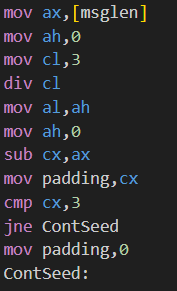
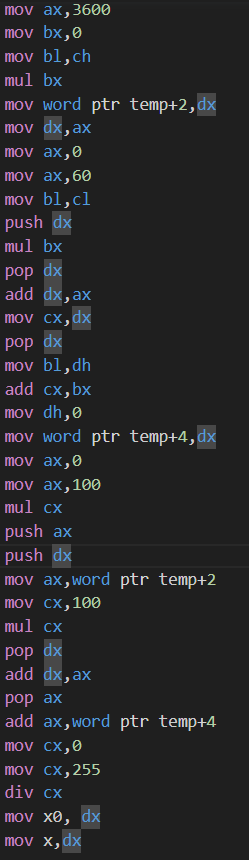
Documentație Temă Arhitecturi

1. Eticheta SEED:

* In Figura 1 am folosit variabila padding pentru a salva, in functie de lungimea mesajului din fisierul de intrare, lungimea padding-ului necesar.



Figura



In aceasta secventa de cod am implementat calculul variabilei x0.

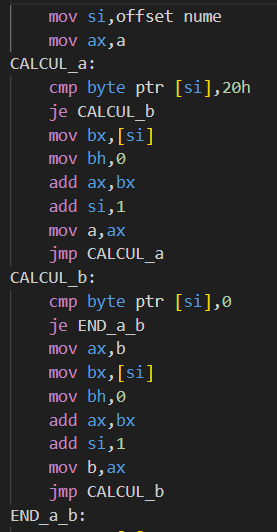
Aici am folosit variabila temp pentru a stoca temporar valoarea lui dx. Am folosit-o pentru ca stiam ca este un loc liber, care la un moment dat va fi suprascris in functiile care nu trebuiau modificate.

Am calculat x0 prin realizarea tuturor calculelor, apoi la final am facut si operatia de mod 255.

Am avut grija ca atunci cand ora sistemului este 19 sau mai mare, sa tin cont si de octetii din dx de dupa inmultirea 19x3600=68.400 (1 0B30 h). Am rezolvat aceasta problema prin stocarea octetului 1 din dx in temp+2, de unde l-am preluat in ax pentru a face si inmultirea acestuia cu 100.

Figura

1. Eticheta ENCRYPT:

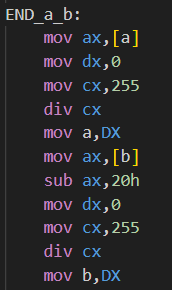


In aceasta secventa am calculat variabilele a si b.

Aici am verificat daca am ajuns la caracterul spatiu pentru a sti cand am terminat cu variabila a.

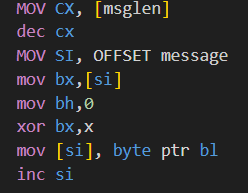
La aceasta linie am verificat daca am ajuns la final ul variabilei pentru a sti ca am terminat de adunat valorile ASCII din nume.

Figura



Pentru finalizarea calcularii variabilelor a si b am facut operatia de mod 255 pentru fiecare.

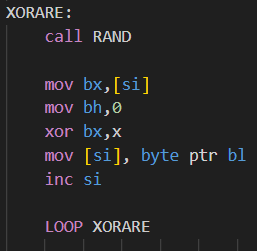
Figura



In aceasta secventa am facut pregatirea pentru codificarea propriu-zisa, aceasta constand in:

* Mutarea in cx a dimensiuniii mesajului
* Mutarea registrului SI la inceputul mesajului
* Prima xorare (a primului octet din mesaj cu x=x0)

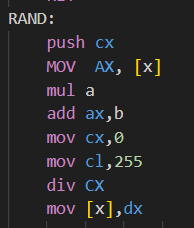
Figura



Aici am realizat operatia de xor a fiecarui termen din mesaj cu xn-ul respective.

Functia RAND este prezentata in Figura 7.

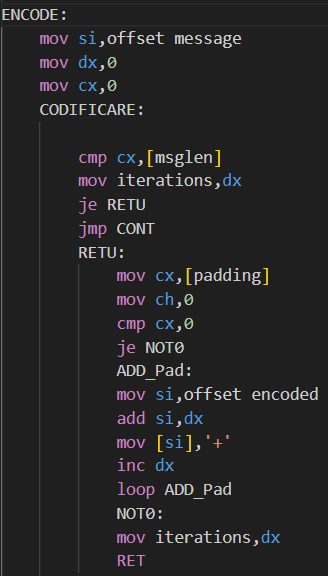
Figura



In aceasta functie am calculat x-ul urmator.

Figura

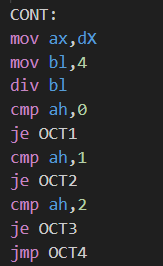
1. Eticheta ENCODE:



Din functia ENCODE dau return atunci cand valoarea din registrul cx ajunge la sfarsitul mesajului.

In cx tin numarul caracterului la care am ajuns din mesaj, iar in dx tin numarul caracterului la care am ajuns din mesajul codificat.

In cadrul RETU adaug padding-ul necesar, la final de encoded, apoi dau return.



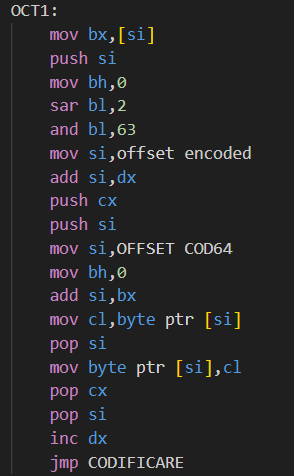
Am gandit sa fac codarea astfel:

* Fiecare octet din encoded este codificat in felul sau, dar procesul se repeta din 4 in 4 octeti
* Asa ca in etichetele OCT1-OCT4 am facut codificarea pentru fiecare dintre acestia
* Ma deplasez in paralel atat pe mesaj cat si pe encoded

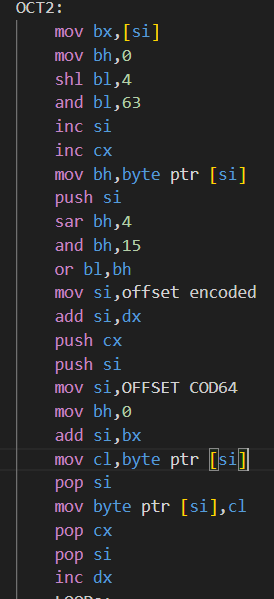
Figura

Explicatie:

* am luat octetul din mesaj
* am shiftat cu 2 octetul pentru a obtine cei mai importanti 6 biti din acesta
* am facut and cu 63 (0011 1111) pentru a lua in registrul bl doar primi 6 biti
* am luat caracterul corespunzator din COD64 si l-am concatenat la encoded



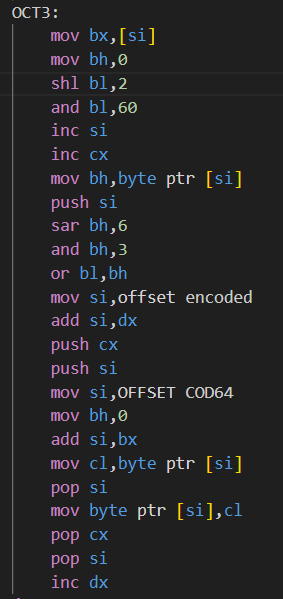
Figura



Explicatie:

* Am shiftat la stanga cu 4 si am facut and cu 63 pentru a lua ultimi 2 cei mai putin semnificanti biti din primul octet al mesajului
* Am trecut la al doilea octet al mesajului
* Am shiftat la dreapta cu 4 si am facut and cu 15 (0000 1111) pentru a lua cei mai semnificanti 4 biti din acest octet
* Am facut “or bl,bh” pentru a combina intr-un octet cele doua secvente de biti
* Am luat caracterul corespunzator din COD64 si l-am concatenat la encoded

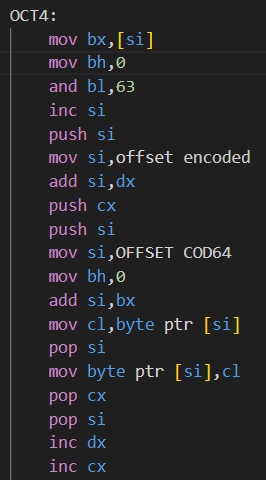
Figura



Explicatie:

* Am salvat in bl octetul din mesaj
* Am facut shiftare cu 2 la stanga si am dat and cu 60 (0011 1100) pentru a lua cei mai putin semnificanti 4 biti din acesta
* Am trecut la urmatorul caracter din mesaj
* Am luat in bh caracterul si am dat shift cu 6 la dreapta apoi am dat and cu 3(0000 0011) pentru a lua cei mai semnificanti 2 biti din acesta
* Am combinat bl si bh prin operatia or
* Am concatenat caracterul din COD64 ce corespunde valorii rezultate

Figure



Explicatie:

* Am luat in bx caracterul din mesaj
* Am facut and cu 63 (0011 1111)pentru a lua cei mai putin semnificanti 6 biti
* Am concatenat caracterul din COD64 ce corespunde valorii rezultate

Figure

