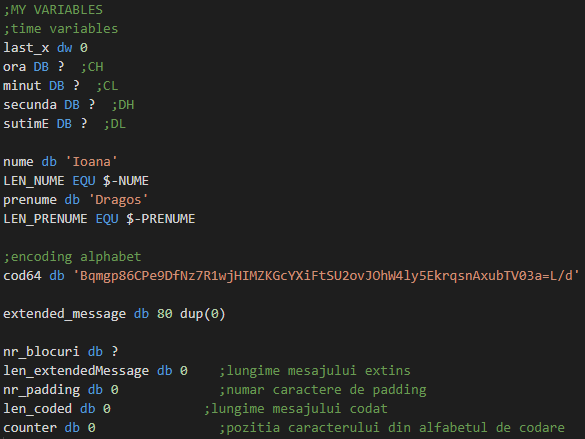
**Proiect**

**Arhitectura sistemelor de calcul**

1. **Variabile declarate**



* Variabilele **ora,minuta,secunda,milisecunda** vor retine valorile momentului de timp curent dupa subrutina **2CH** din intreruperea **21H.**
* Variabila **last\_x** va retine valoarea lui **Xn-1** ,adica a ultimul x folosit la criptarea mesajului
* **Cod64** contine intreg alfabetul de codarea
* **Extended\_message** este o variabila de 80 de octeti initializati cu 0,cu octetii pentru padding,in care se va copia mesajul criptat si care se va folosi pentru codarea mesajului
* **Nr\_blocuri** va retine numarul de blocuri din **extended\_message** ,adica numarul de grupari de cate 3 caractere din **extended\_message(extended\_message** retine mesajul impreauna cu bitii adaugati pentru padding**)**

**Nr\_blocuri=(strlen(message)+number\_of\_padding\_caracters)**

**3**

**Nr\_blocuri=strlen(encoded)**

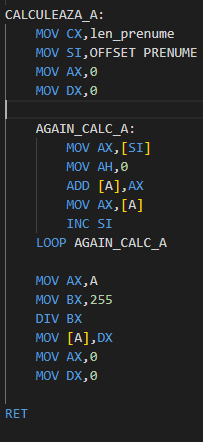
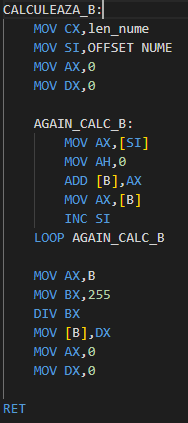
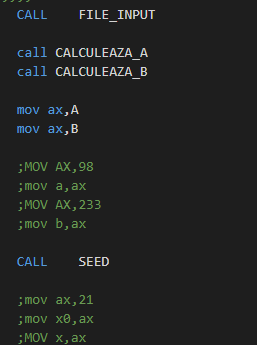
**4**

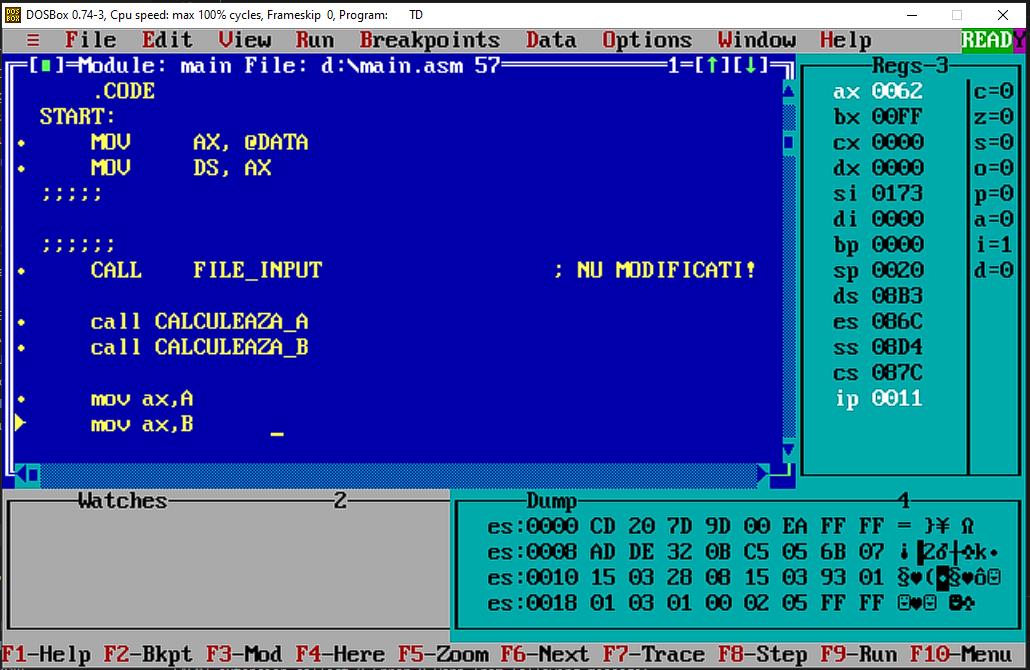
* **Nr\_padding** retine numarul de caractere de padding din mesajul codat
* **Len\_codded** va retine lungimea mesajului codat
* **Counter** va fi folosit ca si offset in alfabetul de codare

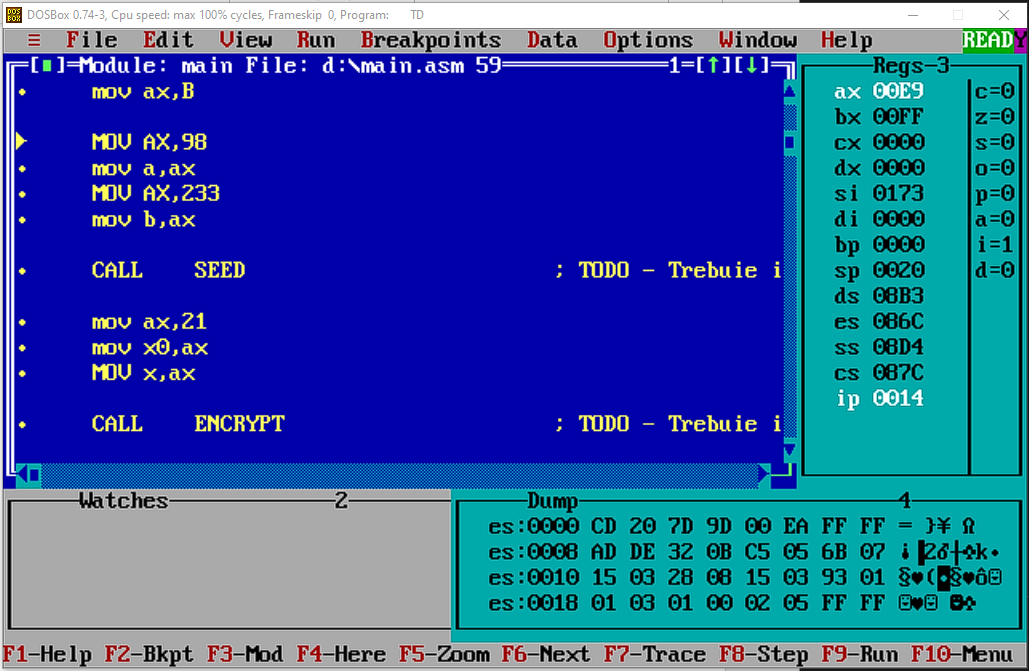
1. **Calcului parametrilor necesari(a,b,X0,X)**

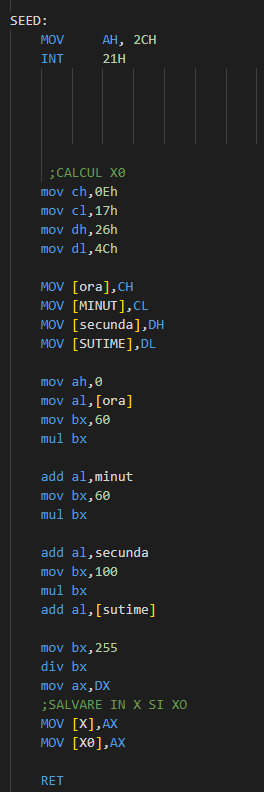
* **Calculul lui A** si **B**

Dupa citirea mesajul din fisierul **in.txt** se vor calcula valorile **A** si **B** pe baza valorilor introduse in variabilele **nume** si **prenume** si a algoritmului prezentat in exercitiu.

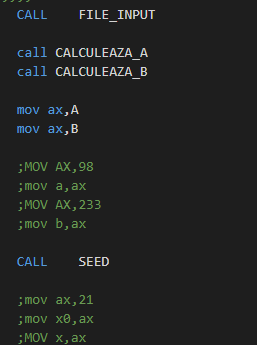
Variabilele **len\_nume,len\_prenume** retin numarul de caractere din nume respeciv din prenumele setat.

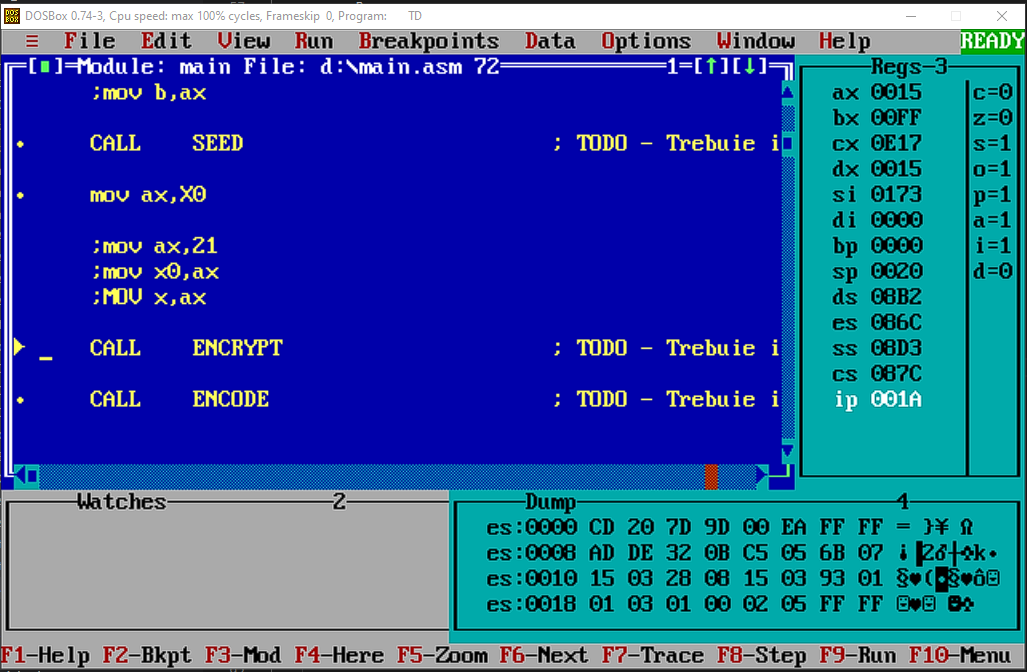
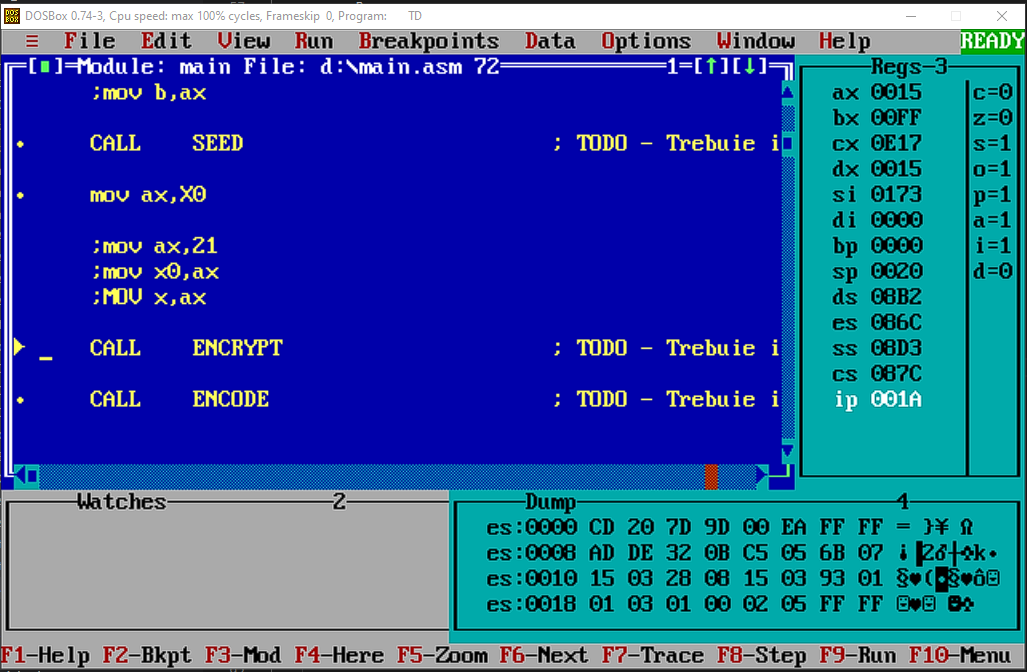




* **Calculul lui X0**

Calculul lui **X0** se executa in functia **SEED** dupa algoritmul prezentat in sarcina.Pentru exemplificarea se vor folosi valorile din sarcina proiectului

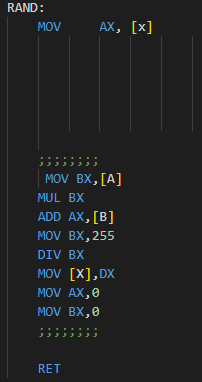




1. **Criptarea mesajului**

Critarea mesajului se efectueaza aplicand operatia **XOR** pe fiecare caracter dim mesajul din fisierul de input impreuna cu valoarea **X** generata la iteratia respectiva

de catre functia **RAND** pornind de la valoarea initiala **X0**.



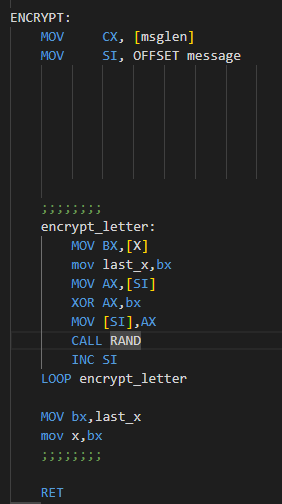
* Functia **RAND**

Fiecare x se obine din x precedent folosind formula:

**xn = (a ∗ xn−1 + b) mod 255**

Functia salveaza valoarea noului x in variabila **x.**

La final registrii **AX** si **BX** se vor seta pe 0 pentru a nu afecta operatiiile urmatoare.



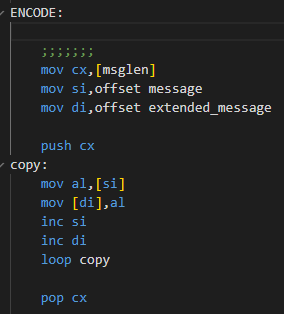
* Cripatarea mesajului

In interiorul loop-ului se extrage pe rand fiecare litera din mesaj in regitrul AX.In registrul BX se va salva valoarea

X-ului curent iar variabila **last­\_x** va memora fiecare valoare **Xn** folosita la criptare.Astfel la sfarsitul loop-ului

In **last\_x** se va retine ultima valoare folosita criptare din sirul **[X]** care apoi se va salva in variabila **X.**

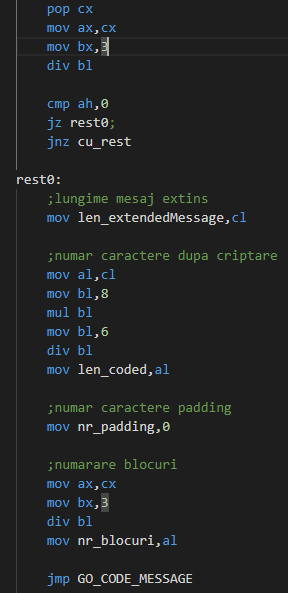
1. **Criptarea mesajului**



In registrul CX se va retine lungimea mesajului din fisierul de input

Intai se va copia mesajul in variabila **extended\_mesage** care are 80 de octeti initializati cu 0,asfel dupa copiere **extended\_message** va avea

80-[msglen] octeti de 0.

Se verifica daca se vor adauga octeti de padding prim impartirea **msglen** la 3.

**Daca restul impartirii la 3 este 0** nu se vor lua biti de padding:

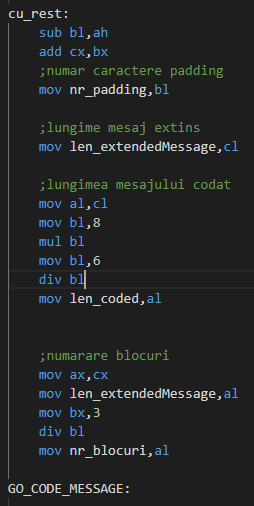
**nr\_padding=0**

Lungimea mesajului extins va fi considerate egal cu lungimea mesajului .Adica din toate caractere din **extended\_message** vom coda doar primele **len\_extendedMessage** caractere:

**len\_extendedMessage=msglen**

Iar numarul de blocuri(**iterations**) va fi catul impartirii mesajului la 3:

**nr\_blocuri=msglen div 3**

**Daca restul este diferit de 0** se vor adauga octetii de padding.

**nr\_padding=3 - (msglen mod 3)**

Lungimea numarul de caractere va fi suma dintre lungimea mesajului si nr de caractere de padding.

Adica se vor coda (msglen+3-rest) caractere din **extended\_Message**

**Len\_extendedMessage = msglen + nr\_padding**

Numarul de blocuri(**iterations**) va fi:

**nr\_blocuri = len\_extendedMessage div 3**

Numarul de carctere codate va fi:

**len\_coded = (len\_extendedMessage div 8) mul 6**

* **Logica criptarii**

In variabila **extended\_Message** se afla mesajul de input urmat de **80-msglen** octeti cu valoarea 0.

Din **extende\_Message** vom lucra doar pe primele **len\_extendedMessage** caractere pe care le vom codifica

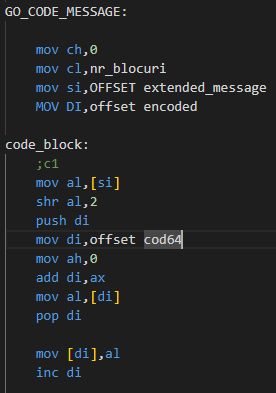
Codificarea se va executa secvential la nivelul unui **bloc** care reprezinta un calup de 3 octeti.Acesti 3 octeti vor fi codificati in 4 caractere din alfabetul **cod64** prin extragerea tuturor secventelor de 6 biti din cei 3 octeti

Fiecare caracter din **cod64** are valoare sa in 6 biti echivalenta cu pozitia sa in variabila.

Astfel ca ,atunci cand extragem o secventa de 6 biti,aflam valoarea acelei secvente, si introducem in variabila oferita de scheletul proiectului,**encoded,**

caracterul din **cod64** care se afla la offset-ul oferit de valorea celor 6 biti.

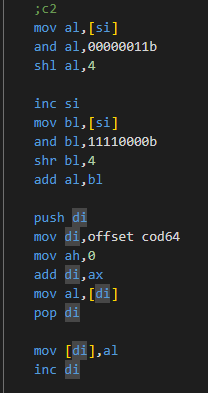
* **Obtinerea caracterelor codate**



Primul caracter codat se obtine din cei mai semnificativi 6 bitii din primul caracter al **blocului**

Se extrag acesti biti si se shift-eaza la dreapta cu 2 pozitii.La final se introduce in encoded caracterul din

**cod64** aflati la offset-ul dat de valoare celor 6 biti.



Al doilea caracter se obtine din cei mai putin semnificativi 2 biti din primul octet si cei mai semnificativi 4 biti din al doilea octet.

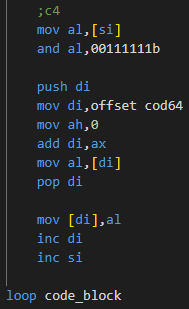
Ca sa obtinem aceasta valoare salvam cei 2 biti din primul octet in **al** si facem o shiftare la stanga cu 4 pozitii.La **al** adaugam **bl** care contine cei mai semnificativi 4 biti din al 2-lea octet shiftati la dreapta cu 4 pozitii.In final se adauga pe urmatoarea pozitie din **encoded** caracterul de pe pozitia valorii retinute in **al** din **cod64.**

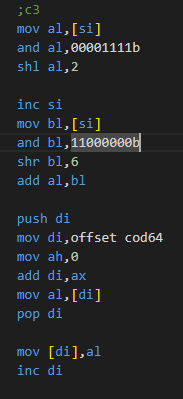
Al treilea caracter se obtine din ce mai putin semnificativi 4 biti din al doilea octet si cei mai semnificativi 2 biti din al treilea octet.

Ca sa obtinem aceasta valoare salvam cei 4 biti din al 2-lea octet in **al** si facem o shiftare la stanga cu 2 pozitii.La **al** adaugam **bl** care contine cei mai semnificativi 2 biti din al 3-lea octet shiftati la dreapta cu 6 pozitii.In final se adauga pe urmatoarea pozitie din **encoded** caracterul de pe pozitia valorii retinute in **al** din **cod64.**

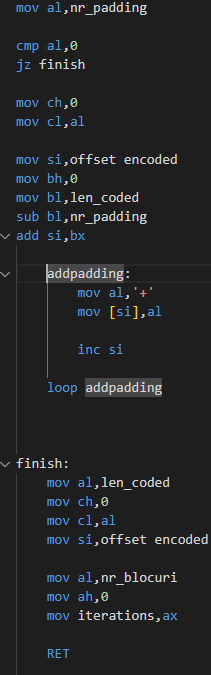
Ultimul caracter codat din **bloc** se obtine din cei mai nesemnificativi 6 biti din al 3-lea octet.

Valoarea acestor biti este stocata in **al.**Pe urmatoarea pozitie din **encoded** se adauga caracterul din **cod64** de la pozitia **al.**



Acest algoritm se aplica pentru fiecare **bloc** din **extended\_Message**.

Dupa aceasta codare trebuie sa fie setate si caractere de padding.



Daca din calculele facute anterior rezulta ca exista caractere de padding acestea se vor seta pe ultime valori

din **encoded** .

Registrul  **SI** va fi setat pe pozitia primul caracter de padding prin scaderea din numarul total de caractere obtinut dupa codare,numarul de caractere de padding

**SI = offset encoded + (len\_coded – nr\_padding)**

Daca nu exista caractere de padding nu se va modifica **encoded** si se va sari la sfarsitul programului unde in variabila **iterations** va fi pusa valoarea din **nr\_blocuri.**