Arhitectura sistemelor de calcul

Curs 2 - registrii + tip de date

Registrii sunt capacități de memorare, foarte mici ca și capacitate de memorare (8, 16, 32, 64 biți) și foarte rapide ca viteză de acces utilizate pentru stocarea temporară a operanzilor cu care operează un procesor. Cei 8 registrii (EAX, EBX, ECX, EDX, ESP, EBP, EDI, ESI) se numesc generali deoarece programatorul are libertatea de a îi utiliza în orice scop dorește, totuși fiecare are un rol predefinit care se manifestă într-un anumit context.

EAX: registrul acumulator este folosit când facem operații aritmetice deoarece este întotdeauna unul dintre operanzi.

EBX: registrul de bază este folosit pentru a marca un început (de exemplu la o zonă de memorie pentru localizarea unui element precum A[7], locul unde începe A-ul (adresa lui A) este baza, iar 7 este indexul)

ECX: registrul contor este folosit pentru structuri repetitive precum loop.

EDX: registrul de date este folosit atunci când dimensiunea

rezultatului depășește dimensiunea de reprezentare a operandului

de exemplu op1 - repr pe 16 biți și op2 - repr pe 16 biți =>

repr rezultatului calculului op1 · op2 = m + n = 16 + 16 = 32 biți

=> rezultatul se află în DX : AX.

La nivelul programării pe 32 de biți procesorul lucrează cu 3 tipuri de date: byte, word, doubleword.

Tipul de date este o combinație între o structură și operații asociate (OOP). Pentru microprocesor există 3 tipuri de dată care înseamnă dimensiunea de reprezentare.

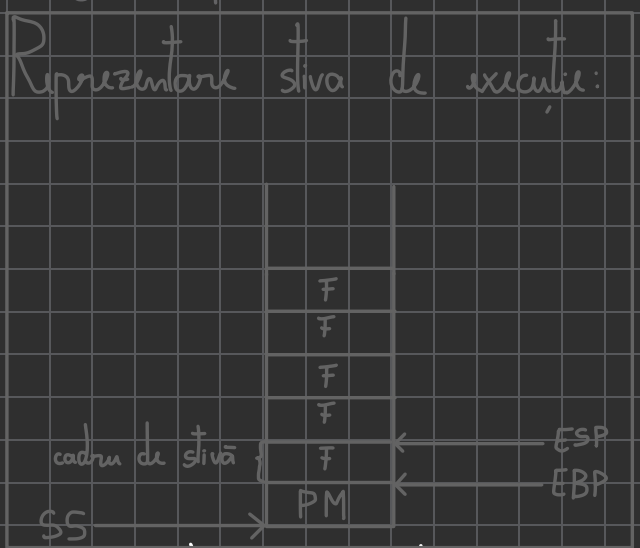
Directivele de definire a datelor sunt db, dw, dd care reprezintă dimensiunea de reprezentare.

Rolul directivelor de definire în NASM nu este de a preciza tipul de dată a variabilelor definite, ci doar de a genera octeții corespunzători acelor zone de memorie în conformitate cu directiva specificată și respectând ordinea de plasare de tip little-endian. Deci variabilele nu sunt de un anumit tip, ci sunt doar offseturile unor zone de memorie.

Structurile de date au o structură organizatorică și un mecanism de acces: coada - FIFO, stiva - FILO

Mecanismul de execuție al oricărui program respectă disciplina de execuție LIFO (stivă). (exemplu avem un program main care apelează de mai multe ori recursiv o funcție pentru a calcula factorialul unui număr)

Reprezentare stiva de execuție:



SS: adresa începutului zonei de stivă

EBP: adresa unde a fost stocat primul element introdus în stivă.

ESP: adresa unde a fost stocat ultimul element introdus în stivă.

ESP-ul scade cu 4 când facem operația push deoarece se rezervă 4 octeți ca să se poată stoca valoarea la noua adresă ESP sub forma little-endian pentru ca atunci când facem pop să se poată reconstrui valoarea.

Exemplu, dorim să punem pe stivă valoarea 12345678h:

```
; ESP = 00400008
push 12345678h
; ESP = 00400004
```

ADDRESS	Hex dump
00400000	0 0 0 0 0 0 0 0
00400004	7 8 5 6 3 4 1 2
00400008	0 0 0 0 0 0 0 0

```
pop EAX
; EAX = [ESP] +
        + [ESP+1] << 8
        + [ESP+2] << 16
        + [ESP+3] << 24
```

EDI și ESI sunt regiștrii de index utilizați pentru manipularea șirurilor de octeți, cuvinte sau dublucuvinte (MOVS, CMPS, LODS, STOS, SCAS).

Fiecare dintre registrii generali (EAX, EBX, ECX, EDX, ESP, EBP, EDI, ESI) au capacitatea de 32 de biți, fiecare fiind format din concatenarea a doi subregistrii de 16 biți. Subregistrul superior nu are denumire și nu poate fi accesat. Subregistrul inferior poate fi accesat având registrii de 16 biți (AX, BX, CX, DX, și BP, DI, SI), dintre care AX, BX, CX și DX sunt la rândul lor formați din concatenarea a alți doi subregistrii a câte 8 biți. Există astfel AH, BH, CH și DH conținând cei 8 biți superiori și AL, BL, CL și DL conținând cei 8 biți inferiori.

RAM

- variabilele de memorie ram sunt definite de db, dw sau dd
- timpul de acces la orice zonă de memorie din RAM (random access memory) este același indiferent de poziția față de începutul memoriei.
- spre deosebire de memoriile ROM (read only memory care permit doar citirea, ex: mașina de spălat) suportă citiri și scrieri în mod aleator.