UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI
Instrumente Structurale 2007-2013

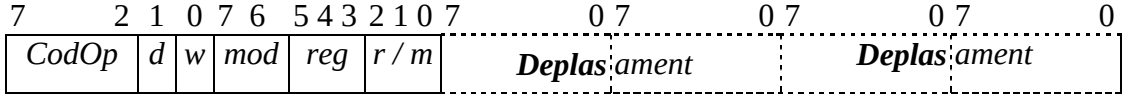
Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

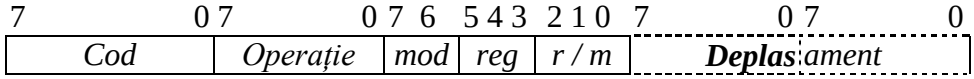
Programare în limbaj de asamblare

16. Formatul instrucțiunilor (codificare, moduri de adresare).

Formatul instrucțiunilor

Instrucțiunile procesorului 80286 au o lungime variabilă şi pot include 1 ÷ 6 octeți. Formatul general al unei instrucțiuni este următorul:





Formatul instrucțiunii la 286

Codificarea instrucțiunilor

În figura anterioară sunt prezentate două tipuri de format corespunzătoare cazurilor în care codul operației este reprezentat pe un octet pentru primul format, respectiv pe doi octeți pentru cel de-al doilea format. Semnificația biților este următoarea:

d – direction – specifică direcția rezultatului operației, şi anume:

d = 0, r/m r/m [Op] reg (registru)

d = 1, reg reg [Op] r/m (registru sau memorie).

w – word bit – indică tipul operanzilor, astfel:

w = 0, operand de tip octet;

w = 1, operand de tip cuvânt (doi octeți); la 386/486, această valoare înseamnă operand de dimensiune completă (16/32 de biți în funcție de modul de lucru).

În ceea ce privește câmpul „mod“, acesta reprezintă o codificare a modului de calcul al adresei efective sau lungimea deplasamentului şi este utilizat pentru a determina adresa efectivă împreună cu câmpul „r/m“.

Dacă mod = 11, câmpul r/m este tratat ca un câmp registru. În acest caz, precum şi pentru câmpul „reg“, adresele utilizate pentru referirea (codificarea) registrelor sunt prezentate în tabelă:

Adresa	Registru	
	Cuvânt	Octet
	W=1	W=0
000	AX	AL
001	CX	CL
010	DX	DL
011	BX	BL
100	SP	AH
101	BP	CH
110	SI	DH
111	DI	BH

Celelalte valori ale câmpului „mod“ precizează prezența câmpului deplasament în instrucțiune şi dimensiunea acestuia:

MOD	*Semnificație*
00	Câmpul deplasament nu este prezent. Câmpul *r/m* specifică un mod de adresare indirect prin registre, sau un mod de adresare bazat/ indexat, mai puțin pentru *r/m=110*, care denotă adresare directă, cu deplasament.
01	Câmpul deplasament are 8 biți (low). Câmpul *r/m* specifică un mod de adresare indexat sau un mod de adresare bazat/ indexat, cu deplasament.
10	Câmpul deplasament are dimensiune completă, adică are 16 biți, în modul de lucru pe 16 biți, şi respectiv 32 biți, în modul pe 32 biți; deplasamentul este cu semn şi urmează după câmpul *r/m*. Câmpul *r/m* specifică un mod de adresare indexat sau un mod de adresare bazat/ indexat, cu deplasament.
11	Câmpul *r/m* specifică un registru, şi utilizează aceeaşi codificare cu câmpul *reg*.

Câmpul „*r/m*„ conține fie adresa unui registru, care conține operandul – dacă mod = 11, fie o codificare care este utilizată pentru calculul adresei efective a operandului, astfel:

r/m	**Adresa efectivă**	**Adresa efectivă (386/486)**
000	**[BX] + [SI] + deplasament**	**[EAX] + scala*index + depl**
001	**[BX] + [DI] + deplasament**	**[ECX] + scala*index + depl**
010	**[BP] + [SI] + deplasament**	**[EDX] + scala*index + depl**
011	**[BP] + [DI] + deplasament**	**[EBX] + scala*index + depl**
100	**[SI] + deplasament**	**[ESP] + scala*index + depl**
101	**[DI] + deplasament**	**[EBP] + scala*index + depl**
110	**[BP] + deplasament**	**[ESI] + scala*index + depl**
111	**[BX]+ deplasament**	**[EDI] + scala*index + depl**

Codificarea modului de adresare

Există şi o excepție de la modurile de adresare din tabelă, şi anume: dacă *mod=00,* adresa efectivă nu se calculează utilizând registrul BP, ci doar un deplasament (care este prezent în acest caz) de 16, respectiv 32 de biți (pentru operarea pe 16 biți, pentru *r/m=110,* AE= deplasament de 16 biți, iar pentru operarea pe 32 de biți, pentru *r/m=101*, AE= deplasament de 32 de biți).

Majoritatea instrucțiunilor cu doi operanzi permit fie ca memoria sau un registru să fie un operand, fie un registru sau o constantă, conținută în instrucțiune, să fie utilizat ca al doilea operand. Operațiile cu ambii operanzi în memorie sunt excluse (cu excepția operațiilor pe şiruri şi a operațiilor cu stiva). Operanzii din memorie pot fi adresați direct, cu un offset (deplasament) de adresă de 16 biți, sau indirect, cu un registru de bază (BP sau BX) şi/sau registre index (SI sau DI) şi, eventual, adunate cu o constantă de deplasament, de 8 sau 16 biți.

Toate operațiile cu doi operanzi, cu excepția înmulțirii, împărțirii şi a operațiilor cu şiruri, permit ca operandul sursă să apară în instrucțiune ca dată imediată. Operanzii imediați de 16 biți,

având octetul mai semnificativ extensia bitului de semn a octetului mai puțin semnificativ, pot fi abreviați la 8 biți. Operanzii imediați de 16 biți sunt memorați, după constantele de deplasament, în ordinea obișnuită pentru date: primul este octetul mai puțin semnificativ, iar cel de-al doilea este octetul mai semnificativ.

Unitatea de execuție (EU) are acces la operanzii imediați și registre; când are nevoie de un operand din memorie, transmite la AU offsetul acestuia și registrul segment, iar AU determina adresa fizică a operandului, în funcție de modul de adresare, pe care o transmite către BU.

Există, în principal, șase moduri de adresare:

- *directă*, adresa efectivă (AE) a operandului este reprezentată de deplasamentul conținut în instrucțiune;
- *indirectă*, AE se află într-unul din registrele de bază sau index;
- *bazată*, AE este suma dintre deplasament și conținutul unui registru de bază (BX sau BP);
- *indexată*, AE este suma dintre deplasament și conținutul unui registru index (SI sau DI);
- *bazată și indexată*, AE este suma dintre conținutul a două registre: unul de bază și unul index;
- *bazată și indexată cu deplasament*, AE se obține ca suma a unui registru de bază, a unui registru index și un deplasament.

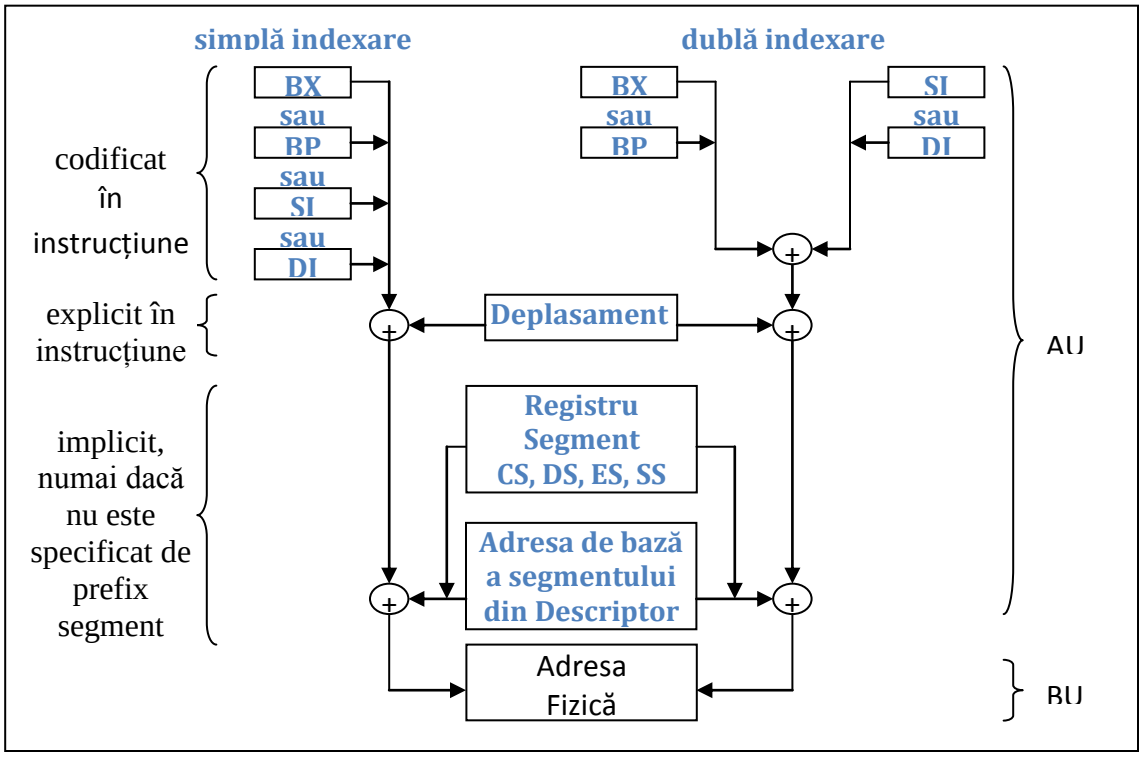
Mai pot fi puse în evidență și următoarele moduri de adresare:

- *imediată*, operandul este conținut în instrucțiune;
- *la registre*, operandul se află într-un registru.

În afara acestor moduri de adresare de bază, mai există și două moduri de adresare speciale:

- adresarea șirurilor;
- adresarea porturilor de I/O.

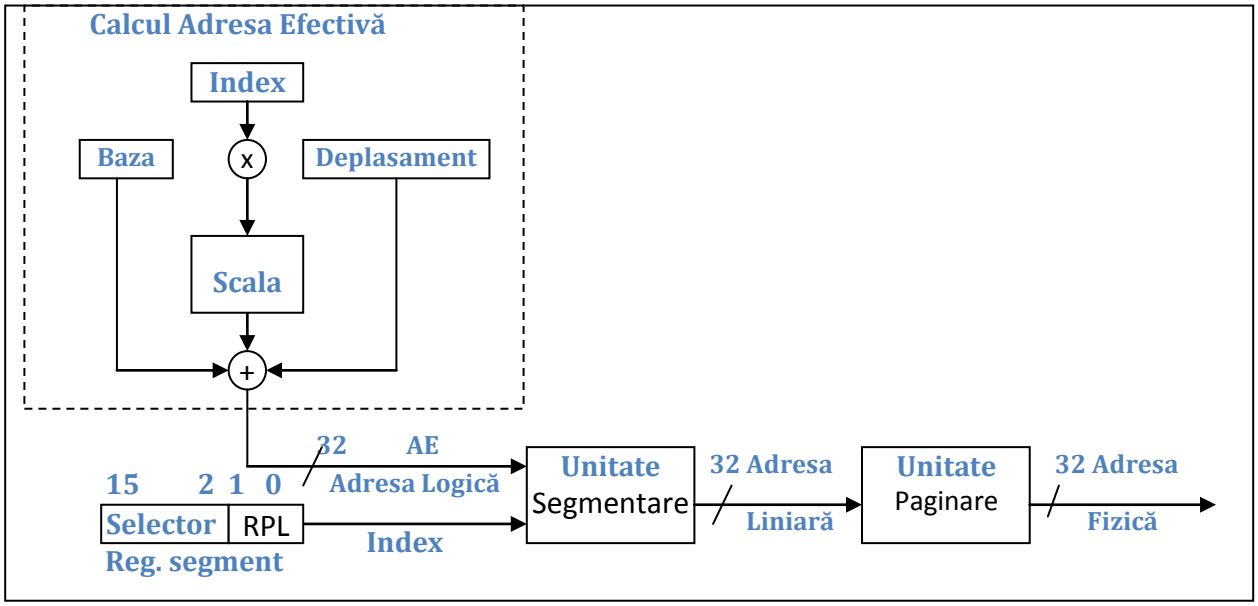
Modul de calcul al adresei fizice a operandului este reprezentat schematic în figură.



Generarea adresei fizice

În funcție de modul de lucru, real sau protejat, pentru determinarea adresei fizice se utilizează fie unul din registrele segment, fie adresa de bază dintr-un descriptor de segment. Pentru procesoarele 386/486, această adresă este denumită adresă liniară şi, dacă adresarea paginată nu este activă, atunci ea reprezintă chiar adresa fizică. Dacă adresarea paginată este activă, atunci unitatea de paginare o transformă intr-o adresă fizică; modul de obținere al adresei fizice pentru aceste procesoare este prezentat în figură.

Adresa fizică determinată de AU din adresa logică se calculează din surse diferite, în funcție de tipul referinței:



Determinarea adresei fizice pentru 386/486

Tipul referinței la memorie	Adresa de bază a segmentului		**Offset**
	Implicit	Alternativ	
Fetch instrucțiune	CS	-	IP
Operații cu stiva	SS	-	SP
Variabile de memorie	DS	CS , ES , SS	Adresa Efectivă
Operații pe şiruri - şirul sursă - şirul destinație	 DS ES	 CS , ES , SS -	 SI DI
BP utilizat ca registru de bază	SS	CS , ES , SS	Adresa Efectivă

Registrele utilizate pentru calculul adresei fizice

Pentru adresarea într-un alt segment decât cel implicit se va preceda instrucțiunea respectivă cu un prefix de segment (override) care va specifica, în mod explicit, registrul segment utilizat. Formatul instrucțiunilor la procesoarele 386/486 şi Pentium este asemănător:

Cod operație	mod reg r/m	SIB	Deplasament	Operand
1–2 octeți	0–1 octeți	0–1 octeți	0, 1, 2, 4 octeți	0, 1, 2, 4 octeți

În plus față de formatul anterior, în instrucțiune poate fi prezent octetul SIB (Scale×Index + Base), ce specifică registrul index utilizat (care poate fi oricare dintre registrele EAX, EBX, ECX, EDX, EBP, ESI sau EDI, deci nu poate fi utilizat ca registru index ESP), factorul de scalare cu care se înmulțește registrul index (care poate avea una din valorile 1, 2, 4, 8) şi registrul de bază (oricare dintre registrele EAX,..., EDI) utilizat pentru calculul adresei efective. Dacă în cazul procesorului 286 o instrucțiune nu poate fi precedată decât de prefixe de segment (care specifică registrul segment utilizat pentru determinarea adresei fizice) sau prefixe instrucțiune (de repetare sau de blocare a accesului altui procesor la magistrala sa), instrucțiunile procesorului 386/486 pot fi precedate de următoarele prefixe (de câte un octet fiecare):

Prefix instrucțiune	**Prefix segment**	**Prefix dimensiune adresă**	**Prefix dimensiune operand**

În cadrul octetului sau al celor doi octeți de cod de operație, pot fi definite câmpuri codificate mai mici, care diferă în funcție de clasa operației. Aceste câmpuri definesc diverse informații : direcția operației, dimensiunea deplasamentului, codificarea registrelor sau extensia de semn. Aproape toate instrucțiunile cu referire la un operand din memorie au un octet de mod de adresare, plasat după primul octet de cod de operație. Unele codificări ale acestui octet (mod, r/m) indică un al doilea octet de adresare, octetul scală-index-bază SIB.

Octetul SIB conține, codificat, următoarele informații:

- factorul de scală (ce poate fi 1, 2, 4 sau 8), pe primii doi biți;
- registrul index utilizat, pe următorii 3 biți (ca registre index pot fi utilizate 7 dintre cele 8 registre generale, mai puțin registrul ESP);
- registrul de bază utilizat, pe ultimii 3 biți.

Dacă octetul SIB este prezent în instrucțiune, el specifică doar indexul şi factorul de scală (după cum se poate vedea şi în tabela 7.4). Când octetul SIB nu este prezent, câmpul „registru de bază“ alături de celelalte câmpuri din acest octet (SIB) este utilizat doar pentru *r/m=100* (în cazurile *mod=00, 01* şi *10*, adică pe poziția utilizării registrului ESP pentru adresare, în locul utilizării registrului ESP, când octetul SIB este, totuşi, prezent în instrucțiune), la operarea pe 32 de biți; pentru celelalte valori ale câmpului *r/m*, octetul SIB nu este prezent.

Deplasamentul, dacă este prezent în instrucțiune, poate fi de 8, 16 sau 32 de biți (în funcție de dimensiunea implicită a adresei).

Mai există un câmp de un bit, care permite extensia de semn a datelor de 8 biți la 16/32 biți, în funcție de tipul destinației. În cadrul instrucțiunilor de test sunt codificate şi condițiile de test.

Setul de instrucțiuni de la 386/486 este extins în două direcții:

- pentru a permite date pe 32 biți;
- pentru a permite moduri de adresare pe 32 biți.

Extensia aceasta se face în funcție de bitul D (Default) din descriptorul de segment de cod şi prin utilizarea celor două prefixe: dimensiune operand şi dimensiune adresă.

Modul de operare implicit al instrucțiunilor, fie pe 16, fie pe 32 de biți, depinde de bitul D din descriptorul de segment de cod. Bitul D specifică dimensiunea implicită (0 pentru 16 şi 1 pentru 32 de biți) atât pentru operanzi, cât şi pentru adresa efectivă. Cele două prefixe, pentru

dimensiune adrese şi dimensiune operanzi, permit prefixarea şi deci modificarea dimensiunii operandului şi adresei peste selecţia implicită (a bitului D).

În modurile de lucru real şi virtual 8086 (V86) nu se utilizează descriptori de segment, dar valoarea lui D (egală cu 0) este presupusă intern de 386/486. Cele două prefixe permit selectarea individuală a bitului D. Aceste prefixe pot preceda orice cod de operaţie şi afectează numai instrucţiunea ce o preced. Utilizarea celor două prefixe va determina dimensiunea operandului sau a adresei la valoarea „opusă" faţă de starea bitului D. De exemplu: dacă dimensiunea operandului era de 32 de biţi, prezenţa prefixului pentru dimensiune operand va determina operarea cu date de 16 biţi, iar dacă dimensiunea implicită a adresei efective este de 16 biţi, prezenţa prefixului de dimensiune adresă efectivă va determina calculul adresei efective pe 32 de biţi.

Extensiile la 32 de biţi sunt posibile în toate modurile, inclusiv în modul real sau virtual 8086. În aceste moduri, implicit se lucrează pe 16 biţi, astfel că sunt necesare prefixe pentru a specifica adrese şi operanzi de 32 de biţi. Utilizarea modului extins de adresare (adică utilizarea registrelor extinse) va determina asamblorul să insereze prefixul, deci nu este nevoie de a specifica explicit instrucţiunea prefix.

Bitul *w* are semnificaţia, la 386/486, de operare pe 8 biţi (dacă este 0) sau pe 16/32 biţi, adică dimensiunea completă a operandului (dacă este 1).

În ceea ce priveşte codificarea registrelor (câmpurile *reg* şi respectiv *r/m*) la 386/486, aceasta se face în mod asemănător:

- câmpul *w* este prezent în instrucţiune:
  - pentru operaţiile pe 16 biţi sunt aceleaşi adrese;
  - pentru operaţiile pe 32 biţi, pentru *w*=0 la fel ca în tabelă, iar dacă *w*=1 atunci se adresează în aceeaşi ordine registrele EAX, EBX,... EDI;
- câmpul *w* nu este prezent în instrucţiune:
  - pentru operaţiile pe 16 biţi se adresează cu aceleaşi adrese registrele AX, BX,..., DI;
  - pentru operaţiile pe 32 biţi se vor adresa registrele de 32 biţi EAX, EBX,..., EDI.

Pentru selecţia registrului segment se utilizează doi biţi la 286, pentru selecţia unuia din cele patru registre segment: CS, DS, ES, SS, respectiv 3 biţi la 386/486, pentru a putea selecta şi registrele de date suplimentare FS, GS.

Codificarea modurilor de adresare la 486 este aceeaşi ca la 286, dacă se lucrează cu adrese de 16 biţi, iar dacă se utilizează adrese de 32 biţi, atunci se pot utiliza ca registre de bază şi index oricare dintre registrele EAX, EBX,..., EDI (cu excepţia registrului ESP, care nu poate fi utilizat ca registru index). Calculul adresei efective se face în mod asemănător, cu deosebirea că pentru mod=10 deplasamentul este pe 32 biţi (în loc de 16), iar dacă se utilizează octetul SIB, atunci la calculul adresei efective se utilizează scala şi registrul index indicat de acest octet (EAX,..., EDI), ca în tabela 7.4.

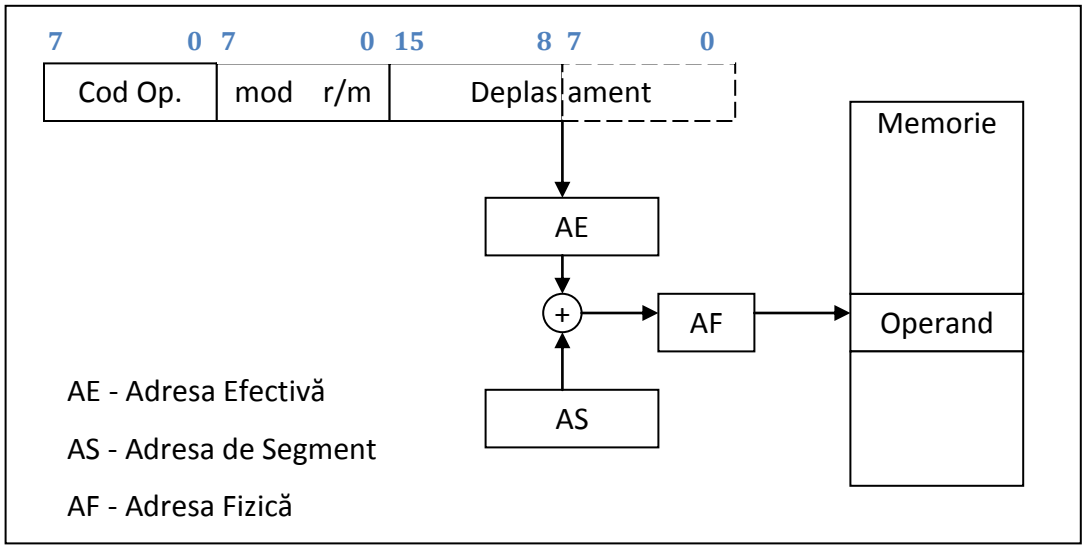
Utilizarea componentelor de bază şi index se face pentru referirea la:

- parametrii şi variabilele locale din stivă ale unei proceduri;
- o structură, pentru mai multe apariţii ale aceluiaşi tip de structură sau într-un vector de structuri;
- un vector cu una sau mai multe dimensiuni;
- un vector alocat dinamic.

Toate completările aduse la procesoarele 386/486 sunt valabile şi pentru procesoarele Pentium.

7.2. Moduri de adresare

■ *Adresarea directă* nu implică nici un registru; adresa efectivă este specificată chiar în codul instrucțiunii, prin deplasament, ca în figură:

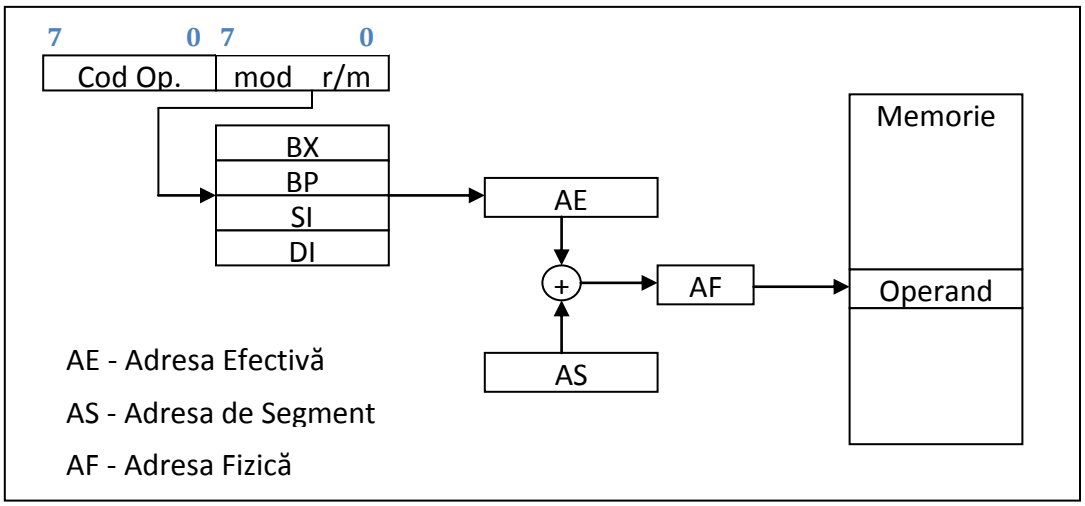


Adresarea directă

Exemple de instrucțiuni:

```
mov     ax, adr_w       ; adr_w – adresa operand cuvant
mov     adr_w[2], si    ; transfer la adresa adr_w + 2
```

■ *Adresarea indirectă prin registre* face referire la memorie prin intermediul registrelor index sau de bază, care vor conține, în acest caz, adresa efectivă a operandului, ca în figură:



Adresarea indirectă, prin registre

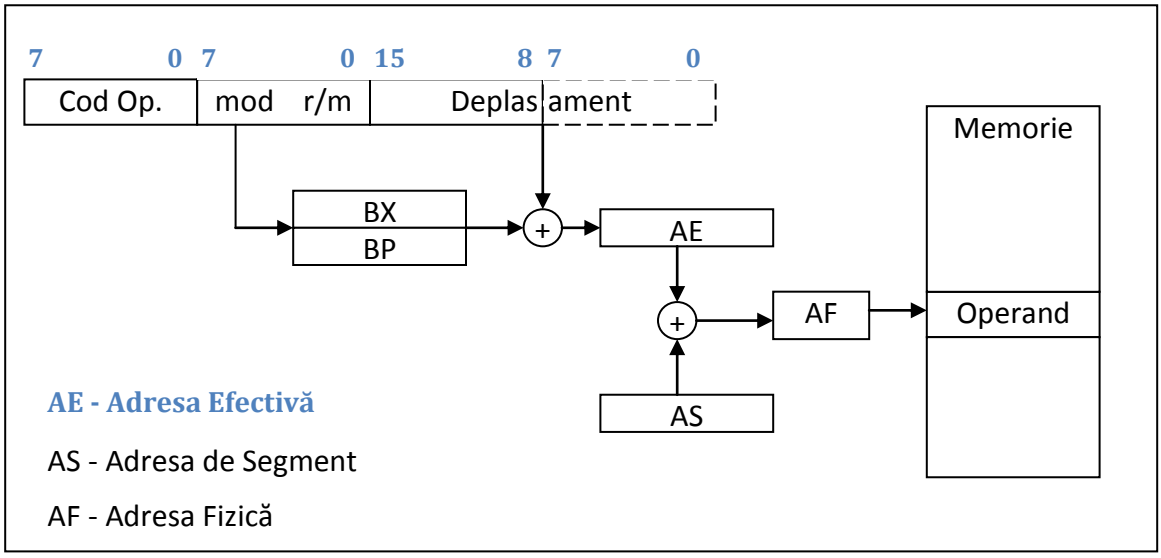
Sintaxa instrucțiunilor în limbaj de asamblare utilizează pentru adresarea indirectă operatorul index [].

Exemple de instrucțiuni:

```
mov     ax, [bx]
mov     bx, [si]
```

La procesorul 286, pentru adresarea indirectă prin registre pot fi utilizate doar registrele index şi de bază.

■ *Adresarea bazată* determină adresa efectivă adunând conținutul unui registru de bază cu deplasamentul din instrucțiune, ca în figură:

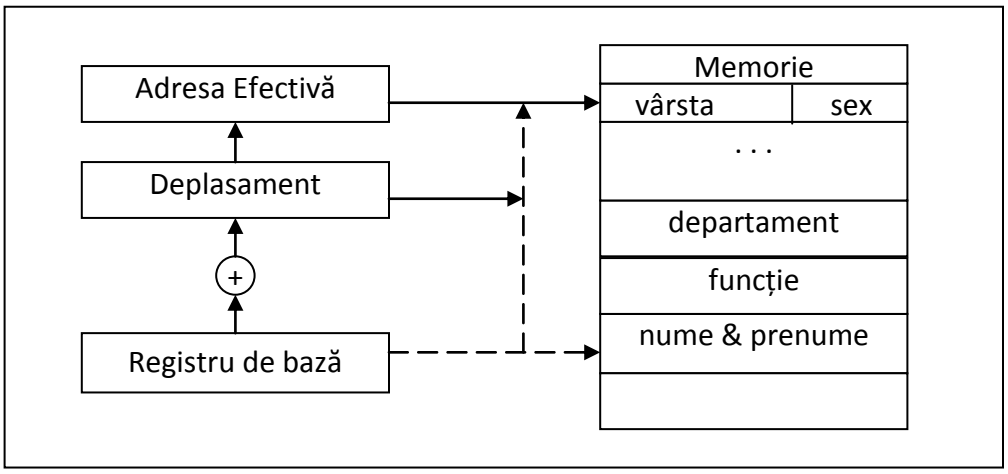


Adresarea bazată

Dacă se utilizează registrul BP ca registru de bază, atunci se pot obține operanzi din segmentul curent de stivă la care face referire SS (dacă nu este prezent un alt prefix segment). În acest mod se poate face referire la datele din stivă fără a descărca stiva.

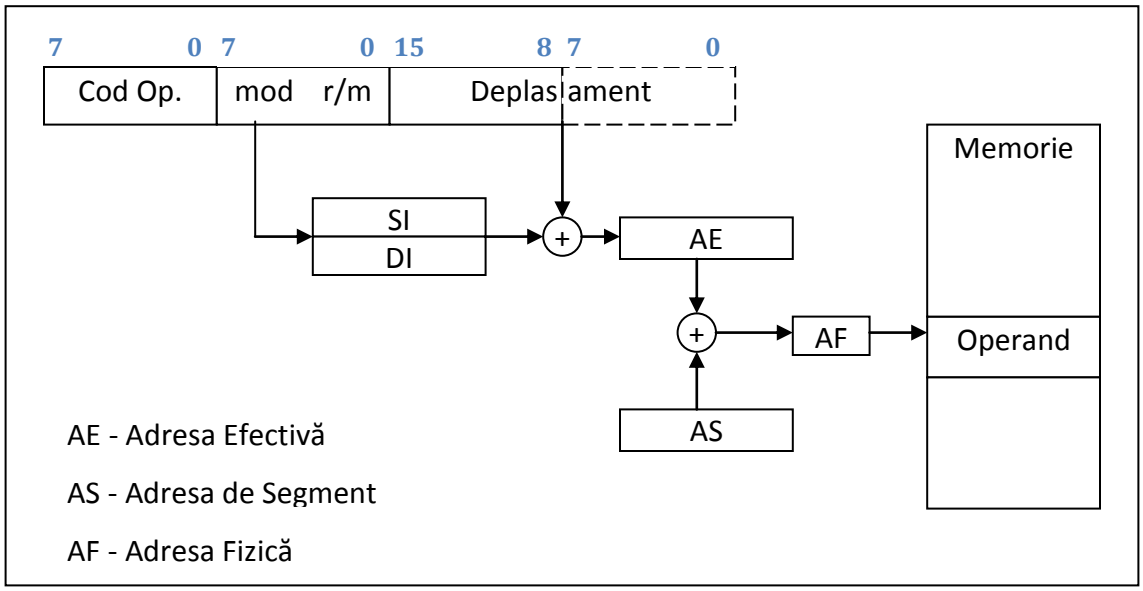
Adresarea bazată furnizează, de asemenea, un mod convenabil pentru a adresa structuri de date similare, care pot fi memorate la adrese diferite de memorie. În acest caz, registrul de bază se va referi la baza structurii, iar elementele structurii vor fi adresate prin deplasamentul lor față de bază. Aceleaşi câmpuri ale structurii pot fi accesibile prin simpla modificare a registrului de bază cu adresa de început a noii structuri. Prin modificarea doar a deplasamentului se poate face referire la alte câmpuri ale structurii (figura următoare). Exemple de instrucțiuni:

```
mov     ax, depl[bx]
mov     ax, [depl + bx]
mov     ax, [bx] + [depl]
```



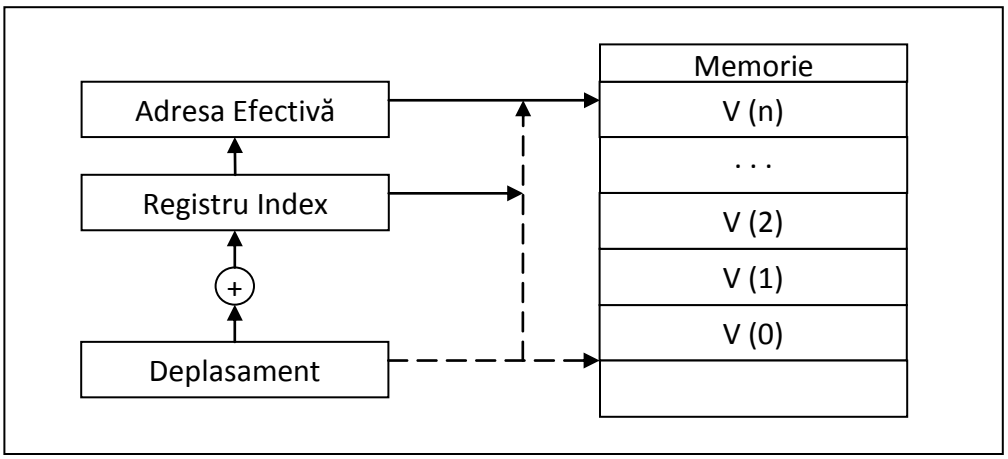
Adresarea elementelor unei structuri

■ *Adresarea indexată* este asemănătoare cu adresarea bazată, întrucât adresa efectivă se obține tot ca o sumă între un registru, de această dată index SI sau DI, şi deplasamentul din instrucțiune.



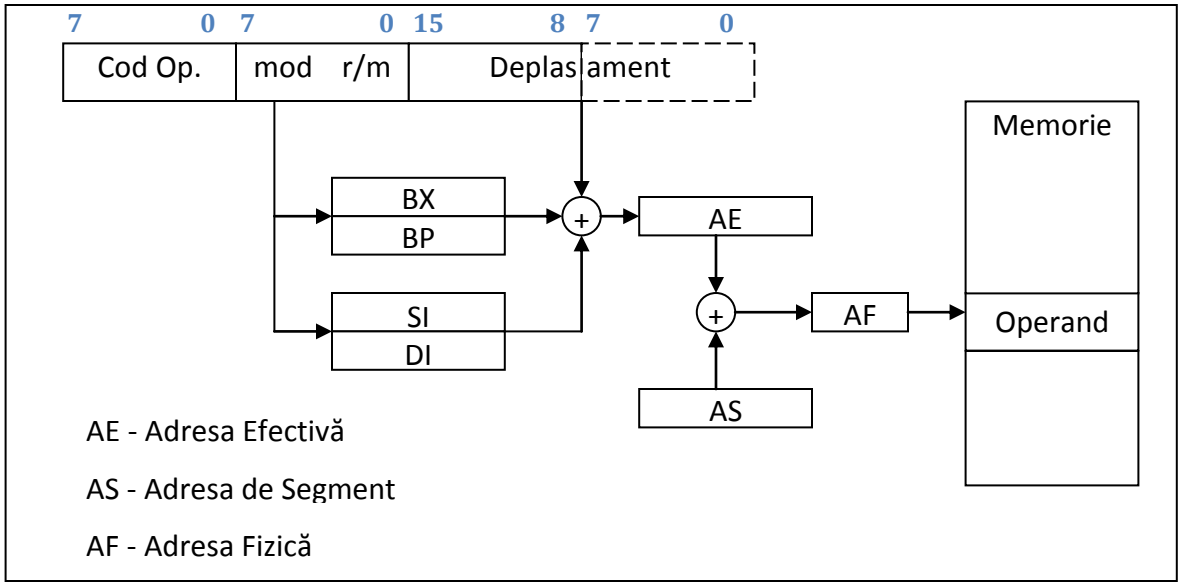
Adresarea indexată

Acest mod de adresare este utilizat pentru referirea la elementele unui vector. Deplasamentul va marca începutul vectorului, iar registrul index va selecta elementul, prin poziția sa relativă în cadrul vectorului. Deoarece toate elementele vectorului sunt de aceeaşi lungime, prin operații aritmetice elementare aplicate registrului index se va selecta orice element; din acest motiv, la procesoarele 386/486 se poate specifica un factor de scală (1,2,4,8) pentru index, în vederea referirii la vectori având componente de lungime fixă, de 1, 2, 4, 8 octeți.



Adresarea elementelor unui vector

■ *Adresarea bazată şi indexată* utilizează pentru calculul adresei efective două registre, de bază şi index, şi un deplasament, ca în figură:



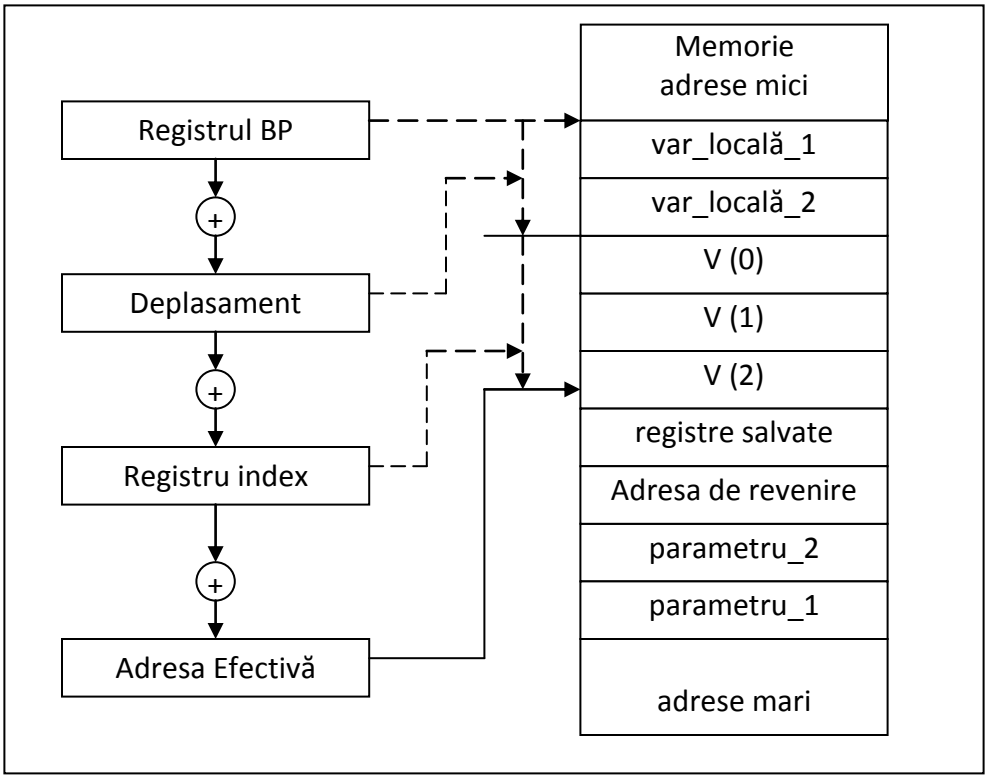
Adresarea bazată şi indexată

Este un mod flexibil de adresare, deoarece pot fi modificate două componente. În acest mod pot fi adresate diferite structuri sau parametrii (de tip structurat) transmişi prin stivă, respectiv variabilele locale unei proceduri, cum este prezentat în figura următoare. Registrul BP va conține deplasamentul unui punct de referință în stivă, de obicei vârful acesteia, după ce procedura a salvat registrele, şi alocă memorie locală variabilelor procedurii. Offsetul unui vector din stivă, față de începutul stivei, poate fi exprimat prin valoarea deplasamentului, iar un registru index este utilizat pentru a adresa elementele individuale ale vectorului. În mod asemănător pot fi adresate şi alte structuri (de exemplu matrice etc.).

Exemple de instrucțiuni:

```
mov     ax, aw[bx][si]
```

mov ax, depl[bp][di]



Adresarea unei structuri din stivă

În cazul adresării elementelor unui vector, adresa (deplasamentul) unui element, de exemplu pentru un vector bidimensional, se obține astfel:

adr_element = adr_baza + (index_linie*dimens_coloana +
index_coloana)*dimens_element

unde *adr_element* este adresa primului element din vector, *v[0][0]*, *dimens_element* este dimensiunea unui element din vector (în octeți), *dimens_linie* este numărul de linii ale vectorului, iar *index_coloană* şi *index_linie* reprezintă indecşii elementului curent adresat.

Pentru un vector tridimensional, adresa unui element se va obține astfel:

*adr_elem=adr_baza+((index_1*dimens_2+index_2)*dimens_3+index_3)* dimens_element*

ceea ce ar corespunde, în pseudocod, unei secvențe de forma:

```
for index_1 = 1 to dimens_1 do
    for index_2 =1 to dimens_2 do
        for index_3 = 1 to dimens_3 do
            v[index_1][index_2][index_3].....
```

În mod asemănător, pentru un vector cu 4 elemente, adresa unui element va fi:

*adr=baza+(((index_1*dim_2+index_2)*dim_3+index_3)*dim_4+index_4)*dim_element*

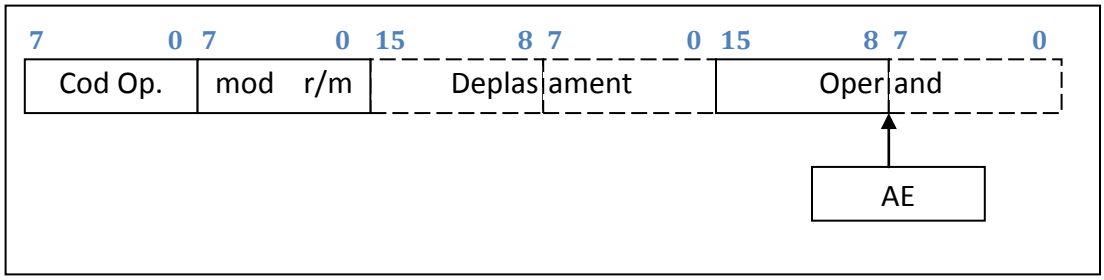
şi în acest mod se poate generaliza pentru un vector de orice dimensiune, *n*.

■ *Adresarea imediată* presupune că operandul se află chiar în instrucțiune – octetul 3 sau octeții 3-4, dacă operandul are 16 biți şi dacă nu avem deplasament, respectiv octeții 5 şi respectiv 5-6, dacă instrucțiunea are şi deplasament.

Exemple de instrucțiuni:

```
mov     ax, 100h
add     beta [bx][si], 0ce43h
mov     alfa [bp][di], 0f7h
```

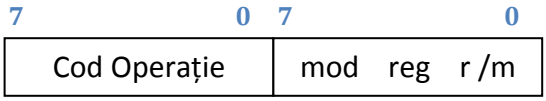
De fapt, se poate realiza o combinație între acest mod de adresare şi celelalte moduri (bazată, indexată, bazată şi indexată), după cum se poate vedea şi din ultimele exemple.



Adresarea imediată

■ *Adresarea la registre*

În acest caz, adresa efectivă a operandului este adresa unui registru general, adică operandul se găseşte într-un registru.



Adresarea la registre

Exemple de instrucțiuni:

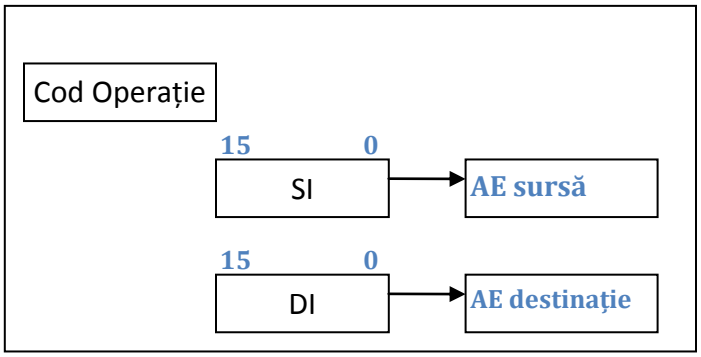
```
mov     ax, si;
mov     ah, cl;
mov     ds, bx
```

■ *Adresarea şirurilor*

Instrucțiunile pe şiruri nu utilizează modurile normale de adresare a memoriei pentru a adresa operanzii şirurilor. Se utilizează, în mod implicit, registrele index pentru a determina adresa efectivă, iar pentru a determina adresa de segment se utilizează registrele segment ES

pentru şirul destinație, respectiv DS pentru şirul sursă (dacă nu este prefixat un alt registru segment pentru sursă). Dacă operația pe şir se execută în mod repetat, atunci registrele index sunt ajustate automat pentru a face referire la elementul următor din şir. Exemple de instrucțiuni:

```
movs        sir_dest, sir_sursa
movsb
cmpsw
```



Adresarea şirurilor

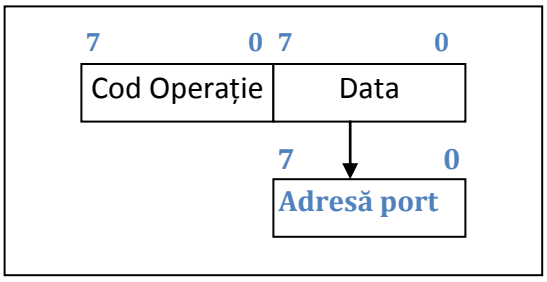
■ *Adresarea porturilor de intrare/ieşire (I/O)*

Pentru a adresa porturile din spațiul de I/O se pot utiliza două moduri:

- *adresarea directă a porturilor*, care presupune că adresa portului de I/O se găseşte în instrucțiune, pe 8 biți, şi deci pot fi adresate porturile din spațiul de adrese 0-255.

Exemple de instrucțiuni:

```
in          al, port_octet; citire de port de tip octet
in          ax, port_cuvant; citire de la port cuvant
out         port_oct, al; scriere la port octet
out         port_cuv, ax; scriere la port cuvant
```

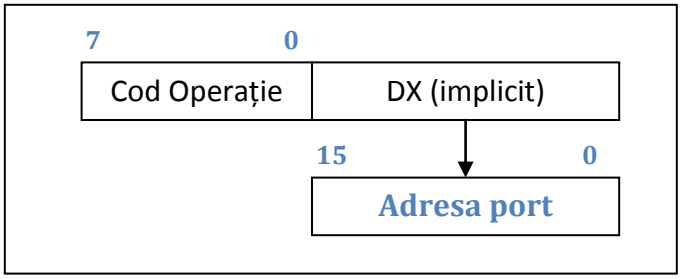


Adresarea directă a porturilor de I/O

- *adresarea indirectă a porturilor* de I/O, care se realizează în mod implicit prin registrul DX, care conține adresa efectivă a portului (instrucțiunea are un singur octet). În acest mod pot fi adresate toate porturile din spațiul de I/O: 0 – 65535. Exemple:

```
in          al, dx ; citire de la port octet
```

```
in          ax, dx ; citire de la port cuvant
out         dx, al ; scriere la port octet
out         dx, ax ; scriere la port cuvant
```



Adresarea indirectă a porturilor de I/O

Trebuie menționat că pentru adresarea porturilor de I/O, deci a spațiului de I/O, nu se utilizează registrele segment. Adresa furnizată de instrucțiune este adresa efectivă a portului respectiv în spațiul de I/O, de 64 Ko.