3. Automatul SEQ (procesorul didactic)

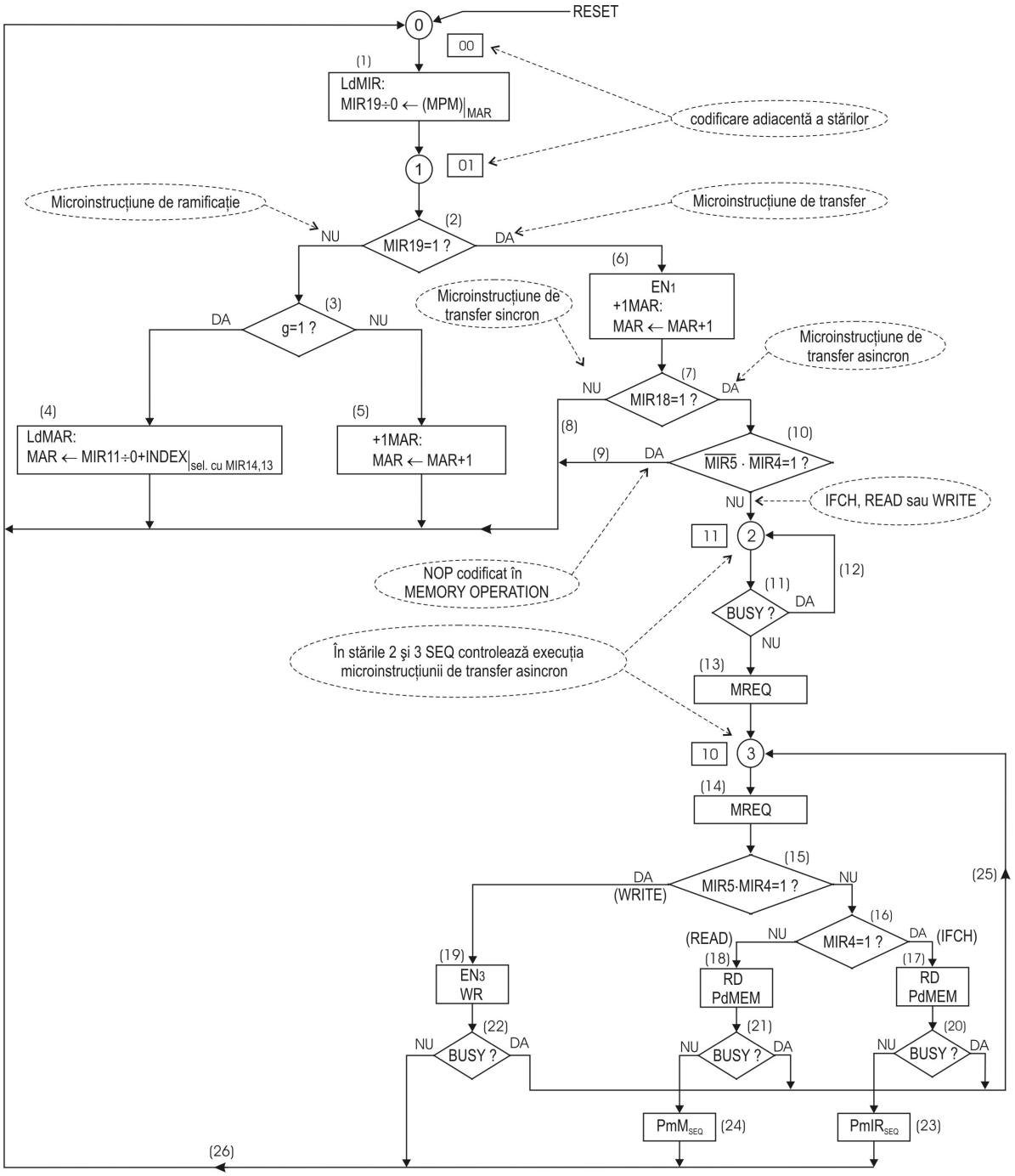

Fig 3.7 Organigrama de funcţionare a automatului SEQ

Deoarece opereaza cu un singur tip de microinstructiune (microinstructiunea generala), procesorul CISC definit la proiect revendica un automat SEQ simplificat. Simplificarile vizeaza partea de organigrama care corespunde starii 1 a automatului SEQ.

4. Proiectarea blocurilor *hardware* din componenta unitatii de control microprogramate

A. Blocul de selecţie index (procesor didactic)

Se proiecteaza in acord cu indecsii definiti in microinstructiunea de ramificatie:

MIR_{14}	MIR_{13}	Denumire INDEX	Valoare INDEX (valoare binară)											
0	0	$INDEX_1$	0	0	0	0	0	0	0	0	0	0	0	0
0	1	$INDEX_2$	0	0	0	0	0	IR_{14}	IR_{13}	IR_{12}	IR_{11}	IR_{10}	0	0
1	0	$INDEX_3$	0	0	0	0	0	0	0	0	IR_9	IR_8	0	0
1	1	$INDEX_4$	(combinaţie neutilizată; disponibilă pentru dezvoltări viitoare)											

Tabelul 3.3 Indecşii selectaţi de câmpul $MIR_{14,13}$ din microinstrucţiune

BLOCUL DE SELECŢIE INDEX intervine în execuţia succesorilor JUMPI şi respectiv JUMP având rolul de a genera cei trei indecşi definiţi la 3.4.1 (tabelul 3.3). Deşi indecşii definiţi în tabelul 3.3 ($INDEX_1$, $INDEX_2$ şi $INDEX_3$) sunt pe 12 biţi, blocul de selecţie index poate fi implementat cu doar 5 multiplexoare 4:1 (figura 3.37). Această simplificare este posibilă deoarece cei trei indecşi definiţi în tabelul 3.3 diferă doar pe zona biţilor mediani ($IND_6 \div IND_2$). Biţii IND_{11}, IND_{10}, IND_9, IND_8, IND_7, IND_1 şi IND_0 au valoarea zero în toţi indecşii definiţi în tabelul 3.3 (vezi observaţia de la 3.6).

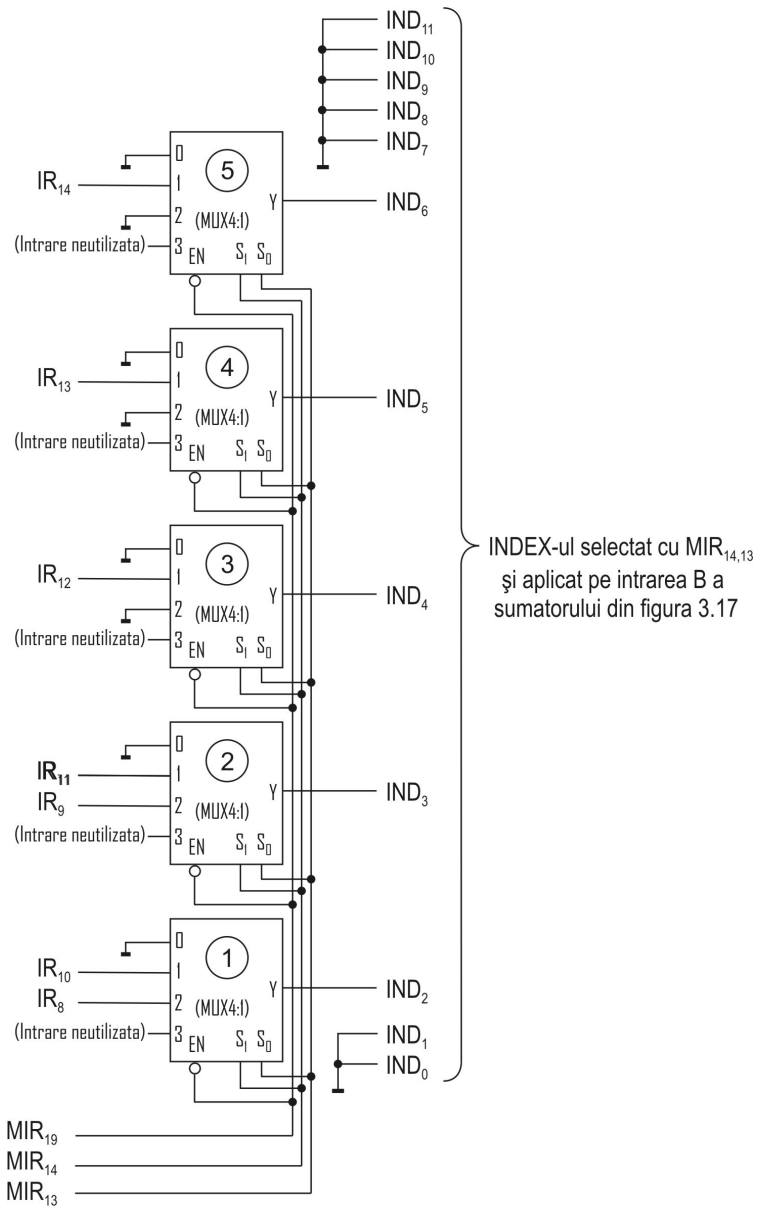

Fig 3.37 Blocul de selecţie index

Succesorii JUMPI şi JUMP pot fi codificaţi doar în microinstrucţiunile de ramificaţie (vezi tabelele 3.1 şi 3.2 de la 3.4.1). Prin urmare, blocul de selecţie index trebuie validat cu MIR_{19}=0 (dacă în MIR se află o microinstrucţiune de ramificaţie). Din acest motiv, intrarea de validare $\overline{EN}$ (activă în "0") aferentă celor 5 multiplexoare din figura 3.37 este acţionată de bitul MIR_{19}.

Cei doi biţi ai câmpului SELECŢIE INDEX (biţii MIR_{14}, MIR_{13}), aplicaţi pe intrările de selecţie (S_1 şi respectiv S_0) ale celor 5 multiplexoare 4:1, selectează cei trei indecşi definiţi la 3.4.1 (tabelul 3.3). Pe primele trei intrări aferente celor 5 multiplexoare 4:1 sunt aplicaţi biţii corespondenţi din cei trei indecşi definiţi în tabelul 3.3 ($INDEX_1$, $INDEX_2$ şi respectiv $INDEX_3$); Pentru că cel de-al patrulea index ($INDEX_4$) este nedefinit în tabelul 3.3, ultima intrare în cele 5 multiplexoare (intrarea 3) este neutilizată.

B. Blocul de selecţie a condiţiei de ramificaţie (procesor didactic)

Se proiecteaza in acord cu succesorii definiti in microinstructiunea de ramificatie (tabelele 3.1 si 3.2):

$MIR_{18\div15}$	Mnemonica microinstrucțiunii	Funcția *f* testată
0 0 0 0	JUMP μADR (μADR=$MIR_{11\div0}$)	$f=\overline{MIR}_{12}$, (*g*=I), salt necondiționat
0 0 0 1	IF (N)IOP JUMP μADR else STEP	f=IOP= $\overline{IR}_{15}$ (clasa de instrucțiuni IOP)
0 0 1 0	IF (N)AM JUMP μADR else STEP	$f=\overline{IR}_9 \cdot \overline{IR}_8$ (adresare imediată)
0 0 1 1	IF (N)AD JUMP μADR else STEP	$f=\overline{IR}_9 \cdot IR_8$ (adresare directă)
0 1 0 0	IF (N)AI JUMP μADR else STEP	$f=IR_9 \cdot \overline{IR}_8$ (adresare indirectă)
0 1 0 1	IF (N)AX JUMP μADR else STEP	$f=IR_9 \cdot IR_8$ (adresare indexată)
0 1 1 0	IF (N)INTR JUMP μADR else STEP	f=INTR (cererea globală de întrerupere)
0 1 1 1	IF (N)ACLOW JUMP μADR else STEP	f=ACLOW (căderea tensiunii de alimentare)
1 0 0 0	IF (N)CIL JUMP μADR else STEP	f= CIL (cod ilegal)
1 0 0 1	IF (N)C JUMP μADR else STEP	f=C (*flag*-ul *Carry* din registrul FLAG)
1 0 1 0	IF (N)Z JUMP μADR else STEP	f=Z (*flag*-ul *Zero* din registrul FLAG)
1 0 1 1	IF (N)S JUMP μADR else STEP	f=S (*flag*-ul *Sign* din registrul FLAG)
1 1 0 0	IF (N)V JUMP μADR else STEP	f=V (*flag*-ul *Overflow* din registrul FLAG)
1 1 0 1	Neutilizată (rezervată pentru dezvoltări)	-
1 1 1 0	Neutilizată (rezervată pentru dezvoltări)	-
1 1 1 1	Neutilizată (rezervată pentru dezvoltări)	-

Tabelul 3.1 Succesorii JUMP (codificaţi în câmpul $MIR_{18\div15}$), dacă $MIR_{14,13}$=0,0)

Dacă $MIR_{14,13}\neq0,0$ atunci se va selecta un index diferit de zero (vezi tabelul 3.3). În acest caz saltul se va face la MICROADRESA DE SALT+INDEX (microadresa de bază + indexul selectat). Precizăm din nou că microadresa de bază este codificată în câmpul $MIR_{11\div0}$ iar indexul este selectat cu câmpul $MIR_{14,13}$. Succesorii devin în acest caz JUMPI (salturi indexate). Succesorii JUMPI sunt prezentaţi în tabelul 3.2

$MIR_{18\div15}$	Mnemonica microinstrucțiunii	Funcția *f* testată
0 0 0 0	JUMPI μADR+INDEX (μADR=$MIR_{11\div0}$)	$f=\overline{MIR}_{12}$, (*g*=I), salt necondiționat
0 0 0 1	IF (N)IOP JUMPI μADR+INDEX else STEP	f=IOP= $\overline{IR}_{15}$ (clasa de instrucțiuni IOP)
0 0 1 0	IF (N)AM JUMPI μADR+INDEX else STEP	$f=\overline{IR}_9 \cdot \overline{IR}_8$ (adresare imediată)
0 0 1 1	IF (N)AD JUMPI μADR+INDEX else STEP	$f=\overline{IR}_9 \cdot IR_8$ (adresare directă)
0 1 0 0	IF (N)AI JUMPI μADR+INDEX else STEP	$f=IR_9 \cdot \overline{IR}_8$ (adresare indirectă)
0 1 0 1	IF (N)AX JUMPI μADR+INDEX else STEP	$f=IR_9 \cdot IR_8$ (adresare indexată)
0 1 1 0	IF (N)INTR JUMPI μADR+INDEX else STEP	f=INTR (cererea globală de întrerupere)
0 1 1 1	IF (N)ACLOW JUMPI μADR+INDEX else STEP	f=ACLOW (căderea tensiunii de alimentare)
1 0 0 0	IF (N)CIL JUMPI μADR+INDEX else STEP	f=CIL (cod ilegal)
1 0 0 1	IF (N)C JUMPI μADR+INDEX else STEP	f=C (*flag*-ul *Carry* din registrul FLAG)
1 0 1 0	IF (N)Z JUMPI μADR+INDEX else STEP	f=Z (*flag*-ul *Zero* din registrul FLAG)
1 0 1 1	IF (N)S JUMPI μADR+INDEX else STEP	f=S (*flag*-ul *Sign* din registrul FLAG)
1 1 0 0	IF (N)V JUMPI μADR+INDEX else STEP	f=V (*flag*-ul *Overflow* din registrul FLAG)
1 1 0 1	Neutilizată (rezervată pentru dezvoltări)	-
1 1 1 0	Neutilizată (rezervată pentru dezvoltări)	-
1 1 1 1	Neutilizată (rezervată pentru dezvoltări)	-

Tabelul 3.2 Succesorii JUMPI (codificaţi în câmpul $MIR_{18\div15}$), dacă $MIR_{14,13}\neq0,0$)

Blocul de selecţie a condiţiei de ramificaţie (figura 3.38) genereză funcţia globală de ramificaţie **g**.

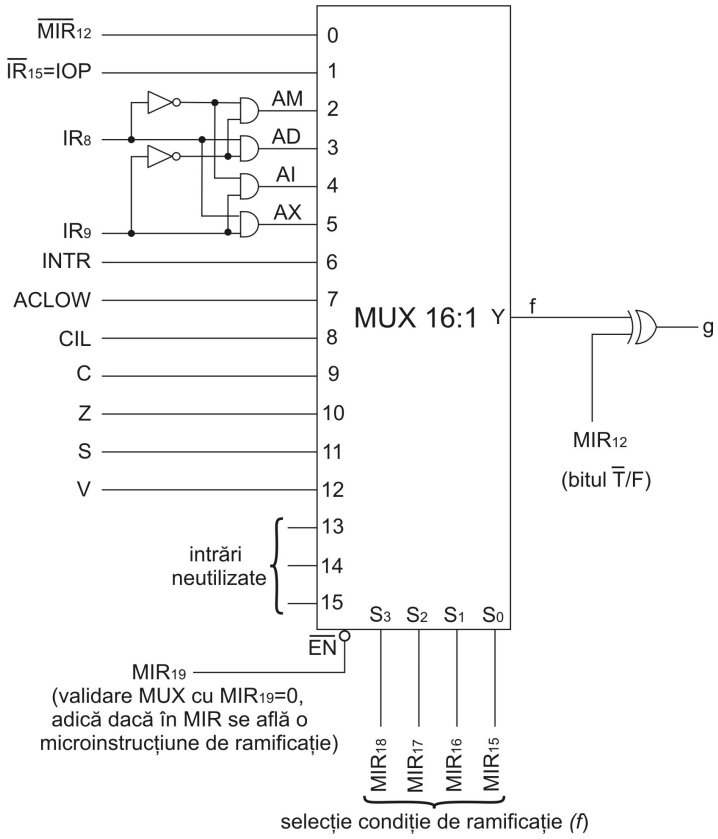

Fig 3.38 Blocul de selecţie a condiţiei de ramificaţie

Acest bloc are rol în execuţia succesorilor condiţionaţi din cadrul microinstrucţiunilor de ramificaţie (tabelele 3.1 şi respectiv 3.2). Prin urmare, blocul este validat cu MIR_{19}=0 (dacă în MIR se află o microinstrucţiune de ramificaţie).

Multiplexorul 16:1 din figura 3.38 generează funcţia de ramificaţie **f** (selectând întocmai condiţiile cuprinse în tabelele 3.1 şi respectiv 3.2). Pe baza funcţiei **f** şi a bitului MIR_{12} ($\overline{T}/F$) se generează funcţia globală de ramificaţie **g**=f$\oplus MIR_{12}$ (vezi 3.4.1). De altfel, tabelele 3.1 şi 3.2 sunt identice din punctul de vedere al condiţiilor testate; diferenţa între cele două tabele o face INDEX-ul (care are valoarea zero în cazul tabelului 3.1 şi respectiv diferită de zero în cazul tabelului 3.2)

În tabelele 3.1 şi 3.2, în afară de primul succesor, toţi ceilalţi succesori sunt condiţionaţi. Succesorii condiţionaţi au forma generică:

SUCCESOR1 else SUCCESOR2

Dacă condiţia testată (pe fals sau pe adevărat) este îndeplinită, atunci BLOCUL DE SELECŢIE CONDIŢIE DE RAMIFICAŢIE generează *g*=1. În consecinţă, se execută SUCCESOR1. Dacă condiţia testată este neândeplinită, atunci BLOCUL DE SELECŢIE CONDIŢIE DE RAMIFICAŢIE generează *g*=0. În consecinţă, se execută SUCCESOR2.

Pentru toate cele 12 microinstrucţiuni în discuţie (tabelele 3.1 şi respectiv 3.2), SUCCESOR1 este JUMPI µADR+INDEX (respectiv JUMP µADR dacă INDEX-ul selectat are valoarea zero), iar SUCCESOR2 este STEP.

După cum s-a arătat la 3.4.1.B, execuţia succesorului de salt indexat JUMPI µADR+INDEX se realizează prin operaţia:

$$MAR \leftarrow MIR_{11\div0}+INDEX\,|_{\text{selectat cu } MIR_{14,13}}$$

În acord cu cele prezentate la 3.4.1.A, execuţia succesorului JUMP µADR se realizează prin operaţia:

$$MAR \leftarrow MIR_{11\div0}$$

iar, execuţia succesorului STEP se realizează prin operaţia:

MAR←MAR+1

Funcţia g este testată de către automatul SEQ (vezi figurile 3.7 şi 3.17). Dacă g=1, SEQ va activa comanda LdMAR (necesară execuţiei succesorilor JUMPI µADR+INDEX şi respectiv JUMP µADR). Dacă *g*=0 (dacă condiţia testată este neândeplinită), SEQ va activa comanda +1MAR (necesară execuţiei succesorului STEP); se va trece astfel secvenţial la următoarea microinstrucţiune.

Observaţie:
Primul succesor din tabelele 3.1 şi respectiv 3.2 este necondiţionat (JUMP µADR în tabelul 3.1 şi respectiv JUMPI µADR+INDEX în tabelul 3.2). Pentru ca aceste salturi (necondiţionate!) să fie executate corect, SEQ trebuie să încarce microadresa de salt în MAR, prin activarea comenzii LdMAR. În conformitate cu organigrama din figura 3.7, SEQ activează comanda LdMAR numai dacă funcţia globală de ramificaţie g are valoarea 1. Prin urmare, pentru aceşti succesori de salt necondiţionat, în figura 3.38 se generează (se forţează) g=1, printr-un mic artificiu care poate fi rezumat astfel:

-Cei doi succesori necondiţionaţi (primii succesori din tabelele 3.1 şi 3.2) sunt selectaţi cu $MIR_{18\div15}=0000_2$

-Dacă $MIR_{18\div15}=0000_2$*, pe ieşirea multiplexorului din figura 3.38, rezultă* $f=\overline{MIR}_{12}$

-În aceste condiţii, la ieşirea porţii SAU-EXCLUSIV (din figura 3.38) rezultă:

$$g=\overline{MIR}_{12}\oplus MIR_{12}=1$$

Prin acest artificiu, primii succesori din tabelele 3.1 şi respectiv 3.2 devin necondiţionaţi.

C. Decodificatoarele de microcomenzi (procesorul didactic)

Se proiecteaza in acord cu campurile generatoare de microcomenzi definite in microinstructiunile de transfer sincron si respectiv asincron:

19	18	17 16 15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
1	**0**	**SURSA SBUS**	**SURSA DBUS**	**OPERAȚIE ALU**	**SURSA RBUS**	**DESTINAȚIE RBUS**	**„OTHER OPERATIONS"**
		NONE	NONE	SBUS	NONE	NONE	NOP
		PdIR[OP]	PdM	$\overline{SBUS}$	PdALU	PmIR	(Cin,PdCOND)
		PdIR[IND]	Pd $\overline{M}$	DBUS	PdIVR	PmA	(INTA,-2SP)
		PdA	PdRG	SUM	PdFLAG	PmSP	PdCOND
		PdSP	Pd $\overline{RG}$	AND	Pd ADR	PmM	Cin
		PdDs	PdPC	OR	–	PmRG	PmFLAG
		Pd-Is	PdDd	XOR	–	PmPC	+2SP
		–	Pd-Id	INV	–	PmADR	–2SP

A. Formatul microinstrucţiunii de transfer sincron

Microcomenzile (C_L) se generează prin decodificarea câmpurilor microinstrucţiunilor de transfer (sincron şi asincron). Microcomenzile generate din microinstrucţiunea de transfer sincron sunt activate în starea ST_1 a automatului SEQ. Comanda pe care am prevăzut-o pentru validarea (activarea) acestor microcomenzi este EN_1 (vezi organigrama SEQ descrisă la 3.5.1). Microinstrucţiunea de transfer asincron generează microcomenzi în stările ST_1 (comenzile codificate în câmpul SHIFT & OTHER OPERATIONS) şi respectiv ST_3 (comenzile codificate în cele patru câmpuri: SURSA SBUS, SURSA DBUS, OPERAŢIE ALU şi SURSA RBUS, câmpuri care au semnificaţie numai dacă în câmpul MEMORY OPERATION este codificată microcomanda WRITE). Pentru validarea (activarea) microcomenzilor pe durata stării ST_3 a automatului am prevăzut comanda EN_3 (vezi organigrama SEQ descrisă la 3.5.1). La implementarea BLOCULUI DE DECODIFICARE MICROCOMENZI trebuie ţinut cont de ambele surse de generare. Remintim faptul că şi automatul SEQ, în stările ST_2 şi ST_3, activează în mod direct câteva microcomenzi prin care controlează transferul procesor-memorie (vezi figurile 3.11 şi 3.7 cu descrierile aferente).

Exemplu: Generarea microcomenzilor pentru selecţia sursei pe SBUS, DBUS, RBUS şi pentru ALU

Microcomenzile pentru selecţia sursei pe SBUS, DBUS, RBUS şi pentru ALU sunt codificate în microinstrucţiunile de transfer sincron şi respectiv asincron, în câmpurile: SURSA SBUS, SURSA DBUS, SURSA RBUS şi OPERAŢIE ALU (vezi figurile 3.3 şi 3.4 şi tabelele 3.4, 3.5, 3.6 şi 3.7).
Toate aceste microcomenzi sunt de tip nivel şi se generează din două surse:

a) din microinstrucţiunea de transfer sincron, în starea ST_1 a automatului SEQ.

b) din microinstrucţiunea de transfer asincron, în starea ST_3 a automatului SEQ, dacă în câmpul MEMORY OPERATION din această microinstrucţiune este codificată microcomanda WRITE ($MIR_{5,4}$=1,1). În acest caz, comenzile în discuţie vor emite pe RBUS datele de scris în memorie (vezi paragraful C3.3 de la 3.5.1).

Pentru decodificarea celor patru câmpuri (SURSA SBUS, SURSA DBUS, OPERAŢIE ALU şi SURSA RBUS) vom utiliza decodificatoare 3:8 (figura 3.32). Cele patru decodificatoare decodifică , în ordine, cele patru câmpuri menţionate. Validarea celor patru decodificatoare este realizată cu semnalul:

$$EN=EN_1+EN_3 \qquad (3.26)$$

Ecuaţia (3.26) reuneşte cele două surse de generare descrise mai sus; EN_1 validează sursa a) iar EN_3 validează sursa b). Comenzile EN_1 şi EN_3 sunt activate de automatul SEQ, în stările ST_1 şi respectiv ST_3 (vezi 3.5.1).

Observaţii:

1. *Validarea* EN_1 *operează (în starea* ST_1 *a automatului SEQ) atât pentru microinstrucţiunea de transfer sincron cât şi pentru cea de transfer asincron (vezi 3.5.1). Validarea* EN_3 *operează (în starea* ST_3 *a automatului SEQ, numai pentru microinstrucţiunea de transfer asincron şi numai dacă în această microinstrucţiune este codificată microcomanda* WRITE*. Prin urmare, în cazul unei microinstrucţiuni de transfer asincron, comenzile codificate în cele patru câmpuri sunt activate de două ori succesiv: mai întâi în starea* ST_1 *a automatului SEQ şi apoi în starea* ST_3*, când trebuie emise pe RBUS (MBUS) datele de scris în memorie. Validarea în* ST_1 *nu este necesară dar nu deranjează. Noi chiar vom exploata această succesiune de două validări în microrutinele de execuţie aferente instrucţiunilor* INC src *şi respectiv* DEC src*, plasate la etichetele* INC *şi respectiv* DEC *în cadrul microprogramului de emulere (vezi 3.8.2). În cadrul celor două microrutine, printr-o singură microinstrucţiune de transfer asincron, care conţine microcomenzile* WRITE *(codificată în câmpul* MEMORY OPERATION*) şi respectiv* PdCOND *(codificată în câmpul* SHIFT& OTHER OPERATIONS*), vom reuşi să scriem în memorie (microcomanda* WRITE*) valoarea src+1 (la instrucţiunea* INC*), respectiv src-1 (la instrucţiunea* DEC*) şi să poziţionăm flag-urile de condiţii (microcomanda* PdCOND*) în acord cu rezultatul incrementării;* PdCOND *operează în starea* ST_1 *a automatului SEQ iar* WRITE *în starea* ST_3*.*

2. *Câmpurile* SURSA SBUS*,* SURSA DBUS*,* SURSA RBUS *şi* OPERAŢIE ALU *ocupă aceleaşi poziţii binare atât în microinstrucţiunea de transfer sincron cât şi în cea de transfer asincron. Această suprapunere a avut drept scop simplificarea implementării hardware a decodificatoarelor aferente celor patru câmpuri (figura 3.32).*

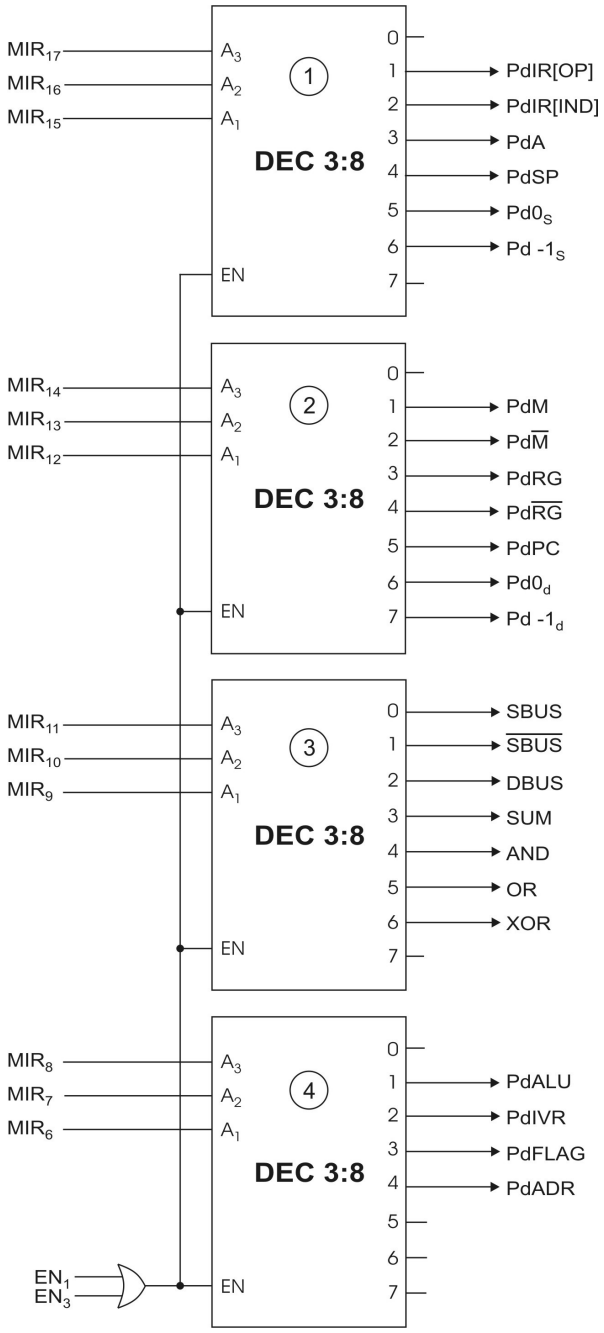

Fig 3.32 Generarea microcomenzilor pentru selecţia sursei pe SBUS, DBUS, RBUS şi a microcomenzilor pentru unitatea ALU