Tema arhitecturi

Primul task indeplinit in cadrul temei a fost construirea lui x0 folosind intreruperea 2Ch in functia SEED asftel:

    MOV     AH, 2CH                     ; BIOS Int - Get System Time

    INT     21H

    ;3600\*CH

    MOV AL,CH

    MOV AH,0

    MOV BX,3600

    PUSH DX

    MUL BX

    ;Rezultatul se afla in AX

    ;60\*CL

    MOV SI,AX

    MOV AL,CL

    MOV AH,0

    MOV BX,60

    MUL BX

    ;Adunarea celor doua

    ADD SI,AX

    ;+DH

    POP DX

    MOV AX,0

    MOV AL,DH

    MOV AH,0

    ADD SI,AX

    MOV CX,DX

    ;Gata prima paranteza

    ;\*100

    MOV AX,SI

    MOV BX,100

    MUL BX

    ;+DL

    MOV BX,0

    MOV BL,CL

    ADD AX,BX

    ;MOD 255

    MOV BX,255

    DIV BX

    ;restul impartirii ramane in DX

    MOV [x0],DX

    MOV AX,[x0]

    MOV [x],AX

Astfel, am pregatit variabila x0 pentru XOR-are si construirea in continuare a sirului

In continuare, tot in functia SEED am calculate variabilele a si b, folosindu-ma de variabilele nume si prenume initializate anterior, impreuna cu lungimea acestora calculata la declarare

    ;Calculare a

    MOV AX,0

    MOV DX,0

    MOV CX,0

    MOV SI, OFFSET prenume

    MOV CX,lenm

    LOOP\_A:

    MOV BX,[SI]

    MOV BH,0

    ADD AX,BX

    INC SI

    LOOP LOOP\_A

    MOV BX,255

    DIV BX

    MOV [a],DX

    ;Calculare b

    MOV AX,0

    MOV DX,0

    MOV CX,lenn

    MOV SI,OFFSET nume

    LOOP\_B:

    MOV BX,[SI]

    MOV BH,0

    ADD AX,BX

    INC SI

    LOOP LOOP\_B

    MOV BX,255

    DIV BX

    MOV [b],DX

    RET

Astfel, pentru fiecare rulare a programului a va fi F6h, iar b 27h

In continuare, am completat functia rand astfel incat sa modifice x-ul cu urmatorul termen din sir

RAND:

    MOV     AX, [x]

    MOV BX,[a]

    MUL BX

    MOV BX,[b]

    ADD AX,BX

    MOV BX,255

    DIV BX

    MOV [x],DX

    RET

Iar in functia ENCRYPT am facut XOR cu fiecare termen al sirului

ENCRYPT:

    MOV     CX, [msglen]

    MOV     SI, OFFSET message

    XOR\_EN:

    MOV AX,[SI]

    MOV BH,AH

    MOV AH,0

    XOR AX,[x]

    MOV AH,BH

    MOV [SI],AX

    INC SI

    CALL RAND

    LOOP XOR\_EN

    RET

In continuare,pentru a realiza codarea mesajului criptat, am creat o copie acestuia pe care voi lucra

COPY\_MESSAGE:

    MOV CX,[msglen]

    MOV SI,OFFSET message

    MOV DI,OFFSET message\_copie

    LOOP\_COPY:

    MOV AX,[SI]

    MOV [DI],AX

    INC SI

    INC DI

    LOOP LOOP\_COPY

    RET

Codare nu am reusit sa o realizez decat pentru siruri a caror nume de caractere este divizibil cu 3(adica nu au nevoie de padding), deoarece, pe final, am avut niste erori la padding pe care nu am reusit sa le elimin in timp util.Iata mai jos codul insotit de comentarii explicative al codarii:

ENCODE:

    MOV CX,0

    MOV SI,OFFSET message\_copie

    MOV AX,[msglen]

    MOV BL,3

    DIV BL

    ;restul ramane in AH,catul in AL

    MOV CL,AL

    MOV [iterations],0

    MOV DI,OFFSET alfabet

    CMP AH,0

    JE L\_3BLOCK

    L\_3BLOCK:

    ;aducem primul octet din mesajul criptat si shiftam la dreapta cu 2 pozitii

    MOV AX,[SI]

    MOV AH,0

    MOV BX,AX

    ;shiftam la dreapta cu 2 biti, pentru a retine primul calup de 6 biti

    SHR AL,2

    ;acum avem primii 6 biti, adica primul caracter din encoded

    ADD DI,AX

    MOV AX,[DI]

    MOV AH,0

    PUSH SI

    MOV SI,OFFSET encoded

    ADD SI,[iterations]

    MOV DX,[DI]

    MOV DH,0

    MOV [SI],DX

    POP SI

    ;salvam in BX care acum contine AX-ul initial,ultimii doi biti din AX

    AND BL,00000011b

    ;aducem urmatorul octet

    INC SI

    MOV AX,[SI]

    MOV AH,0

    ;salvam in DX acest octet

    MOV DX,AX

    ;shiftam cu 4 pozitii la dreapta

    SHR AL,4

    ;adaugam ultimii 2 biti din primul octet

    SHL BL,4

    ADD AL,BL

    ;am format al doilea calup de 6 biti

    MOV DI,OFFSET alfabet

    ADD DI,AX

    PUSH SI

    MOV SI,OFFSET encoded

    ADD SI,[iterations]

    INC SI

    MOV BX,[DI]

    MOV BH,0

    MOV [SI],BX

    POP SI

    ;aducem si al treilea octet

    INC SI

    MOV AX,[SI]

    MOV AH,0

    ;shiftam cu 6 poztii, pentru a retine primii 2 biti

    SHR AL,6

    ;adaugam ultimii 4 biti de la precedentul octet

    AND DL,00001111b

    SHL DL,2

    ADD DL,AL

    MOV DH,0

    ;iar acum avem al treilea calup de 6 biti in AX

    MOV DI,OFFSET alfabet

    ADD DI,DX

    CALL ADD\_PENULTIMUL\_OCTET

    CALL ULTIMUL\_OCTET

    ADD [iterations],3

    LOOP L\_3BLOCK

    JMP JMP\_RET

    ADD\_PENULTIMUL\_OCTET:

        MOV DI,OFFSET alfabet

    ADD DI,DX

    PUSH SI

    MOV SI,OFFSET encoded

    ADD SI,[iterations]

    ADD SI,2

    MOV BX,[DI]

    MOV BH,0

    MOV [SI],BX

    POP SI

    RET

    ULTIMUL\_OCTET:

    MOV AX,[SI]

    MOV AH,0

    AND AL,00111111b

    MOV AH,0

    MOV DI,OFFSET alfabet

    ADD DI,AX

    MOV BX,[DI]

    MOV BH,0

    PUSH SI

    MOV SI,OFFSET encoded

    ADD SI,[iterations]

    ADD SI,3

    MOV [SI],BX

    ADD [iterations],1

    POP SI

    RET

JMP\_RET:

O rulare a programului cu debug

Setul de date dupa producerea intreruperii 2Ch:



AX\*3600



Adunat cu 60\*CL=60\*14h



Rezultatul primei paranteze:



X0 final:



ENCRYPT

Primul octet(initial 41 dupa XOR cu x1=x0)



X2=(F6h\*1Dh+27h)%FFh=21h

Iar parcurgerea continua analog pentru urmatorii pasi

Sirul criptat:

(t-ul provine de la out.txt)

ENCODE

Mutarea in CX a numarului de blocuri de cate 3 caractere al sirului:



Sirul criptat final:





Sirul criptat in binar:

(010111)(00 0101)(0011 10)(011111)

C64:Z 8 z t