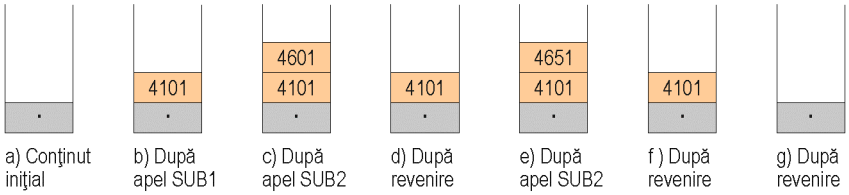


**Figura 5.6.** Utilizarea stivei pentru păstrarea adreselor de revenire ale subrutinelor din Figura 5.5.

La apelul unei subrutine, trebuie să i se transmită acesteia anumiți *parametri*. Aceştia pot fi plasați în *registre* înainte de apel, pot fi depuşi într-o *zonă de memorie*, sau pot fi transmişi prin *stivă*. Ultima metodă este cea mai utilizată de compilatoarele limbajelor de nivel înalt. În acest caz, pe lângă adresa de revenire, se depun în stivă şi parametrii transmişi subrutinei.

### 5.4.5. Instrucțiuni de control al sistemului

Acestea sunt de obicei instrucțiuni *privilegiate*, care pot fi executate numai dacă UCP este într-un mod privilegiat sau execută un program aflat într-o zonă specială de memorie. De obicei, aceste instrucțiuni sunt rezervate sistemului de operare.

Instrucțiunile de control pot citi sau înscrie anumite registre speciale, numite registre de control, sau pot modifica o cheie de protecție a memoriei.

### 5.4.6. Instrucțiuni de I/E

Aceste instrucțiuni permit transferul datelor între un periferic şi UCP. Comunicația cu perifericele se realizează prin registre ale interfețelor de I/E. Aceste registre pot fi:

- Registre de *stare*, care permit citirea stării perifericului;
- Registre de *comandă*, prin care se pot transmite diferite comenzi perifericului;
- Registre de *date*, prin intermediul cărora sunt transferate datele între UCP şi periferic.

## 5.5. Moduri de adresare

Modurile de adresare indică felul în care se specifică operanzii unei instrucțiuni. Aceste moduri de adresare influențează numărul referințelor la memorie pentru obținerea operanzilor, complexitatea calculului adreselor şi flexibilitatea operațiilor care pot fi executate.

Se utilizează următoarele *notații*:

A Conținutul unui câmp de adresă al instrucțiunii;

AE *Adresa efectivă* a locației sau a registrului care conține operandul la are se face referire;

(X) Conținutul locației X.

Fiecare calculator permite mai multe moduri de adresare. Diferite coduri de instrucțiuni pot utiliza diferite moduri de adresare. În cadrul formatului instrucțiunii, există de obicei un *câmp al modului de adresare*, care indică modul de adresare utilizat.

### 5.5.1. Adresarea imediată

Este cea mai simplă formă de adresare, în care operandul se află în cadrul instrucțiunii, în câmpul de adresă:

OPERAND = A

Acest mod poate fi utilizat pentru definirea şi utilizarea constantelor, sau setarea valorii inițiale a variabilelor (Figura 5.7). CO reprezintă codul operației.

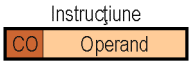


**Figura 5.7.** Adresarea imediată.

Avantajul adresării imediate este că pentru obținerea operandului nu sunt necesare referiri suplimentare la memorie, pe lângă încărcarea instrucțiunii din memorie. Dezavantajul este că dimensiunea operandului este limitată la cea a câmpului de adresă.

### 5.5.2. Adresarea directă

Câmpul de adresă conține adresa efectivă a operandului (Figura 5.8):

AE = A

Acest mod de adresare necesită o singură referire la memorie, şi nu necesită un calcul de adresă. Dezavantajul este că permite un spațiu de adresare limitat, deoarece lungimea câmpului de adresă este de obicei mai mică decât lungimea cuvântului.

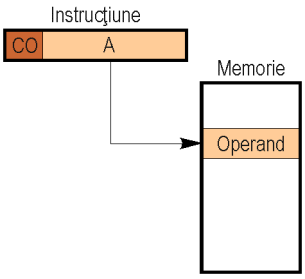


**Figura 5.8.** Adresarea directă.

### 5.5.3. Adresarea indirectă

Pentru a extinde spațiul de adresare, o soluție constă în utilizarea câmpului de adresă din instrucțiune ca o referință la un cuvânt de memorie, care conține adresa completă a operandului. Aceasta reprezintă *adresarea indirectă*:

$$AE = (A)$$

Avantajul este că pentru un cuvânt de lungime $N$, este disponibil un spațiu de adresare de $2^N$. Dezavantajul este că execuția instrucțiunii necesită două referiri la memorie pentru obținerea operandului, una pentru citirea adresei operandului, şi alta pentru valoarea acesteia (Figura 5.9).

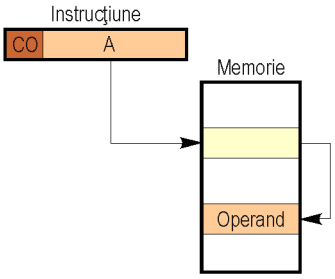


**Figura 5.9.** Adresarea indirectă.

O variantă a adresării indirecte este adresarea indirectă *multinivel*:

$$AE = ( \ldots (A) \ldots)$$

În acest caz, există în cadrul adresei un bit *indicator de indirectare I*. Dacă acesta este 0, adresa reprezintă o adresă efectivă. Dacă este 1, este necesar un alt nivel de indirectare. Dezavantajul este că sunt necesare trei sau mai multe referiri la memorie pentru încărcarea operandului.

### 5.5.4. Adresarea registrelor

Este similară cu adresarea directă. Diferența constă în faptul că adresa se referă la un registru şi nu la memoria principală (Figura 5.10):

AE = R

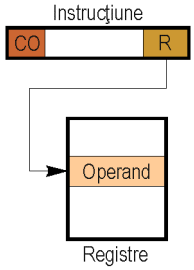


**Figura 5.10.** Adresarea registrelor.

De obicei, un câmp de adresă destinat registrelor are 3 sau 4 biți, astfel că se pot adresa 8 sau 16 registre generale.

Avantajele acestui mod de adresare sunt că este necesar un câmp de adresă de lungime mică, şi nu sunt necesare referiri la memorie pentru citirea operandului. Dezavantajul este că spațiul de adresare este foarte limitat.

### 5.5.5. Adresarea indirectă prin registru

Acest mod de adresare este similar cu adresarea indirectă, dar câmpul de adresă se referă la un registru (Figura 5.11). Astfel,

AE = (R)

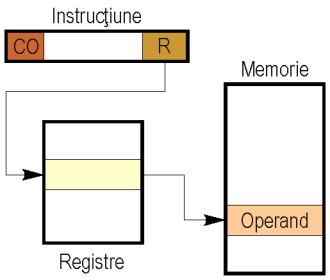


**Figura 5.11.** Adresarea indirectă prin registru.

Avantajele şi limitările adresării indirecte prin registru sunt aceleaşi ca şi pentru adresarea indirectă. Adresarea indirectă prin registru necesită mai puține referiri la memorie decât adresarea indirectă.

## 5.5.6. Adresarea cu deplasament

Acest mod de adresare combină posibilitățile adresării directe cu cele ale adresării indirecte prin registru. Este cunoscut prin diferite denumiri, dar metoda de bază este aceeaşi (Figura 5.12). Adresa efectivă se calculează prin relația:

$$AE = A + (R)$$

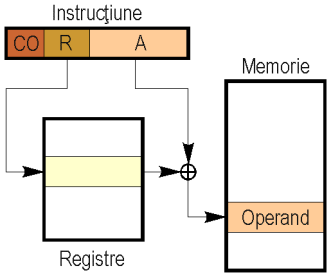


**Figura 5.12.** Adresarea cu deplasament.

Adresarea cu deplasament necesită ca instrucțiunea să aibă două câmpuri de adresă, din care cel puțin unul să fie explicit. Valoarea conținută în primul câmp (A) este utilizată direct. Al doilea câmp de adresă, sau o referință implicită bazată pe codul operației, se referă la un registru al cărui conținut este adunat la A pentru a genera adresa efectivă.

Se vor descrie trei din utilizările cele mai frecvente ale adresării cu deplasament.

### 5.5.6.1. Adresarea relativă

În acest caz, registrul referit în mod implicit este contorul de program (PC). Deci, adresa instrucțiunii curente este adunată la conținutul câmpului de adresă pentru a genera AE. De obicei, câmpul de adresă este tratat ca un număr în complement față de 2 pentru această operație. Deci, câmpul de adresă este un deplasament relativ la adresa instrucțiunii.

Deoarece majoritatea referințelor la memorie sunt relativ apropiate de instrucțiunea curentă, utilizarea adresării relative reduce numărul de biți de adresă din instrucțiune.

### 5.5.6.2. Adresarea bazată

Registrul la care se face referire, numit *registru de bază*, conține o adresă de memorie, iar câmpul de adresă conține un deplasament față de acea adresă. Deplasamentul este de obicei un întreg fără semn. Referința la registru poate fi explicită sau implicită.

Acest mod de adresare permite implementarea segmentării memoriei. În anumite implementări, se utilizează un singur registru de bază al segmentului, care este referit în mod implicit. În alte cazuri, se poate selecta un registru care va păstra adresa de bază a unui segment, referirea la acest registru realizându-se în mod explicit.

### 5.5.6.3. Adresarea indexată

Câmpul de adresă conține o adresă de memorie, iar registrul referit, numit *registru de index*, conține un deplasament pozitiv față de acea adresă. Această utilizare este inversă față de cea de la adresarea bazată. Deoarece câmpul de adresă este considerat ca o adresă de memorie, în general conține un număr mai mare de biți decât un câmp de adresă dintr-o instrucțiune cu adresare bazată. Totuşi modul de calcul al adresei efective este acelaşi în ambele cazuri.

O utilizare importantă a indexării este de a asigura un mecanism eficient pentru executarea *operațiilor iterative*. De exemplu, considerăm o listă de numere memorate începând de la adresa A. Presupunem că trebuie să se adune o valoare la fiecare element al listei. Secvența adreselor efective necesare este A, A+1, A+2, …, până la ultima locație a listei. Dacă se utilizează indexarea, adresa A este depusă în câmpul de adresă al instrucțiunii, iar registrul de index este inițializat cu 0. După fiecare operație, registrul de index este incrementat cu 1.

Deoarece registrele de index sunt utilizate în mod frecvent pentru asemenea operații iterative, ele fiind incrementate sau decrementate după fiecare utilizare a lor, anumite sisteme pot executa în mod automat incrementarea sau decrementarea. Această variantă a adresării indexate se numeşte *autoindexare*. Dacă anumite registre sunt dedicate exclusiv indexării, autoindexarea poate fi implicită, fiind executată automat. Dacă se utilizează registre generale, operația de autoindexare trebuie indicată printr-un bit al instrucțiunii.

Autoindexarea prin incrementare poate fi descrisă astfel:

$$AE = A + (R)$$
$$(R) \leftarrow (R) + 1$$

La anumite calculatoare, este posibilă atât adresarea indirectă, cât şi indexarea, şi ele pot fi utilizate în aceeaşi instrucțiune. Există două posibilități: indexarea este executată înainte de indirectare, sau este executată după indirectare.

Dacă indexarea este executată după indirectare, ea se numeşte *postindexare*.

$$AE = (A) + (R)$$

Întâi, conținutul câmpului de adresă este utilizat pentru accesul la o locație de memorie care conține adresa operandului (o adresă directă). Această adresă este apoi indexată prin conținutul registrului.

Această tehnică este utilă pentru accesul la conținutul unui bloc de date cu o structură fixă. Registrul de index va conține deplasamentul în cadrul blocului.

În cazul *preindexării*, indexarea este executată înainte de indirectare:

$$AE = (A + (R))$$

Câmpul de adresă este indexat întâi cu conținutul registrului. Adresa rezultată este utilizată pentru accesul la locația de memorie care conține adresa operandului.

Un exemplu de utilizare a acestei tehnici este pentru construirea unei *tabele de salt.* Într-un anumit punct al programului, poate exista un salt la un număr de locații, în funcție de o anumită condiție. Se poate construi o tabelă cu toate adresele de salt, adresa de început a tabelei fiind indicată în câmpul de adresă al instrucțiunii. Prin indexare în cadrul acestei tabele, se poate găsi adresa de salt necesară.

### 5.5.7. Adresarea stivei

Stiva este implementată de obicei în memoria internă, într-o zonă rezervată de program. Adresa vârfului stivei este conținută într-un registru numit indicator de stivă (SP). Astfel, referirile la locațiile din memoria stivă sunt realizate de fapt utilizând adresarea indirectă prin registru (Figura 5.13).

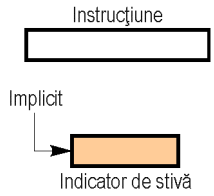


**Figura 5.13.** Adresarea stivei.

Adresarea stivei este o formă a adresării implicite. Instrucțiunile cu stiva nu trebuie să conțină o referință la memorie, operând implicit asupra elementelor din vârful stivei.

## 5.6. Formatul instrucțiunilor

Formatul unei instrucțiuni defineşte structura acesteia, din punct de vedere al părților componente. Acest format trebuie să conțină un cod de operație şi unul sau mai mulți operanzi, definiți implicit sau explicit. Fiecare operand definit explicit poate fi accesat utilizând unul din modurile de adresare descrise. Formatul trebuie să indice, în mod implicit sau explicit, modul de adresare pentru fiecare operand. Pentru cele mai multe seturi de instrucțiuni, se utilizează mai multe formate de instrucțiuni.