11. AUTOMATE COMPLEXE

În cazul în care un automat prezintă mai mult de 5÷6 variabile de stare şi de intrare, proiectarea prin metodele descrise anterior devine dificilă. Astfel, minimizarea funcțiilor de transfer şi de ieşire utilizând diagrame Karnaugh devine practic inoperantă datorită dificultăților ce apar în determinarea vecinătăților. În astfel de cazuri, proiectarea se face pe baza organigramelor care descriu algoritmii de lucru ai automatelor.

Structura generală a unui automat secvențial descris prin organigramă (figura 11.1) conține două blocuri distincte: elementele funcționale ale aplicației (EFA), care constituie schemele cu rol de execuție şi sistemul de comandă (SC) sau secvențiatorul, care furnizează funcțiile de excitație pentru EFA conform organigramei.

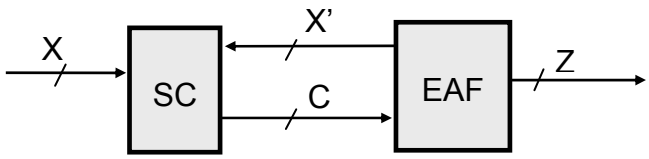


Fig. 11.1. Structura unui automat secvențial descris prin organigramă

11.1 Descrierea automatelor prin organigrame

Un automat poate fi descris printr-o organigramă logică în care sunt evidențiate operațiile prevăzute de algoritmul de lucru şi deciziile ce se iau la un moment dat. Acțiunile automatului se execută în mod secvențial, o *secvență* fiind intervalul de timp în care se execută o operație şi/sau se ia o decizie.

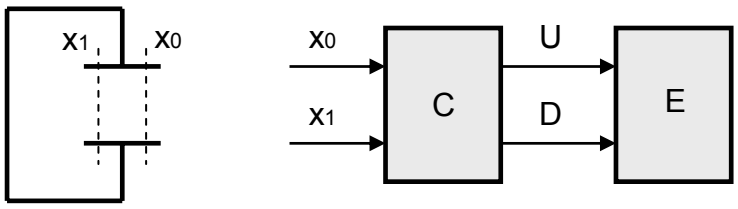


Fig. 11.2. Automat secvențial: problema muzeului

Exemplu:
Automat de contorizare a numărului de persoane aflate într-o incintă, problemă cunoscută ca *problema muzeului* (figura 11.2). În funcție de sensul parcurgerii zonei de acces (in/out) trebuie incrementat sau decrementat un numărător.

C – sistemul de comandă (secvențiatorul)
E – circuitul de evidență

După construirea organigramei (figura 11.3) se pun în evidență secvențele distincte ($s_0 \div s_4$).

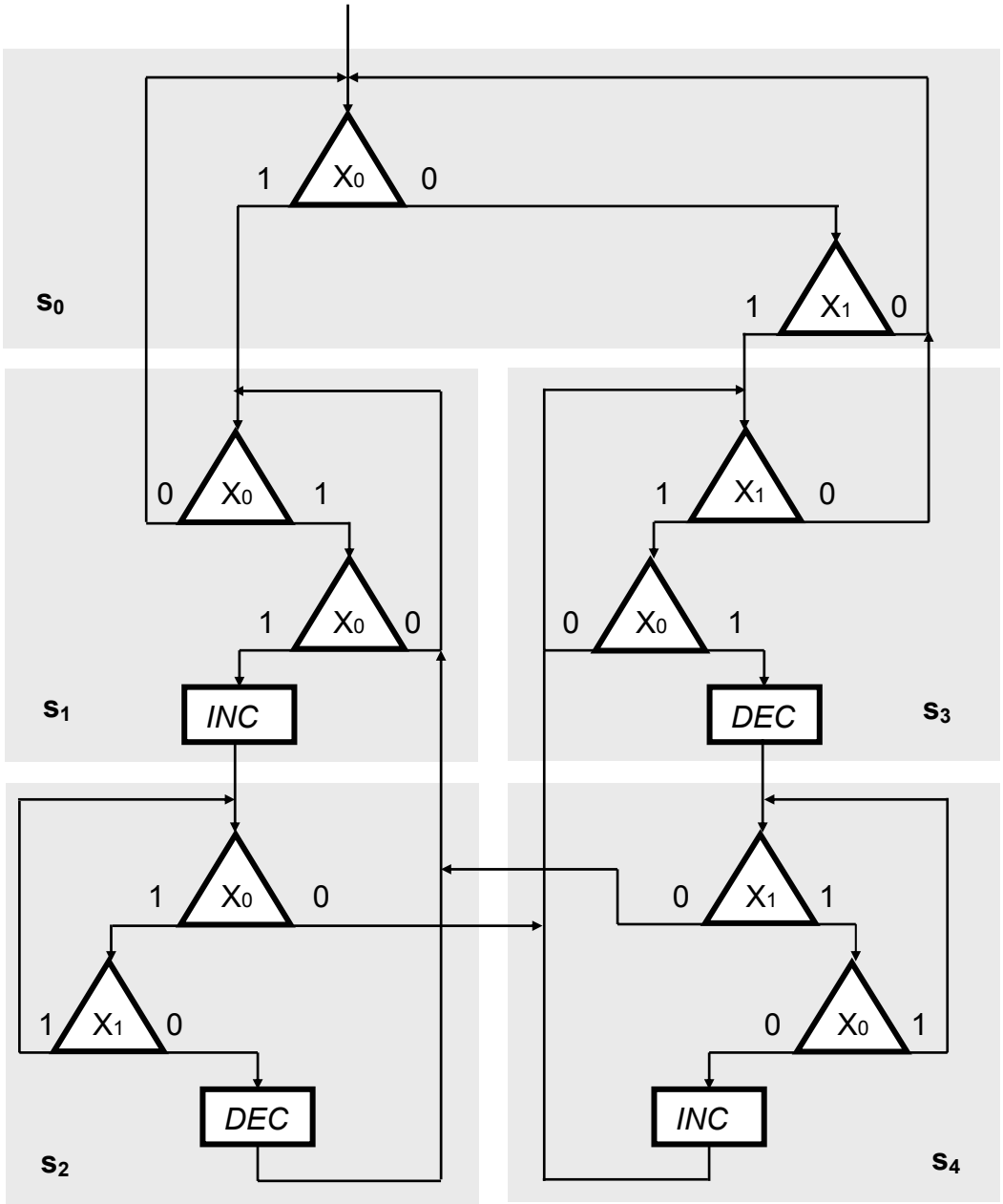


Fig. 11.3 Organigrama problemei muzeului şi evidențierea secvențelor distincte ($s_0 \div s_4$)

11.2 Elementele funcționale de aplicație (EFA)

Organigrama asociată automatului scoate în evidență pe lângă algoritmul de lucru şi elementele ce vor constitui schemele de execuție. Aceste elementele formează aşa numitele *elemente funcționale de aplicație* (EFA). Pentru exemplul considerat, schemele EFA sunt redate în figura 11.4.

Schema EFA conține un numărător zecimal reversibil (74192), ale cărui ieşiri sunt conectate la intrările unui decodificator BCD-7 segmente (7446), care pilotează un afişor "7 segmente" cu LED-uri. Pentru ştergerea numărătorului la punerea sub tensiune sau la o comandă exterioară (SW), schema este prevăzută cu un circuit de inițializare realizat cu o poartă *trigger-schmitt* 7413, care furnizează şi semnalul de inițializare pentru secvențiator, în scopul aducerii acestuia în secvența s_0.

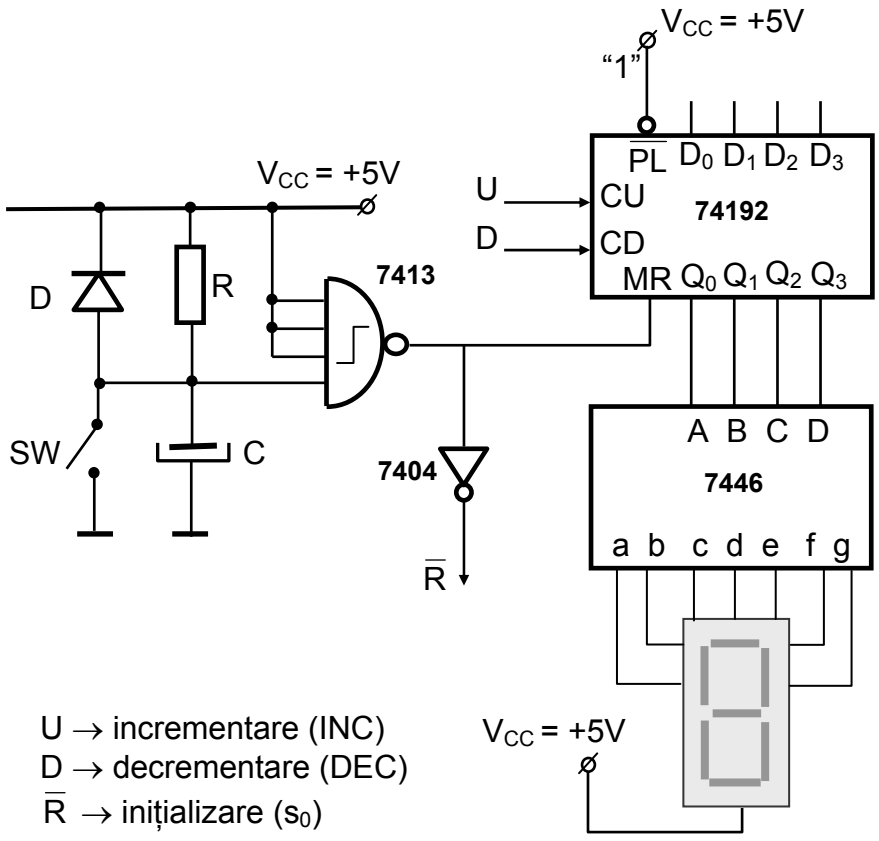


Fig. 11.4 Elementele funcționale de aplicație pentru problema muzeului

11.3 Sistemul de comandă (SC)

Blocul din componența automatului care generează secvențele de lucru şi semnalele de comandă pentru EFA se numeşte *sistem de comandă* (SC) sau *secvențiato*r. În fiecare secvență de lucru se poate executa o anumită operație (numită *micro-operație*) după care se trece la secvența următoare. Această trecere poate fi uneori condiționată.

11.3.1 Secvențiatorul cablat

Această soluție presupune utilizarea unui element de memorare a stării (registru de memorie, numărător presetabil, bistabile etc.) cuplat cu un decodificator, fiecare vector decodificat producând o acțiune specifică asupra EFA.

În vederea proiectării, fiecărei secvențe i se ataşează un vector binar distinct. Organigrama logică se transpune într-o tabelă de adevăr care conține vectorii binari (codurile) corespunzători secvenței specifice. Pentru exemplul considerat tabela de adevăr este prezentată în figura 11.5.

Sinteza variabilelor de stare astfel determinate conduce la obținerea sistemului de relații 11.1.

secvența curentă				secvența următoare			
	C	B	A		C	B	A
s_0	0	0	0	$s_0 \cup s_1 \cup s_3$	0	x_1	$x_0 \oplus x_1$
s_1	0	0	1	$s_0 \cup s_1 \cup s_2$	0	x_1	$x_0 \cdot \overline{x_1}$
s_2	0	1	0	$s_1 \cup s_2 \cup s_3$	0	x_1	$x_0 \oplus x_1$
s_3	0	1	1	$s_0 \cup s_3 \cup s_4$	x_0	$\overline{x_0} \cdot x_1$	$\overline{x_0} \cdot x_1$
s_4	1	0	0	$s_1 \cup s_3 \cup s_4$	$x_0 \cdot x_1$	$\overline{x_0}$	$x_0 \oplus x_1$

Fig. 11.5. Tabela de adevăr a secvențelor

$$\begin{cases} A = (s_0 + s_2 + s_4)(x_0 \oplus x_1) + s_1 x_0 \overline{x_1} + s_3 \overline{x_0} x_1 \\ B = (s_0 + s_1 + s_2)x_1 + s_3 \overline{x_0} x_1 + s_4 \overline{x_0} \\ C = s_3 x_0 + s_4 x_0 x_1 \end{cases} \quad (11.1)$$

Tot pe baza organigramei se determină şi expresiile semnalelor de comandă pentru EFA, ținând cont şi de particularitățile acestora. Expresiile semnalelor de incrementare U (up) respectiv decrementare D (down) au valorile 11.2.

$$\begin{cases} U = s_1 x_1 + s_0 \overline{x_0} \\ D = s_2 \overline{x_1} + s_3 \overline{x_0} \end{cases} \quad (11.2)$$

Dacă se folosesc circuite basculante bistabile de tip D (7474) pentru realizarea elementului de memorare a stării, schema secvențiatorului rezultă de forma din figura 11.6. Pentru implementarea funcțiilor logice C, B, A se folosesc porți logice.

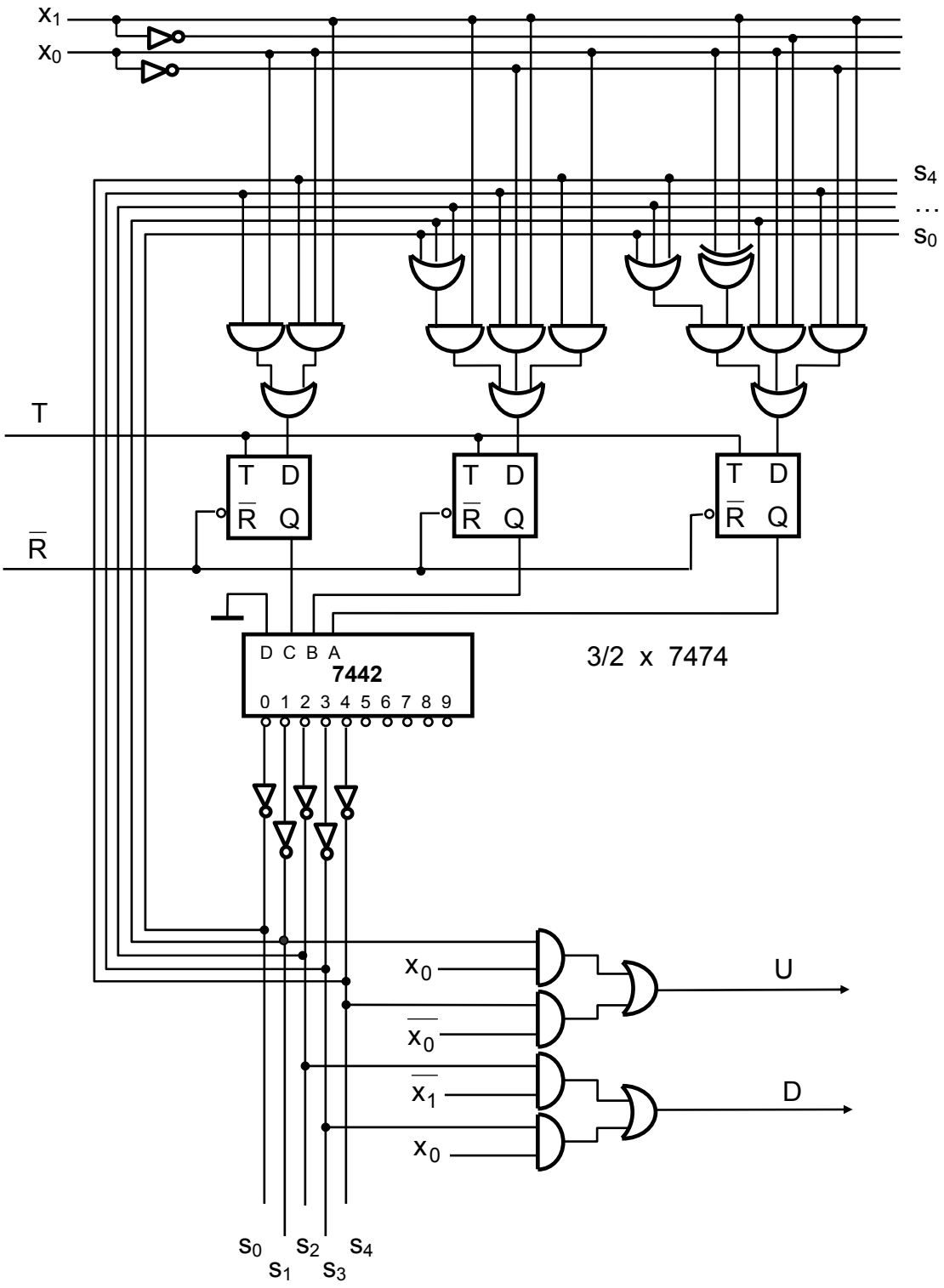


Fig. 11.6. Schema secvențiatorului cu circuite basculante

În anumite situații, trecerea de la o secvență la alta se face prin incrementarea sau prin decrementarea secvenței. Existența acestei situații sugerează posibilitatea implementării elementului de memorare cu ajutorul unui numărător zecimal reversibil şi eventual presetabil, pentru a putea asigura şi încărcarea paralelă a secvențelor neconsecutive.

Din tabela de adevăr a secvențelor se stabilesc ecuațiile funcțiilor de incrementare, decrementare şi încărcare paralelă (relațiile 11.3).

$$\begin{cases} CU = s_0 x_0 \overline{x_1} + s_1 x_1 + s_2 \overline{x_0} x_1 + s_3 x_0 \\ CD = s_1 \overline{x_0}\, \overline{x_1} + s_2 x_0 \overline{x_1} + s_4 x_0 \overline{x_1} \\ PL = s_0 \overline{x_0} + s_1 x_0 \overline{x_1} + s_2 x_0 \overline{x_1} + s_3 \overline{x_0} + s_4 x_0 \end{cases} \qquad (11.3)$$

În cazul în care semnalul pentru comanda încărcării paralele PL este activ, trebuie determinate şi secvențele din tabelul 11.7.

secvența curentă				secvența următoare			
	C	B	A		C	B	A
s_0	0	0	0	$s_0 \cup s_3$	0	x_1	x_1
s_1	0	0	1	s_1	0	0	1
s_2	0	1	0	s_2	0	1	0
s_3	0	1	1	$s_0 \cup s_3$	0	x_1	x_1
s_4	1	0	0	$s_1 \cup s_4$	x_1	0	$\overline{x_1}$

Fig. 11.7. Tabela de adevăr a secvențelor când PL este activ

Pe baza tabelului 11.7 rezultă pentru variabilele A, B, C expresiile 11.4.

$$\begin{cases} A = (s_0 + s_3)x_1 + s_2 + s_4 \overline{x_1} \\ B = (s_0 + s_3)x_1 + s_2 \\ C = s_4 x_1 \end{cases} \qquad (11.4)$$

Schema secvențiatorului în cazul în care pentru elementul de memorare se foloseşte 74193 (numărător binar reversibil presetabil) este cea prezentată în figura 11.8.

Partea de schemă care implementează semnalele de comandă (U, D) rămâne aceeaşi ca la schema precedentă.

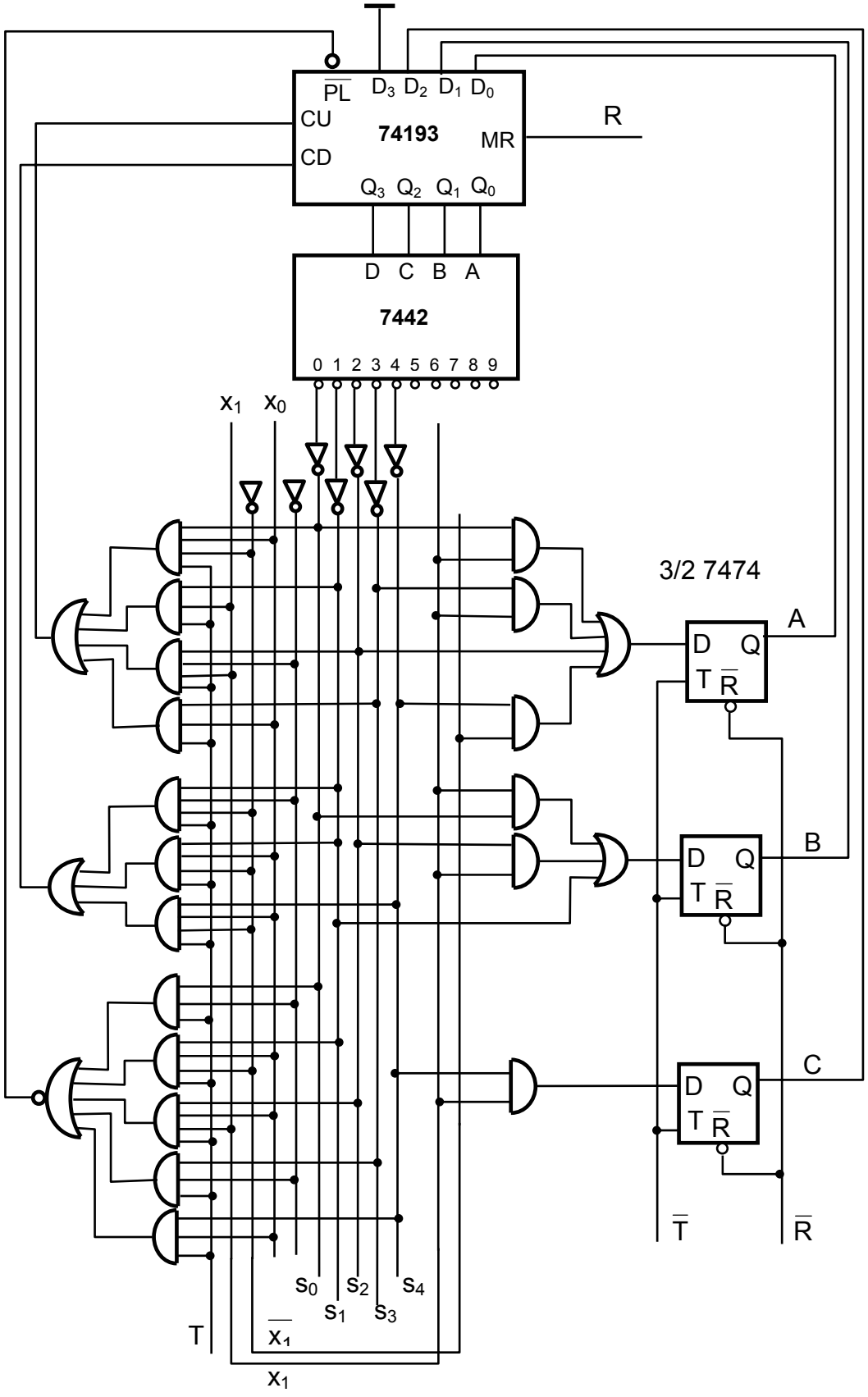


Fig. 11.8. Schema secvențiatorului cu numărător presetabil 74193 ca element de memorare

11.3.2 Secvențiator microprogramat

În cazul secvențiatorului microprogramat, secvențele organigramei sunt furnizate de conținutul locațiilor unei memorii adresată de un element de secvențiere (numărător presetabil). Cuvântul citit de la locația selectată determină acțiuni asupra EFA sau asupra elementului de secvențiere care în acest caz se numeşte *registru de adrese program*.

Una din caracteristicile secvențiatorului micro-programat este lungimea cuvântului din memoria program, cuvânt care se numeşte *micro-instrucțiune*. Adresarea micro-instrucțiunilor se poate face în două moduri: *explicit* (adresa următoare este specificată în câmpul instrucțiunii curente) sau *implicit* (adresa următoare este adresa imediat superioară celei a micro-instrucțiunii curente). Totalitatea micro-instrucțiunilor utilizate de un sistem micro-programat constituie *setul de micro-instrucțiuni*.

Pentru sinteza automatelor micro-programate sunt necesare cel puțin două tipuri de instrucțiuni, corespunzătoare celor două tipuri de adresare:

- instrucțiuni de salt condiționat: dacă este îndeplinită o anumită condiție, se execută un salt la o adresă specificată; în caz contrar, se continuă cu instrucțiunea următoare;
- instrucțiuni de comandă: se dă o anumită comandă către EFA iar programul continuă cu instrucțiunea următoare.

Câmpul instrucțiunii trebuie deci să conțină 3 zone:

1. instrucțiuni de salt condiționat:

T	C	A

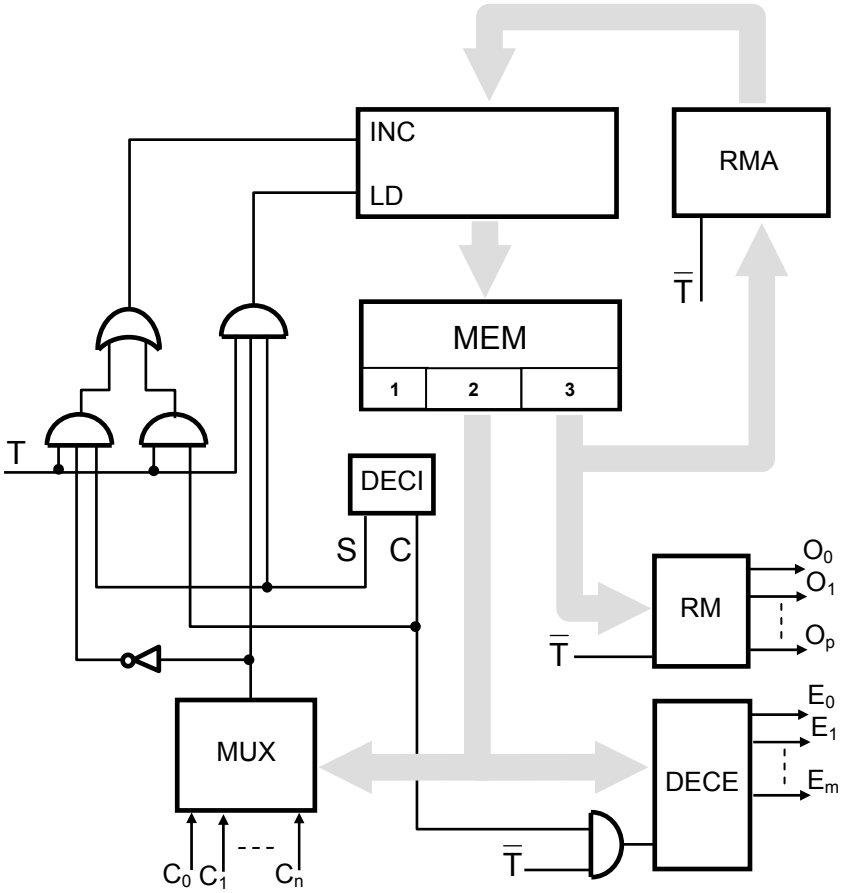
T - tipul instrucțiunii (T = 1 $\rightarrow$ instrucțiune de salt condiționat);
C - codul condiției ce provoacă saltul;
A - adresa de salt.

2. instrucțiune de comandă:

T	E	O

T - tipul instrucțiunii (T=0 $\rightarrow$ instrucțiune de comandă);
E - codul comenzii de executat;
O - operandul comenzii (de exemplu, dacă E codifică încărcarea unui registru de memorie, O este cuvântul ce va fi memorat).

Având în vedere cele expuse, schema bloc a unui secvențiator micro-programat este prezentată în figura 11.9. Un circuit de memorie MEM este adresat prin intermediul unui registru de adrese program RAP. Memoria furnizează la ieşire cele trei câmpuri ale instrucțiunii, acestea fiind la rândul lor interpretate de decodificatorul de instrucțiuni (DECI), decodificatorul de comenzi (DECE) şi selectorul de condiții (MUX) şi de registrul de memorie pentru operandul comenzii (RM). Circuitul mai conține un registru de memorie pentru adresa de salt (RMA) şi logica necesară pentru trecerea la instrucțiunea următoare.



RAP	- registru de adrese program
RMA	- registru de memorie pentru adresa de salt
MEM	- memorie program
DEC1	- decodificator de instrucțiuni
MUX	- selector de condiții
DECE	- decodificator de comenzi
RM	- registrul de memorie pentru operandul comenzii

Figura 11.9 Secvențiator micro-programat, schemă bloc

Programul de funcționare al automatului este descris de secvența de micro-instrucțiuni înscrisă în memoria programului. Pentru exemplul considerat programul este cel din figura 11.10.

Există două tipuri de instrucțiuni (IP, EX), deci tipul instrucțiunii poate fi codificat pe un bit::

JP : 1 (11.5)
EX: 0

		secvență	adresă	binar	hexa
S0	JP, $\overline{x_1}\ \overline{x_o}$, S0	S0	0000	10000000	80
	JP, $x_1\ \overline{x_o}$, S3		0001	10101001	A9
S1	JP, $\overline{x_1}\ \overline{x_o}$, S0	S1	0010	10000000	80
	JP, $\overline{x_1}\ x_0$, S1		0011	10010010	92
	EX, INC, -		0100	00000000	00
S2	JP, $x_1\ x_0$, S2	S2	0101	10110101	B5
	JP, $x_1\ \overline{x_o}$, S3		0110	10101001	A9
	EX, DEC, -		0111	00010000	10
	JP, NC, S1		1000	11000010	C2
S3	JP, $\overline{x_1}\ \overline{x_o}$, S0	S3	1001	10000000	80
	JP, $x_1\ \overline{x_o}$, S3		1010	10101001	A9
	EX, DEC, -		1011	00010000	10
S4	JP, $x_1\ x_0$, S4	S4	1100	10111100	BC
	JP, $\overline{x_1}\ x_0$, S1		1101	10010010	92
	EX INC, -		1110	00000000	00
	JP, NC, S3		1111	11001001	C9

Figura 11.10 Program pentru secvențiatorul micro-programat

Condițiile de salt se pot codifica pe 3 biți:

$\overline{x_1}\ \overline{x_o} = 1$ 000 (11.6)

$\overline{x_1}\ x_0 = 1$ 001

$x_1\ \overline{x_o} = 1$ 010

$x_1\ x_0 = 1$ 011

NC 100

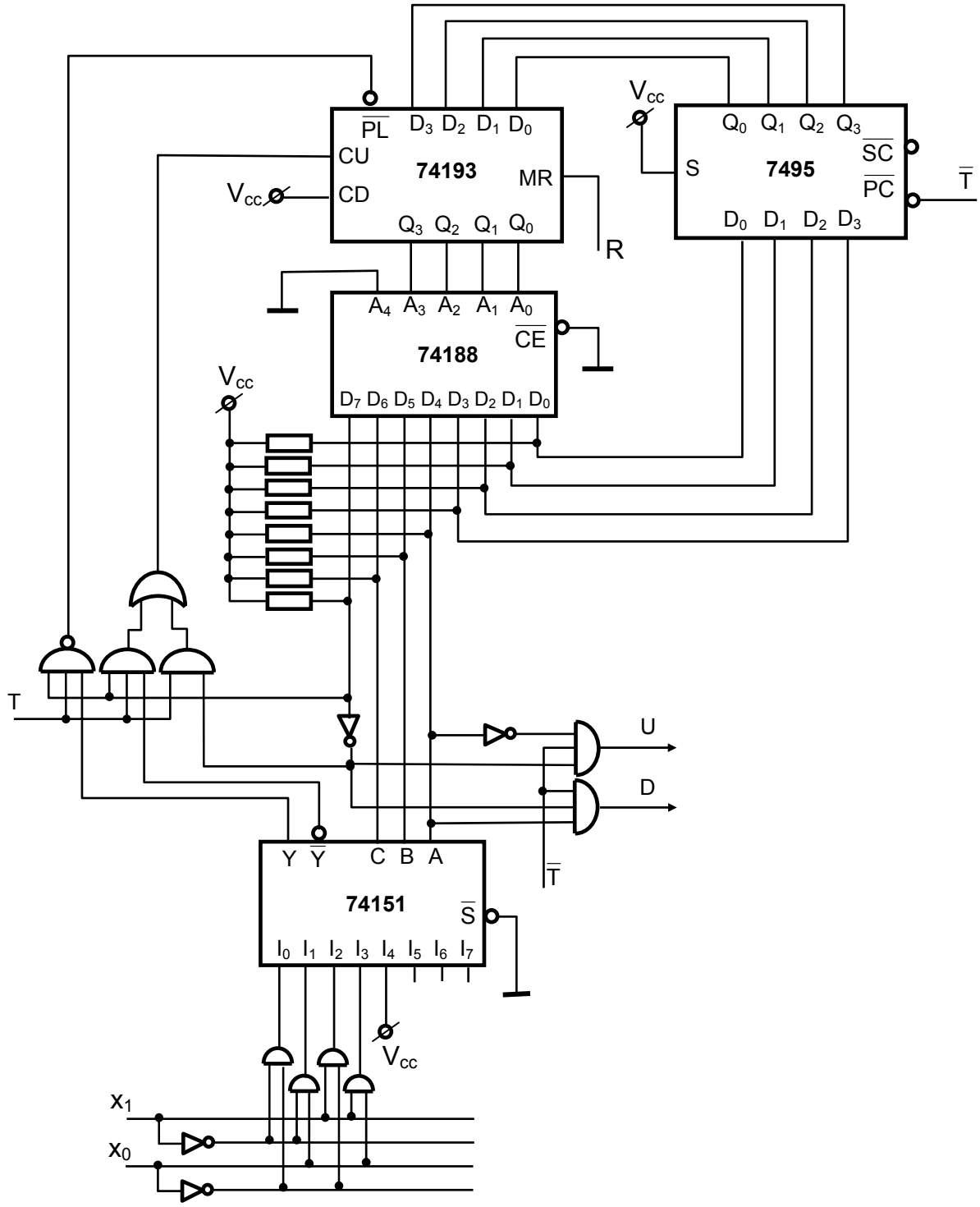


Fig. 11.11 Structura secvențiatorului microprogramat

Comenzile furnizate de secvențiator (U, D) pot fi codificate pe un singur bit, dar lungimea zonei 2 a câmpului instrucțiunii este dictată de lungimea codului condițiilor de salt (3 biți):

U (INC) 000
D (DEC) 001

Comenzile nu necesită operanzi.

Adresele de salt au lungimea de 4 biți (programul cuprinde 16 instrucțiuni plasate între adresele 0000 şi 1111).

Programul scris în cod binar şi în cod hexazecimal este prezentat deasemenea în tabelul din figura 11.10 (coloanele din dreapta), folosindu-se codificările descrise.

Pentru implementarea memoriei program este necesar un circuit de memorie cu dimensiunile 16×8. Se poate folosi un circuit 74188 (PROM 32×8 cu ieşiri cu colector în gol). Cu aceste specificații, schema secvențiatorului este cea prezentată în figura 11.11.

Se observă că acest secvențiator poate fi utilizat şi pentru comanda altor elemente funcționale ale aplicației (EFA-), dacă se schimbă programul (alt PROM), cu condiția să nu existe mai mult de 7 condiții de salt şi mai mult de 2 ieşiri de comandă. Numărul ieşirilor de comandă poate fi mărit (maximum 8) dacă se foloseşte un decodificator (7442) pentru obținerea semnalelor de comandă.

11.3.3 Secvențiator programat

Între funcțiile logice şi structura unei organigrame există o echivalență directă (figura 11.12).

Această echivalență conduce la ideea înlocuirii porților logice şi a conexiunilor dintre ele cu secvențe de program corespunzătoare, având în vedere şi faptul că structurile logice de bază pot fi combinate pentru a produce orice funcție.

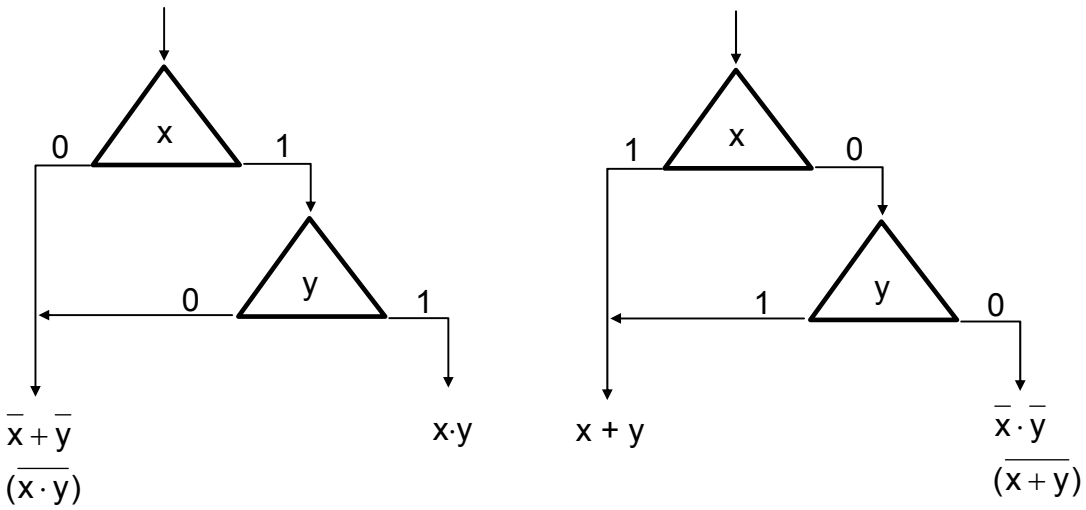


Fig. 11.12 Echivalența între funcțiile logice şi structura unei organigrame

Pe structurile prezentate (figura 11.12), funcțiile x şi y sunt generate la momente diferite de timp, folosind acelaşi suport hardware (procesorul). Din acest motiv, logica programată este mai lentă, conferind însă o flexibilitate maximă datorită posibilității implementării oricărei funcții logice.

Pentru a implementa o funcție logică solicitată de o anumită aplicație, cu ajutorul unui microcalculator, trebuie proiectată o interfață între proces şi microcalculator şi trebuie proiectat programul care să realizeze funcția respectivă. Interfața constă într-o serie de porturi de intrare/ieşire prin care se realizează comunicația proces-microcalculator. Interfața trebuie să asigure citirea mărimilor de intrare şi a condițiilor furnizate de proces, precum şi transmiterea de comenzi către proces. Interfața comunică cu microcalculatorul prin magistrala de adrese, magistrala de date şi magistrala de comandă.

Pentru exemplul considerat (problema muzeului), interfața trebuie să asigure îndeplinirea următoarelor cerințe:

- citirea mărimilor de intrare x_1, x_0; este deci necesar un port de intrare de 2 biți, cu ieşiri *three-state* pentru a putea fi conectat la magistrala de date a microcalculatorului;
- transmiterea comenzilor U, D către EFA; este necesar un port de ieşire de 2 biți;
- validarea/selectarea celor două porturi; este necesară o logică de decodificare.

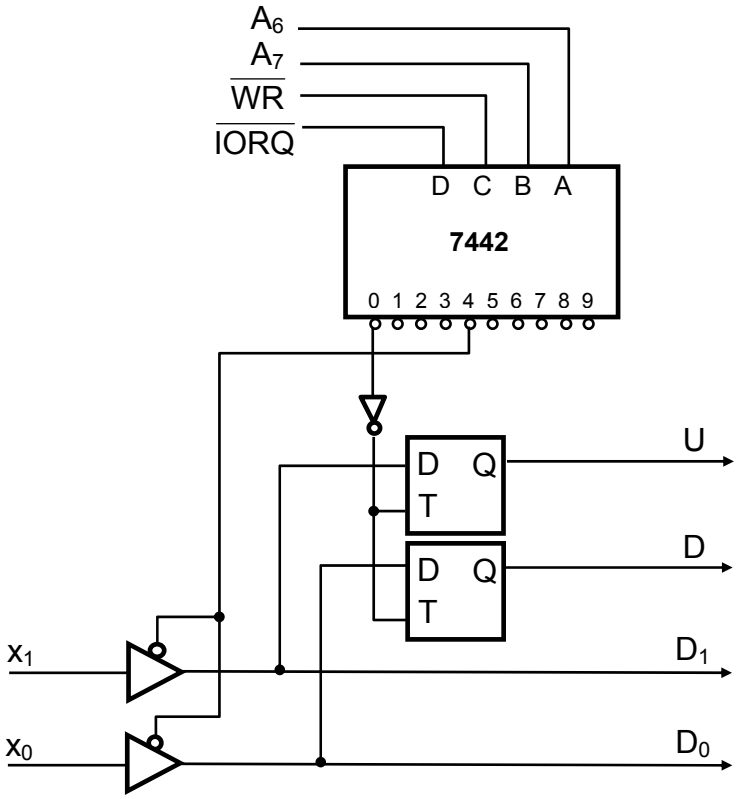


Fig. 11.13 Interfața microcalculator-proces

Având în vedere aceste cerințe, schema interfeței are structura din figura 11.13. În figura 11.14 este prezentată logica de selecție a portului de intrare (PI), respectiv ieşire (PO). Pe porturile de intrare se citesc mărimile de intrare furnizate de senzorii x_0 şi x_1 iar pe portul de ieşire se transmit comenzile de incrementare (U) şi decrementare (D). Organigrama corespunzătoare este cea din figura 11.15.

Port	Adresă	Conținut	$\overline{WR}$	Acțiune
PI	0 0 x x x x x x	x x x x x x x_1 x_0	1	Citire
PO	0 0 x x x x x x	x x x x x x 1 0	0	U (INC)
	0 0 x x x x x x	x x x x x x 0 1	0	D (DEC)

Fig. 11.14. Validarea / selectarea celor două porturi ale interfeței dintre proces şi microcalculator

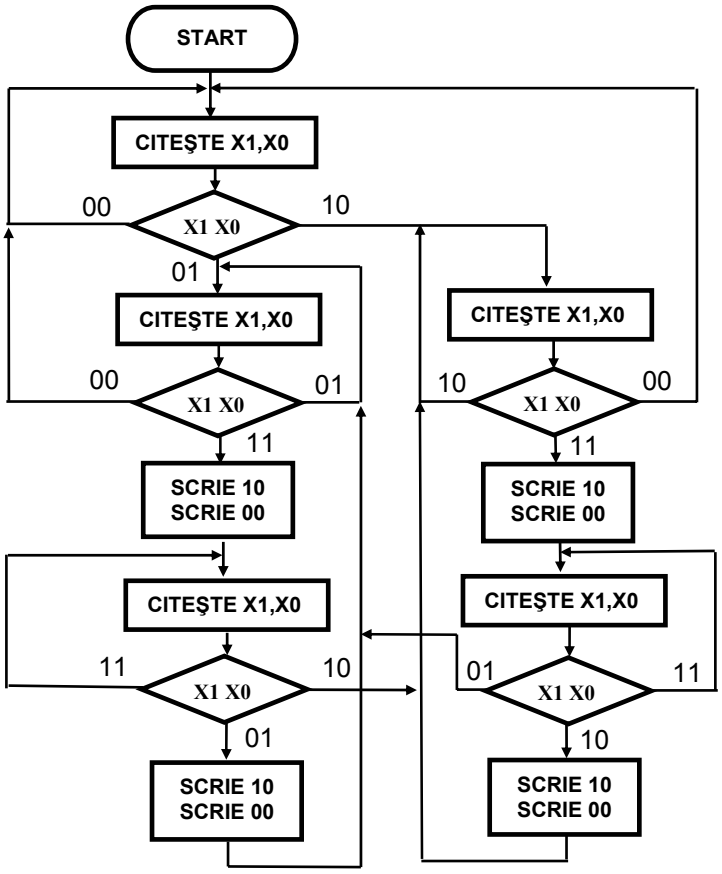


Fig. 11.15 Organigrama programului

Programul în limbaj de asamblare pentru microprocesorul Z80 este următorul (figura 11.16).

```
S0:  IN    A, (00)
     AND   03
     CP    01
     JR    C, S0
     JR    Z, S1
S3:  IN    A, (00)
     AND   03
     CP    02
     JR    C, S0
     JR    Z, S3
     LD    A, 01
     OUT   (00), A
     RES   0, A
     OUT   (00), A
S4:  IN    A, (00)
     AND   03
     CP    02
     JR    C, S1
     JR    Z, C1
     JR    S4
C1:  LD    A, 02
     OUT   (00), A
     RES   1, A
     OUT   (00), A
     JR    S3
S1:  IN    A, (00)
     AND   03
     CP    01
     JR    C, S0
     JR    Z, S1
     LD    A, 02
     OUT   (00), A
     RES   1, A
     OUT   (00), A
S2:  IN    A, (00)
     AND   03
     CP    02
     JR    C, C2
     JR    Z, S3
     JR    S2
C2:  LD    A, 01
     OUT   (00), A
     RES   0, A
     OUT   (00), A
     JR    S1
```

Fig. 11.16 Programul în limbaj de asamblare Z80

11.3.4 Criterii de alegere a tehnicii de realizare a unui secvențiator

Realizarea unui secvențiator se poate face în una din următoarele variante:

1. *logică cablată:* se foloseşte pentru automate mici (maximum 20÷40 circuite integrate) care nu solicită modificări, necesită viteze mari de lucru şi la care nu este necesară vehicularea unui volum mare de date; această soluție conferă viteză maximă de lucru, cost minim de proiectare, dar are un grad redus de flexibilitate.
2. *logică microprogramată:* această tehnică de realizare conferă secvențiatorului posibilitatea de a fi modificat fără modificarea totală a structurii (uneori trebuie modificat doar PROM-ul); viteza de lucru este mai mică decât la secvențiatorul cablat datorită microinstrucțiunilor de salt.
3. *logică programată cu microcalculator:* se foloseşte în aplicațiile ce nu necesită viteze mari de lucru, dar care solicită calcule aritmetice şi/sau memorarea/prelucrarea unui volum mare de date.