2.6.6. Adrese FAR şi NEAR

Pentru a adresa o locaţie din memoria RAM sunt necesare două valori: una care să indice segmentul, alta care să indice offsetul în cadrul segmentului. Pentru a simplifica referirea la memorie, microprocesorul derivă, în lipsa unei alte specificări, adresa segmentului din **unul dintre registrii de segment CS, DS, SS sau ES**. Alegerea implicită a unui registru de segment se face după nişte reguli proprii instrucţiunii folosite.

Prin definiţie, o adresă în care se specifică doar offsetul, urmând ca segmentul să fie preluat implicit dintr-un registru de segment poartă numele de *adresă NEAR* (adresă apropiată). O adresă NEAR se află întotdeauna în interiorul unuia din cele patru segmente active.

O adresă în care programatorul indică explicit un selector de segment poartă numele de *adresă FAR* (adresă îndepărtată). O adresă FAR este deci o SPECIFICARE COMPLETA DE ADRESA şi ea se poate exprima la nivelul unui program în trei moduri:

- $s_3s_2s_1s_0$: specificare_offset unde $s_3s_2s_1s_0$ este o constantă;
- registru_segment : specificare_offset, registru segment fiind CS, DS, SS, ES, FS sau GS;
- FAR [variabilă], unde variabilă este de tip QWORD şi conţine cei 6 octeţi constituind adresa FAR. (ceea ce numim variabila pointer in limbajele de nivel inalt)

Formatul intern al unei adrese FAR este: la adresa mai mică se află offsetul, iar la adresa mai mare cu 4 (cuvântul care urmează după dublucuvântul curent) se află cuvântul ce conţine selectorul care indică segmentul.

Reprezentarea adreselor respectă principiul reprezentării little-endian expus în capitolul 1, paragraf 1.3.2.3: partea cea mai puţin semnificativă are adresa cea mai mică, iar partea cea mai semnificativă are adresa cea mai mare.

2.6.7. Calculul offsetului unui operand. Moduri de adresare

În cadrul unei instrucţiuni există 3 moduri de a specifica un operand pe care aceasta îl solicită:

- *modul registru*, dacă pe post de operand se află un registru al maşinii; mov eax, 17
- *modul imediat*, atunci când în instrucţiune se află chiar valoarea operandului (nu adresa lui şi nici un registru în care să fie conţinut); mov eax, 17
- *modul adresare la memorie*, dacă operandul se află efectiv undeva în memorie. În acest caz, offsetul lui se calculează după următoarea formulă:

$$\textit{adresa\_offset} = [\text{ bază }] + [\text{ index} \times \text{scală }] + [\text{ constanta }]$$

Deci *adresa_offset* se obţine din următoarele (maxim) patru elemente:

- conţinutul unuia dintre regiştrii EAX, EBX, ECX, EDX, EBP, ESI, EDI sau ESP ca bază;
- conţinutul unuia dintre regiştrii EAX, EBX, ECX, EDX, EBP, ESI sau EDI drept index;
- factor numeric (scală) pentru a înmulţi valoarea registrului index cu 1, 2, 4 sau 8
- valoarea unei constante numerice, pe octet, cuvant sau dublucuvânt.

De aici rezultă următoarele moduri de adresare la memorie:

- *directă*, atunci când apare numai *constanta*;
- *bazată*, dacă în calcul apare unul dintre regiştrii bază;
- *scalat-indexată*, dacă în calcul apare unul dintre regiştrii index;

Cele trei moduri de adresare a memoriei pot fi combinate. De exemplu, poate să apară adresare directă bazată, adresare bazată şi scalat-indexată etc

Adresarea care NU este directă se numeste ***adresare indirectă*** (bazată şi/sau indexată). Deci o adresare indirectă este cea pt care avem specificat cel puţin un registru între parantezele drepte.

La instrucţiunile de salt mai apare şi un alt tip de adresare numit adresare *relativă*.

Adresa relativă indică poziţia următoarei instrucţiuni de executat, în raport cu poziţia curentă. Poziţia este indicată prin numărul de octeţi de cod peste care se va sări. Arhitectura x86 permite atât adrese relative scurte (SHORT Address), reprezentate pe octet şi având valori între -128 şi 127, cât şi adrese relative apropiate (NEAR Address), pe dublucuvânt cu valori între -2147483648 şi 2147483647.

```
Jmp MaiJos  ; aceasta instrucţiune se traduce (vezi OllyDbg) de obicei in Jmp [0084]↓
…….
…….
MaiJos:
   Mov eax, ebx
```