INTRODUCERE IN LIMBAJUL DE PROGRAMARE HARDWARE AHPL.

(**AHPL - A Hardware Programming Language**).

In functie de scopul urmarit, un calculator numeric poate fi descris folosind diverse mijloace. Astfel, la nivelul etapei de proiectare, se folosesc adesea diagrame bloc, organigrame, limbaje specializate, tabele de cablaj etc.

In cadrul acestei lucrari calculatorul numeric este descris in termenii unor secvente de transferuri ale datelor intre diferitele primitive functionale ale calculatorului, in vederea implementarii unui algoritm dat. Se considera ca, dupa terminarea acestei etape de proiectare, elaborarea schemelor logice si a schemelor de cablaj reprezinta operatii mecanice, de rutina, care se pot automatiza cu usurinta.

Sistemele numerice de calcul prelucreaza in principal date structurate sub forma de vectori binari, stocati in registre sau memorii organizate sub forma de matrici, cu acces la linii.

In acest context apare evidenta utilitatea unui limbaj prevazut cu facilitati pentru manipularea vectorilor si matricilor.

Un asemenea limbaj a fost propus de K.E. Iverson sub numele APL evoluand intr-o noua ipostaza ca limbajul J.

(A Programming Language). Limbajul APL cunoaste o larga raspindire, ca limbaj conversational in sistemele cu multiacces IBM 360/370 si chiar in cadrul calculatoarelor personale IBM-PC compatibile. Trebuie mentionata, de asemenea, si implementarea realizata in tara noastra pe sistemele FELIX C-256/512.

APL este un limbaj foarte puternic. El permite descrierea concisa a algoritmilor si verificarea lor prin executie, pe calculator. Pentru scopurile urmarite de proiectare, limbajul APL trebuie completat cu instructiuni si alte constructii, care permit o implementare directa in hardware.

Un asemenea limbaj AHPL (A Hardware Programming Language) a fost propus de catre F. Hill si G. Peterson in 1974, fiind apoi perfectionat pe parcurs (1978, 1983). AHPL, ca limbaj de descriere a transferurilor intre registre, asigura o corespondenta directa intre notatia folosita si implementarea ei in hardware.

AHPL foloseste, pe linga constructiile proprii, si un subset de constructii din APL, care isi gasesc un corespondent in hardware. Frecvent, in scopul facilitarii comunicarii si documenarii,

in programele AHPL se mai utilizeaza unele notatii APL, care descriu algoritmul de operare al unui calculator numeric.

In momentul traducerii in hardware a descrierii AHPL a modulului numeric dat, aceste notatii nu sunt luate in consideratie.

Autorii limbajului AHPL au elaborat si un compliator pentru hardware, care are ca intrare descrierea AHPL a sistemului numeric, iar ca iesire - schema sistemului, la nivel de diagrame.

Indiferent de implementarea unui algoritm, este o buna practica aceea de a descrie mai intai algoritmul in APL si de a-l executa pe un calculator, prevazut cu un compliator de APL.

In continuare algoritmul descris in APL poate fi implementat in hardware, firmware (microprogram) sau poate fi executat direct, fiind implementat in software.

Conventii privind operanzii folositi in AHPL.

Operanzii manipulati in limbajul AHPL reprezinta marimi constante sau variabile.

Constantele sunt date numerice sau alfanumerice. Ele se reprezinta prin numere standard, respectiv - prin litere standard sau numere cuprinse intre semnele apostrof.

Variabilele pot fi scalari, vectori, matrici.

Este important de facut distinctie intre variabile si valori.In limbajele conventionale de programare o variabila reprezinta un nume prin care se face o referire la un operand; o valoare reprezinta marimea pe care o ia efectiv operandul. In AHPL, prin variabila se intelege numele unui registru al carui continut este manipulat printr-o instructiune a programului, iar prin valoare - data care se plaseaza in registru.

Tipurile de operanzi in AHPL se vor deosebi prin conventii tipografice: litere mici pentru scalari, majuscule pentru vectori si majuscule ingrosate pentru matrici.

Transferul unei constante (binare - in cazul de fata) intr-un registru AC se noteaza astfel:

$$AC \longleftarrow 0,1,1,1,1,0,1,0$$

Transferul continutului unui registru RD, intr-un registru AC, fara a se modifica continutul lui RD se reprezinta prin instructiunea:

$$AC \leftarrow RD$$

Un vector constituie o colectie de operanzi avand o structura unidimensionala. Numarul componentelor vectorului reprezinta dimensiunea vectorului. Pentru a specifica dimensiunea unui vector oarecare (AC) se va folosi notatia ρAC, unde ρ reprezinta operatorul “dimensiune”.

Daca AC are 16 biti (ranguri binare), atunci ρAC = 16. In aceasta lucrare, in cazul in care nu se vor face mentiuni speciale, pozitiile bitilor individuali vor fi specificate prin indici cu originea 0, plasata in extrema stanga: AC_0 ,AC_1 ,.....,$AC_{\rho AC-1}$

Trebuie observat ca aceasta notatie este destul de greoaie in textele dactilografiate, ceea ce face ca, in multe cazuri, indicii sa fie plasati pe aceeasi linie cu numele registrului, de exemplu: ACi - bitul de rang i din registrul AC. Pentru a specifica un grup de biti adiacenti, dintr-un registru se foloseste notatia: $A_{i:j}$ sau A(i:j), unde sunt inclusi si bitii i si j.

Operanzii de tip matricial se reprezinta sub forma unui tablou bidimensional, constituit din elemente cu indici inferiori si superiori. Indicii inferiori specifica coloanele, iar cei superiori liniile. Se considera matricea **M** avind ρ1**M** coloane si ρ2**M** linii, unde ρ1 si ρ2 reprezinta notatiile pentru operatorii care aplicati lui **M** furnizeaza numarul de coloane si numarul de linii ale tabloului **M**.

$\mathbf{M}^0{}_0$ $\mathbf{M}^0{}_1$ ……...…….....$\mathbf{M}^0{}_{\rho 1\mathbf{M}-1}$
.....……..…….….
.....…… $\mathbf{M}^i{}_j$.....…….
$\mathbf{M}^{\rho 2M-1}{}_0$ $\mathbf{M}^{\rho 2M-1}{}_1$.....…… $\mathbf{M}^{\rho 2M-1}{}_{\rho 1M-1}$

Linia i a matricii **M** se specifica prin notatia $\mathbf{M}^i$ sau **M<i>,** iar coloana j - prin notatia $\mathbf{M}_j$ sau **M**[j]. Liniile succesive i...k, ale matricii **M** se noteaza prin $M^{i:k}$ sau **M**<i:k>, iar coloanele succesive j...l - prin $\mathbf{M}_{j:l}$ sau **M**[j:l].

Conventii privind operatorii APL si AHPL.

Operatorii folositi sunt *operatorii primitivi* si *operatorii de tip mixt*.

Operatorii primitivi manipuleaza, de regula, operanzi de tip scalar, desi ei pot fi extinsi atat la vectori, cat si matrici. Acesti opeatori se pot referi la o singura variabila (operatori unari) sau la doua variabile (operatori binari). In cele ce urmeaza se prezinta operatorii primitivi aritmetici, logici si relationali. In scopul facilitarii comunicarii, in textele ce reprezinta programe AHPL pot fi intalniti toti acesti operatori. Implementari directe in hardware au insa numai operatorii logici. Operatorii aritmetici si relationali sunt specifici limbajului APL.

Operatorii aritmetici manipuleaza numere reale:

x + y adunare; suma algebrica, (APL),

x - y scadere; diferenta algebrica, (APL),

$x * y$ inmultire; inmultire algebrica, (APL),

x % y impartire algebrica; (APL).

|x valoare absoluta; (APL).

Operatorii logici manipuleaza numere binare (vectori binari):

$\overline{x}$ NU (APL si AHPL),

$x \cap y$ SI (APL si AHPL),

$x \cup y$ SAU (APL si AHPL),

$x \oplus y$ SAU-Exclusiv (APL si AHPL).

Operatorii relationali sunt de forma:

$(x \Re y)$ unde $\Re \in \{ <, \leq, =, \geq, > \}$ (APL).

In versiunile existente de APL, incercarea de a folosi alte tipuri de variabile, cu exceptia celor logice, conduce la eroare de domeniu.

In raport cu operatorii aritmetici si relationali nu exista limitari privind folosirea variabilelor.

In exemplele urmatoare se considera:

x = 1, y = -3, W = (4,-5,0,2) si U = (1,2,1,-1).

Instructiune:	**Rezultat:**
$z \leftarrow x + y$	z = -2
$z \leftarrow \overline{x}$	z = 0
$Z \leftarrow W + U$	Z = (5,-3, 1,1)
$Z \leftarrow W * U$	Z = (4,-10,0,-2)
$Z \leftarrow W \% U$	Z = (4,-2.5,0,-2)
$z \leftarrow (x < y)$	z = 0
$z \leftarrow (x > y)$	z = 1

Operatorii de tip mixt sunt extrem de puternici, deoarece permit manipularea unor combinatii de scalari, vectori si matrici. In cele ce urmeaza se vor prezenta *operatorii de tip mixt*, cu specificarea limbajelor in care se utilizeaza.

Notatie	**Denumire**	**Semnificatie**	**Observatii**
X,Y	Inlantuire/ Concatenare	X_0 ,...,$X_{\rho X-1}$,Y_0 ,....,$Y_{\rho Y-1}$	APL si AHPL
M!N	Inlantuire linii.	O matrice cu ρ2**M**+ρ2**N** linii cu liniile lui M peste liniile lui N.	AHPL
k↑X	Extragere	Se extrag primele k elemente din vextorul X.	APL
k↓X	Eliminare	Se elimina primele k elemente din vectorul X.	APL
⊥X	Decodificare	Echivalentul zecimal al vectorului binar X	APL
n⊤p	Codificare binara.	Un vector cu n elemente binare, obtinut prin reprezentarea numarului zecimal p in baza doi.	APL si AHPL
@/X	Reducere	X_0 @X_1 @X_2 @.....@$X_{\rho X-1}$	APL si AHPL
X ← @/**M**	Reducere linie	$X_i \leftarrow @/M^i$	AHPL si APL
X ←@//**M**	Reducere coloana	$X_j \leftarrow @/M_j$	AHPL si APL
X ←U/Y	Comprimare	Vectorul X este obtinut din vectorul Y prin su-primarea rangurilor X_i pentu care U_i =0 U este vector binar.	APL si AHPL in descrierea unitatilor logice combinationale
A ← U/**M**	Comprimare linii	**A** ← U^i/$\mathbf{M}^i$	Idem linii.
A ← U//**M**	Comprimare coloane	$\mathbf{A}_j$← U/$\mathbf{M}_j$	Idem coloane

Nota1: @ este un operator logic sau aritmetic.
Nota2: **A** si **M** sunt matrici, iar U este un vector binar.

In exemplele care urmeaza se ilustreaza utilizarea operatorilor de tip mixt.

Inlantuirea.

Fie: X = (1,2,3,4) si Y = (5,6,7),

daca $Z \leftarrow X,Y$ atunci, rezulta: Z = (1,2,3,4,5,6,7)

Inlantuirea permite descrierea operatiilor de deplasare si rotire ale vectorilor si matricilor.

In cele ce urmeaza se vor prezenta notatiile AHPL pentru deplasari si rotiri de vectori.

Fie vectorul $A = (A_0 ,A_1 ,A_2 ,.....,A_{\rho A-1})$

Deplasare logica - dreapta (cu inserta unui zero in bitul 0).

$A \leftarrow 0, A_0 ,A_1 ,A_2 ,.....,A_{\rho A-2}$

Deplasare logica - stanga (cu insertia unui zero in bitul de rang ρA-1).

$A \leftarrow A_1 ,A_2 ,.....,A_{\rho A-1},0$

Deplasare aritmetica dreapta (cu extinderea bitului de rang 0 in bitul de rang 1).

$A \leftarrow A_0, A_0 ,A_1 ,A_2 ,.....,A_{\rho A-2}$

Deplasarea aritmetica stanga este identica cu deplasarea logica stanga.

Rotire -dreapta.

$A \leftarrow A_{\rho A-1}, A_0 ,A_1 ,A_2 ,.....,A_{\rho A-2}$

Rotire - stanga.

$A \leftarrow A_0 ,A_1 ,A_2 ,.....,A_{\rho A-1,}\ 0$

In operatiile de deplasare si rotire, inaintea bitului de rang 0 si dupa bitul de rang ρA-1, se pot plasa biti individuali reprezentand indicatori de conditii (de exemplu, din unitatea de executie, bitul de transport). Cele mai multe microprocesoare dispun de instructiuni care implementeaza asemenea operatii de deplasare. Ele sunt folosite pentru instructiunile de transfer conditionat al comenzii.

In continuare sint ilustrate operatiile de rotire/deplasare circulara a liniilor unei matrici.

Fie: $\mathbf{M} = \begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{vmatrix}$

Deplasare circulara in sus.

Daca $\mathbf{N} \leftarrow \mathbf{M}^{1:2}\ !\ \mathbf{M}^{0}$, atunci rezulta: $\mathbf{N} = \begin{vmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{vmatrix}$

Deplasare circulara in jos.

Daca $\mathbf{N} \leftarrow \mathbf{M}^2 \,!\, \mathbf{M}^{0:1}$, atunci rezulta: $\mathbf{N} = \begin{vmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix}$

Extragerea.

Fie: X = (1,2,3,4,5,6)

daca $Y \leftarrow 4\uparrow X$, atunci rezulta: Y = (1,2,3,4) in APL

Eliminarea.

daca, $Y \leftarrow 4\downarrow X$ atunci rezulta: Y = (4,5,6) in APL.

In AHPL notatiile echivalente vor fi urmatoarele:

$Y \leftarrow X_{0:3}$ (extragere) si respectiv $Y \leftarrow X_{4:6}$ (eliminare)

Codificare binara.

Fie: x = 13 si y = 17

daca: $X \leftarrow \top 13$ si $Y \leftarrow \top 17$, atunci: X = (1,1,0,1) si Y = (1,0,0,0,1)

Decodificarea binara.

Fie: X = (1,1,0,1) si Y = (1,0,0,0,1)

daca: $x \leftarrow 4\perp X$ si $y \leftarrow 5\perp Y$, atunci: x = 13 si y = 17.

Reducere.

Fie @ un operator binar care se aplica unui vector binar X,rezultatul operatiei va fi de forma:

$x \leftarrow (....((X_0@X_1)@X_2)@.....X_{\rho X-1})$

Expresia:

$x \leftarrow +/X$ este echivalenta cu :

$$x \leftarrow \sum_{i=0}^{\rho X-1} X_i$$

Daca X este un vector logic (cu componente binare), atunci urmatoarele operatii vor furniza informatii privind:

$x \leftarrow +/X$, numarul de unitati din vectorul X;

$x \leftarrow \cup/X$, prezenta a cel putin unei unitati diferita de zero, daca $x <> 0$;

x ← ∩/X, prezenta tuturor componentelor egale cu unu, daca x = 1.

Fie matricea **M**:

$$\mathbf{M} \leftarrow \begin{vmatrix} 1\,0\,0\,1 \\ 0\,1\,0\,0 \\ 1\,0\,1\,1 \end{vmatrix}$$ atunci: +/**M** = (2,1,3); +//**M** = (2,1,1,2,) si +/(+//(**M**)) = 6.

Comprimare.

Fie U = (1,0,0,1) si A = (1,2,3,4), atunci: X ← U/A, conduce la rezultatul: X = (1,4)

Considerind matricea **M** si vectorul U de mai sus, comprimarea pe linii **N** ← U/**M** va furniza matricea **N**, de forma:

$$\mathbf{N} \leftarrow \begin{vmatrix} 1\,1 \\ 0\,0 \\ 1\,1 \end{vmatrix}$$

Comprimarea coloanelor va fi exemplificata cu un vector logic V, de forma: V = (1,1,0).

N ←V//**M** va avea aspectul urmator:

$$\mathbf{N} \leftarrow \begin{vmatrix} 1\,0\,0\,1 \\ \\ 0\,1\,0\,0 \end{vmatrix}$$

Ca un ultim exemplu, penru a ilustra forta limbajului AHPL se va scrie un program pentru cautarea intr-o lista de cuvinte plasate intr-o memorie, in vederea extragerii acelor cuvinte pentru care primii 4 biti sint egali cu 1.

Se considera ca lista de cuvinte binare, de cite 16 biti, se afla intr-o memorie **M**, cu un numar de ρ2**M** cuvinte. Se va forma un vector U cu ρ2TM componente, care vor fi 0 pentru acele cuvinte, care nu indeplinesc conditia impusa si 1, in caz contrar. In final se va genera o matrice **N** obtinuta prin comprimarea coloanelor lui **M** dupa vectorul U.

1. U ← ρ2**M**T0 /* se genereaza valoarea initiala a vectorului U
2. i ← 0 /* se initializeaza contorul i
3. → ((∩/($\mathbf{M}^{i}_{0} \cap \mathbf{M}^{i}_{1} \cap \mathbf{M}^{i}_{2} \cap \mathbf{M}^{i}_{3}$)) = 0)/(5) /* se testeaza conditia
4. U_i ←1 /* se atribuie valoarea 1 componentei care nu indepplineste conditia impus
5. i ←i + 1 /* se incrementeaza contorul i

6. $\rightarrow$ (i < ρ2**M**)/(3) /* transfer conditionat al comenzii

7. **N** $\leftarrow$U//**M**

Se poate observa ca vectorul U se pute genera simplu prin opeatia:

$U \leftarrow (\mathbf{M}_0 \cap \mathbf{M}_1 \cap \mathbf{M}_2 \cap \mathbf{M}_3)$

Deci, tot programul secvential de mai sus se poate inlocui cuexpresia:

$\mathbf{N} \leftarrow (\mathbf{M}_0 \cap \mathbf{M}_1 \cap \mathbf{M}_2 \cap \mathbf{M}_3)//\mathbf{M}$

Conventii AHPL pentru descrierea logicii combinationale.

S-a aratat ca procesul de prelucrare a datelor consta in trans-ferul acestora intre registrele sursa si registrele destinatie prin intermediul unor retele logice combinationale. Acestea din urma reprezinta resurse hardware, care pot fi folosite in mod repetat, cu diverse argumente, in puncte diferite ale secventei AHPL, care descrie algoritmul.

Se considera o retea logica combinationala, care realizeaza operatia NOR (SAU-NU) asupra rangurilor a doi vectori de 4 biti.

$C \leftarrow (A_0 \cap B_0 , A_1 \cap B_1 , A_2 \cap B_2 , A_3 \cap B_3)$

Daca expresia de mai sus apare in mod repetat in secventa AHPL, se poate folosi o notatie prescurtata:

C $\leftarrow$NOR(A;B), care este asemanatoare unei notatii de subrutina intr-un limbaj de programare. In contextul de fata NOR(A;B) va fi mentionata ca o functie sau o unitate logica combinationala.

Intr-un program AHPL, ea va fi descrisa o singura data, dupa care poate fi apelata in mod repetat, cu diferite argumente.

Sub forma generala descriera unei asemenea functii fi delimitata de titlu si de sfirsit (END).

```
UNIT:NOR(A;B)
    conexiuni
    .........
    conexiuni
END
```

Dupa titlu pot urma declaratii privind vectorii manipulati, lungimea lor etc. Conexiunile sunt instructiuni care se traduc direct in hardware sub forma unor legaturi intre intrarile si iesirile unor circuite logice combinationale.

Intr-un alt paragraf se vor da mai multe exemple privind descrierea unor unitati logice combinationale: BUSFN, DCD, ADD etc.