**Arhitectura sistemelor de calcul**

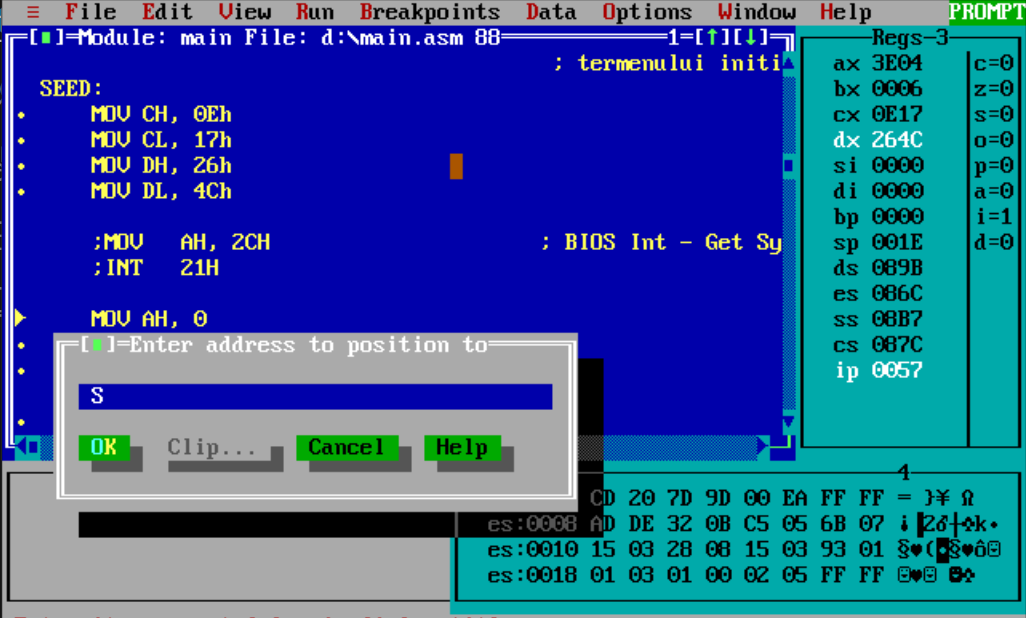
**Tema - Document explicativ**

**Enache Alexandru-Costin**

**Grupa C112-C**

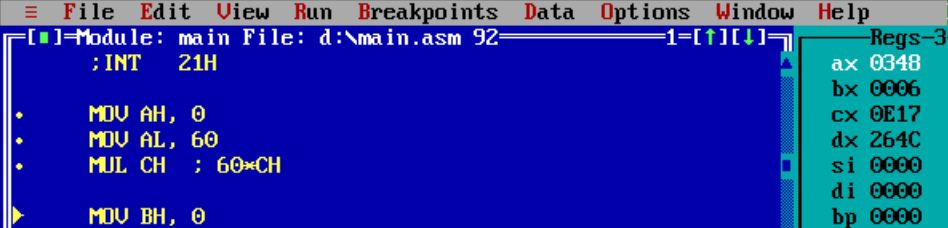
Pentru inceput, o sa prezint modul in care am implementat primul TO-DO (subrutina SEED). In acest task am asignat valorile corespunzatoare variabilelor a, b si x0.

Am considerat de folos sa dau un factor comun 60 in formula “x0 = ((CH ∗ 3600 + CL ∗ 60 + DH) ∗ 100 + DL) mod 255.” si sa ajung la “x0 = ((60\*(CH\*60+CL)+DH)\*100+DL) mod 255.”, iar apoi sa calculez pe bucati pana ajung la rezultatul termenului initial al sirului. Atasez rezultatul dupa fiecare operatie pentru a explica mai amanuntit.

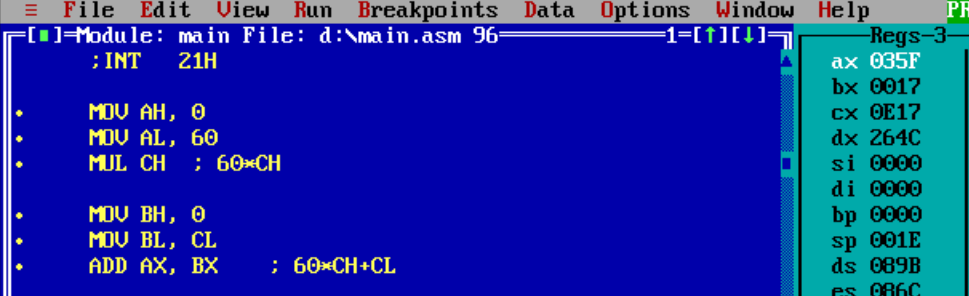


Am comentat functia 2ch ce ofera ora pentru a initializa direct cu valorile oferite drept exemplu in PDF.

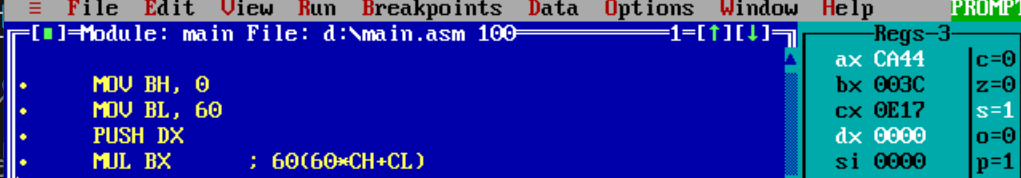
Am inceput cu parantezele interioare pana la modulo (interior spre exterior). Pana a incepe prima inmultire (CH\*60), CH poate fi vazut in poza ca fiind initial 0E, iar dupa inmultire va trebui sa avem valoarea 348h in AX.



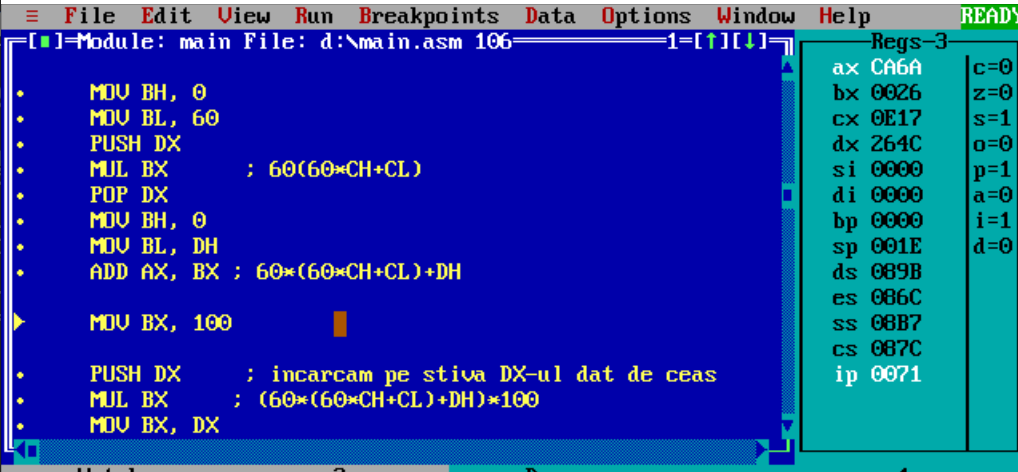
A urmat adunarea cu registrul CL.



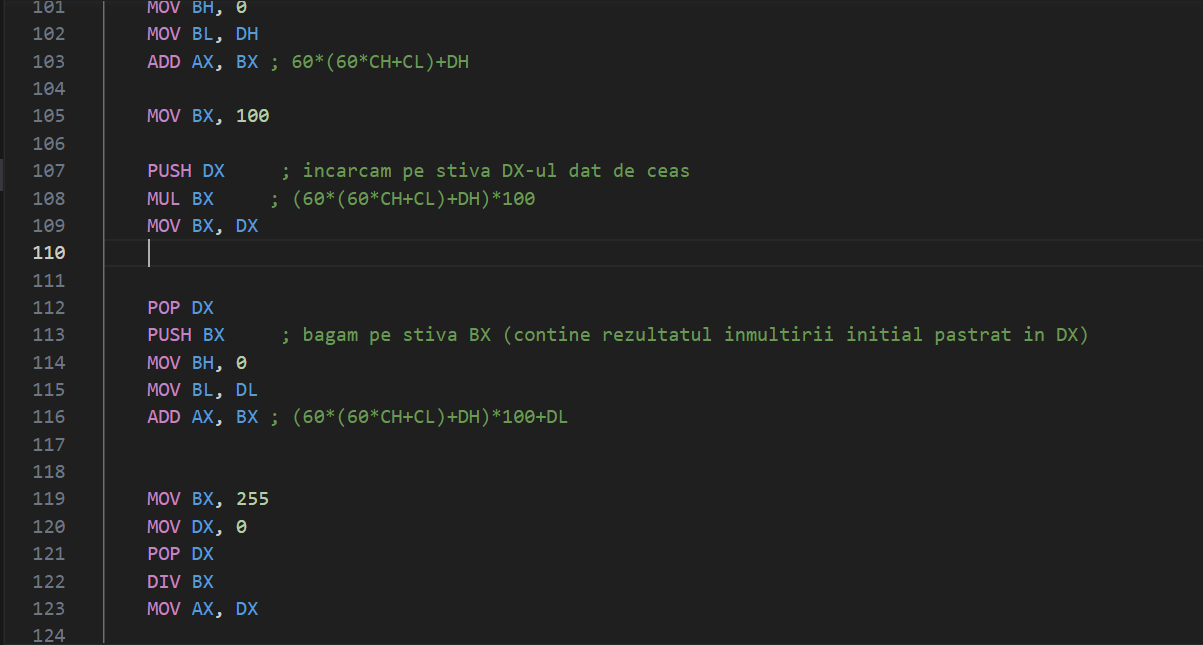
Ulterior, fiind nevoie sa inmultesc cu 60 rezultatul acelei paranteze din interior, am folosit registrul BX ca sa pastreze valoarea pentru inmultire. DX-ul a fost urcat pe stiva pentru a nu se pierde valoarea prin suprascriere datorita functiei MUL.



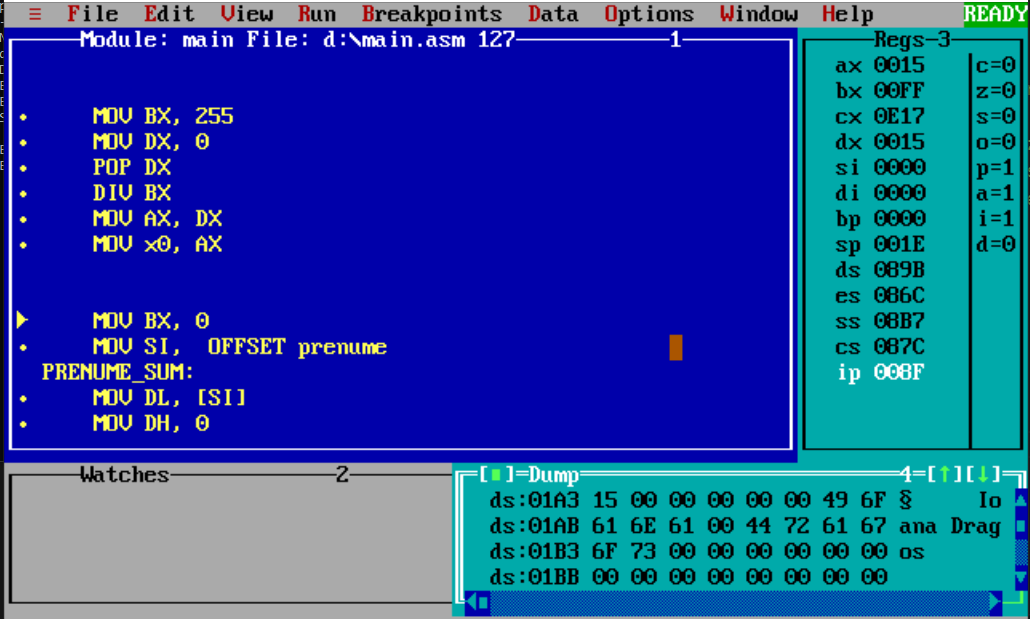
Pentru a aduna rezultatul cu continutul registrului DH, am recuperate valoarea lui DX de pe stiva folosind POP DX, incarcandu-l ulterior in registrul BX si adunandu-l la AX (registrul ce pastreaza rezultatul expresiei). Pentru a inmulti cu 100 ne folosim de registrul BX in mod asemanator ca in situatia anterioara.



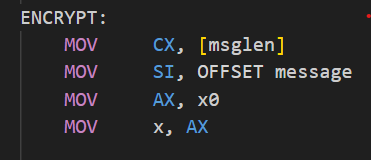
Ne-a ramas sa adunam rezultatul obtinut cu registrul DL. Am plasat BX pe stiva pentru a ne folosi de acesta sa stocam registrul DL in BL (pentru a aduna BX la AX)., astfel am pastrat in BX cei 2 octeti ce nu au putut fi pastrati in AX in urma operatiei MUL (dupa inmultirea cu un registru pe 16 biti, 2 octeti sunt pastrati in AX si 2 in registrul DX). La final, dupa adunare, pastram in BX valoarea 255, scoatem in DX valoarea de pe stiva (cei 2 octeti de overflow) si impartim rezultatul curent (stocat in AX) la valoarea din BX, pastram restul in AX.



La finalul acestor operatii, pentru valorile de test asignate la inceput trebuie sa avem in variabila x0 valoarea 15h.

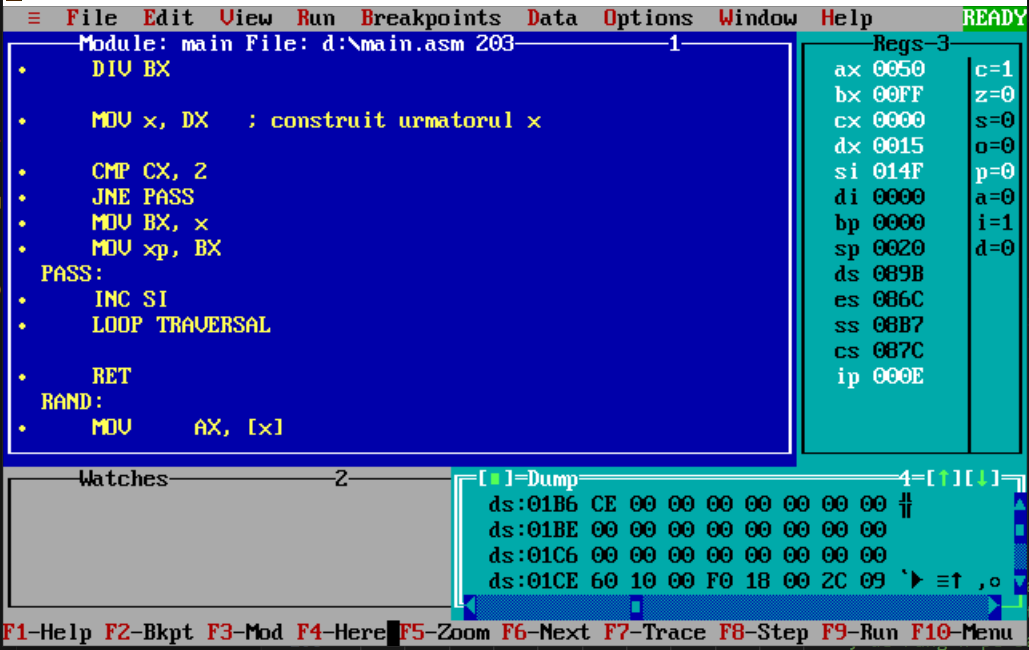


Trecand la etapa de criptare, am plasat pentru inceput in variabila x valoarea initiala a sirului.



Am inceput prin operatia de XOR intre primul octet din sirul de intrare si primul termen al sirului calculat din formula generala, apoi am construit urmatorul termen din formula generala si l-am salvat in variabila x.

Deoarece acest loop se va opri dupa ce va calcula ultimul element din sir, voi salva in variabila xp penultima valoare a sirului (comparand registrul CX cu valoarea 2 pentru a sari la incrementare si la urmatoarea iteratie).



La partea de codificare am intampinat probleme deoarece nu am mai reusit sa gasesc o regula pentru lungimea sirului codificat (daca dau valori random se face codificarea cu tot cu padding buna).

Am gandit codificarea in felul urmator: am lucrat pe cate 3 octeti per iteratie si am separat cei 24 de biti intr-un sir nou, avand astfel cate un octet nou per grupare de 6 biti.In felul acesta la final mi-a fost greu sa gasesc o regula generala pentru lungimea sirului (fiind 3 cazuri, cand restul impartirii este 1 octet, 2 octeti sau niciun octet). Astfel, cu ideea ‘comentata’ in cod am reusit sa afisez cum trebuie cand nu este necesar paddingul, iar cand este necesar cu caracterul final lipsa. Tema mea intampina o alta problema, nu am reusit sa afisez in fisier, desi dupa parerea mea nu am modificat nimic, doar in modul debug am putut vizualiza rezultatele din pacate.