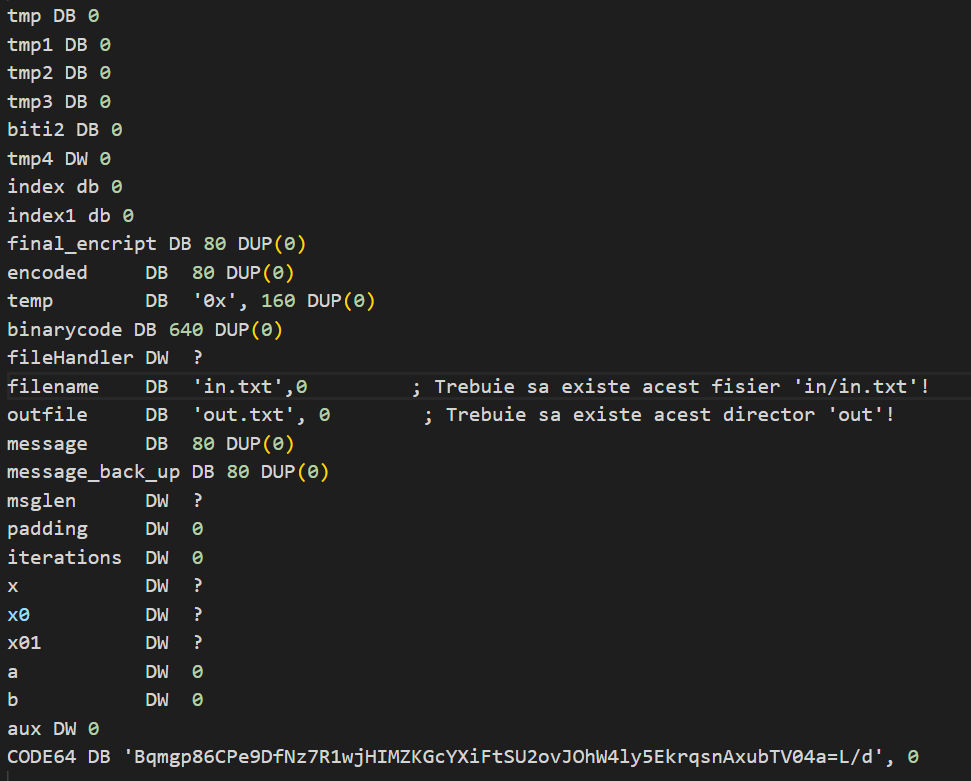
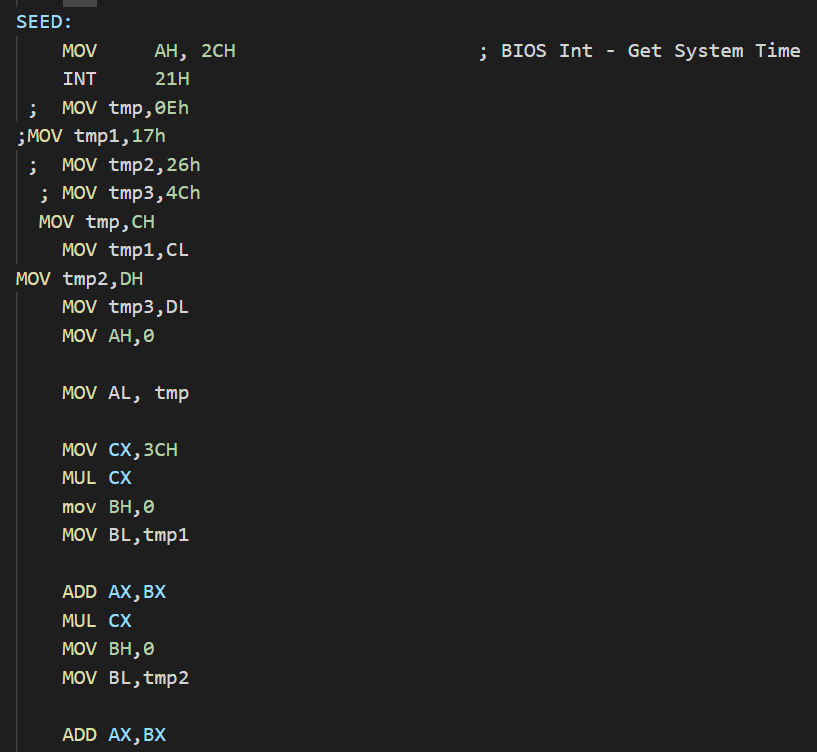
READ ME

Tema este facută integral (functiile a,b,x0,x,encrypted si encoded rezolvate), are 702 linii de cod. Acest cod atunci cand este executat transforma un continut dintr un fisier (in.txt ) intr un fisier de output cryptat(out.txt).



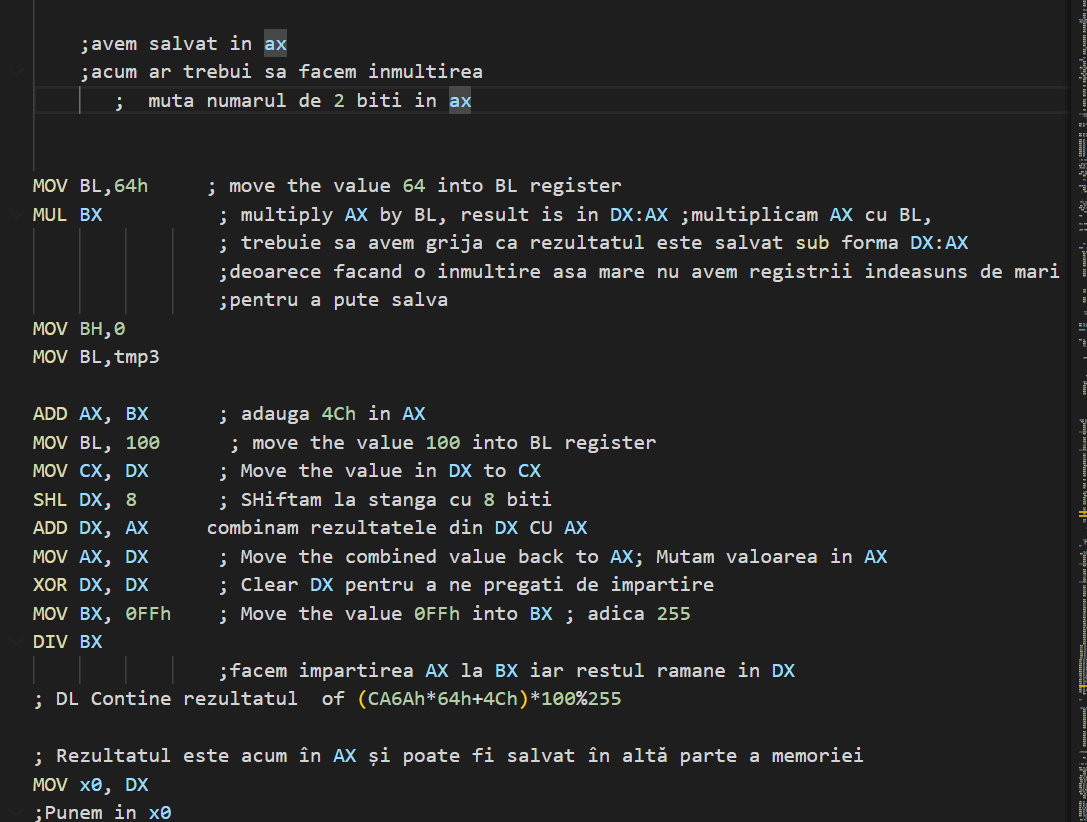
Segmentul de date: tmp,tmp1,tmp2,tmp3 sunt folosite pentru a fi tinute minte ora minutul secunda si sutimile de secunda atunci cand apasam RUN programului.



Primul pas pentru a face criptarea este popularea termenului x0.

Se foloseste formula respectivă:



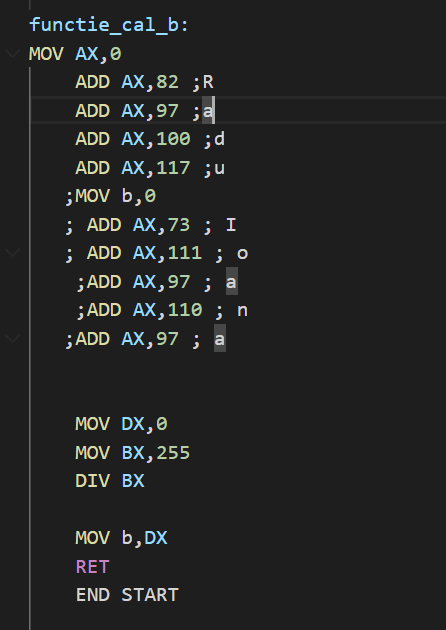
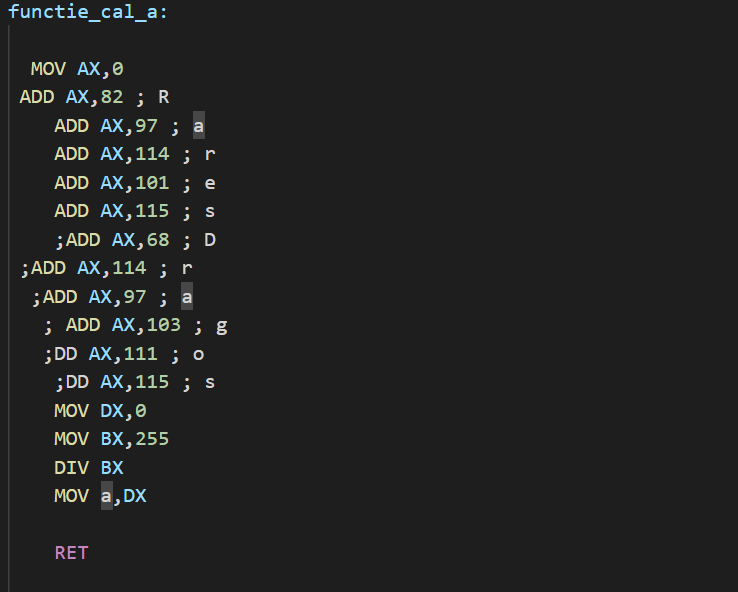




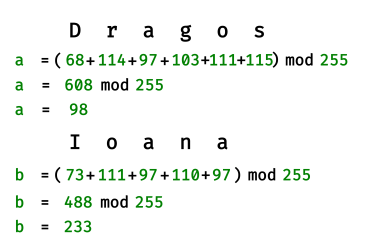


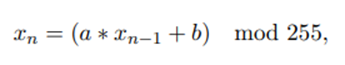
Avem nevoie sa calculam x, pe acesta il calculam in următoarea functie este de encrypt, care transforma stringul din input intr un string. Este prima criptare, criptarea finala se afla in functia de encoded care se foloseste de prima criptare .

Aceasta functie transforma stringul intr o cheie unica deoarece se foloseste de ora exacta salvata in x0 si de numele administratorului in functia encrypted.



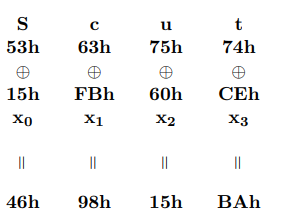
Cele doua functii vor sa populeze a si b in felul acesta

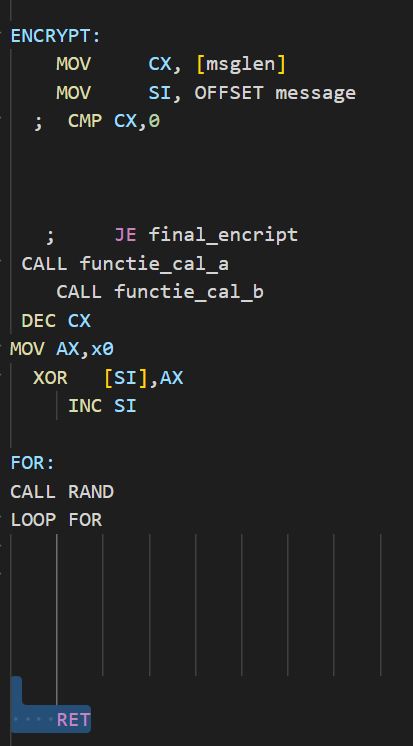




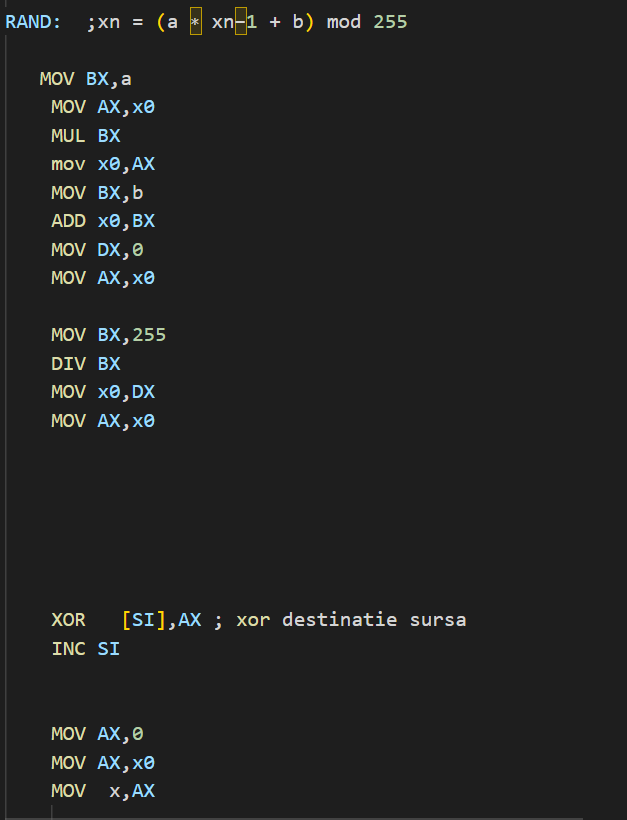
Numarul de iteratii ,n , o sa fie dat de numarul de caractere initial, din file\_input.

Pentru fiecare x se mai face pe deasupra si un XOR intre x de iteratia curenta cu caracterul respectiv



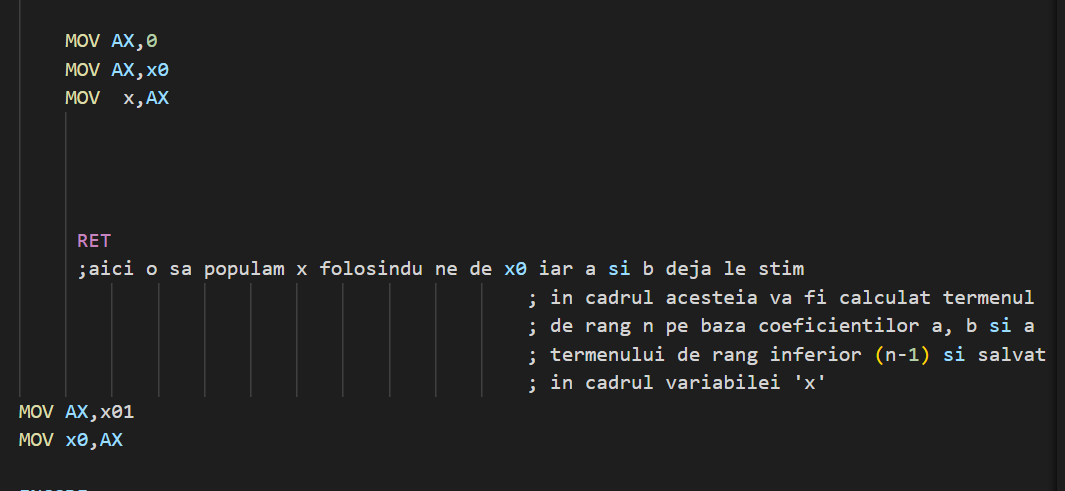


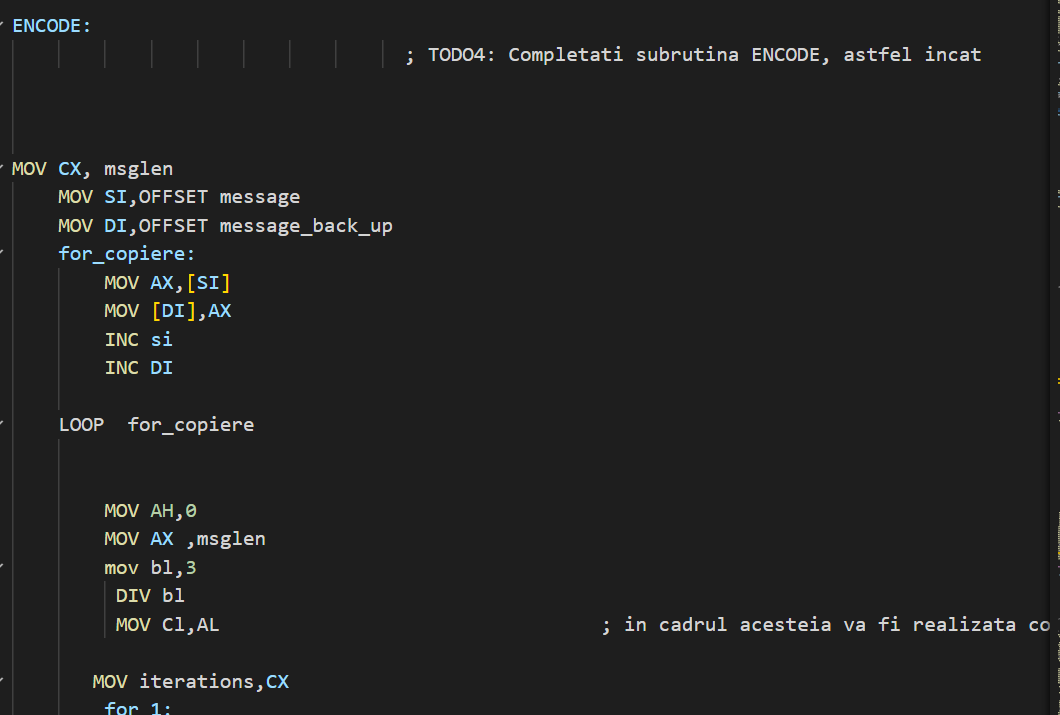
Functia rand calculeaza fiecare x in functie de previous x dat. La inceput se incepe cu x0, aflat in functia de seed.



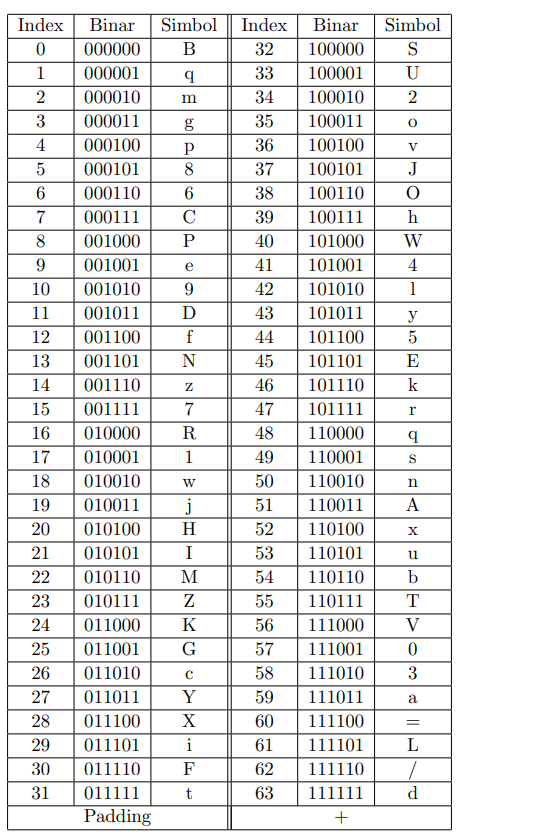
Se foloseste formula iar pentru a salva x-ul recurent o sa il punem in x0.

Pentru fiecare x o sa se face mod FFh(255).



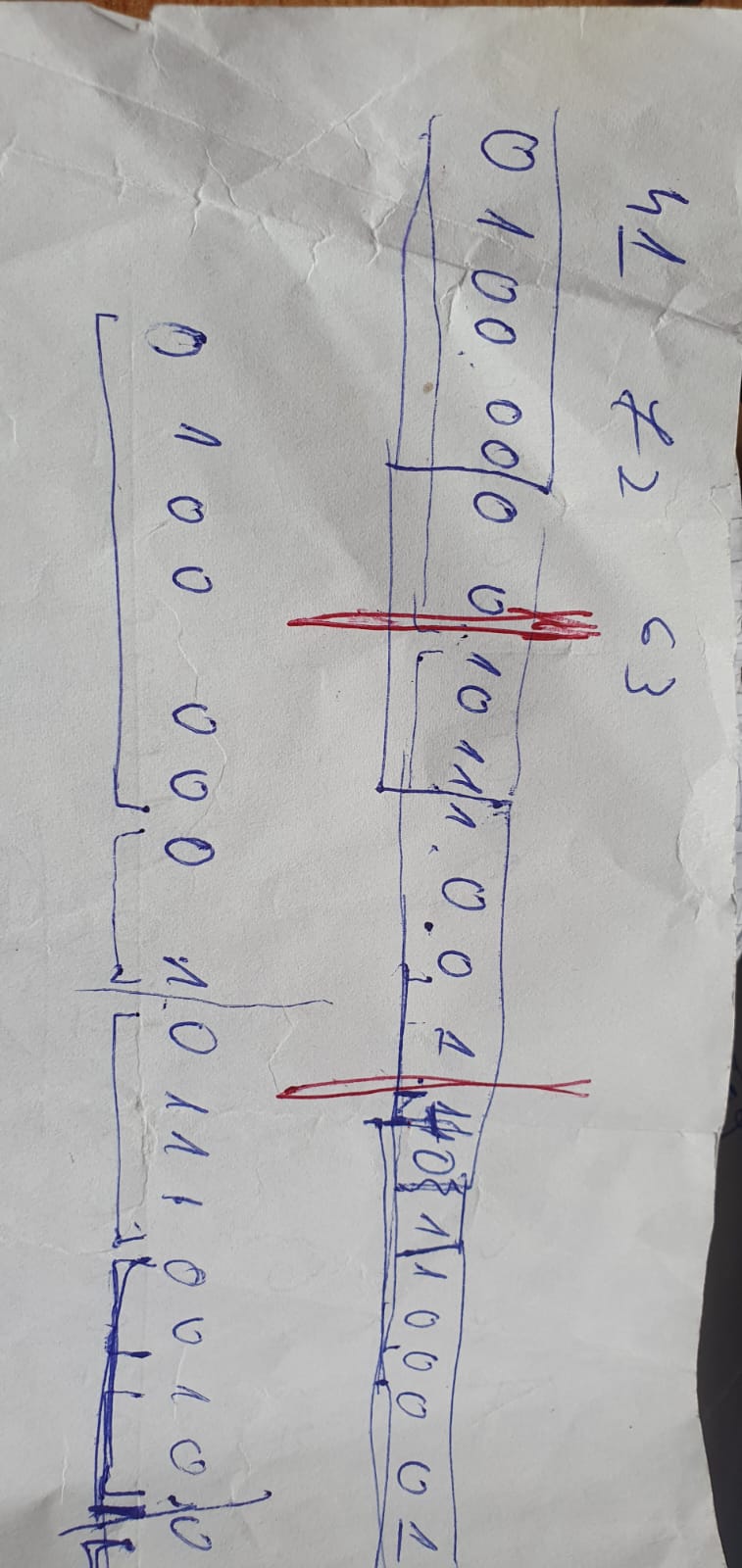


Pentru ultima functie o sa luam bufferul dat din prima criptare iar pentru a face ultima criptare o sa regrupam toti bitii 6 cate 6 iar fiecare calup o sa fie interpretat in functie de tabelul respectiv



Avem trei cazuri, pentru a calcula mai eficient o sa luam grupuri de 3 octeti deoarece 3\*8=24 iar 4\*6=26, din 3 octeti o sa avem 4 octeti mai speciali.

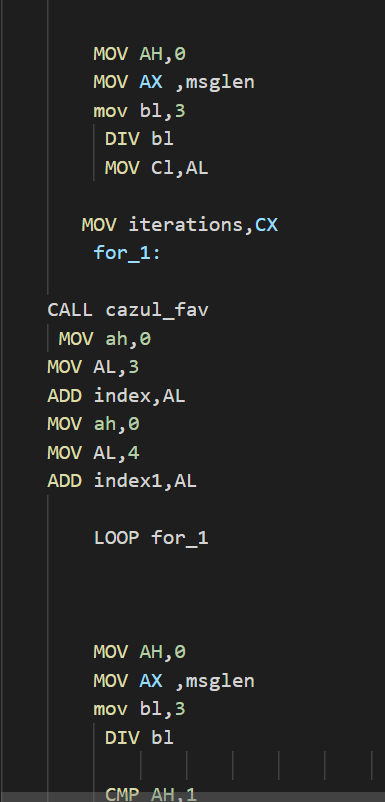
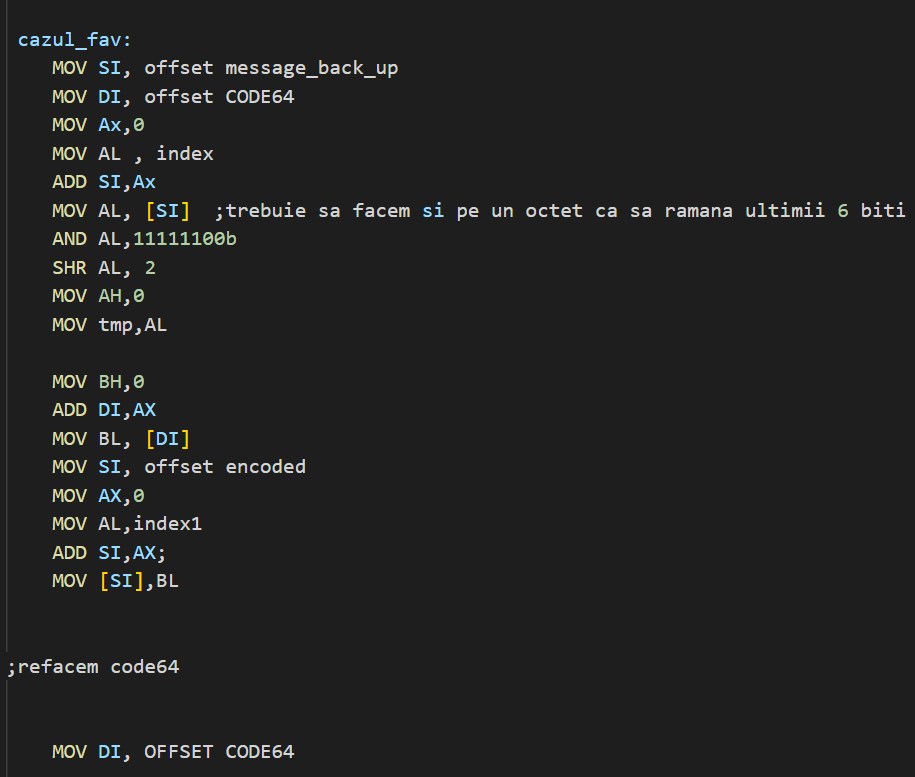
Primul caz numarul de caractere din buffer este divizibil cu 3.



Al doilea caz numarul de caractere nu este divizibil si da restul 1. Ultimul octet o sa transformat in 2 de 6 biti si vom mai adauga inca 2 plusuri. Pentru a face bloc de 3 octeti .

Al treilea caz numarul de caractere nu este divizibil si da restul 2 . Din cei doi octeti o sa transformam in 3 de 6 si vom mai adauga un plus.

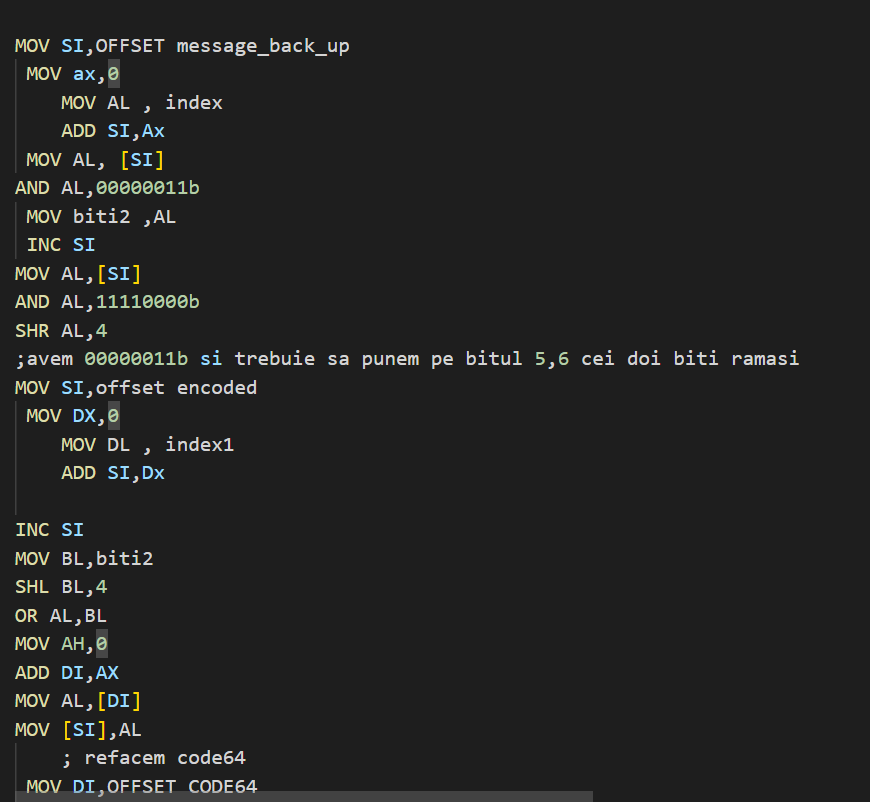
Nu conteaza cat de lung este bufferul deoarece vom face lungimea\_buffer/3 repetari.

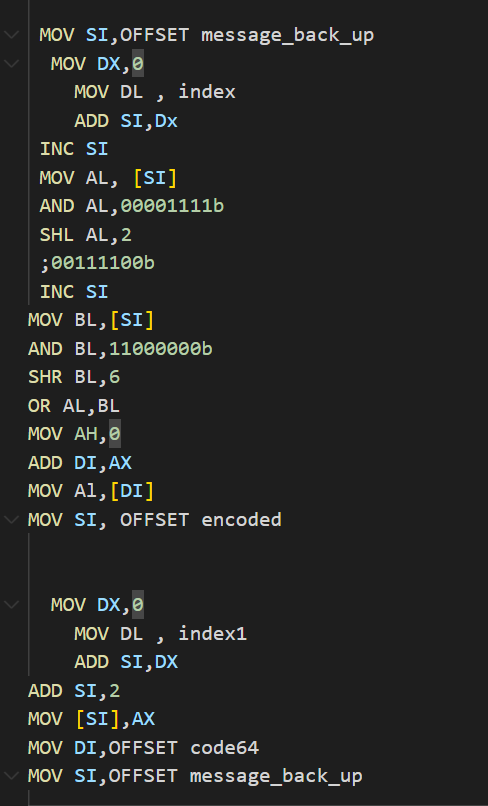
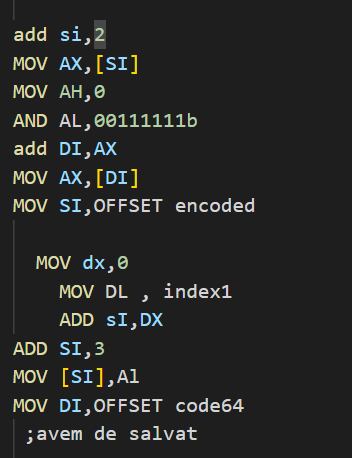
 

Am facut un message\_back\_up pentru ca initial ma gandeam sa modific message odata cu urmatorul bloc de 3 .

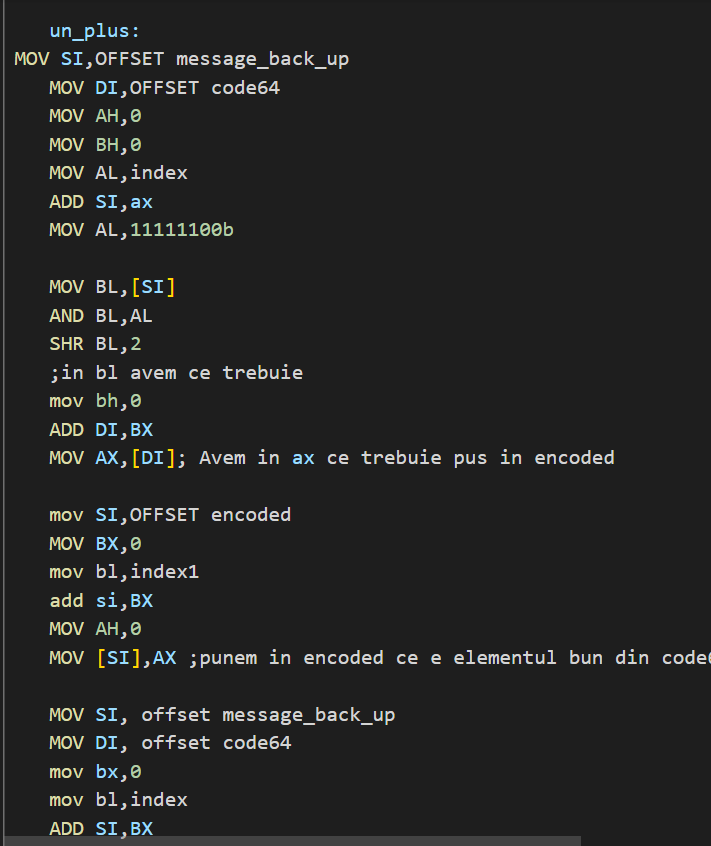
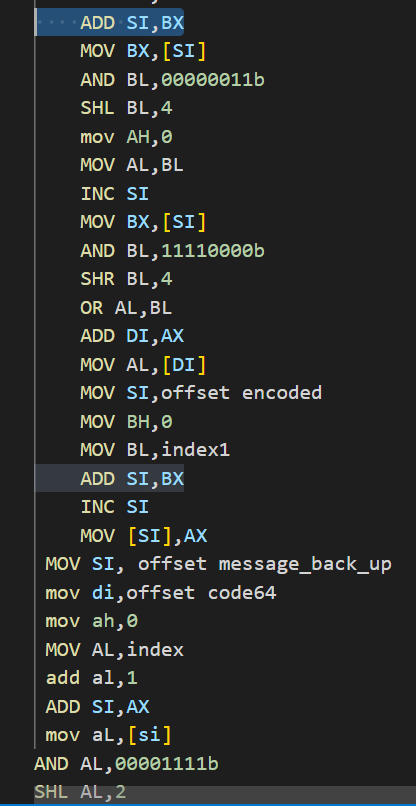
Index si index 1 se ocupa cu bucla pentru a lua din message\_back\_up si de a pune unde trebuie in encode. La inceput amandoua vor fi 0.

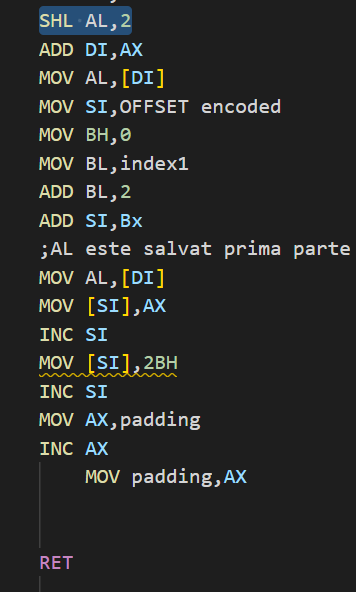
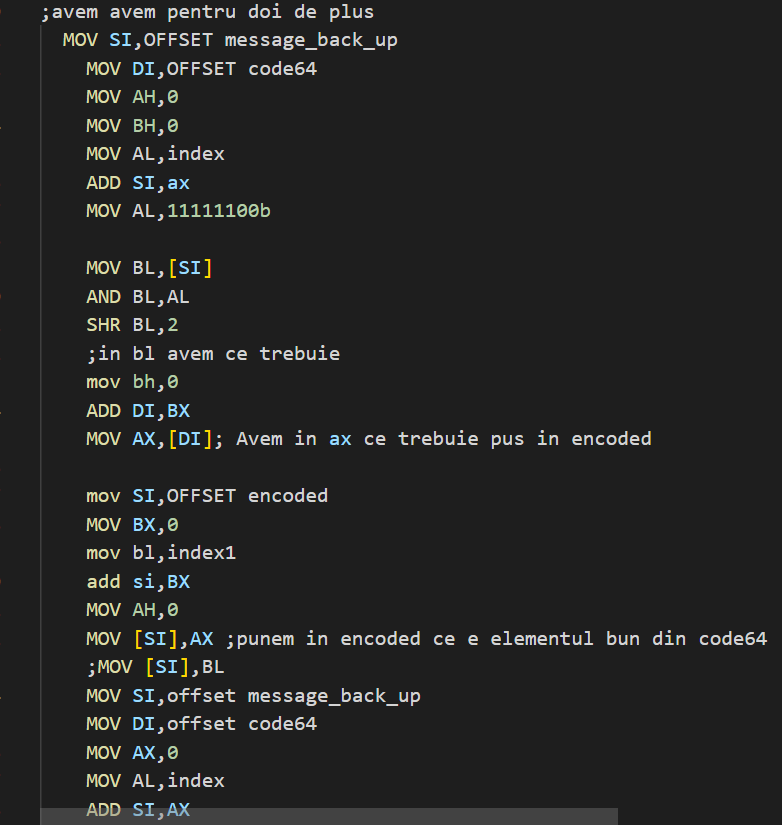
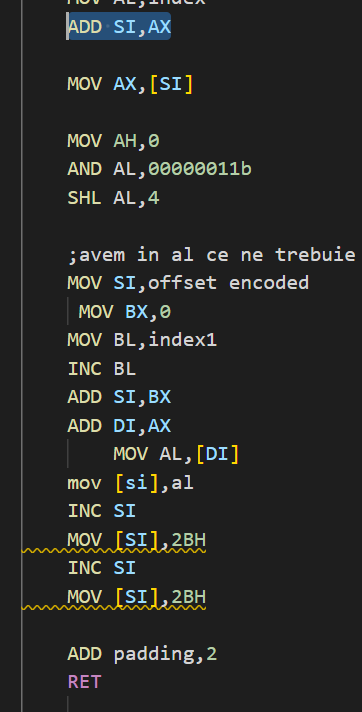
Prima data vom lua primii 6 biti din primul octet facem shiftare iar apoi o sa il adunam cu DI iar elementul de pe pozitia octetului de 6 biti gasiti va fi pus in encoded. Pentru urmatorul caracter luam ultimii doi biti din primul octet al mesajului si primii 4 biti din al doilea octet. II dispunem facand shiftari iar apoi facem un OR pentru a construi octetul de 6 biti, apoi luam iar elementul din code64. Pentru urmatorul element o sa luam ultimii 4 biti din al doilea octet si primii 2 biti din al treilea octet.Pentru ultimul, o sa luam ultimii 6 biti din al 3 lea octet din message. Asa am construit din 3 octeti din message 4 octeti, fiecare luat din code64.



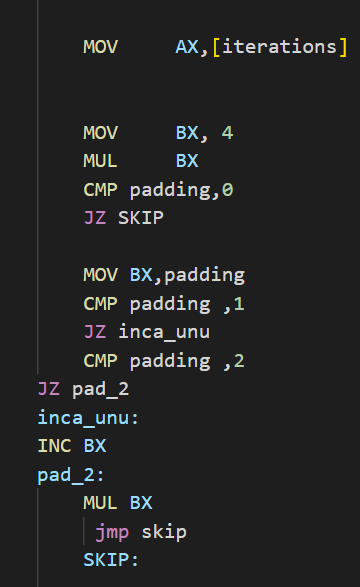


Cazurile 2 si 3 se construiesc dupa cazul 1. In encoded avem salvat bufferul mai putin un octet sau 2 octeti. Octetul/octetii ramasi vor fi manipulati separat. In padding salvam cate plusuri avem, pentru afisare.

Am modificat si functia de write pentru a afisa ce trebui din encoded.



Pentru a arata ca sursa functioneaza am sa iau exemplele date .

