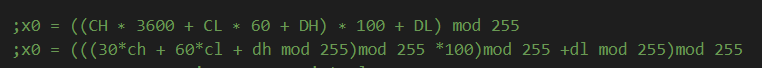
Documentație

Pentru criptare, am început cu calculul pentru x0.



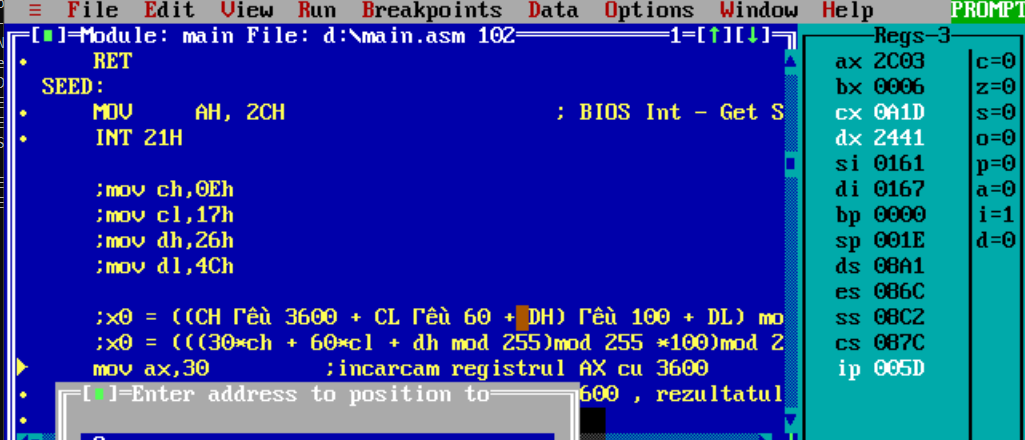
Formula inițială pentru x0 am rescris-o pe baza formulelor :

(a\*b) mod n=[(a\*mod n)(b\*mod n)]mod n

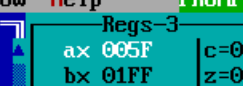
(a+b) mod n=[(a\*mod n)+(b\*mod n)]mod n

Le-am folosit pentru ca așa rezultatele sunt doar pe 1 octet.

În eticheta SEED, am calculat x0 așa cum am descris mai sus.

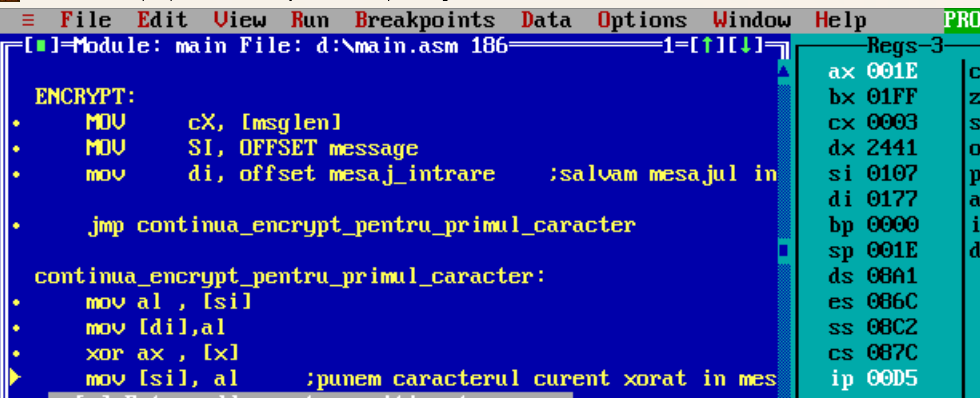


Este ora 10:29, astfel CH=0A, CL=1D, DH=24 și DL=41. Rezultatul final pentru x0 este stocat initial in AX, apoi mutat în variabilele x0 și x, iar acestea în urma calculului vor avea valoarea 5Fh.



După calculul lui x0, am sărit în eticheta ENCRYPT, unde în registrul SI setăm primul caracter din mesajul de intrare pentru a-l xora cu x0. Am făcut o etichetă “continua\_encrypt\_pentru\_primul\_caracter” în care am xorat primul caracter din mesajul de intrare cu x0 ,urmând apoi celelalte caractere să fie xorate cu urmatoarele valori din x .

După prima xorare, vom obține valoarea 1Eh salvată în AX.



Apoi, sărim în eticheta “return\_SEED” unde setam registrul CX cu valoare 6 , care reprezintă lungimea prenumelui meu, și BX cu 0, unde urmează să stocăm suma valorilor literelor ASCII în format hexazecimal din prenumele meu. Lucrurile acestea le fac pentru a calcula a-ul ce mă ajută în calcularea x-ului următor.

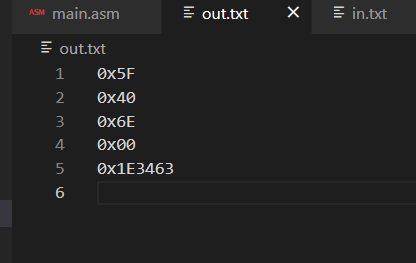
Sărim în RAND unde salvăm primul caracter din prenume în SI și primul din nume pe DI.

Am facut funcții pentru calcularea a-ului și b-ului, iar valorile lor sunt standard 40h respectiv 6Eh.

După calcularea b-ului, continuăm expresia de calcul pentru x, sărind în eticheta “continua\_expresia”. Ultima valoare a lui x va fi afișată în fișierul de ieșire, și anume: 00h.

Eticheta “continua\_encrypt” este pentru xorarea urmatoarelor valori din x cu următoarele elemente ale mesajului. De fiecare dată după ce calculez un x, sar în “continua\_encrypt” pentru a-l xora cu caracterul respectiv.

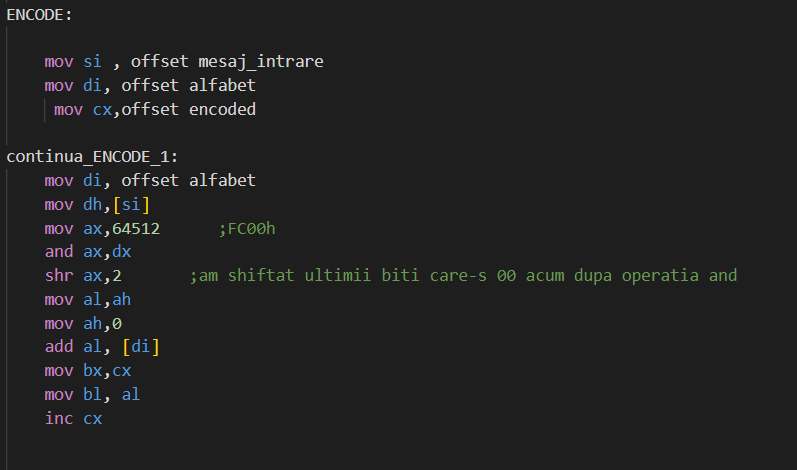
Așa arată output-ul :



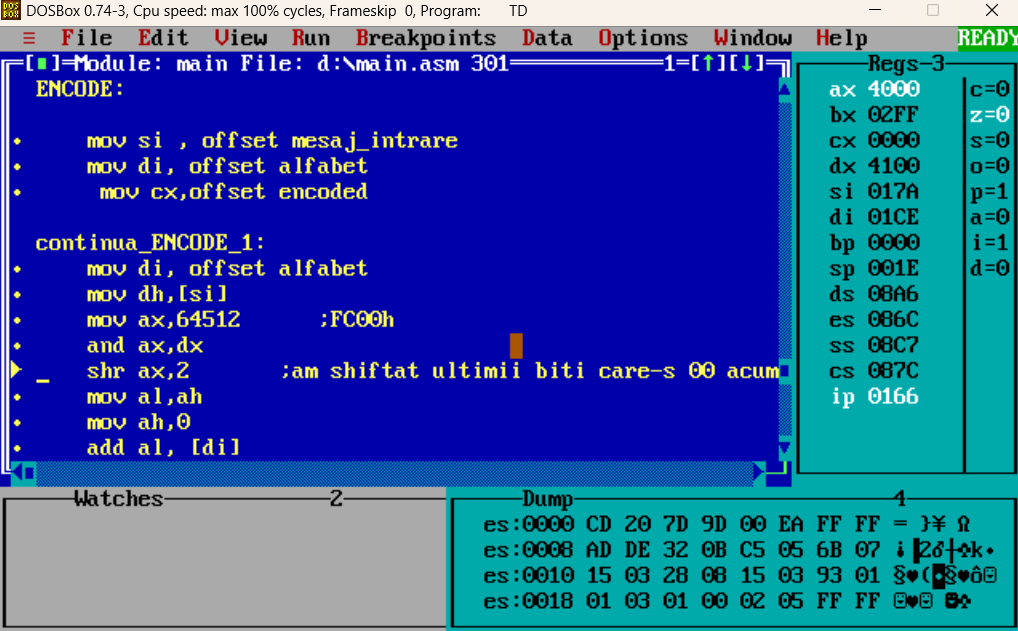
Iar mesajul de intrare este cuvântul Arc.

Pentru funcția ENCODE am creat o variabila numită ”mesaj\_intrare” în care am salvat mesajul din fișierul in.txt înainte să fie criptat. Am făcut în această etichetă 3 etichete mai mici ”continua\_ENCODE\_1”, ”continua\_ENCODE\_2” și ”continua\_ENCODE\_3” pentru că există 3 cazuri în care poți lua cei 6 biți.

continua\_ENCODE\_1:

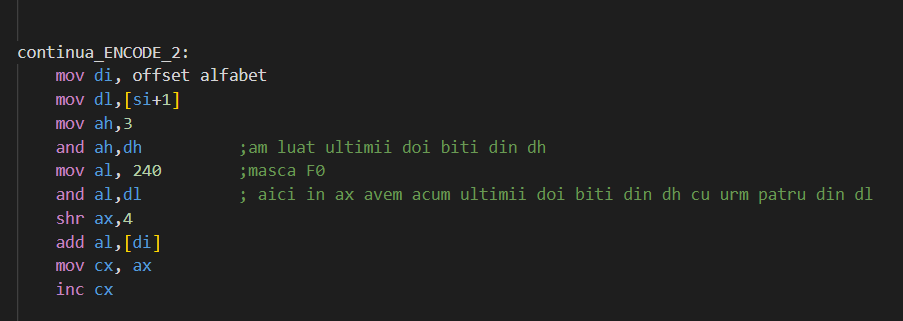


Iau primii 8 biți din mesaj și îi pun în DH. În AX pun masca FC00h și execut operația AND între cele două pentru că vreau să mi se salveze doar primii 6 biți în AX .



După această operație , rezultatul se află în AH însă cu ultimii doi biți 00, iar eu vreau să scap de ei, să-mi rămână doar cei 6 de care am nevoie, de aceea shiftez registrul AX la dreapta cu 2.

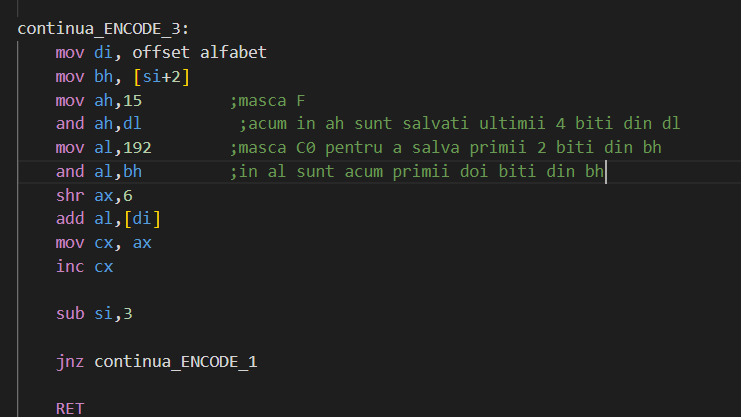
Vreau să mut rezultatul din AH în AL , iar acestuia să-i adaug valoarea de la adresa de început a vectorului encoded, ca să-mi indexeze către litera corespunzătoare în alfabet (am luat o variabilă numită ”alfabet” în care am băgat toate literele din alfabetul dat).

continua\_ENCODE\_2: 

Pun în registrul DL următorul caracter din mesajul de intrare. Pun în AH masca 3 și execut operația AND cu DH pentru a salva ultimii doi biți din DH în AH. Apoi pun masca F0 în AL pentru execuția operației AND cu AL și DL pentru a salva în AL primii 4 biți din DL. Apoi shiftez totul la dreapta cu 4 pentru a stoca pe AL valoarea următoare de 6 biți.

Din nou adaug la registru valoarea de început a alfabetului pentru a obține indexul corespunzător literei.

continua\_ENCODE\_3:



În BH introduc valoarea următorului caracter din mesajul de intrare. În AH mut masca F pentru că vreau să am în el ultimii 4 biți din DL după ce execut ”and ah,dl”.

În AL adaug masca C0 pentru a salva primii doi biți din BH după ”and al,bh”. Acum în AX am cei 6 biți pe care trebuie să-i shiftez la dreapta cu 6 pozitii pentru a salva exact valoarea indicată de acei 6 biți. Din nou, adaug adresa de început a alfabetului la AL, unde sunt salvați biții.

Din SI, registrul care conține adresele caracterelor din mesajul de intrare, scădem 3 pentru a știi programul când să termine de codificat elementele.

Dacă nu a ajuns la finalul mesajul,acesta sare înapoi la ”continua\_ENCODE\_1” pentru că se repetă modul în care grupăm cei 6 biți.

În eticheta ”ENCODE” adăugasem ”mov cx, offset encoded” și adăugam de fiecare data când codificam un caracter din mesaj în vectorul encoded prin ”mov cx,ax ”.