**Lab 5**

Mov al, 10

Jmp Scadere

Adunare:

Add al, 3

Jmp final

Scadere:

Sub al, 5

Jmp Adunare

Final:

Cmp

Dx:ax+ cx:bx

Add ax, bx ; daca exista tr se seteaza CF

JC AdunarecuCarry

JNC AdunarefaraCarry

AdunarecuCarry:

Adc dx, cx

Jmp final

AdunarefaraCarry:

Add dx, cx

Final:

**A = 10 ;-10**

**B = 20 ;-20**

**If a>b then rez = a-b**

**Else rez = b-a**

**Mov al, [a]**

**Mov bl, [b]**

**Cmp al, bl**

**JA etichetathen ;JG**

**JBE etichetaelse ; jmp Etichetaelse: ;JLE**

**Etichetathen:**

**Sub al, bl**

**Mov [rez], al**

**Jmp final**

**Etichetaelse:**

**Sub bl, al**

**Mov [rez], bl**

mov ecx, numar ; numar de iteratii

JECXZ endFor ;skip loop if numar=0

forIndex

; instructiuni

Loop forIndex ; repeat

endFor

**mov al, 1**

**mov ecx, 5**

**jecxz final**

**repeta:**

**add al, 1**

**loop repeta**

**probl:**

**bits 32 ; assembling for the 32 bits architecture**

**; declare the EntryPoint (a label defining the very first instruction of the program)**

**global start**

**; declare external functions needed by our program**

**extern exit ; tell nasm that exit exists even if we won't be defining it**

**import exit msvcrt.dll ; exit is a function that ends the calling process. It is defined in msvcrt.dll**

**; msvcrt.dll contains exit, printf and all the other important C-runtime specific functions**

**; our data is declared here (the variables needed by our program)**

**segment data use32 class=data**

**; ...**

**sir db 1, 1Ah, 1010b, 2, 7**

**lung\_sir equ $ - sir ; =5**

**rez db 0, 0, 0, 0, 0 ; sau rez times lung\_sir db 0**

**k equ 5**

**; our code starts here**

**segment code use32 class=code**

**start:**

**; reg index esi - sir sursa si edi pentru sir destinatie**

**mov esi, 0**

**mov edi, 0**

**mov ECX, lung\_sir**

**repeta:**

**mov al, [sir+esi]**

**add al, k**

**mov [rez+edi], al**

**inc esi ; add esi, 1**

**inc edi ; add edi, 1**

**loop repeta**

**; ...**

**; mov al, 1**

**; mov ecx, 1**

**; repeta:**

**; add al, 1**

**; loop repeta**

**; exit(0)**

**push dword 0 ; push the parameter for exit onto the stack**

**call [exit] ; call exit to terminate the program**