Moduri de adresare

Curs #7

Cuprins

- **Consideraţii generale**
  - Adrese virtuale
  - Adresa efectiva la I8086
  - Rationamente de reprezentare
  - Structura generala a modurilor de adresare
- **Moduri de adresare**
  - Adresare imediată
  - Adresare directă prin registru
  - Adresare indirecta prin registru
  - Adresare cu autoincrementare/autodecrementare
  - Adresare indirectă cu autoincrementare/autodecrementare
  - Adresare indirectă bazată cu deplasament
  - Adresare dublu indirectă bazată cu deplasament
  - Adresare indirectă bazată indexată
  - Adresare indirectă bazată indexată cu autoincrementare/decrementare
  - Adresare indirectă bazată indexată cu deplasament
  - Adresare indirectă indirect bazată indexată cu deplasament
  - Adresare indirectă cu deplasament indirect bazată cu deplasament
  - Adresare indirectă cu deplasament indirect bazată indexată cu deplasament

Adrese virtuale

- Adresele incluse în programele executate de microprocesoarele moderne (după I8086) nu sunt fizice.
- Ele sunt relative la spaţiul de memorie alocat programului la momentul execuţiei de către sistemul de operare.
- Programul conţine, deci adrese virtuale care la momentul execuţiei sunt transformate în adrese fizice.
- Spaţiul de memorie al programului este un spaţiu virtual (memorie virtuală) necesar pentru execuţia programului şi asigurat de sistemul de operare prin infrastructura pusă la dispoziţie de microprocesor.

Adresa efectivă la I8086

- In plus la I8086 dimensiunea magistralei de adrese este de 20 biţi, iar dimensiunea registrelor de adresare este de 16 biţi.
- Pentru a se ajunge la adresa fizică (efectivă) de memorie se realizează următoarele operaţii:
  - **adresa de segment** (AS) care pointează începutul segmentului de memorie este completată cu patru biţi de zero (prin deplasare la stănga), obţinându-se o adresă pe 20 de biţi
  - la această adresă se adună **adresa offset** (deplasamentul din segmentul de memorie)
    Adresa efectivă (pe 20 de biţi) va fi deci: **`AE = AS0000 + AO`**

Raţionamente de reprezentare

- În toate limbajele de programare procedurale, datele prelucrate se păstrează în memorie.
- Operanzii utilizaţi în instrucţiunile microprocesorului sunt specificaţi implicit sau explicit:
  - **implicit** de instrucţiune în cazul instrucţiunilor care se referă la un anumit registru al microprocesorului
  - **explicit** în instrucţiune - cazul în care în instrucţiune apare operandul.
- Modurile de adresare specifica modalitatea explicită de obţinere a operanzilor.

Raţionamente de reprezentare

- Codul instrucţiunii cuprinde două câmpuri:
  - câmpul **CODOP** – ce specifică operaţia de executat
  - câmpul **operand** – ce specifică în mod explicit operandul.

Raţionamente de reprezentare

- Câmpul operand trebuie să indice operanzii care se prelucrează (unul sau doi în funcţie de operaţie) şi locul unde se depune rezultatul operaţiei.
- În cazul extrem câmpul operand ar trebui să codifice doi operanzi şi adresa rezultatului.
- Pentru a reduce dimensiunea instrucţiunii, la unele microprocesoare, inclusiv Intel, rezultatul este depus la locaţia unui operand.
- Astfel se codifică maxim două obiecte şi de aici instrucţiuni cu dimensiune mai mică şi deci programe care ocupă mai puţin spaţiu în memorie.

Raţionamente de reprezentare

- Operanzii se pot păstra în registre sau în memorie.
- Registrele constituie modalitatea optimă de păstrare deoarece se codifică pe mai puţin biţi şi deci ocupă spaţiu redus pentru reprezentare şi operaţiile realizate sunt mai rapide fiind direct legaţi la unităţile de execuţie.

Raţionamente de reprezentare

- Din păcate numărul registrelor este limitat. La microprocesorul studiat, numărul de registre este de 8 registre de 8 biţi: AL, AH, BL, BH, CL, CH, DL, DH.
- Cele 8 registre se codifică pe 3 biţi astfel:

000	AL
001	AH
010	
011	
100	
101	
110	
111	

Structura generală a modurilor de adresare

- Modurile de adresare specifică modalitatea de obţinere a operanzilor.
- Păstrarea operanzilor în memorie oferă flexibilitate programatorului şi posibiltăţi multiple de obţinere a lor.
- Se poate astfel ca un operand să se găsească la o locaţie de memorie sau la acea locaţie să se afle adresa operandului sau adresa adresei operandului.
- În plus la nivelul microprocesorului se oferă suport pentru implementarea structurilor complexe de date utilizate în limbajele de nivel înalt.

- Elementele de bază care se utilizează pentru obţinerea oricărui mod de adresare sunt:
  - obiecte
  - funcţii de bază.

- Obiectele din modurile de adresare sunt:
  - registre ale microprocesorului
  - deplasamente (offset) în segmentele de memorie, specificate în câmpul operand al instrucţiunilor

- Registrele microprocesorului au următoarele funcţii:
  - registru operand – în registru se află valoarea operandului
  - registru indirect – în registru se află adresa operandului
  - registru de bază – în registru se află o adresă de bază la care se adună offsetul pentru a se obţine adresa operandului.
- Deplasamentul din câmpul instrucţiunii poate reprezenta:
  - valoarea operandului
  - adresa operandului
  - valoarea deplasamentului care se adună la adresa de bază pentru obţinerea adresei operandului.

- Funcţiile de bază realizate în modurile de adresare sunt:
  - *adunarea* – specifică adunarea mai multor valori pentru a obţine adresa operandului
  - *indirectarea* – adresa utilizată specifică adresa operandului
  - *deplasarea* – utilizată pentru înmulţire cu 2 (2^1), 4 (2^2), 8 (2^3), etc (ce specifică numărul de octeţi ai elementelor structurilor de date stocate în memorie) şi utilizată pentru referirea elementelor succesive din memorie.

Problematica modurilor de adresare

- Aducerea operanzilor din memorie necesită cicluri suplimentare de citire a memoriei.
- Un ciclu pentru a obţine un operand a cărui adresă se cunoşte, două cicluri dacă se cunoaşte adresa adresei operandului ş.a.
- Moduri de adresare sofisticate duc la timpi suplimentari de execuţie.
- Avantajul acestor moduri de adresare îl reprezintă creşterea în flexibilitate a programului. Astfel dacă modul de adresare este primar şi valoarea operandului se specifică în program, la schimbarea operandului trebuie schimbat şi programul, recompilat şi linkeditat.
- Dacă se cunoşte adresa atunci se poate schimba conţinutul de la adresa respectivă fără a necesita rescrierea programului.
- In plus dacă se lucrează cu şiruri de caractere, operaţiile se repetă pentru fiecare element al şirului, iar pentru adresare reîncărcarea registrului cu deplasamentul elementului curent se poate face utilizând un mod de adresare care facilitează această operaţie.

MODURI DE ADRESARE

Moduri de adresare

- Sunt prezentate in continuare cele mai utilizate moduri de adresare si exemple de folosire a lor.

Se vor folosi următoarele convenţii de notare:

r = **registru**

d = **deplasament**

R(r) = conţinutul registrului r

M(x) = funcţie de adresare indirectă a valorii x

x sh y = delasare la stânga a lui x cu y poziţii binare.

o = **operand**

Adresare imediată

- În instrucțiune apare valoarea operandului
  - `o = d` (valoarea operandului se află în câmpul deplasament al instrucţiunii)

Adresare imediată - exemplu

```
.data
var1   DB   2
var2   DW   5 dup('ab')

.code
mov al, 2                    ;valoarea operandului este in instructiune
mov ax, offset var2          ;valoarea operandului este adresa relativa in
segmentul de date
mov ax, data                 ;valoarea operandului este adresa de segment
pentru data
```

Adresare directă prin registru

- În instrucțiune apare adresa operandului care se încarcă în/dintr-un registru
- `o = R(r)` (valoarea operandului se află în registru)

Adresare directă prin registru - exemplu

```
mov ax,var2 ;var2 reprezinta adresa de memorie de la care se incarca
operandul

mov ax, word ptr var2+1 ;word ptr specifica dimensiunea word (2
octeti) pentru operand, deci se incarca in ax valorile de la adresa
var2+1

mov var2+2,ax
mov var2(2),ax
add cx,[100]
```

Adresare indirectă prin registru

- Operandul se gaseste in registru (**`bx`**, **`si`** sau **`di`**)
- Registrul conţine adresa operandului.
  - `o = M(R(r))` (*adresa operandului se află în registru*)

Adresare indirectă prin registru - exemplu

```
mov ax,[bx] ;operandul se afla la adresa indicata de registrul bx
mov [di],cx
add byte ptr[si],2 ;instructiune echivalenta cu add[si],2
;(cu obs. ca nu se cunoaste dimensiunea operandului)
```

Adresare cu autoincrementare/ autodecrementare

- Adresare utilizată cu registrul contor de instrucţiuni (IP) şi registrul indicator de stivă (SP).
- Registrul conţine adresa operandului şi valoarea acestuia se postincrementează sau predecrementează cu n, unde n reprezintă un număr de octeţi corespunzător operaţiei.
- `o = M(R(r)), R(r) = R(r)+n`
- `R(r) = R(r)-n, o = M(R(r))`

Adresare indirectă cu autoincrementare/ autodecrementare

- Un registru păstrează adresa adresei operandului, iar conţinutul registrului se autoincrementează/ autodecrementează:
  - o =M(M(R(r))), R(r) = R(r)+n
    - R(r) = R(r)-n, M(M(R(r)))
- Adresa se află în câmpul instrucţiunii şi nu se modifică în timpul execuţiei. Se pot astfel referi sirurile stocate în memorie.
- Modul de adresare se utilizează în instrucţiunile cu şiruri.

Adresare indirectă bazată cu deplasament

- Adresa operandului se calculează prin adunarea conţinutului unui registru de bază specificat cu valoarea unui deplasament.
  - o =M(R(r)+d)
- Acest mod de adresare este potrivit pentru adresarea elementelor unei structuri de date.
- Registrul conţine adresa de bază de început a structurii de date, iar deplasamentul numărul de octeţi până la elementul respectiv.

Adresare bazata - exemplu

```
.data
var DW 10 dup(10)
.code
mov bx,5 ;s-a pregatit registrul de baza

mov ax,var[bx]
mov ax,bx[var]
mov ax,[bx+var]
mov ax,[bx].var
;instructiunile sunt echivalente si transfera in ax al 5-lea
element de doi octeti de la adresa var sau
mov bx,offset var ;s-a pregatit registrul de baza
mov ax,[bx]
```

Adresare dublu indirectă bazată cu deplasament

- Adresa operandului se calculează prin adresarea locatiei indicate prin adunarea conţinutului unui registru de bază specificat cu valoarea unui deplasament.
  - `o =M(M(R(r)+d))`
- Acest mod de adresare este potrivit pentru adresarea variabilelor la care referirea se face printr-un pointer. Structura de date este stocată la o adresă relativă de memorie. Registrul conţine adresa de bază de început a structurii de date, iar deplasamentul numărul de octeţi până la elementul respectiv.

Adresare indirectă bazată indexată

- Adresa operandului se calculează prin adunarea adresei de bază dintr-un registru cu indexul elementului ce reprezintă operandul. Indexul se obţine prin shiftare la stânga cu numărul ce reprezintă puterea lui 2 necesară pentru a specifica dimensiunea unui element.
  - o =M(R(r_1)+R(r_2).sh.n)
- În instructiune apar registri de baza si index utilizati pentru obtinerea adresei operandului. Adresa efectiva se obtine adunand valoarea din registrul de baza cu cea din registrul index si adresa relativa continuta in instructiune.

Adresare indexată - exemplu

- Pentru obtinerea adresei operandului se foloseste un registru index (**`si`** sau **`di`**), iar adresa stocata in acesta se aduna cu un deplasament.

- Registrul segment pentru **`si`** este **`ds`**, iar pentru **`di`** este **`es`**.

- Utilizarea este aceeasi cu adresarea bazata

```
mov si,5
mov ax,var[si]
```

Adresare indirectă bazată indexată cu autoincrementare/decrementare

- Adresa operandului se calculează prin adunarea adresei de bază dintr-un registru cu indexul elementului ce reprezintă operandul. Indexul se obţine prin shiftare la stânga cu numărul ce reprezintă puterea lui 2 necesară pentru a specifica dimensiunea unui element.

- Conţinutul registrului de bază se autoincrementează/autodecrementează.

- `o =M(R(r1)+R(r2).sh.n), R(r1)= R(r1)+N`

Adresare indirectă bazată indexată cu deplasament

- Adresa operandului se calculează prin adunarea adresei de bază dintr-un registru cu un deplasament şi cu indexul elementului ce reprezintă operandul. Indexul se obţine prin shiftare la stânga cu numărul ce reprezintă puterea lui 2 necesară pentru a specifica dimensiunea unui element.

- o `=M(R(r1)+d+R(r2).sh.n)`

- Ne putem imagina o adresă de structură de date stocată la o adresă relativă şi care conţine un şir de elemente la un deplasament d de începutul structurii.

```
mov ax,[bx+di+offset var]
```

Adresare indirectă indirect bazată indexată cu deplasament

- Adresa operandului se calculează indirectarea adresei obţinute prin adunarea adresei de bază dintr-un registru cu un deplasament şi adunate cu indexul elementului ce reprezintă operandul. Indexul se obţine prin shiftare la stânga cu numărul ce reprezintă puterea lui 2 necesară pentru a specifica dimensiunea unui element.

- `o =M(M(R(r1)+d)+R(r2).sh.n)`

Adresare indirectă cu deplasament indirect bazată cu deplasament

- Adresa operandului se calculează indirectarea adresei obţinute prin adunarea adresei de bază dintr-un registru cu un deplasament şi adunate cu deplasamentul elementului ce reprezintă operandul.

- `o =M(M(R(r1)+d1)+d2)`

Adresare bazata si indexata - exemplu

- In instructiune apar registri de baza si index utilizati pentru obtinerea adresei operandului. Adresa efectiva se obtine adunind valoarea din registrul de baza cu cea din registrul index si adresa relativa continuta in instructiune.

- mov ax,var[bx][si]
  mov ax,var[bx+si]
  mov ax,[bx+di+offset var]

Adresare indirectă cu deplasament indirect bazată indexată cu deplasament

- Adresa operandului se calculează indirectarea adresei obţinute prin adunarea adresei de bază dintr-un registru cu un deplasament adunată cu deplasamentul structurii în care se află elementul ce reprezintă operandul.

- `o =M(M(R(r1)+d1)+d2+ R(r2).sh.n)`

Concluzii referitoare la modurile de adresare

- Modurile de adresare pot fi împărţite în următoarele categorii:
  - moduri simple
  - moduri complexe (cu deplasament, registru de bază, registru index)
  - moduri complexe cu autoincrementare/autodecrementare.
- Cum adresarea memoriei nu se realizează static, ci dinamic, modurile de adresare complexe se potrivesc sistemelor moderne de operare şi tehnicilor dinamice de programare.
- Autoincrementarea/autodecrementarea se utilizează pentru:
  - extragerea instrucţiunilor de executat de către microprocesor
  - lucrul cu stiva (organizată LIFO)
  - operaţii cu şiruri