fost iniţial, la 55 msec, deci să o reprogramăm cu n0=0.

Pas2: Vom reprograma timerul cu o constanta de divizare de jumătate, deci 65535/2 => noua bază de timp T~=**28 msec**

Daca in rest pastram programul asa cum a fost original **mov SI, 10000/55** , vom obtine o intarziere de doar 5 sec în loc de 10 secunde, deoarece baza de timp este la jumătate, adică practic timpul curge de 2 ori mai repede.

Intrebare1: **Care este secvenţa de program pentru a reprograma baza de timp la 28msec in loc de 55 msec ?**

Răspuns: este aceeaşi secvenţă de program ca la întrebarea 4, Task 1, doar că în loc de 0 vom scrie **65535/2**

Intrebare2: **Care este ordinea în care scriem aceste secvenţe (blocuri de cod) ?**

Răspuns:

```
; programare timer cu o noua constanta de divizare, pentru a asigura deci o noua baza de timp
; pas 1 redirectare de întrerupere
; pas 2 redirectare de întrerupere

; programul principal, în care setăm valoarea lui SI

; pas 3 redirectare de întrerupere
; secvenţa pentru programare timer cu n0=0
; iesire afara din program
```

Task 3: Lucrul cu memoria video

Memoria video este o zonă din memoria sistemului, cu anumite particularităţi, în care, dacă scriem ceva, se va afişa instantaneu pe ecran (nu avem nevoie de nici o întrerupere pentru afişare, pur şi simplu doar scriem ceva în zona memoriei video).

Particularităţi memoria video :

1. **Memoria video începe la adresa fizică 0B8000h (adresă scrisă pe 20 de biţi)**

2. **Conţinutul memoriei video trebuie văzut în perechi de câte 2 octeţi: primul, cel de la adresa pară, va reprezenta conţinutul Ascii al caracterului pe care vrem să îl afişăm pe ecran, iar al doilea octet, cel de la adresa impară următoare, va reprezenta atributul de culoare cu care vrem să se afişeze acel caracter pe ecran**
   Reamintesc aici atributul de culoare:
   primii 4 biţi (b7b6b5b4 = fondul: Blink Red Green Blue)
   ultimii 4 biţi (b3b2b1b0 = culoarea de scriere: Blink Red Green Blue)

3. **Colţul stânga sus al ecranului formatat în mod text – ecranul negru (deci cu 80 coloane şi 25 linii) corespunde fix adreselor 0B8000h şi resp. 0B8001h.**

Task3.1. **Rulaţi următorul program care afişează A colorat cu galben pe fond roşu, folosind memoria video:**

```
.model small
.stack
.data
.code

main:
  mov AX, 0B800h
  mov ES, AX            ; folosim registrul ES ca sa accesam segmentul
  mov BX, 10 + 5*2*80 + 2*2 ; cu 10 controlam punctul de inceput (coloana a 5-a)
                            ; cu 5*2*80 controlam pozitia pe linie (deplasare),
                            ; cu 2*2 controlam pozitia pe coloana (deplasare),
  mov ES:[BX], 0CE41h ; culoarea CEh se va scrie la adresa 0B8001h,
                      ; iar caracterul 'A'=41h, se va scrie la adresa 0B8000h

    mov AX,4C00h        ; iesirea afara din program
    int 21h
END main
```

Task3.2. **Modificati programul anterior, astfel incat sa afiseze 3 caractere B, colorate cu verde pe fond alb.**

Task 4.1: **In cadrul sablonului programului isr1c.asm din arhiva L6, scrieţi secvenţa de cod prin care să se afişeze câte un caracter B, colorat cu verde pe fond alb, la fiecare intrare in RTI.**

Task 4.2. **Modificati rutina RTI, a.î. să se afişeze un caracter la fiecare a 10-a intrare in RTI.**

```
isr1Ch:
    dec SI

    mov CL,10
    mov AX,SI
    div CL        ; impartim valoarea lui SI la 10 si verificam daca restul este 0
    cmp AH,0
    jne nu        ; daca restul NU este 0, sarim peste instructiunile cu mem video

      mov ES:[BX], 0CE41h ; daca restul este 0, vom scrie in memoria video continut
      add BX, 2

nu:
    iret
```

Task 5. Adaugati sunet la fiecare afisare a caracterului, urmat de o pauza.