12 Adresarea memoriei

Modurile de adresare constituie un instrument principal pentru reprezentarea în memorie a imaginii datelor, aşa cum este aceasta văzută de programatorul în limbaj de nivel înalt. Ele permit programatorului să construiască şi să utilizeze structuri complexe de date, în timp ce echipamentul poate trata doar matrice lineare cu elemente de dimensiune fixă. Pentru accesul la un element al unei structuri complexe de date trebuie evaluată adresa de memorie ce corespunde începutului zonei de memorie în care se păstrează acel element. De regulă, această operaţie se efectuează prin mai multe instrucţiuni, utilizând mai multe moduri de adresare.

Cu cât calculele efectuate în cadrul modului de adresare sunt mai complexe, cu atât numărul mediu de instrucţiuni, cerut de accesul la un element al unei structuri complexe de date, este mai mic.

12.1 Elemente de bază privind modurile de adresare

Modurile de adresare utilizate de diferite microprocesoare sunt foarte variate. Ele prezintă însă o caracteristică generală: fiecare mod se obţine ca o combinaţie între un număr relativ mic de obiecte şi funcţii (operaţii) de bază.

Se consideră că un obiect sau funcţie sunt fundamentale dacă, în absenţa acelui obiect sau funcţie unitatea de adresare nu este capabilă să genereze nici un mod de adresare.

În consecinţă, obiectele fundamentale sunt registre şi deplasamente sau offset - uri conţinute în instrucţiune iar funcţiile fundamentale sunt adunarea, deplasarea şi adresarea indirectă. Operaţia de deplasare ar putea fi înlocuită cu adunări sau scăderi, însă timpul consumat pentru adresare ar fi prea mare, aşa încât este considerată funcţie fundamentală. Adunarea include şi operaţiile de incrementare şi decrementare, cazuri particulare de adunări.

Registre. Sunt situate în unitatea centrală de prelucrare, în unitatea aritmetică şi în cea de gestiune a memoriei. Principalele roluri ale registrelor în ceea ce priveşte modurile de adresare sunt:

- *registru operand*: conţinutul registrului este chiar operandul la care se face referire;

- *registru adresă* : conținutul registrului este adresa operandului la care se face referire (operand în memorie);
- *registru de bază* : registrul conține o adresă care, pentru obținerea adresei complete a operandului, va fi utilizată împreună cu conținutul unui alt registru sau cu un deplasament;

Offset -uri. Sunt conținute în instrucțiune (câmp *imediat*) şi sunt utilizate împreună cu conținutul unui registru pentru calculul adresei unui operand din memorie.

Câmpul *imediat* este un câmp opțional al unei instrucțiuni şi acesta poate fi:

- chiar operandul utilizat de instrucțiune (operand *imediat*);
- adresa completă a operandului (adresă *imediată*);
- un deplasament (offset) care va fi implicat în adresare.

Adunarea. Toate modurile de adresare, cu excepția celor mai simple, necesită adunarea a două sau mai multe valori pentru obținerea adresei; operațiile de incrementare / decrementare sunt considerate cazuri particulare de adunare.

Adresarea indirectă. Este o funcție al cărei argument este dat de rezultatele unor calcule iar valoarea este adresa operandului din memorie. Modurile complexe de adresare pot folosi de mai multe ori adresarea indirectă pentru generarea adresei finale a operandului.

Deplasarea. Este mutarea cu una, două sau mai multe poziții binare, spre stânga, a valorii binare a unui parametru de adresare. Se utilizează în cadrul operației de indexare, care la rândul său se foloseşte în adresarea tablourilor. Indexarea produce deplasamentul (offset-ul) unui element din tablou pe baza valorilor indicilor săi.

Fie *a[i], i = 0, 1, 2, . . . , n*, un tablou unidimensional format din elemente de lungime *w* (în octeți); atunci adresa relativă a elementului *a[i]* față de baza tabloului *a[0]* va fi *offset(a[i]) = i * w* . Cum lungimea unui element este de regulă 1, 2, 4 sau 8 octeți, deplasarea se face respectiv cu 0, 1, 2 sau 3 poziții binare, ceea ce este echivalent cu înmulțirea cu 1, 2, 4 sau 8 a valorii lui *i*. Operația de deplasare se face mult mai rapid decât înmulțirea, ceea ce o recomandă pentru includerea în funcțiile de bază pentru adresare.

12.1.1. Formatul instrucțiunilor

Instrucțiunile procesorului Intel 8086 sunt codificate pe un număr de octeți cuprins între 1 şi 6. Formatul sau structura instrucțiunii arată rolul fiecărui octet şi al câmpurilor de biți conținute în fiecare octet.

Primul octet conține codul operației (pe 6 biți), adică tipul prelucrărilor ce vor fi efectuate la execuția instrucțiunii (operație aritmetică, logică, transfer de date, salt etc.).

Al doilea octet conține informații privind adresarea memoriei, registrele în care se află operanzii, lungimea acestora, lungimea deplasamentului.

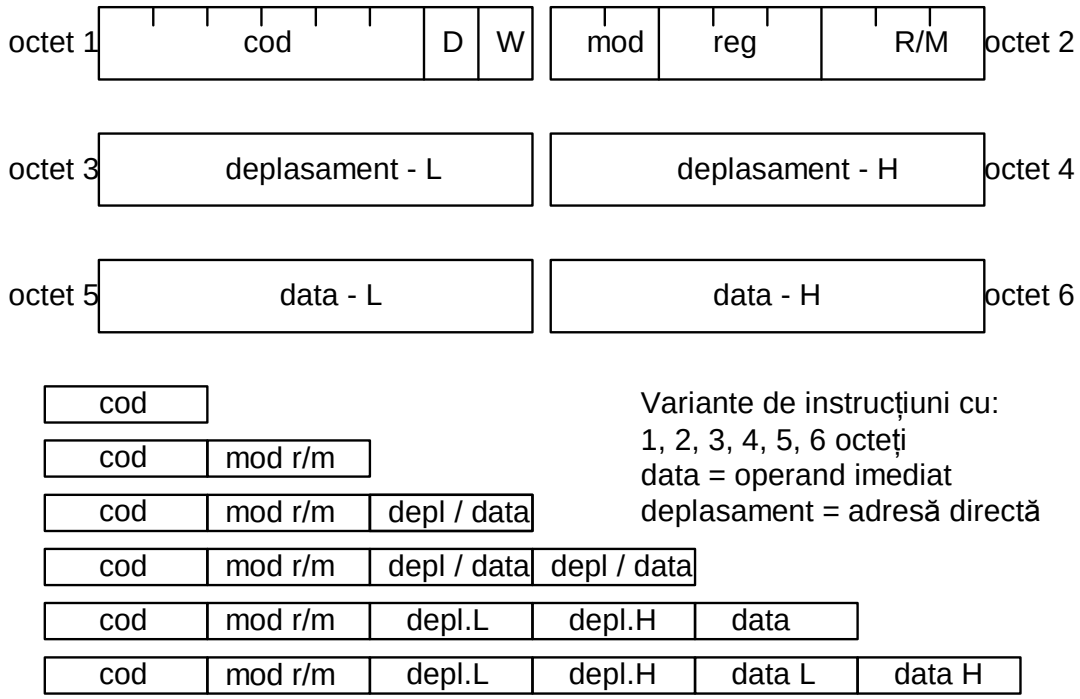

Fig.1 Formatul instrucțiunilor la Intel 8086

Octeții 3, 4, 5 şi 6 pot avea semnificația de deplasament pe 8 sau 16 biți, date pe 8 sau 16 biți sau deplasament şi date, conform figurii 1.

Pe lângă formatul propriu-zis, mai poate exista un prefix de segment sau un prefix de repetare.

La instrucțiunile cu doi operanzi, un operand este în mod obligatoriu într-un registru, precizat de câmpul REG iar celălalt operand este în registru sau în memorie, variantele fiind determinate de câmpurile R/M şi MOD.

Semnificația câmpurilor:

- **D** (*destination*) = destinație; codifică sensul operației:
  D = 0 câmpul REG indică operandul destinație;
  D = 1 câmpul REG indică operandul sursă.
  De exemplu, codurile instrucțiunilor

```
add  [cx], ax
add  ax,  [cx]
```

diferă numai prin câmpul D (aceeaşi operație, aceiaşi operanzi dar diferă sensul de transfer).

- **W** (*word*) = cuvânt; codifică lungimea operanzilor (octet sau cuvânt):
  W = 0 operație la nivel de octet;
  W = 1 operație la nivel de cuvânt.
- **REG** - indică registrul care conține unul din operanzi. Dimensiunea registrului specificat depinde de bitul W.

REG	W = 0	W = 1
0 0 0	AL	AX
0 0 1	CL	CX
0 1 0	DL	DX
0 1 1	BL	BX
1 0 0	AH	SP
1 0 1	CH	BP
1 1 0	DH	SI
1 1 1	BH	DI

- R/M (*register / memory*) - câmp registru / memorie; dă informații despre al doilea operand fiind dependent de câmpul MOD. Dacă MOD = 11, atunci R/M indică registrul care conține al doilea operand iar în celelalte cazuri R/M indică registrul implicat în formarea adresei efective pentru al doilea operand.

MOD	11		0 0	0 1	1 1
R/M	W = 0	W = 1			
0 0 0	AL	AX	(BX) + (SI)	(BX) + (SI) + d8	(BX) + (SI) + d16
0 0 1	CL	CX	(BX) + (DI)	(BX) + (DI) + d8	(BX) + (DI) + d16
0 1 0	DL	DX	(BP) + (SI)	(BP) + (SI) + d8	(BP) + (SI) + d16
0 1 1	BL	BX	(BP) + (DI)	(BP) + (DI) + d8	(BP) + (DI) + d16
1 0 0	AH	SP	(SI)	(SI) + d8	(SI) + d16
1 0 1	CH	BP	(DI)	(DI) + d8	(DI) + d16
1 1 0	DH	SI	BP	(BP) + d8	(BP) + d16
1 1 1	BH	DI	(BX)	(BX) + d8	(BX) + d16

În tabelul de mai sus, d8 şi d16 reprezintă deplasament pe 8 biți, respectiv pe 16 biți; acestea sunt specificate explicit în instrucțiuni.

12.2 Modurile de adresare a memoriei

Instrucțiunile microprocesorului 8086 pot avea unul sau doi operanzi care pot fi conținuți în registre sau în memorie. Când există doi operanzi, unul este în mod obligatoriu într-un registru (nu pot fi ambii operanzi în memorie).

Dacă ambii operanzi sunt în registre, nu este necesară adresarea memoriei. **Modurile de adresare** specifică modul în care se calculează adresa operandului din memorie. Se utilizează denumirea de **adresă de segment** (AS) pentru adresa de început a segmentului de memorie în care se află operandul şi **adresă efectivă** (AE), pentru deplasamentul (*offset*-ul) operandului în cadrul segmentului. Adresa de segment este furnizată de unul din cele 4 registre de segment.

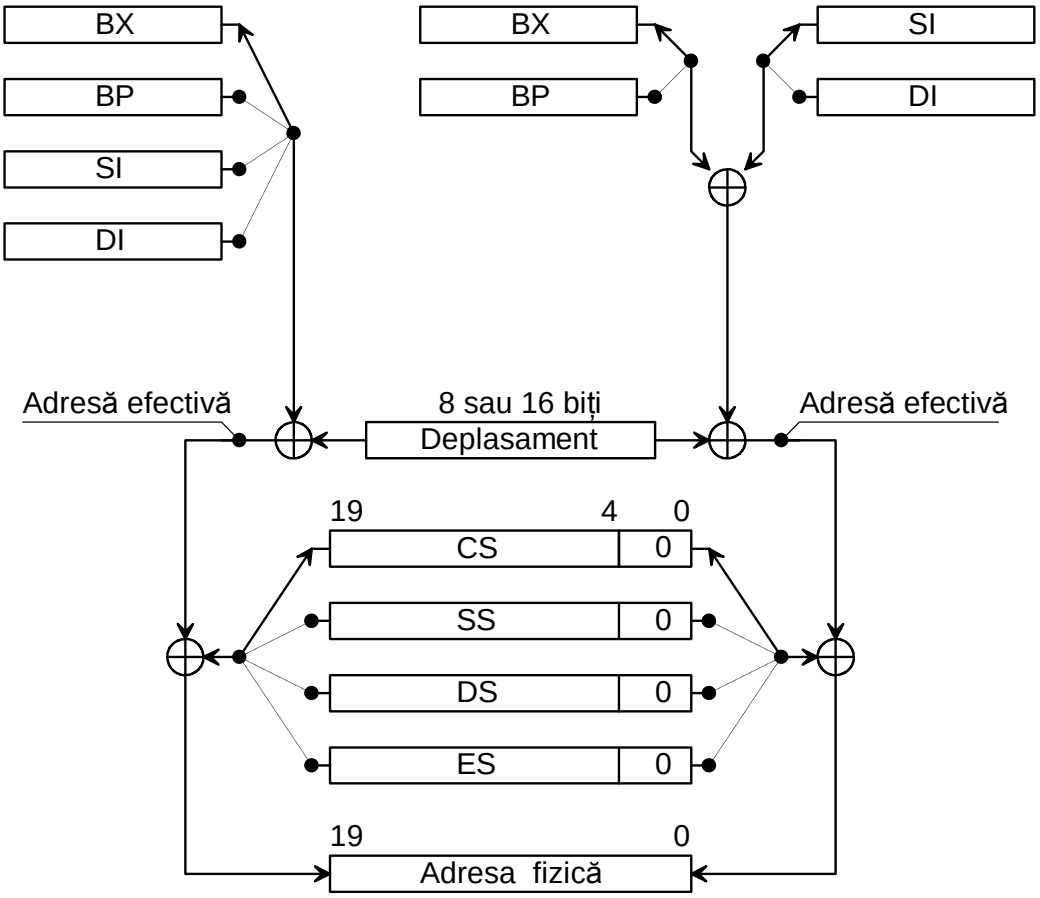

Fig. 2 Schema generală a modurilor de adresare la Intel 8086

- **Adresare imediată**

În acest mod de adresare, operandul este conținut în instrucțiune. Nu este necesar accesul la memoria externă, deoarece toți octeții instrucțiunii sunt încărcați în procesor la extragerea ei din memorie, din segmentul de cod. Exemplu:

```
mov  ax,  07FF
add   bx,  044C
```

- **Adresarea directă**

Adresa efectivă a operandului se obține din câmpul deplasament al instrucțiunii care poate fi pe un octet sau pe doi octeți; este un deplasament în interiorul segmentului curent de date.

```
.data
     DEP dw   1200H
.code
     mov  bx,  DEP  ; la asamblare DEP se va înlocui cu offset-ul în
                    ; cadrul segmentului de date; în final bx=1200H.
     add   cx,  [200] ; deplasament explicit în instrucțiune
```

Acest mod de adresare utilizează implicit registrul segment DS.

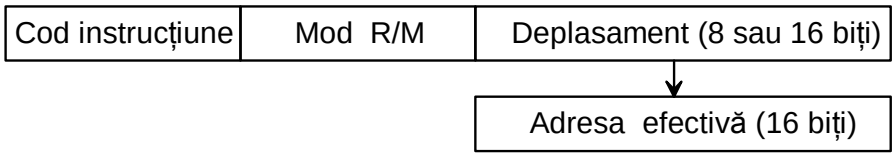

Fig.3 Adresarea directă: adresa este conținută în instrucțiune

- **Adresarea indirectă (prin intermediul registrelor)**

Adresa efectivă a operandului este dată de conținutul registrelor BX, BP, SI sau DI. Se utilizează ca registru segment implicit SS cu BP şi DS în celelalte cazuri.

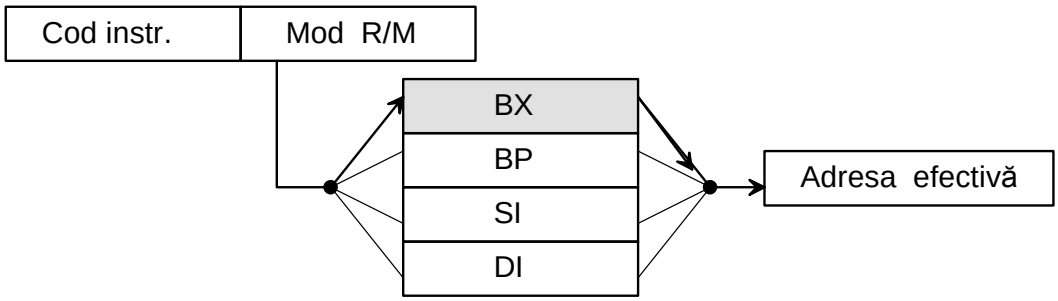

Fig.4 Adresarea indirectă: adresa efectivă este conținută într-un registru de bază sau index

Exemple:

```
mov  cx,  [bx]          ;pune în cx cuvântul cu adresa în bx
mov  [di],  cx          ;pune cx în memorie, la adresa din  di
add   byte ptr   [si], 7    ;adună 7 la octetul cu adresa în  si
```

În primele două instrucțiuni operandul din memorie este considerat de 16 biți datorită registrului de 16 biți implicat în transfer. În instrucțiunea a treia, operandul este octet; dacă s-ar fi scris :

```
add [si],  7
```

asamblorul nu ar şti dacă operandul din memorie este pe 8 sau pe 16 biţi şi se generează un mesaj de eroare la asamblare. Considerând operandul pe 8 biți se poate obține un rezultat diferit (decât cu operand pe 16 biți), în cazul în care prin adunarea cu 7 se depăşeşte valoarea 255 sau FFH.

- **Adresarea indirectă cu deplasament (bazată sau indexată)**

Adresa efectivă se obține prin adunarea la unul din registrele de bază (BX, BP) sau index (SI, DI), un deplasament constant pe 8 sau 16 biți. Registrul de segment implicit este DS (dacă se folosesc BX, SI sau DI) şi SS dacă se foloseşte pentru adresare BP.

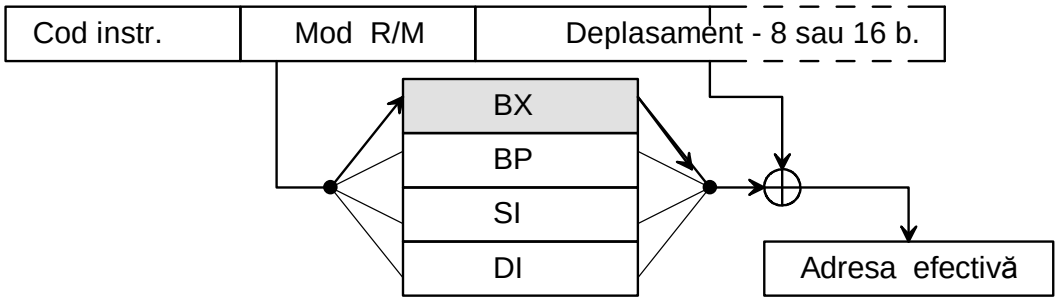

Fig.5 Adresarea indirectă cu deplasament

Exemplu:

```
.data
    TAB     dw  10  dup (0)   ;definire tablou de 10 elem.
.code
    mov bx,  5
    mov ax,  TAB [bx]
    mov cx,  bx [TAB]
    mov dx,  [bx +TAB]
    mov bp,  [bx].TAB
```

Se utilizează diverse forme de scriere pentru acelaşi efect; în bx este, din punct de vedere al mecanismului de adresare, o adresă de bază iar TAB este un deplasament constant. Pentru programul care accesează elementele tabloului, semnificaţia este inversă: TAB este adresa de început a tabloului (deci adresă de bază) iar bx este un indice (deplasament).

Un alt mod de acces la elementele tabloului este prin încărcarea în bx a adresei de bază (folosind operatorul *offset*) şi un indice explicit:

```
    mov bx,  offset TAB     ; adresa de bază este adresa de
                            ; început a tabloului
    mov ax,  5 [bx]
    mov ax,  bx [5]
    mov ax,  [bx + 5]
```

```
mov ax,   [bx] . 5
```

Observație:

În textul sursă, toate formele de adresare se scriu cu operatorul de indexare (paranteze drepte).

```
mov cx,   [bx]          ; adresare indirectă
mov cx,   [200]         ; adresare directă
mov cx,   [bx + 200]    ; adresare bazată
```

Folosirea registrului BP cu un deplasament nul este o scriere asemănătoare cu adresarea indirectă dar care este codificată intern ca adresare bazată:

```
mov ax,   [bx]      ; adresare indirectă
mov ax,   [bp]      ; adresare bazată cu deplasament 0
```

- **Adresarea bazată şi indexată**

Adresa efectivă se obține prin adunarea unui registru de bază (BX sau BP) cu un registru index (SI sau DI) şi cu un deplasament de 8 sau 16 biți.

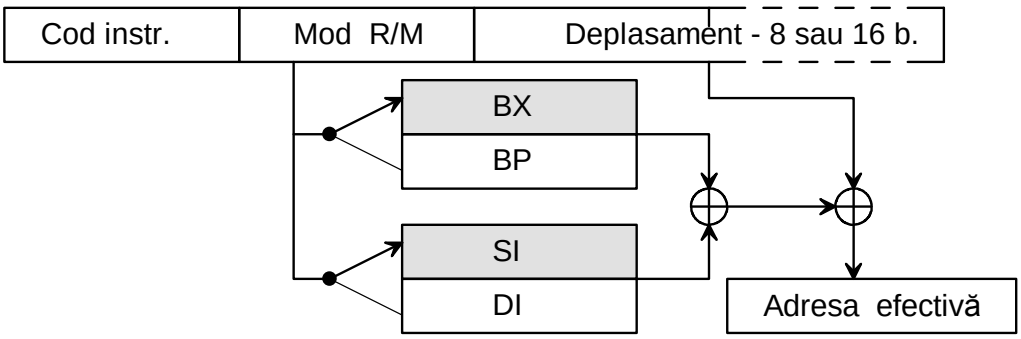

Fig.6 Adresarea bazată şi indexată

Registrele segment utilizate implicit sunt: DS (în cazul folosirii lui BX cu SI sau DI) şi SS (dacă se foloseşte BP cu SI sau DI); deplasamentul poate fi nul.

```
mov ax,   [bx] [si]
mov ax,   [bx + si + 9]
mov ax,   [bx + si] . 9
mov ax,   [bp] [di] [9]
```

În descrierea modurilor de adresare a fost precizat de fiecare dată registrul segment care participă în mod implicit la formarea adresei fizice. Regula generală este următoarea: în toate modurile de adresare în care nu

participă BP, registrul segment implicit este DS; dacă participă BP, registrul segment implicit este SS.

Regula de mai sus poate fi ignorată dacă se utilizează prefixele de segment în textul sursă (segment explicit):

```
mov bx,   ds: [bp + 7]          ; registrul segment explicit DS
mov ax,   cs: [si] [bx + 5]     ; registrul segment explicit CS
mov cx,   ss: [bx]              ; registrul segment explicit SS
```

În adresarea memoriei (cu un registru segment implicit sau explicit), controlul asupra datelor ce se află în memorie în segmentul respectiv cade în sarcina programatorului, care trebuie să aibă în vedere permanent "harta memoriei", adică modul de organizare a datelor în memorie.