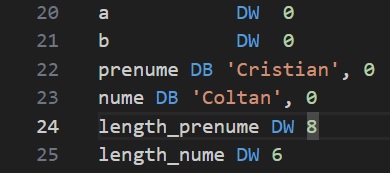
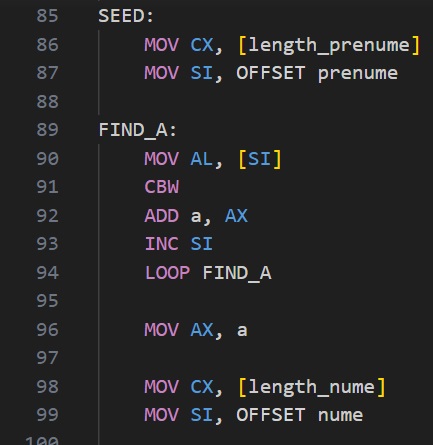
# ***Detaliere + Rationament lucru***

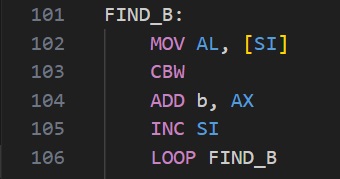
Proiectul complet se poate rula din sursa: ***source.asm.***

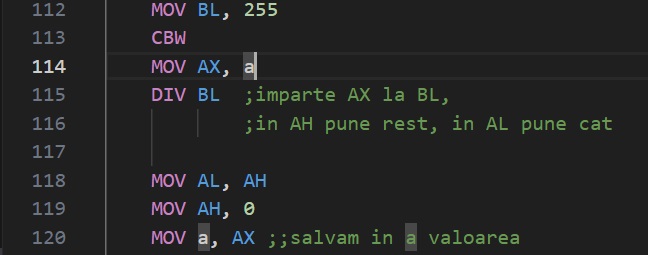
In primul rand am calculat valorile a si b.

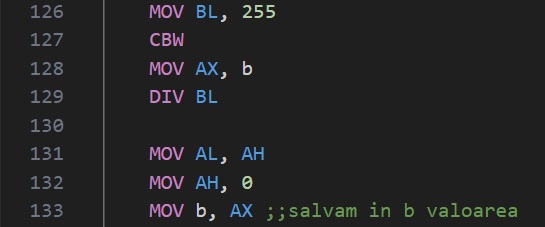


M- am folosit de sirurile **prenume** si **nume** cat si de **length\_prenume**, **length\_nume** pentru a calcula rapid a si b. Pun in registrul CX lungimea sirului, iar in SI offset ul, iar apoi folosesc un loop pentru a aduna valorile hexa in variabila a. La fel procedez si pentru variabila b. Aplic modulo 255 si salvez rezulatul in a, respectiv b.



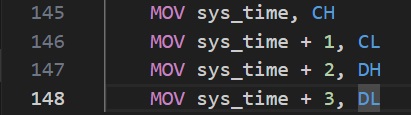






Apoi a trebuit sa calculez x0. Am folosit intreruperea 2CH. M- am folosit de un sir **sys\_time** pentru a- mi fi mai usor sa manipulez cele 4 valori. Apoi am trecut la calculul in sine, unde m- am ajutat de **xaux** pentru a calcula pe x0.





;CH \* 3600

    MOV AX, 0

    MOV AL, [sys\_time]

    MOV BX, 3600

    MUL BX

    MOV [xaux], DX

    MOV [xaux+2], AX

;CL \* 60

    MOV AX, 0

    MOV AL, [sys\_time+1]

    MOV BX, 60

    MUL BX

;CH \* 3600 + CL \* 60

    MOV DX, [xaux]

    ADD AX, xaux+2

    ADC DX, 0

    MOV[xaux],DX

    MOV[xaux+2],AX

;DH

    MOV AX, 0

    MOV AL, [sys\_time+2]

    MOV DX, [xaux]

;CH \* 3600 + CL \* 60 + DH

    ADD AX, xaux+2

    ADC DX, 0

    MOV [xaux], DX

    MOV [xaux+2],AX

;(CH \* 3600 + CL \* 60 + DH) \* 100

    MOV BX, 100

    MOV CX, DX

    MUL BX

    MOV [xaux+2], AX

    MOV [x0], DX

    MOV AX,CX

    MUL BX

    ADD DX, AX

    ADD DX, x0

    MOV [xaux], DX

;(CH \* 3600 + CL \* 60 + DH) \* 100 + DL

    MOV AX, 0

    MOV DX, xaux

    MOV AL, [sys\_time+3]

    ADD AX, xaux+2

    ADC DX, 0

    MOV [xaux], DX

    MOV [xaux+2],AX

;(CH \* 3600 + CL \* 60 + DH) \* 100 + DL mod 255

    MOV CX, 255

    DIV CX

    MOV x0, DX

    MOV xx0, DX

Urmeaza calcularea a **msglen**  **- 1** iteratii pentru a afla X(n-1). In acelasi timp aplic si operatia de XOR pentru a afla sirul CRIPTAT. Prima data am calculat pentru X0 valoarea criptata, apoi am automatizat printr -un loop.

  MOV SI, OFFSET message

    MOV CX, msglen

    DEC CX ;facem msglen -1 iteratii

    MOV x, DX

    ;MOV x, 15h ;- test 1 (Scut)

    MOV AX, 0

    MOV AL, [SI]

    MOV BX, [x]

    XOR AX, BX

    MOV [SI], AL

    INC SI

RANDD:

    MOV AX, [x0]

    MOV BX, a ;a ;62h- test 1(Scut) ;;a=40h-Cristian

    MUL BX

    MOV BX, b ;b ;233- test 1(Scut) ;;b=63h-Coltan

    ADD AX, BX

    MOV BX, 255

    DIV BX

    MOV [x], DX

    MOV AX, 0

    MOV AL, [SI]

    MOV BX, [x]

    XOR AX, BX

    MOV [SI], AL

    INC SI

    MOV x0, BX

    LOOP RANDD

    MOV AX,[xx0]

    MOV [x0],AX

Dupa ce am criptat sirul, am trecut la codificarea acestuia. M- am folosit de tabelul de la pagina 9, transformand- ul intr- un sir pentru a putea codifica sirul **message**.

tabel DB 'Bqmgp86CPe9DfNz7R1wjHIMZKGcYXiFtSU2ovJOhW4ly5EkrqsnAxubTV03a=L/d', 0

Am aflat cati octeti pentru padding trebuie sa folosesc si i-am adaugat la sirul **message**.

MOV AX, msglen

    MOV BX, 3

    DIV BX

    MOV [iteratii\_padd], DX

    MOV padding, DX

    MOV iterations, AX

    MOV CX, AX

    CMP DX, 1

    JE PADDING2

    CMP DX, 2

    JE PADDING1

    JMP endd

    PADDING2:

    MOV SI, OFFSET message

    MOV AX, msglen

    ADD SI, AX

    MOV byte ptr [SI], 0

    MOV byte ptr [SI+1], 0

    INC AX

    INC AX

    MOV msglen, AX

    JMP endd

PADDING1:

    MOV SI, OFFSET message

    MOV AX, msglen

    ADD SI, AX

    MOV byte ptr [SI], 0

    INC AX

    MOV msglen, AX

    JMP endd

Am trecut la codarea in sine, unde am observat ca procesul de codare se repeta din 3 in 3 octeti, rezultand 4 caractere. Am hardcodat acest lucru, folosind masti pe fiecare octet in parte alaturi de shiftari la stanga si la dreapta, rezultand codarea propriu- zisa.

MOV DI, OFFSET encoded

    MOV SI, OFFSET message

criptare\_loop:

    MOV AL, [SI]

    AND AL, 252

    SHR AL, 2

    PUSH SI

    MOV SI, OFFSET tabel

    CBW

    ADD SI, AX

    MOV BL, [SI]

    MOV [DI], BL

;pana aici am criptat bit 1

    INC DI

    POP SI

    MOV AL, [SI]

    AND AL, 3

    SHL AL, 4

    INC SI

    MOV BL, [SI]

    AND BL, 240

    SHR BL, 4

    OR AL, BL

    PUSH SI

    MOV SI, OFFSET tabel

    CBW

    ADD SI, AX

    MOV BL, [SI]

    MOV [DI], BL

;pana aici am criptat bit 2

    INC DI

    POP SI

    MOV AL, [SI]

    AND AL, 15

    SHL AL, 2

    INC SI

    MOV BL, [SI]

    AND BL, 192

    SHR BL, 6

    OR AL, BL

    PUSH SI

    MOV SI, OFFSET tabel

    CBW

    ADD SI, AX

    MOV BL, [SI]

    MOV [DI], BL

;pana aici am criptat bit 3

    INC DI

    POP SI

    MOV AL, [SI]

    AND AL, 63

    PUSH SI

    MOV SI, OFFSET tabel

    CBW

    ADD SI, AX

    MOV BL, [SI]

    MOV [DI], BL

;pana aici am criptat bit 4

    INC DI

    POP SI

    INC SI

LOOP criptare\_loop

MOV DX, iteratii\_padd

CMP DX, 0

JNZ criptare\_padding

JMP fara\_padd

criptare\_padding: ;aici daca avem 000000 o sa inlocuim cu +

    MOV AX, iterations

    INC AX

    MOV iterations, AX

    MOV AX,0

    MOV AL, [SI]

    AND AL, 252

    SHR AL, 2

    PUSH SI

    MOV SI, OFFSET tabel

    CBW

    ADD SI, AX

    MOV BL, [SI]

    MOV [DI], BL

;pana aici am criptat bit 1

    INC DI

    POP SI

    MOV AL, [SI]

    AND AL, 3

    SHL AL, 4

    INC SI

    MOV BL, [SI]

    AND BL, 240

    SHR BL, 4

    OR AL, BL

    CMP AL, 0

    JE zero1

    PUSH SI

    MOV SI, OFFSET tabel

    CBW

    ADD SI, AX

    MOV BL, [SI]

    MOV [DI], BL

    JMP continue1

zero1:

    PUSH SI

    MOV [DI], 43

continue1:

;pana aici am criptat bit 2

    INC DI

    POP SI

    MOV AL, [SI]

    AND AL, 15

    SHL AL, 2

    INC SI

    MOV BL, [SI]

    AND BL, 192

    SHR BL, 6

    OR AL, BL

    CMP AL, 0

    JE zero2

    PUSH SI

    MOV SI, OFFSET tabel

    CBW

    ADD SI, AX

    MOV BL, [SI]

    MOV [DI], BL

    JMP continue2

zero2:

    PUSH SI

    MOV [DI], 43

continue2:

;pana aici am criptat bit 3

    INC DI

    POP SI

    MOV AL, [SI]

    AND AL, 63

    CMP AL, 0

    JE zero3

    PUSH SI

    MOV SI, OFFSET tabel

    CBW

    ADD SI, AX

    MOV BL, [SI]

    MOV [DI], BL

    JMP continue3

zero3:

    PUSH SI

    MOV [DI], 43

continue3:

;pana aici am criptat bit 4

    INC DI

    POP SI

    INC SI

fara\_padd:

Pentru inceput am vazut de cate ori se repeta o secventa de 3 caractere pentru a initializa loop- ul principal. Am salvat cu ajutorul operatiei **DIV**, catul si restul, pentru a vedea numarul de secvente de 3 caractere si numarul de octeti pentru padding. Daca numarul de caractere se impartea exact la 3, nu am mai adaugat padding.

Am verificat daca a fost adaugat padding, iar daca raspunsul a fost afirmativ, am realizat codarea cu padding, unde 000000 in loc sa se transforme in **B**, se transforma in **+**.