De obicei, memoriile ROM utilizează organizarea 2D, toți biții unui cuvânt fiind în cadrul aceluiași circuit. Memoriile RAM sunt organizate de multe ori ca având un singur bit într-un circuit integrat.

8.4. Memoria stivă

Memoria stivă este o listă liniară de date elementare, în care inserarea, eliminarea şi accesul la elementele de date se efectuează la un singur capăt al acesteia. Se mai numeşte listă de tip LIFO (*Last-In, First-Out*).

În funcție de modul de implementare, stiva poate fi de mai multe tipuri:

- Stivă *software*, organizată în memoria internă;
- Stivă *cablată* (hardware);
- Stivă *parțial cablată*.

Indiferent de modul de implementare, o stivă trebuie să permită efectuarea următoarelor operații:

- Introducere (inserare): PUSH;
- Extragere (eliminare): POP;
- Acces la elementul din vârful stivei.

8.4.1. Tipuri de memorii stivă

8.4.1.1. Stiva implementată în memorie

Stiva poate fi simulată în memoria internă a calculatorului, utilizându-se adresarea convențională. Gestionarea stivei se poate realiza prin software, existând instrucțiuni speciale şi registre dedicate pentru operațiile cu stiva (Figura 8.5).

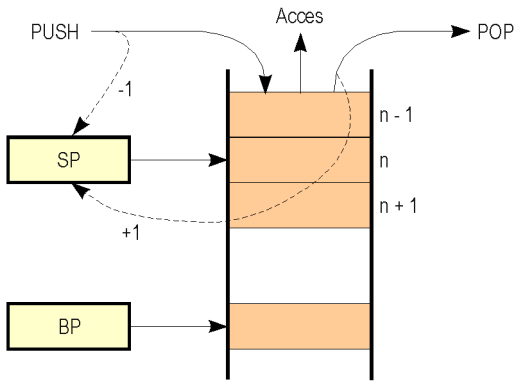

Figura 8.5. Stivă implementată în memorie.

Stiva poate fi definită cu ajutorul a două registre. Registrul BP indică baza stivei, iar registrul SP indică vârful stivei. Adresa de bază a stivei rămâne fixă, în timp ce adresa care indică vârful stivei se modifică la fiecare operație de introducere și eliminare din stivă. Instrucțiunile speciale de lucru cu stiva modifică automat conținutul registrului SP.

De obicei, stivele cresc spre adrese mici, elementul din vârful stivei având adresa cea mai mică, dar există și stive care cresc spre adrese mari. Indicatorul de stivă poate conține adresa ultimului cuvânt ocupat din stivă (elementul din vârful stivei), sau adresa primului cuvânt liber din stivă.

Presupunând că stiva crește spre adrese mici, iar SP indică elementul din vârful stivei, instrucțiunea de introducere a unui element în stivă decrementează registrul SP și copiază elementul respectiv în stivă. Instrucțiunea de extragere a unui element din vârful stivei copiază acest element într-un registru sau un cuvânt de memorie și incrementează registrul SP.

În cazul stivei simulate în memorie, pe lângă operațiile obișnuite cu stiva, se pot citi sau modifica elemente oarecare din stivă, prin adresarea lor relativă față de registrul de bază al stivei (BP).

8.4.1.2. Stiva cablată

Spre deosebire de stiva simulată în memorie, la care vârful stivei se deplasează în timp ce informațiile din stivă rămân fixe, în cazul stivei cablate există un vârf fix și informațiile din stivă sunt translatate la fiecare operație de introducere sau eliminare din stivă (Figura 8.6).

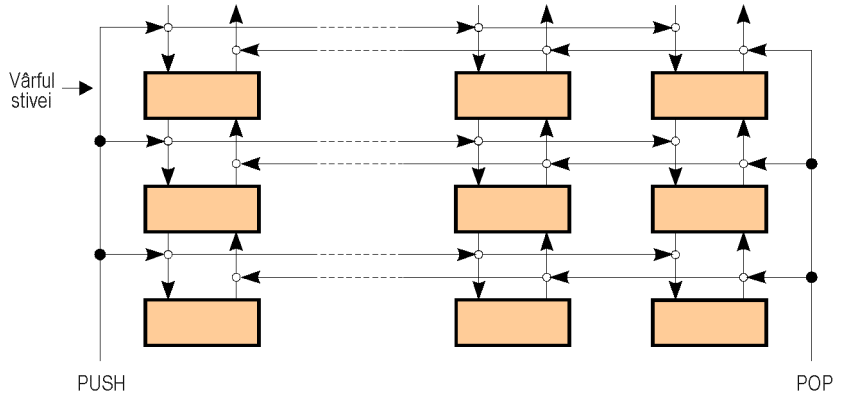

Figura 8.6. Stivă cablată.

Stivele cablate pot avea un *registru de stare* asociat, care indică dacă stiva este goală (*EMPTY*) sau plină (*FULL*). La încercarea de extragere a unui element din stiva goală se poate activa un semnal de eroare *UNDERFLOW*, iar la încercarea de introducere a unui element în stiva plină se poate activa un alt semnal, *OVERFLOW*.

Stivele cablate au avantajul unei viteze ridicate de efectuare a operațiilor. Dezavantajul lor este dimensiunea limitată.

8.4.1.3. Stiva parțial cablată

Dacă dimensiunea stivei cablate este redusă, este indicată completarea acesteia cu o stivă simulată în memoria internă, pentru cazurile de depășire de capacitate a stivei cablate (Figura 8.7).

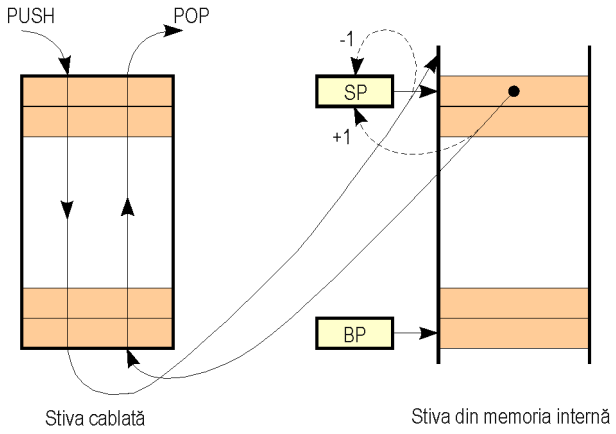

Figura 8.7. Stivă parțial cablată.

8.4.2. Calculatoare orientate pe stivă

Calculatoarele obișnuite sunt de tip *von Neumann*, fiind denumite și mașini cu registre. Acestea utilizează un limbaj mașină în care instrucțiunile adresează explicit sau implicit registrele UCP și în mod explicit locații de memorie, fiind posibile mai multe formate de instrucțiuni, cu 1 până la 3 sau 4 adrese.

Utilizarea limbajelor evoluate necesită calculatoare care simplifică compilarea și execuția programelor scrise în aceste limbaje. În acest sens, calculatoarele orientate pe stivă, care utilizează o stivă cablată în UCP, în locul registrelor, rezolvă în mod eficient o serie de probleme legate de compilarea și execuția programelor scrise în limbaje evoluate.

Aceste calculatoare utilizează instrucțiuni fără adresă, care pot specifica extragerea a doi operanzi din stivă, efectuarea unei operații asupra lor și depunerea rezultatului în stivă (Figura 8.8).

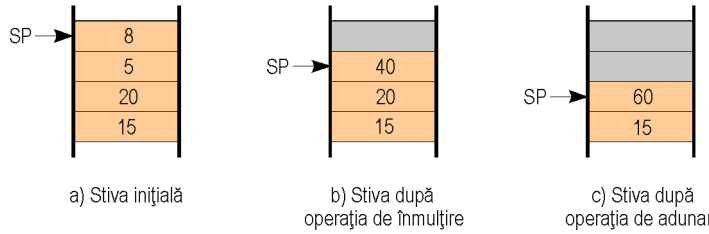

Figura 8.8. Utilizarea stivei pentru păstrarea operanzilor şi a rezultatului unor operaţii aritmetice.

Dintre problemele care se rezolvă în mod eficient de calculatoarele orientate pe stivă, se amintesc:

- Transformarea expresiilor aritmetice din forma *infixată* în forma *postfixată*;
- Evaluarea expresiilor aritmetice.

8.4.2.1. Notația poloneză pentru expresiile aritmetice

Forma expresiilor aritmetice în care un operator apare între operanzi corespunde *notației infixate*. De exemplu:

$A * (B + C)$

Notația poloneză este o tehnică de scriere a expresiilor aritmetice care a fost introdusă de logicianul polonez *J. Lukasiewitz* (1958). Există notația poloneză *prefixată* şi notația *postfixată*.

Fie E_1 şi E_2 două expresii aritmetice. Dacă E este o expresie aritmetică, $E = E_1$ o E_2, unde o este un operator aritmetic, atunci forma poloneză prefixată a expresiei E este o$E'_1E'_2$, unde E'_1 şi E'_2 sunt formele poloneze prefixate ale expresiilor E_1 şi E_2.

Forma poloneză postfixată a expresiei E este $E''_1E''_2$o, unde E''_1, E''_2 sunt formele poloneze postfixate ale expresiilor E_1 şi E_2. Forma poloneză prefixată şi postfixată a unui operand coincid cu operandul însuşi.

Forma poloneză postfixată, numită şi forma poloneză inversă, este cea mai utilizată. Ea are câteva avantaje față de notația infixată:

- Expresiile pot fi scrise fără paranteze;
- Regulile de precedență ale operatorilor infixați, reguli stabilite în mod arbitrar, nu mai sunt necesare.
- Evaluarea expresiilor se realizează mai simplu la calculatoarele care dispun de memorie stivă.

Se prezintă în Tabelul 8.2 câteva exemple de expresii infixate şi expresiile postfixate echivalente.

Tabelul 8.2. Exemple de expresii infixate şi postfixate.

Infix	Postfix
A ∗ (B + C)	A B C + ∗
A ∗ B + C	A B ∗ C +
A ∗ B + C ∗ D	A B ∗ C D ∗ +
(A + B) / (C - D)	A B + C D - /
((A + B) ∗ C + D) / (E + F)	A B + C ∗ D + E F + /

Ordinea operanzilor este aceeaşi la ambele notaţii. Transformarea expresiilor se reduce la amplasarea operatorilor în poziţiile corespunzătoare.

8.4.2.2. Evaluarea expresiilor postfixate

Pentru evaluare se poate utiliza următorul algoritm:

1. Se examinează fiecare simbol din expresia postfixată, începând de la stânga, până când se ajunge la un operator.
2. Se efectuează operaţia indicată asupra celor doi operanzi de la stânga operatorului, şi se memorează rezultatul.
3. Se şterg operanzii şi operatorul, în locul lor trecându-se rezultatul obţinut anterior.
4. Dacă expresia constă acum dintr-o singură valoare, aceasta reprezintă rezultatul şi algoritmul se termină. În caz contrar, se continuă cu pasul 1.

Exemplul 8.1

Expresia infixată: `(8 + 2 * 5) / (1 + 3 * 2 - 4)`
Expresia postfixată: `8 2 5 * + 1 3 2 * + 4 - /`

Etapele evaluării sunt următoarele:

```
8 2 5 * + 1 3 2 * + 4 - /
8 10 + 1 3 2 * + 4 - /
18 1 3 2 * + 4 - /
18 1 6 + 4 - /
18 7 4 - /
18 3 /
6
```

Se consideră o expresie postfixată formată din *n* simboluri, fiecare din ele reprezentând o constantă sau un operator. Pentru evaluarea acestei expresii prin utilizarea unei stive, se poate utiliza algoritmul următor.

1. Se inițializează o variabilă *k* cu 1.
2. Se atribuie unei variabile *s* valoarea simbolului cu numărul de ordine *k* din expresia postfixată (de exemplu, dacă *k* = 1, *s* ia valoarea primului simbol al expresiei).
3. Dacă *s* este o constantă, se memorează în stivă şi se trece la pasul 5.
4. Dacă *s* este un operator, se extrag cele două elemente din vârful stivei, şi se efectuează asupra lor operația indicată de *s*, astfel încât elementul din vârful stivei reprezintă operandul din dreapta operatorului. Rezultatul se memorează în stivă.
5. Dacă *k* = *n* (s-a ajuns la ultimul simbol al expresiei), operația se termină, rezultatul fiind în vârful stivei. În caz contrar, se incrementează *k* şi se continuă de la pasul 2.

Exemplul 8.2

Expresia infixată: `(8 + 2 * 5) / (1 + 3 * 2 - 4)`
Expresia postfixată: `8 2 5 * + 1 3 2 * + 4 - /`

Se prezintă în Figura 8.9 stiva pentru fiecare pas al operației.

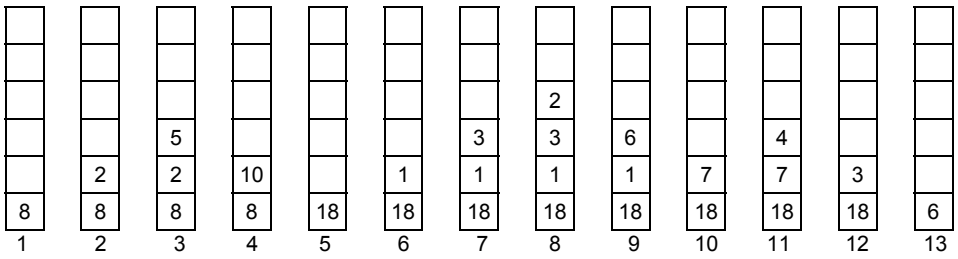

Figura 8.9. Conținutul stivei la evaluarea unei expresii postfixate.

8.5. Memoria cache

8.5.1. Principiul memoriei cache

Viteza UCP este superioară vitezei memoriilor, astfel că după inițierea unui ciclu de acces la memorie, UCP trebuie să rămână inactivă un timp, aşteptând răspunsul acesteia.