CURS 2

2. Tipuri de comenzi AHPL

Instrucțiuni de atribuire

Formatul unei instrucțiuni de atribuire este:

<destinație> <operator_de_atribuire> <expresie>

Expresiile pot fi formate utilizând:

- operatori logici: ($\wedge$;$\vee$; $^-$);
- operatori de sincronizare: (SL ; SYN (semnal_asincron));
- operatori de selecție: (A_j; $A_{m:n}$; $\mathbf{M}^j$; $\mathbf{M}^{m:n}$);
- operator de selecție prin comprimare: (X / Y);
- operator de selecție prin comprimare şi reducere: (**M** * F);
- operatori de concatenare: (A,B ; A!B);
- operator de codificare: (n T p);
- operatori de decodificare: ($\perp^X$; DCD);
- operatori relaționali.

Instrucțiuni de conexiune

Dacă se notează cu:

MAG - o magistrală predeclarată
Z - un vector de ieşire;
VLCO - un vector de funcții logice combinaționale de ieşire;
MLCO - o matrice de vectori logici combinaționali VLCO;
F - un vector de constante binare sau funcții logice

instrucțiunile de conexiune pot fi specificate prin una din următoarele forme:

Z = VLCO
MAG = VLCO
Z = **MLCO***F
MAG = **MLCO***F

Instrucțiuni de transfer

Variabila destinație este un bistabil sau un vector ce are ca echivalent un registru sau un cuvânt de memorie. Înscrierea valorii expresiei în operandul destinație se face sincronizat cu un semnal de tact. Acest semnal de tact poate fi specificat explicit printr-o conexiune sau implicit ca fiind tactul unității de comandă.

Forma generală a unei instrucțiuni de transfer este:

VD $\leftarrow$VLCO
VD $\leftarrow$ **MLCO***F
MD*F$\leftarrow$VLCO

unde: VD este un vector destinație iar
MD este o matrice destinație

Prima instrucțiune specifică un transfer necondiționat.

Cea de a doua formă poate fi descompusă în următoarele operații elementare: conexiune la intrările vectorului destinație a vectorului rezultat prin evaluarea expresiei şi apoi înscrierea acestuia în VD.

Cea de a treia formă este echivalentă cu: valoarea lui VLCO se transferă în toate liniile matricei rezultate prin comprimarea şi reducerea cu vectorul de selecţie F.

9. Descrierea functiilor in AHPL

Pentru a facilita scrierea secvenţelor de descriere a circuitelor logice combinaţionale oferind şi o flexibilitate de utilizare a acestora în diferite module, limbajul AHPL permite descrierea separată ca module de program distincte sub forma:

UNIT: nume_funcţie <lista de parametri>
<declaraţii>
<conexiuni>
END

Dacă o schemă combinaţională apare (este apelată) de mai multe ori cu aceeaşi parametri efectivi ea va fi generată fizic o singură dată cu condiţia să nu se depăşească fan-out-ul ieşirilor.

Dacă însă este apelată de mai multe ori, cu parametri diferiţi se poate proceda în două moduri:

- se implementează câte o copie a schemei pentru fiecare apelare;
- se implementează schema o singură dată dar se prevede schema de selecţie a parametrilor de intrare corespunzători fiecărei apelări.

Alegerea uneia dintre modalităţi se face prin analiza complexităţii funcţiei combinaţionale şi a logicii de selecţie a parametrilor.

Apelul unei unităţi logice combinaţionale se face prin:
nume_funcţie(lista_de_parametri_efectivi).

Deci în expresii AHPL pot să apară nume de unităţi logice combinaţionale definite.

20. Operatori de selectie si de reducere in AHPL

- **operatori de reducere:**

$\wedge/$; $\vee/$; +/ realizează funcţia logică specificată între componentele vectorului asupra căruia acţionează. Rezultatul este un scalar.

Exemple:

$$\wedge/A = A_0 \wedge A_1 \wedge A_2 \wedge ... \wedge A_n$$

- **operator de selecţie prin comprimare:**

X/Y elimină componentele din Y corespunzătoare componentelor egale cu 0 din X

Exemplu:

Fie A = (0, 1, 1, 0, 1, 0, 0, 1)
B = (B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7)
A / B = (B_1, B_2, B_4, B_7)

- **operator de selecţie prin comprimare şi reducere**:

M*F realizează eliminarea liniilor din matricea M corespunzătoare componentelor egale cu 0 din vectorul F

Exemplu:

Fie

$$M = \begin{bmatrix} M^0 \\ M^1 \\ M^2 \end{bmatrix}$$ şi F=[0,0,1]

rezultă:

M*F selectează linia 2, adică $\mathbf{M}^2$

3. Descrierea sumatorului elementar complet

Sumatorul elementar complet realizează operația de adunare a unor operanzi de lungime un bit, ținând seama de transportul de intrare şi generează pe lângă sumă şi eventualul transport.

În Fig.2.6 se prezintă schema bloc a unui sumator elementar complet. Intrările sunt cei doi operanzi şi transportul din exterior iar ieşirile sunt suma şi transportul spre exterior.

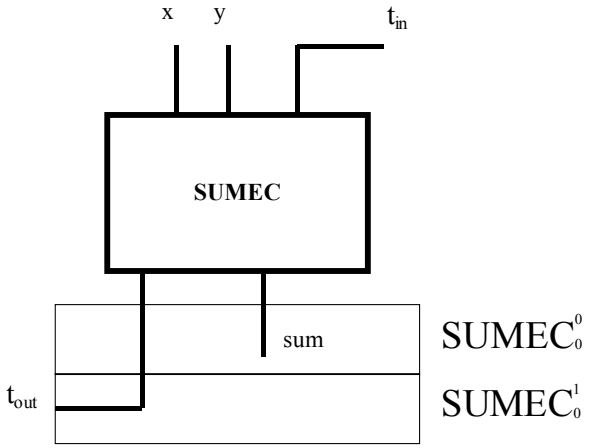


Fig. 2.6. Sumator elementar complet

Tabela de adevăr, diagramele Karnaugh şi ecuațiile logice sunt prezentate în continuare. Ecuațiile logice rezultate sunt :

$sum = x \oplus y \oplus t_{in}$

$t_{out} = ((x \oplus y) \wedge t_{in}) \vee (x \wedge y)$

Aceste expresii logice pot fi descrise în AHPL astfel:

```
UNIT: SUMEC  (x;y;tin)
        INPUTS: (x;y;tin)
        OUTPUTS: SUMEC [2;1]
        1. a = x ⊕ y
        2. b = x ∧ y
        3. sum = a ⊕ tin
        4. c = a ∧ tin
        5. tout = b ∨ c
        6. SUMEC0^0 = sum
        7. SUMEC0^1 = tout
```

Tabela de adevăr:

x	y	t_{in}	suma	t_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

END

Implementarea unității logice combinaționale a sumatorului elementar complet este prezentată în Fig. 2.7.

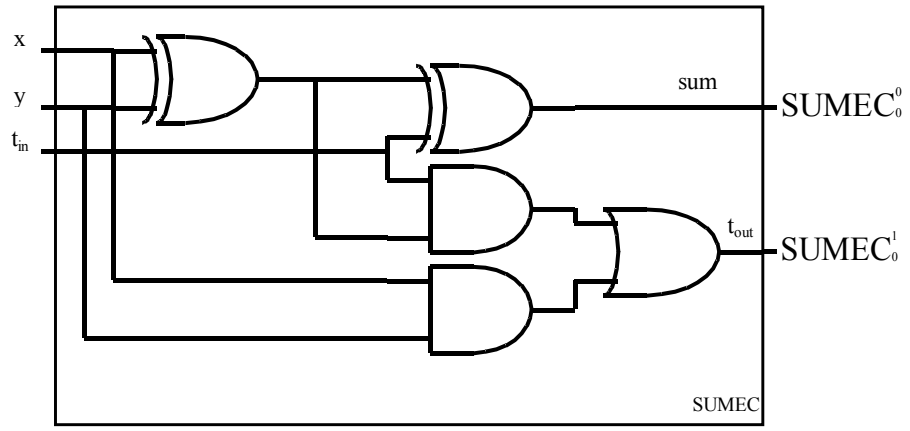


Fig. 2.7. Schema sumator elementar complet

CURS 3

1.Reprezentarea numerelor in virgula fixa

În virgulă fixă numerele se pot reprezenta în funcție de poziția virgulei în trei moduri:

-numere întregi: poziția virgulei fiind interpretată după cifra cea mai putin semnificativă x_0

$x = x_n x_{n-1} \dots x_1 x_0,$

iar valoarea numărului $V_x = \sum_{i=0}^{n} x_i * 2^i$

-numere subunitare: poziția virgulei fiind interpretată înainte de cifra cea mai semnificativă x_{-1}

$x = ,x_{-1}x_{-2} \dots x_{-n+1}x_{-n}$

iar valoarea numărului $V_x = \sum_{i=0}^{n} x_{-i} * 2^{-i}$

-numere reale: poziția virgulei fiind interpretată în cadrul reprezentării. Există un grup de biți care specifică partea întreagă şi un număr de biți care specifică partea fracționară. Această reprezentare combină cele două reprezentări anterioare. În practică se utilizează numai în structuri numerice dedicate, în care se conoaşte foarte bine domeniul de valori al datelor.

$x = x_k x_{k-1} \dots x_1 x_0, x_{-1}x_{-2} \dots x_{-m}$

iar valoarea numărului $V_x = \sum_{i=0}^{k} x_i * 2^i + \sum_{i=0}^{m} x_{-i} * 2^{-i}$

În marea majoritate a calculatoarelor se utilizează reprezentarea în virgulă fixă numere întregi.

10. Adunarea in cod complementar (c4)

Fie :

$x = x_s x_{-1} x_{-2} \dots x_{-n}$

$y = y_s y_{-1} y_{-2} \dots y_{-n}$ cu condiția ca $|x| + |y| < 1$

două numere în virgulă fixă, subunitare, reprezentate în cod complementar.

Operația de adunare constă în:

- se adună bit cu bit începând cu rangul cel mai putin semnificativ, inclusiv bitul de semn.
- Eventualul transport din bitul de semn se neglijează.

Justificare:

1. în cazul în care x>0, y>0 şi |x|+|y|<1

$[x]_c + [y]_c = |x|+|y| = [(|x|+|y|)]_c$

2. în cazul în care x>0, y<0

cazul 1: |x|>|y|

$[x]_c + [y]_c = |x|+2-|y| = 2 + (|x|-|y|)$

deoarece |x|-|y|>0,

atunci 2+(|x|-|y|) generează un transport din bitul de semn, care conform regulii enunțate se neglijează.

În acest caz:

$[x]_c + [y]_c = (|x|-|y|) = [(|x|-|y|)]_c$

cazul 2: |x|<|y|

$[x]_c + [y]_c = |x|+2-|y| = 2 - (|y|-|x|)$

deoarece |y|-|x|>0,

atunci 2-(|y|-|x|) nu generează un transport din bitul de semn.

În acest caz:

$[x]_c + [y]_c = 2 - (|y|-|x|) = [-(|y|-|x|)]_c$

3. în cazul în care x<0, y<0 şi |x|+|y|<1

$[x]_c + [y]_c = 2 - |x| + 2 - |y| =$

$= 2 + 2 - (|x|+|y|) =$

$= 2 - (|x|+|y|) = [-(|x|+|y|)]_c$

Deoarece (|x|+|y|)>0,

atunci 2 - (|x|+|y|) nu generează un transport din bitul de semn.

În schimb 2 + 2 - (|x|+|y|) generează un transport care se neglijează.

În acest caz:

$[x]_c + [y]_c = [-(|x|+|y|)]_c$

11. Etapele inmultirii prin metoda lui Booth (c4)

Pentru a realiza operația de înmulțire a unor numere reprezentate în complement față de 2, prin adunări repetate, este necesar să evaluăm înmulțitorul. Evaluarea înmulțitorului reprezentat în complement față de 2 se realizează utilizând formula lui Booth.

Dacă

$y = y_s y_{-1} y_{-2} ... y_{-n}$ este un număr în complement față de 2 valoarea reprezentată de acest număr este:

$$V_y = -y_s + \sum_{i=1}^{n} y_{-i} * 2^{i}$$ formula lui Booth

Justificare:

a) dacă y este pozitiv $y = 0y_{-1}y_{-2}...y_{-n}$ are valoarea

$$V_y = 0 + \sum_{i=1}^{n} y_{-i} * 2^{-i}$$

ceea ce reprezintă într-adevăr evaluarea unui număr pozitiv reprezentat în cod direct.

b) dacă y este negativ, $y = 1y_{-1}y_{-2}...y_{-n}$, valoarea sa este obținută evaluând $[-y]_c$.

Dar ştim că $[y]_c = [y]_i + 2^{-n}$, deci

$$V_{[-y]_c} = \sum_{i=1}^{n} 2^{-i} y_{-i} - \left(\sum_{i=1}^{n} y_{-i} * 2^{-i} \right) + 2^{-n}$$

unde $y_{-i} = 1 - y_{-i}$

$$V_{[-y]_c} = \left(\sum_{i=1}^{n} y_{-i} * 2^{-i} \right) + 2^{-n}$$

$$V_{[-y]_c} = \left(\sum_{i=1}^{n} \left(1 - y_{-i} * 2^{-i}\right) \right) + 2^{-n}$$

$$V_{[-y]c} = 1 - \sum_{i=1}^{n} y_{-i} * 2^{-i}$$

Dar $V_v = -V_{[-y]c}$ deci $V_{[-y]c} = -1 - \sum_{i=1}^{n} y_{-i} * 2^{-i}$ ceea ce justifică formula lui Bc[illegible]

Pentru a realiza z = x * y este acelaşi lucru cu a realiza z = x * V_y

Booth a propus ca o cifrele de reprezentare ale înmulțitorului, în cod complementar, să se înlocuiască cu diferența a două cifre adiacente:

$y_{-j} \leftarrow y_{-j-1} - y_{-j}$ $1 \le j \le n$

Înlocuirea propusă de Booth nu se realizează efectiv, ci conduce numai la analiza a doi biți adiacenți din reprezentarea înmulțitorului (nu a unui singur bit cum se realiza la înmulțirea directă). Astfel dacă

y_{-j}	Y_{-j-1}	
0	0	se deplasează produsul parțial cu o poziție la dreapta;
0	1	se adună deînmulțitul la produsul parțial şi apoi se deplasează rezultatul cu o poziție la dreapta;
1	0	se scade deînmulțitul din produsul parțial şi apoi se deplasează rezultatul cu o poziție la dreapta;
1	1	se deplasează produsul parțial cu o poziție la dreapta;

După ultima operație, în care participă şi bitul de semn, nu se mai efectuează operația de deplasare.

19. Reprezentarea numerelor cu semn in virgula mobila

În virgulă mobilă, numerele se reprezintă prin două componente:

- mantisă;
- exponent.

Un grup de biți sunt utilizați pentru reprezentarea mantisei, în virgulă fixă numere subunitare, iar alt grup de biți sunt utilizați pentru reprezentarea exponentului în virgulă fixă numere întregi.

$x = E,M$

$x = x_k x_{k-1} \ldots x_1 x_0, x_{-1} x_{-2} \ldots x_{-m}$

unde:

$E = x_k x_{k-1} \ldots x_1 x_0,$

$M = ,x_{-1} x_{-2} \ldots x_{-m}$

iar valoarea numărului $V_x = V_M * b^{VE}$

unde b constituie baza de reprezentare.

În general baza b este 2, însă sunt calculatoare care utilizează ca bază b = 8, sau b = 16.

În reprezentarea în virgulă mobilă interpretarea poziției virgulei se face în funcție de valoarea exponentului.

În calculatoarele numerice semnul ocupă o poziție în reprezentare. Se consideră convenția ca bitul de semn să aibă următoarea semnificație:

$x_s = 0$ semnul este pozitiv

$x_s = 1$ semnul este negativ

iar poziția sa este în general în partea din stânga a reprezentării.

$x = x_s x_n x_{n-1} \ldots x_1 x_0,$ în virgulă fixă, numere întregi

$x = x_s, x_{-1} x_{-2} \ldots x_{-n+1} x_{-n}$ în virgulă fixă, numere subunitare

4. Etapele principale ale operatiei de inmultire in cod direct (c4)

Metoda de înmulțire directă este utilizată în cazul reprezentării în cod direct (mărime şi semn).

Fie:

$x = x_s x_{-1} x_{-2} \ldots x_{-n}$

$y = y_s y_{-1} y_{-2} \ldots y_{-n}$

două numere în virgulă fixă, subunitare, reprezentate în cod direct.

Pentru a realiza înmulțirea **z = x * y** este necesar ca rezultatul să fie reprezentat pe un număr dublu de biți sau să se realizeze o rotunjire a rezultatului, în cazul în care se reprezintă pe acelaşi număr de biți.

Operația de înmulțire constă din trei etape principale:

a) - determinarea semnului;

b)- înmulțirea efectivă;

c)- rotunjirea rezultatului;

a) **Determinarea semnului**

$\mathbf{z_s = x_s \oplus y_s}$

b) **înmulțirea efectivă**

Înmulțirea efectivă are ca scop determinarea modulului produsului: $|z| = |x| * |y|$

$$|z| = y_{-1}*2^{-1}|x| + y_{-2}*2^{-2}|x| + \ldots + y_{-n}*2^{-n}|x|$$

c) **Rotunjirea rezultatului.** Dacă |z| trebuie să fie depus într-un cuvânt de n+1 biți, atunci trebuie făcută o rotunjire prin adaus ce constă în a aduna 2^{-n} la rezultat dacă partea la care se renunță are bitul cel mai semnificativ 1.

CURS 5

26. Schema generala a calculatorului didactic

Memoria M , $\rho_1 M=16$; $\rho_2 M=65536$

Memoria M [65536;16] este o matrice de elemente de memorare organizată într-un spațiu de adresare unic de 65536 cuvinte a câte 16 biți fiecare.

Memoria este utilizată pentru a păstra informații neinterpretate reprezentând date sau instrucțiuni. Oricare două celule de memorie sunt accesibile în mod echivalent. Totuşi, memoria poate fi logic împărțită în segmente de instrucțiuni, date, stivă, şi segmente de memorie disponibilă. Pentru segmentul de instrucțiuni se poate prevedea şi o memorie cu conținut permanent de tip "numai citire".

Citirea şi scrierea se fac asincron sub controlul unității de comandă.

Registrul AM, $\rho AM=16$

Registrul de adresare a memoriei, AM, păstrează adresa celulei de memorie la care se face acces la un moment dat. Lungimea acestui registru se alege astfel ca $2^{\rho AM} \geq \rho_2 M$. Adresa calculată (adresa efectivă) este memorată în AM selectând, prin decodificare, cuvântul din memorie la care se va face accesul.

Registrele RG, $\rho_1 RG=16$; $\rho_2 RG=8$

Deoarece timpul de acces la memoria M este relativ mare (150ns-400ns) se va prevedea o memorie rapidă organizată sub forma a 8 registre de câte 16 biţi fiecare. Registrele RG conţin unul sau ambii operanzi necesari pentru execuţia instrucţiunilor calculatorului didactic. Unele din aceste registre vor fi utilizate şi pentru calculul adresei efective a operanzilor din memorie (Astfel, BA,BB sunt utilizate ca registre de bază; XA,XB sunt utilizate ca registre index iar IS este utilizat ca indicator pentru adresarea unor structuri de date de tip stivă. RA, RB, RC sunt utilizate numai pentru păstrarea operanzilor.)
Adresa registrului selectat este specificată în codul instrucţiunii. Selectarea registrelor generale va fi prezentată detaliat în paragraful 5.4.

Registrul CP, $\rho CP=16$
Registrul contor program CP este utilizat pentru păstrarea adresei instrucţiunii ce urmează să se execute după terminarea execuţiei instrucţiunii curente. Lungimea registrului CP se alege astfel ca $2^{\rho CP} \geq \rho_2 M$. Registrul CP poate fi iniţializat cu o valoare dată la iniţializarea sistemului, cu o valoare oarecare prin execuţia instrucţiunilor de transfer control (vezi paragraful 5.3) sau poate fi incrementat în cazul execuţiei instrucţiunilor (operaţionale) ce nu specifică transferul controlului la o altă secvenţă.

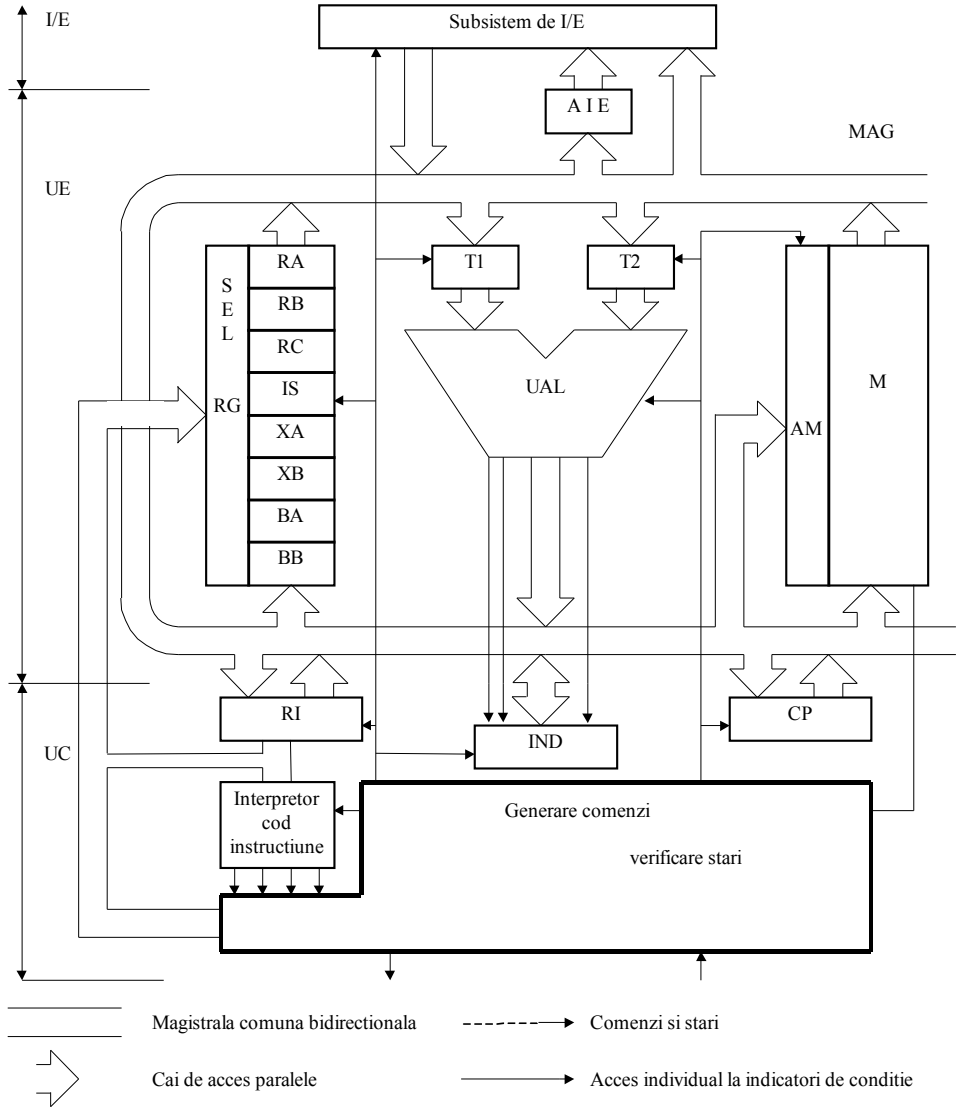
Unitatea aritmetică logică UAL, $\rho UAL=16$
Unitatea aritmetică logică (UAL) realizează operaţiile aritmetice şi logice ale calculatorului didactic şi este utilizată pentru prelucrarea datelor şi pentru calculul adresei efective. Unitatea aritmetică logică este de tip paralel, prelucrează operanzi pe 16 biţi reprezentaţi în cod complementar. Unitatea aritmetică logică implementează direct toate operaţiile elementare necesare execuţiei instrucţiunilor aritmetice şi logice ale calculatorului didactic. Condiţiile în care s-a efectuat o operaţie în unitatea aritmetică logică şi caracteristicile rezultatului sunt păstrate într-un registru de indicatori IND.

Registrele T1,T2, $\rho T1=\rho T2=16$
Registrele temporare T1 şi T2 sunt utilizate pentru a păstra operanzii unei operaţii executate în unitatea aritmetică logică , rezultate intermediare la calcularea adresei efective şi nu sunt accesibile în mod explicit de programator.

Indicatorii de condiţii IND, $\rho IND = 16$
Registrul de indicatori constituie o grupare a unor bistabili cu funcţii individuale, poziţionaţi la execuţia instrucţiunilor în funcţie de rezultatul din unitatea aritmetică logică . Registrul IND permite alegerea unei secvenţe de execuţie următoare unei operaţii aritmetice/logice în funcţie de rezultatul acestei operaţii. Lungimea registrului IND este de 16 deşi numai o parte din aceştia sunt utilizaţi în mod efectiv. Funcţiile acestor bistabili indicator vor fi prezentate în paragraful 5.3.2.

Registrul RI, $\rho RI = 16$
Registrul de instrucţiuni RI păstrează codul instrucţiunii în curs de execuţie. Conţinutul său este decodificat şi transmis secţiunii de generare comenzi/verificare stări din unitatea de comandă. În RI se păstrează şi informaţiile necesare pentru selecţia registrelor generale în funcţie de instrucţiunea în curs de execuţie.

Magistrala MAG, $\rho MAG = 16$
Interconectarea resurselor prezentate mai sus se realizează prin intermediul unei magistrale multiplexate în timp, MAG, care constituie suportul fizic de comunicaţie între aceste resurse. Dimensiunea magistralei este de 16 şi este formată din 16 linii de interconectare, fiind astfel în totalitate pasivă. Fiecare resursă conectată la magistrală va include şi circuitele de interfaţă necesare cuplării la magistrală. Transmisia pe MAG se face astfel încât un singur cuvânt de informaţie circulă pe magistrală la un moment dat.



Schema bloc a calculatorului didactic

14. Modurile de adresare la Calculatorul Didactic

Modul de adresare reprezintă modalitatea în care se calculează adresa efectivă (AE) a operanzilor implicați în instrucțiunea curentă. Instrucțiunile calculatorului didactic pot prelucra maxim doi operanzi. Aceştia se pot găsi :

-ambii în registrele generale RG ;

-unul în registrele generale RG şi altul în memorie ;

-unul în registrele generale RG şi altul în cadrul instrucțiunii respective (operand imediat);

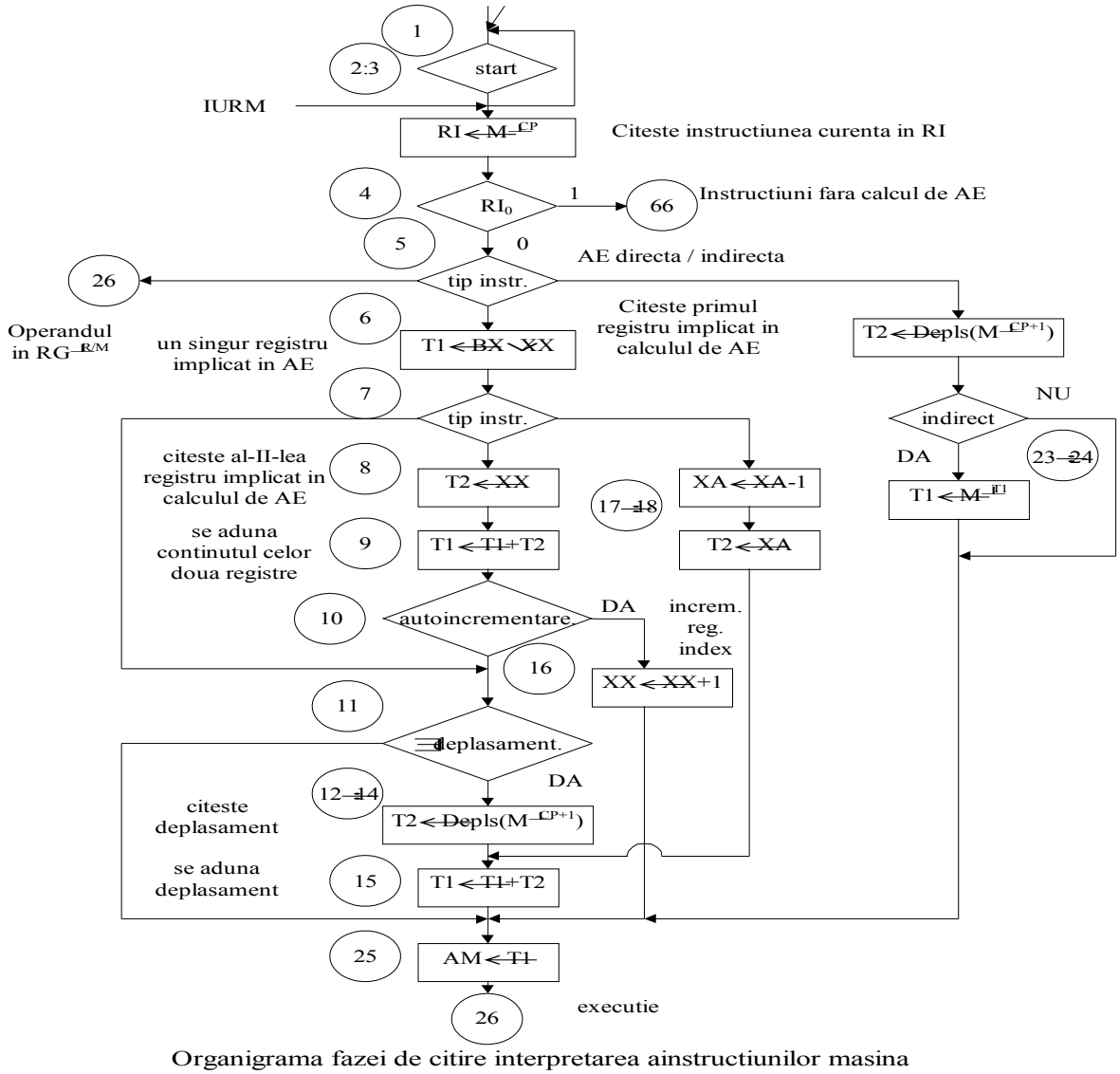
-unul în memorie şi altul imediat.

În funcție de codificarea câmpurilor MOD şi RM se precizează modul în care se calculează adresa efectivă (AE) a operandului.

1. **Adresare directa** mov RA,adresă
2. **Adresare indirecta** mov RA,[adresa]
3. **Adresare indirectă prin registru** mov RA,[BA]
4. **Adresare indirectă prin sumă de registre** mov RA,[BA] [XA]
   mov RA,[BA + XA]

5. **Adresare indirectă prin sumă de registre cu autoincrementare a registrelor index după calculul adresei efective.** mov RA,[BA][XA+]
6. **Adresare indirectă prin sumă de registre cu autodecrementare a registrului index înainte de calculul adresei efective.** mov RA,[BA][XA-]
7. **Adresare bazată** mov RA,[BA]+adresă
   mov RA,adresa[BA]
   mov RA,[BA+adresa]
8. **Adresare indexată** mov RA,[XA]+adresa
   mov RA,adresa[XA]
   mov RA,[XA+adresa]
9. **Adresare bazată indexată** mov RA,[BA][XA]+adresa
   mov RA,adresa[BA][XA]
   mov RA,[BA+XA+adresa]
   mov RA,[BA][XA].adresa
10. **Adresare imediată** mov RA,7
11. **Adresare directă la registru** mov RA,RB

17. Organigrama generala a fazei de citire si interpretare a calculatorului didactic



Organigrama fazei de citire interpretarea ainstructiunilor masina

12. Instructiuni care afecteaza indicatorii de conditie

Vom prezenta indicatorii de condiții numai pentru unitatea care lucrează cu numere reprezentate în complement față de doi.

Indicatorii propuşi sunt următorii:

S semn, reprezintă valoarea bitului de semn al rezultatului.

Z zero, este poziționat în unu dacă rezulatul este zero, şi este poziționat în zero când rezultatul este diferit de zero.

Valoarea lui Z se stabileşte $Z = \overline{\vee / REZULTAT}$ astfel:

D depăşire, este poziționat în unu, când cele două numere care se adună sunt pozitive şi există transport spre bitul de semn, sau dacă operanzii sunt negativi şi nu există transport spre bitul de semn.

Considerând x_s, y_s semnele celor doi operanzi şi t_s transportul spre bitul de semn, care atunci când apare modifică pe z_s, semnul rezultatului astfel că valoarea lui D se stabileşte cu ecuația:

$$D = x_s \wedge y_c \wedge z_c \wedge x_c \wedge y_c \wedge z$$

T transport, se poziționează pe 1 în cazul în care există un transport din bitul de semn spre stânga.

P paritate, se poziționează pe 1 în cazul în care rezulatul are un număr par de unități.

16. Modurile de adresare bazata si indexata

Adresare bazată

Adresa efectivă se obține prin adunarea conținutului celui de-al doilea cuvânt al instrucțiunii cu unul din registrele bază, Fig. 5.9.

AE = [BA / BB] + deplasament

Exemplu:
```
mov RA,[BA]+adresă
mov RA,adresa[BA]
mov RA,[BA+adresa]
```

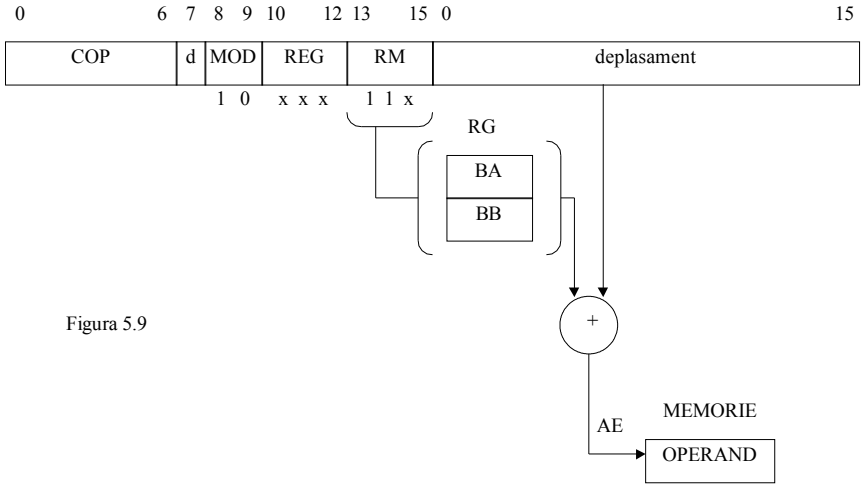


Figura 5.9

Adresare indexată

Adresa efectivă se obține prin adunarea conținutului celui de-al doilea cuvânt al instrucțiunii cu unul din registrele index, Fig. 5.10.

$$AE = \begin{bmatrix} XA \\ XB \end{bmatrix} + \text{deplasament}$$

Exemplu: mov RA,[XA]+adresa
mov RA,adresa[XA]
mov RA,[XA+adresa]

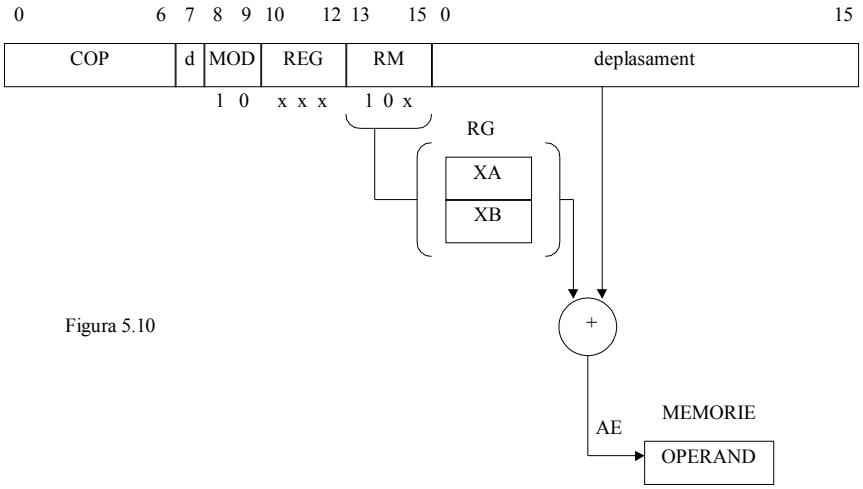


Figura 5.10

CURS 7

6. Resursele principale ale unei interfete de I/E

Cele 256 porturi de I/E, ale calculatorului didactic, pot fi asimilate cu 256 de registre plasate într-un spațiu de adresare separat de cel al registrelor generale şi al memoriei, care pot fi citite sau înscrise individual prin intermediul instrucțiunilor IN şi OUT. Plecând de la aceste premize se poate dezvolta un model simplu de subsistem de I/E bazat pe următoarele considerații :

1. vom considera echipamentele periferice fie de intrare fie de ieşire ;
2. fiecarui echipament periferic i se asociază un registru de date (din cele 256) în care unitatea centrală de prelucrare înscrie datele ce trebuie transmise către echipamentul periferic de ieşire sau din care unitatea centrală de prelucrare preia datele înscrise de un echipament periferic de intrare;
3. fiecărui echipament periferic i se asociază un registru de comenzi în care unitatea centrală de prelucrare înscrie comenzile transmise spre echipamentul periferic şi un registru de stări în care echipamentul periferic înscrie starea ce va fi preluată de unitatea centrală de prelucrare. Din punctul de vedere al implementării este convenabil să se considere 2 registre separate, unul de comenzi selectat numai la execuția instrucțiunilor OUT şi unul de stări selectat numai la execuția instrucțiunilor IN, cele două registre având aceeaşi adresă;
4. registrele de I/E sunt conectate la magistrala MAG a unității centrale de prelucrare;
5. fiecare interfață îşi recunoaşte adresele asociate;
6. datele sunt structurate pe cuvinte;

7. ca modalitate de transfer se alege transferul programat (transferul prin modulul de acces direct la memorie se tratează ulterior) ;
8. interacțiunea şi sincronizarea unității centrale de prelucrare cu echipamentul periferic se realizează prin citirea ciclică a stării echipamentului periferic utilizând instrucțiunile IN, TEST, Jcondiție, etc (interacțiunea prin întreruperi se tratează în capitolul 8);
9. magistrala MAG trebuie prevăzută, pe lângă liniile de date, cu un set minim de linii de comenzi şi stări care asigură un dialog corect între unitatea centrală de prelucrare şi registrele de I/E;
10. pentru echipamentele periferice complexe se pot prevedea mai multe registre de I/E.

Resursele memoriei tampon de tip FIFO sunt:

MT - memorie de tip RAM de 256 cuvinte a 8 biți ;

RSMT - registrul de adrese pentru operația de scriere în memoria tampon, ce indică adresa celulei în care unitatea centrală de prelucrare înscrie un nou caracter ;

RCMT - registrul de adrese pentru operația de citire din memoria tampon, ce indică adresa celulei de unde interfața preia un caracter pentru a-l transfera imprimantei ;

RDATE - registrul de date al memoriei tampon în care se citeşte caracterul ce se va transfera imprimantei.

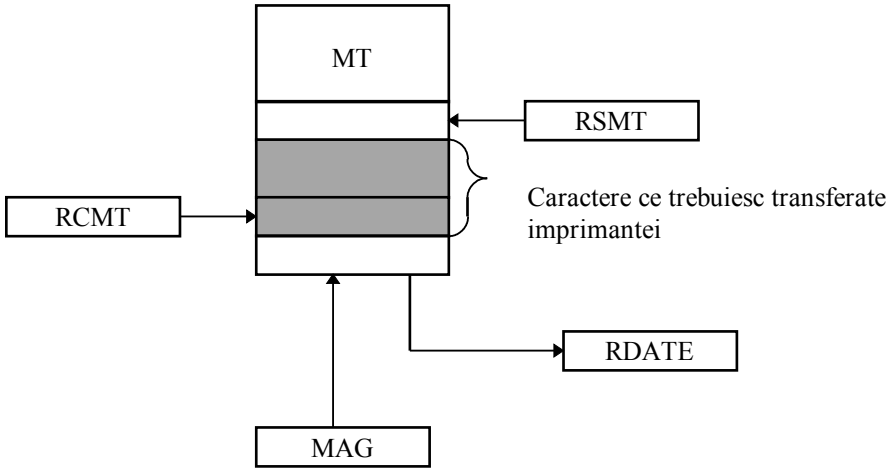


Figura 7.5 Organizarea memoriei tampon

18. Resursele minime necesare pentru interfata I/O

Idem sub.6

22. Principiul de transfer de date prin canal I/O

Prin canal de I/E vom înțelege un procesor specializat capabil să execute "programe de canal" scrise într-un limbaj maşină specializat în operații de I/E.

Canalul conține o unitate de comandă proprie pentru controlul echipamentelor periferice şi a accesului la memorie în mod independent.

Unitatea centrală de prelucrare interacționează cu canalul de I/E prin construcții ale limbajului mașină asociat, denumite comenzi de canal (inițializare, citire stare, lansare program de canal, etc.)

Programele de canal sunt generate de către programele de asistență ce se execută în unitatea centrală de prelucrare (sistemele de operare) şi sunt plasate în memoria internă pentru a fi accesibile canalului de I/E.

Un program de canal trebuie să specifice:

*0adresele zonelor de memorie implicate în transfer;

*1numărul şi dimensiunile blocurilor de date care trebuie transferate;

*2echipamentele periferice implicate şi caracteristicile transferului;

*3directive privind modul de tratare, de către canal, a unor evenimente apărute în cursul transferului;

*4operațiile care trebuie controlate de canal.

Prin comenzi de canal, unitatea centrală de prelucrare transmite canalului de I/E adresa programului de canal şi lansarea în execuție a acestuia.

În continuare unitatea centrală de prelucrare continuă execuția în paralel cu canalul. Concurența la memorie se rezolvă în aceeaşi manieră ca şi în cazul transferului prin modulul de acces direct la memorie.

Prin înlănțuirea comenzilor şi programelor de canal se pot transfera mai multe blocuri de date fară intervenția unității centrale de prelucrare.

Canalul este capabil să acționeze în mod "inteligent" în cazul apariției unor evenimente în cursul transferului (ex: erori de poziționare a capetelor la disc, porțiuni de suport magnetic deteriorat, erori de citire tranzitorii, etc.) rezolvând fară intervenția unității centrale de prelucrare aceste situații.

La terminarea execuției tuturor programelor de canal sau la apariția unor erori fatale (iremediabile) se invocă intervenția unității centrale de prelucrare.

În general, din motive de eficiență, un canal controlează transferul cu mai multe echipamente periferice şi după modul în care se asigură căile de acces direct la memorie a acestora se disting două tipuri de canale de I/E:

*5de tip selector, care controlează un singur periferic la un moment dat;

*6de tip multiplexor, care controlează mai multe periferice simultan.

Formatul unui program de canal pentru canalul de I/E de tip selector în vederea cuplării a patru unități de discuri flexibile poate fi de forma:

- octet	1	-	cuvânt de comandă canal;
- octet	2	-	instrucțiuni pentru echipamentul de I/E (cod operație);
- octet	3	-	numărul de înregistrari de transferat;
- octet	4	-	adresă pistă;
- octet	5	-	adresă sector (primul sector din bloc);
- octeții	6,7	-	adresa zonei de memorie implicată în transfer;
- octet	8	-	număr program de canal curent;
- octeții	9,10	-	adresa următorului program de canal, dacă este cazul.

CURS 8

7. Principiul de functionare al unui sistem de intreruperi

După modul în care se face activarea secvenței (a rutinei) de tratare a unei întreruperi se disting următoarele tipuri de sisteme de întreruperi:

*0nevectorizat;

*1vectorizat.

Sistemele de întreruperi nevectorizate sunt acelea în care toate cererile de întrerupere forțează transferarea controlului la o locație fixă denumită celulă capcană, unde se află punctul de intrare (sau legătura cu rutina de tratare) în programul de tratare a tuturor întreruperilor. Sursa de întrerupere şi activarea rutinei specifice se determină prin citirea unui registru de stare, denumit registrul cererilor de întrerupere (sau a mai multor registre).

Dacă există mai multe cereri prioritatea de servire a acestora se stabileşte prin program. Timpul de răspuns între o cerere şi lansarea în execuție a rutinei specifice de tratare poate fi destul de mare la acest tip de sistem de întrerupere.

Au avantajul unei complexități reduse a resurselor hardware necesare, dar tratarea unor cereri care apar în timpul tratării altor cereri este dificilă.

Sistemele de întreruperi vectorizate sunt acelea în care legătura dintre cereri şi rutinele de tratare se realizează prin intermediul unui vector de adrese (denumite şi celule capcană) cu o componentă directă pentru fiecare nivel. Fiecare componentă conține adresa, sau informații privind calculul adresei rutinei de tratare astfel că printr-un salt indirect prin această locație se inițiază execuția rutinei de tratare.

Sistemul de întreruperi furnizează spre unitatea centrală de prelucrare codul nivelului care trebuie tratat în acel moment, cod ce va fi utilizat pentru selecția componentei asociate din vectorul de adrese al rutinelor de tratare.

Prioritatea între niveluri este stabilită de către sistemul de întreruperi la nivel fizic, timpul de răspuns al sistemului este mai mic decât în cazul precedent.

În practică se utilizează frecvent o soluție ce combină avantajele şi dezavantajele celor două tipuri de sisteme de întreruperi şi anume sisteme vectorizate pe niveluri şi nevectorizate pe subniveluri.

15. Caracteristicile generale ale sistemului de intreruperi

La elaborarea unui sistem de întreruperi trebuie avute în vedere mai multe aspecte cum ar fi:

*0pentru a facilita tratarea lor, cererile de întrerupere trebuie partiționate în clase, după sursa de generare, denumite în general niveluri de întreruperi;

*1în funcție de natura lor (de proveniență) există cereri care trebuie tratate imediat, chiar dacă sunt în curs de tratare alte cereri, ceea ce impune ordonarea după o anumită schemă de priorități a nivelurilor de întrerupere;

*2pe un nivel se pot conecta mai multe cereri, subniveluri de acceaşi prioritate. În acest caz trebuie să existe posibilitatea identificării fiecărui nivel şi subnivel;

*3deoarece prin mecanismul întreruperilor este posibilă execuția concurentă a mai multor programe, se impune necesitatea protejării față de întreruperi a unor secvențe de cod maşină indivizibile, denumite secțiuni critice. Aceasta se poate realiza prin dezactivarea globală sau individuală (mascare) a întreruperilor pe diferite durate de timp;

*4sistemul de întreruperi trebuie prevăzut cu facilități de salvare şi restaurare a contextului programului întrerupt, într-o manieră cât mai eficientă;

*5trebuie prevăzută posibilitatea unui dialog cu subansamblele (echipamentele) care au generat cereri de întrerupere privind tratarea cererii;

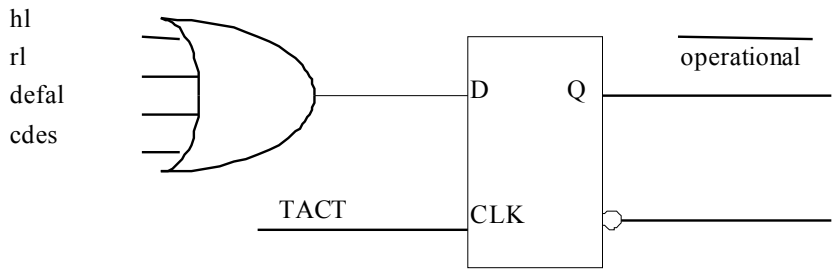
*6timpul de răspuns al sistemului de întreruperi (SI), definit ca intervalul dintre lansarea unei cereri de întrerupere spre sistemul de întreruperi şi momentul execuției primei instrucțiuni din secvența de prelucrare efectivă a cererii, trebuie să fie cât mai mic.

După modul în care se face activarea secvenței (a rutinei) de tratare a unei întreruperi se disting următoarele tipuri de sisteme de întreruperi:

*7nevectorizat;

- vectorizat.

21. Scrierea unui caracter in transferul programat cu intreruperi



Modul de lucru al interfeței, pe baza căruia se va proiecta hardware-ul necesar, se poate rezuma la următoarele caracteristici:

- interfața aşteaptă comenzi, date de la unitatea centrală de prelucrare sau citirea stării de către unitatea centrală de prelucrare ;
- dacă nu este activă nici o operație cu unitatea centrală de prelucrare , se analizează dacă în memoria tampon există caractere şi dacă da, se tipăresc şi se actualizează adresa de citire din memoria tampon;
- pentru operația de citire stare pune pe magistrala MAG cuvântul de stare;
- pentru operația de transfer comenzi, preia comenzile trimise de unitatea centrală de prelucrare în bistabilii de comandă;
- pentru operația de transfer date, înscrie data în memoria tampon, actualizează adresa de scriere în memoria tampon şi verifică dacă nu s-a umplut memoria tampon.

13. Succesiunea operatiilor pentru scrierea unui caracter

Caracterul se presupune în registrul RB.

```
;Rutina de scriere caracter la imprimantă RTIP
;Intrări          :codul ASCII al caracterului de tipărit în registrul RB
;Ieşiri           :se tipăreşte caracterul pe foaia de imprimantă
;Funcție          :tipăreşte caracter în modul de lucru transfer programat (bucla de aşteptare)
;Afectează        :registrul RA şi indicatorii de condiție
RTIP:
RCSI        EQU     0FBH
RDATI       EQU     0FAH
AST:        IN      RCSI        ;citeşte starea
            TEST    RA,8000H    ;verifică dacă este operațional
            JNZ     ERR         ;salt la tratarea erorii
            TEST    RA, 100H    ;dacă memoria tampon a interfeței
            JNZ     AST         ;nu poate primi caracter atunci asteaptă
            MOV     RA, RB      ;se transferă caracterul în RA
            OUT     RDATEI      ;se transmite în memoria tampon
            RET

ERR:                            ; rutina de analiza a erorii
```

CURS 7

5. Principiul de transfer date prin DMA

Transferul datelor este efectuat fară intervenţia unităţii centrale de prelucrare. Pentru aceasta, modulul de acces direct la memorie conţine toate resursele necesare pentru efectuarea autonomă a transferului (registru de adresare a memoriei, contor de cuvinte transferate, mecanismul de acces la memorie, întreruperi, etc.)

Soluţia de rezolvare a concurenţei la memorie dintre unitatea centrală de prelucrare şi modulul de acces direct la memorie conduce la două moduri de transfer prin DMA şi anume :

*0transfer prin furt de ciclu;

*1transfer în mod rafală.

În cazul transferului prin furt de ciclu concurenţa la memorie se rezolvă la nivel de ciclu elementar de acces la memorie. Pentru fiecare cuvânt transferat modulul de acces direct la memorie îşi dispută cu unitatea centrală de prelucrare accesul la memorie. În cazul unor cereri simultane de acces la memorie, modulul de acces direct la memorie are prioritate şi în felul acesta "fură" de la unitatea centrală de prelucrare un ciclu de acces.

Cu excepţia operaţiei de acces efectiv la memorie, activitatea modulului de acces direct la memorie şi a unităţii centrale de prelucrare se desfăşoară în paralel, mărind astfel productivitatea sistemului.

În cazul transferului cu echipamentele periferice critice, care necesită servirea imediată a unei cereri de acces la memorie, pentru o funcţionare corectă se utilizează transferul în mod rafală.

Acest mod de transfer constă în faptul că după iniţierea unei operaţii de transfer, activitatea unităţii centrale de prelucrare este suspendată până la terminarea întregului tranfer de date. Productivitatea sistemului în acest caz este mai mică, în special în cazul transferului de blocuri cu lungimi mari. O astfel de soluţie este utilizată în cazurile în care transferul nu ar putea avea loc sub controlul unităţii centrale de prelucrare şi nici prin modulul de acces direct la memorie dacă accesul la memorie s-ar disputa la fiecare cuvânt.

Iniţierea explicită a tuturor parametrilor transferului şi intervenţia unităţii centrale de prelucrare la apariţia unor evenimente apărute în timpul sau la terminarea transferului menţin un grad destul de mare de interacţiune între unitatea centrală de prelucrare şi modulul de acces direct la memorie.

23. Resursele principale ale unui modul DMA

Modulului de acces direct la memorie i se asociază 8 registre (porturi) de I/E conform cu tabelul Tab. 7.1, prezentat în cele ce urmează. Adresa de bază a grupului de 8 adrese asociate modulului de acces direct la memorie poate fi selectată oriunde în spaţiul de 256 de adrese.

Tabelul 7.1

ADRESA	DENUMIRE	FUNCŢIE
Baza +0	$CSEP_0$	Registrul de comenzi/stări pentru EP 0
Baza +1	$CSEP_1$	Registrul de comenzi/stări pentru EP 1
Baza +2	$CSEP_2$	Registrul de comenzi/stări pentru EP 2
Baza +3	$CSEP_3$	Registrul de comenzi/stări pentru EP 3
Baza +4	CSDMA	Registrul de comenzi/stări pentru modulul de acces direct la memorie
Baza +5	AMDMA	Registrul adresare memorie pentru modulul de acces direct la memorie

Baza +6	RCDMA	Registrul contor (lungime bloc) pentru modulul de acces direct la memorie
Baza +7	ITDMA	Registrul comenzi pentru inițializarea modulului de acces direct la memorie

- Registrele $CSEP_{0:3}$ - sunt scrise/citite de unitatea centrală de prelucrare şi unitatea de comandă a modulului de acces direct la memorie şi au un format nespecificat la proiectarea modulului de acces direct la memorie. Aceste registre se asociază echipamentelor periferice interfațate prin modulul de acces direct la memorie, structura lor fiind determinată în funcție de caracteristicile echipamentului periferic. Modulul de acces direct la memorie face doar decodificarea adreselor acestor registre şi generează semnalele corespunzătoare de citire/scriere a conținutului acestora, ele fiind localizate în interfața cu echipamentul periferic. Comenzile transmise de către unitatea centrală de prelucrare spre aceste registre sunt preluate de echipamentul periferic de pe magistrala MAG[16] . Stările actualizate de echipamentul periferic sunt citite de unitatea centrală de prelucrare prin activarea acestora pe MAG[16], iar datele de ieşire din echipamentul periferic sunt scrise în memorie direct prin intermediul magistralei MAG[16]. Astfel, operațiile de acces la memorie, controlate de modulul de acces direct la memorie, se pot efectua în paralel cu oricare alte operații, exceptând operațiile de acces efectiv la memorie a unității centrale de prelucrare . Concurența la memorie este rezolvată prin funcțiile de control asociate modulului de memorie, UCM, ceea ce implică modificarea corespunzătoare a secvențelor de citire, interpretare şi execuție a instrucțiunilor calculatorului didactic.
- Registrul AMDMA - este scris/citit de unitatea centrală de prelucrare şi conține adresa de memorie la care se face acces în ciclul curent. Ieşirile acestui registru, multiplexate cu ieşirile registrului AM, formează adresa locației de memorie implicată într-un transfer cu acces direct la memorie. AMDMA este incrementat de către unitatea de comandă a modulului de acces direct la memorie după fiecare operație de acces la memorie efectuată de modulul de acces direct la memorie.
- Registrul contor al modulului de acces direct la memorie (RCDMA) - este scris/citit de unitatea centrală de prelucrare şi conține numărul de cuvinte ce mai trebuie transferate în cadrul blocului curent. După fiecare cuvânt transferat, RCDMA este decrementat de unitatea de comandă a modulului de acces direct la memorie, iar în momentul în care conținutul său devine egal cu zero se consideră terminarea transferului întregului bloc. Terminarea transferului va fi semnalată unității centrale de prelucrare prin modificarea corespunzătoare a stării sau prin generarea unei întreruperi.
- Registrul CSDMA - este scris/citit de unitatea centrală de prelucrare şi constituie registrul de comenzi şi stări pentru modulul de acces direct la memorie.

24. Caracteristicile principale ale interfetei seriale

*0Transmisia/recepția cuvintelor prin intermediul interfeței seriale se face utilizând modalitatea de transfer programat;

*1Rata de transfer este funcție de semnalul de tact aplicat interfeței. Frecvența semnalului de tact va fi de 16 ori mai mare decât frecvența de transfer, pentru a asigura o bună funcționare a automatului de recepție;

*2Interfața lucrează în mod "full duplex" (poate primi şi transmite cuvinte simultan);

*3Generează semnalele de dialog conform standardului de transmisie serială CCITT V.24;

*4Transferă cuvinte de lungime egală cu 5, 6, 7, 8 biți;

*5Analizează/generează unul sau doi biți de stop;
*6Analizează/generează paritate pară sau impară;
*7Interfața detectează erorile de paritate, depăşire de ritm şi biți de stop incorecți;

CURS 9

8. Codificarea microinstructiunilor

modalități de codificare a acestora şi anume:
*0codificare verticală sau maximală ;
*1codificare orizontală sau cu control direct ;
*2codificare minimală ;
*3codificare cu control rezidual ;
*4codificare cu control prin adrese ;
*5codificare mixtă.

Codificare verticală

În cadrul codificării verticale, fiecare microinstrucțiune operațională specifică o singură microoperație. Setul de microoperații (MO) necesar pentru controlul primitivelor funcționale se codifică în [$\log_2$ | (MO)] biți, care constituie lungimea cuvântului din MC.

Pentru identificarea microoperației specificate de μI se utilizează un decodificator, Fig. 9.10

Această codificare reprezintă un caz extrem, deoarece elimină orice posibilitate de desfăşurare paralelă a operațiilor elementare.

Din punctul de vedere al minimizării cuvântului de control, codificarea verticală implică numărul cel mai mic de biți. Dimensiunea mare a decodificatorului face ca realizarea fizică a acestuia să aibă loc pe mai multe niveluri, ceea ce conduce la introducerea de întârzieri.

Un dezavantaj major al codificării maximale îl reprezintă eliminarea controlului paralel asupra resurselor precum şi inflexibilitatea dezvoltării sau completării sistemului în ceea ce priveşte introducerea de noi microoperații. Este aplicabilă numai în sisteme dedicate care au o structură specifică.

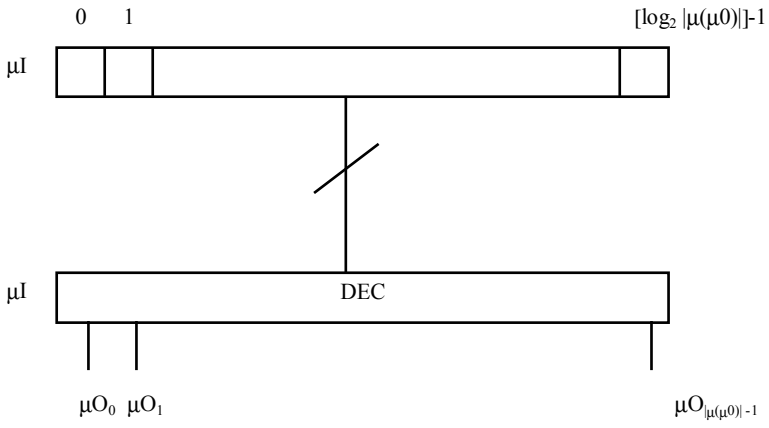


Figura 9.10
Codificarea verticală a microoperațiilor

Codificarea orizontală

În cadrul acestei codificări, fiecare microoperație din setul (MO) este pusă în corespondență cu un bit din cadrul cuvântului de control. Controlul microoperațiilor se face în mod direct, Fig. 9.11.

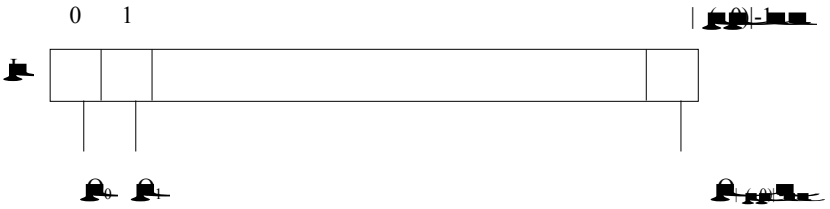


Figura 9.11 Codificarea orizontală

Codificarea orizontală realizează controlul tuturor microoperațiilor paralele, posibile, ce se pot desfăşura în sistem.

Deşi oferă o flexibilitate mare şi asigură paralelismul maxim, utilizarea acestei codificări este un caz extrem din cauza folosirii ineficiente a memoriei de control.

Codificarea minimală

Combină flexibilitatea şi paralelismul potențial oferite de codificarea orizontală cu eficiența codificării verticale.

Ideea de bază este de a grupa în clase de compatibilitate setul de microoperații care se exclud reciproc (μO dintr-o clasă de compatibilitate nu se vor efectua niciodată simultan).

Microinstrucțiunea este împărțită în câmpuri. Un câmp corespunde unei clase de compatibilitate. La nivel de câmpuri se realizează o codificare orizontală iar în cadrul câmpurilor se realizează o codificare verticală, Fig 9.12 a).

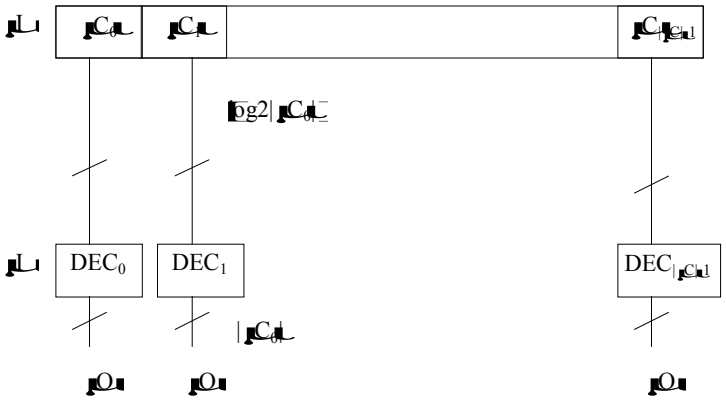


Figura 9.12 a)Codificare minimală

Pentru un câmp μC_j care codifică $|\mu C_j|$ microoperații sunt necesari $|\log_2(|\mu C_j|+1)|$ biți, deoarece trebuie să se prevadă şi posibilitatea de a nu specifica nici o microoperație din cadrul câmpului.

O variantă a acestei codificări o reprezintă codificarea pe două niveluri sau indirectă.

În codificarea pe două niveluri, validarea unor câmpuri depinde de valoarea altui câmp de control din microinstrucțiune, Fig 9.12 b).

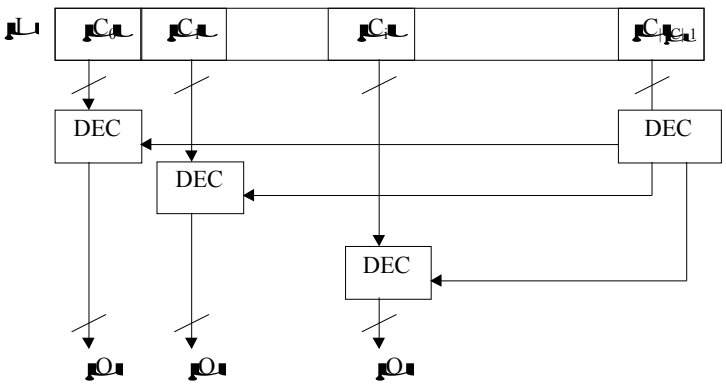


Fig. 9.12 b) Codificare minimală pe două niveluri

Codificarea cu control rezidual

Această metodă de codificare foloseşte registre de control rezidual pentru controlul primitivelor funcționale. Cuvântul de control nu controlează resursele direct ci, prin intermediul registrelor de control rezidual încărcate sub acțiunea microinstrucțiunilor, Fig. 9.13

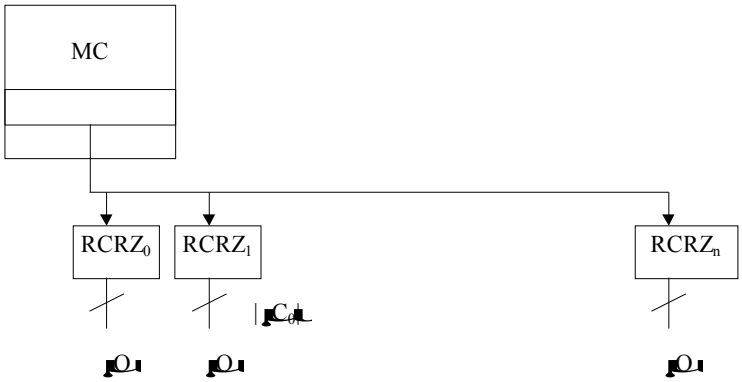


Figura 9.13 Codificare cu control rezidual

Microinstrucțiunile pot să înlocuiască sau să modifice valoarea unuia sau mai multor registre de control. Această tehnică, a controlului rezidual, asigură o economie de memorie de control atunci când unele primitive funcționale realizează aceeaşi operație în mod repetat sau când un set de microoperații este activ o perioadă mare de timp, iar alte seturi de microoperații se modifică.

Registrele de control rezidual $RCRZ_j$, care specifică microoperațiile de control al resurselor hardware, pot fi manevrate cu ajutorul unor microinstrucțiuni de dimensiuni reduse.

Codificarea cu control prin adrese

O modalitate de implementare a microinstrucțiunilor operaționale este aceea în care nu se specifică direct microoperațiile care trebuie să se desfăşoare, ci se specifică o adresă în cadrul unei memorii, unde sunt memorate toate microinstrucțiunile distincte posibile ce controlează sistemul, Fig. 9.14.

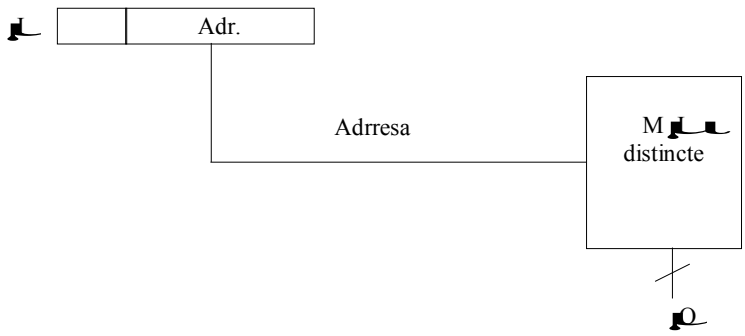


Figura 9.14 Codificare cu control prin adrese

Această modalitate reprezintă o formă simplificată a conceptului de nanoprogramare.

Numărul de microinstrucțiuni distincte nu depinde de numărul de resurse controlate ci de numărul de μO disticte, de mărimea μP şi de numărul de variabile de stare testate .

Memoria care păstrează microinstrucțiunile distincte va avea lungimea cuvântului suficient de mare pentru a controla toate microoperațiile care se pot efectua simultan. Trebuie notat că fiecare μI este memorată o singură dată.

O astfel de implementare face ca microprogramul să fie format dintr-o secvență de adrese care apelează μI păstrate în memoria de μI.

Un dezavantaj al acestei metode constă în faptul că necesită două accese la memorie în cadrul unui ciclu de microinstrucțiune, dar în schimb se realizează o economie importantă de memorie.

Codificare mixtă

O variantă utilizată mult în practică este aceea în care microinstrucțiunea este împărțită în câmpuri. Unele câmpuri controlează direct microoperațiile (sub formă codificată sau directă) iar altele specifică adrese de memorie ce conțin un subset de microinstrucțiuni distincte, Fig. 9.15.

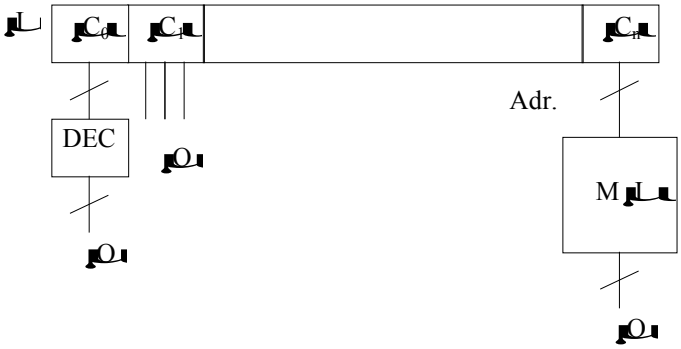


Figura 9.15 Codificare mixtă

25. Rolul microsecventiatorul ui intr-o structura microprogramata

μS - microsecvențiatorul reprezintă unitatea de comandă convențională care asigură citirea, interpretarea şi validarea acțiunilor microinstrucțiunilor. Pe baza tipului microinstrucțiunii curente, a stării primitivelor funcționale ale calculatorului didactic precum şi a codului operație al instrucțiunii maşină curente (ce interpretează şi se

execută), μS formează adresa μI următoare şi validează (activează spre resursele unității de execuție) microoperațiile din μI curentă.

μS - microsecvențiatorul, unitatea de comandă convențională, elementară, care asigură citirea interpretarea şi execuția microinstrucținilor din memoria de control precum şi înlănțuirea acestora pe baza registrului de instrucțiuni maşină RI şi a stării primitivelor funcționale.

Având în vedere structura microinstrucțiunilor se poate descrie funcționarea microsecvențiatorului (μS) care asigură citirea interpretarea şi execuția microinstrucțiunilor (activarea microoperațiilor spre unitatea de execuție a CD).

Descrierea microprogramului care implementează unitatea de comandă microprogramată a calculatorului didactic se poate face utilizând un limbaj de microasamblare. Pentru aprofundarea relației între cuvântul de control şi semnalele de comandă ce controlează primitivele funcționale ale unității de execuție se exemplifică descrierea unei secțiuni de microprogram specificând direct biții din microinstrucțiune.

Trebuie remarcat faptul că stabilindu-se microinstrucțiuni de tip operațional şi de tip condițional, paşii AHPL care specifică acțiune şi salt vor fi implementați în două microinstrucțiuni distincte iar paşii AHPL care specifică salturi multiple se implementează prin mai multe microinstrucțiuni se ramificație.

Microsecvențiatorul (μS) testează starea primitivelor funcționale (caracterizată prin valoarea indicatorilor de condiții) şi biții din registrul de instrucțiuni (cod operație, mod de adresare).