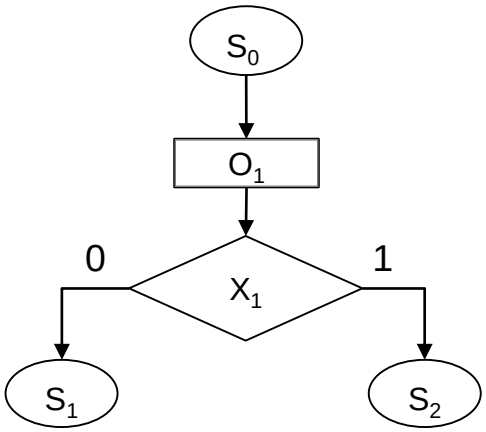
Proiectare Logică

Curs 16: **Structuri microprogramate**

Mariana Mocanu mariana.mocanu@upb.ro: 1 CB
Costin Chiru costin.chiru@upb.ro: 1 CA & CD
Anca Morar anca.morar@upb.ro: 1 CC

Microinstrucțiuni cu format fix (1)

- Informații asociate unei stări:
  - adresa stării următoare;
  - variabila de intrare care condiționează tranzacția;
  - ieșirile activate

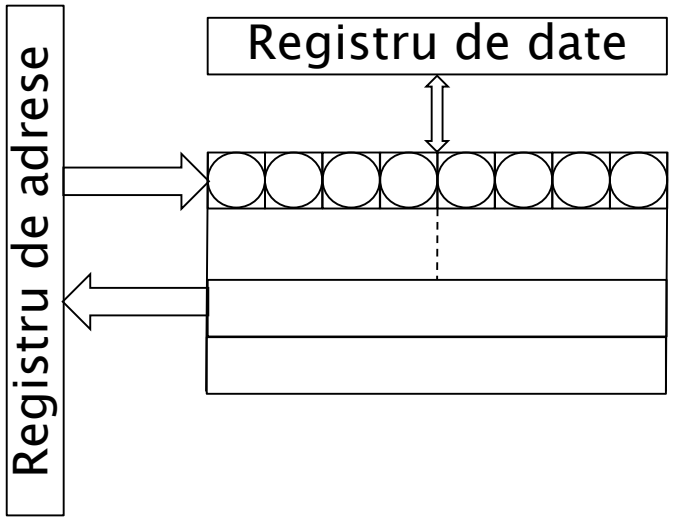


- Structura microinstrucțiunii:

Input Code	ADR 0	ADR 1	OUT

- $L_{\mu i} = n_{ci} + 2 * n_{ADR} + n_{OUT}$

- RA selectează adresa;
- RD conține datele scrise/citite în/din memorie



Microinstrucțiuni cu format fix (2)

Input Code	ADR 0	ADR 1	OUT

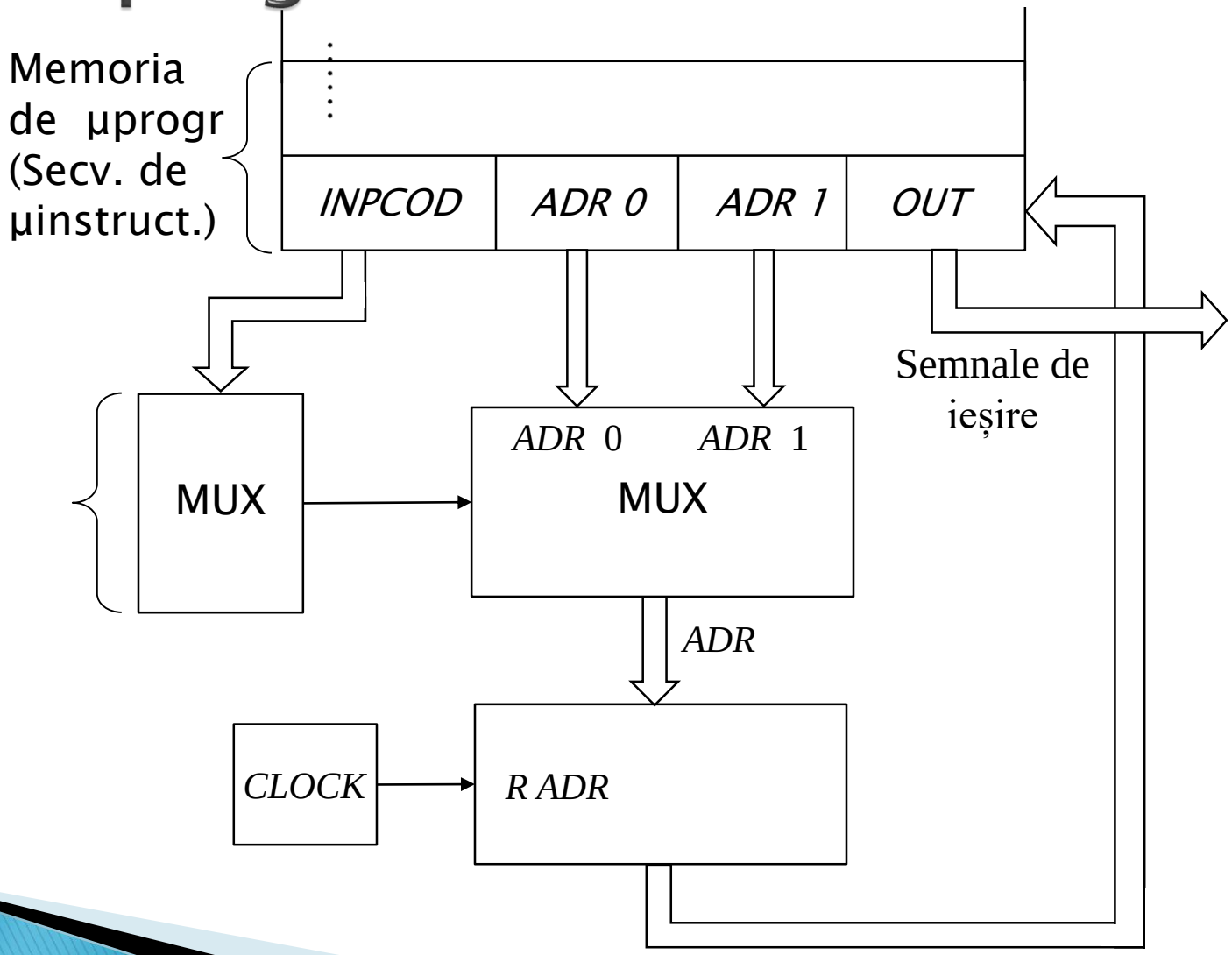
- Cele n intrări sunt codificate, iar codul lor e folosit pentru a stabili care dintre intrări declanșează tranziția
- ADR0, ADR1 – adresele pe 0/1 = locul unde se „sare” cu fluxul de control al aparatului
- OUT = ieșirile, nu se codifică → numărul de ieșiri = numărul de biți OUT necesari
- $L_{\mu i}$ este suma numărului de biți necesari pentru fiecare câmp în parte

Microinstrucțiuni cu format fix (3)

Avem 2 situații posibile:

1. Când tipul de memorie este impus
   - numărul de biți pentru RA se determină în funcție de tipul de memorie folosit.
   - lungimea microinstrucțiunii nu depinde de tipul de memorie (se poate obține prin concatenarea mai multor circuite)
2. Pornind de la organigramă → numărul de stări → numărul de variabile de stare → numărul de biți al RA.
   - dacă numărul de stări este mai mic decât capacitatea memoriei (numărul de cuvinte) atunci rămân zone nefolosite;
   - stările următoare nu trebuie să fie adiacente ca la implementarea automatelor cu CBB-uri!

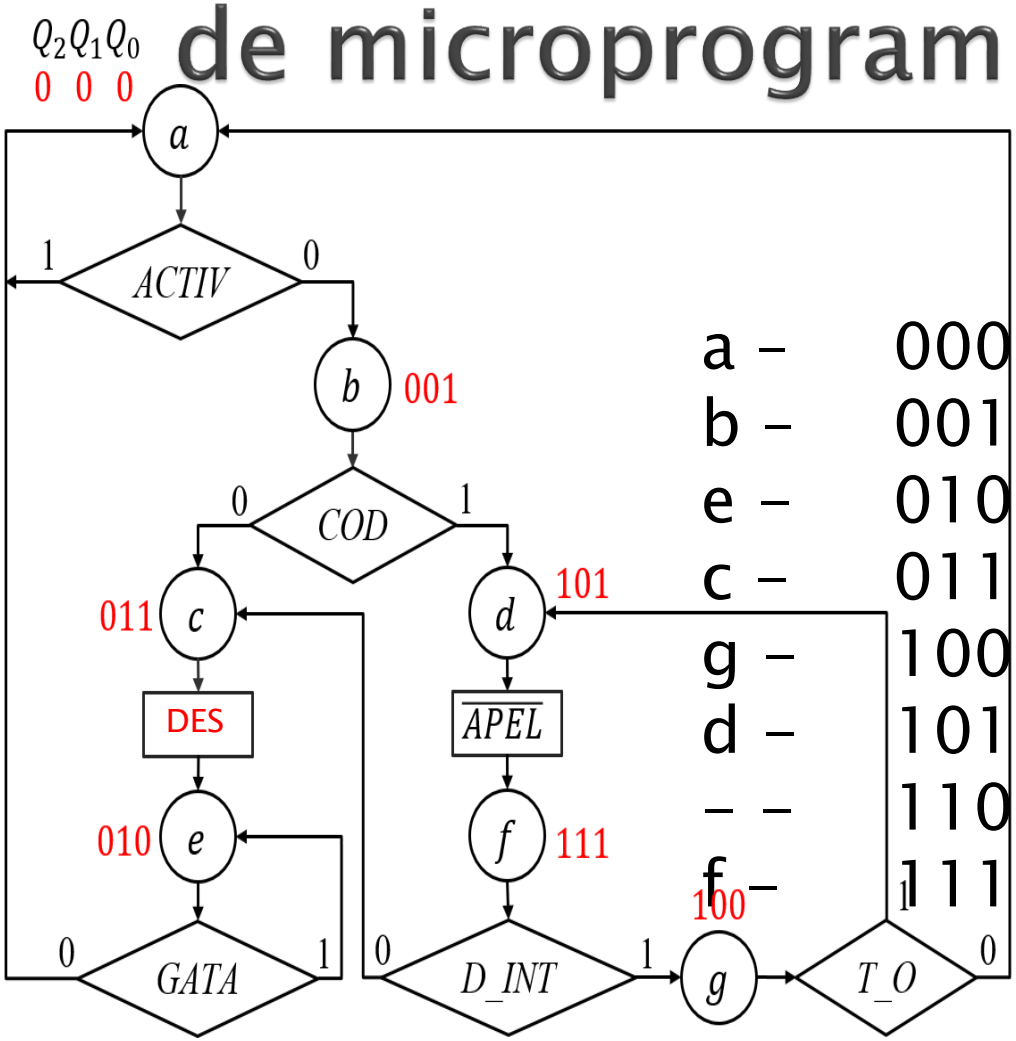
Structura de comandă microprogramată



Dezavantaj microinstrucțiuni cu format fix

- Lungimea microinstrucțiunii poate să devină prea mare
  - Interfon: 7 stări, $L_{\mu i} = 11$ → dacă avem circuite de memorare pe 4 biți avem nevoie de 3, dacă sunt pe 8 biți avem nevoie de 2
- Câmpul destinat ieșirilor conține puțină informație relevantă:
  - În exemplul nostru, informația utilă apare doar in 2 stări: când !APEL devine 0 și când DES devine 1

Problema cu interfonul – memoria de microprogram (3)



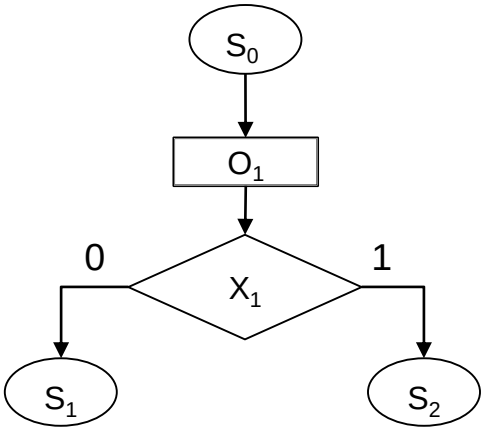
a – 000
b – 001
e – 010
c – 011
g – 100
d – 101
– – 110
f – 111

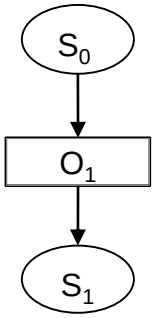
ACTIV – 001, COD – 010, GATA – 011, D-INT – 100, T_0 –101

INPCOD	ADR0	ADR1	DES	!APEL
001	001	000	0	1
010	011	101	0	1
011	000	010	0	1
000	010	010	1	1
101	000	101	0	1
000	111	111	0	0
xxx	xxx	xxx	x	x
100	011	100	0	1

Microinstrucțiuni cu format variabil

- În cazurile semnalate se pierde spațiul de memorie fie prin specificarea de ieșiri inutile, fie prin dublarea adresei următoare → încercăm să optimizăm memoria folosită prin folosirea microinstrucțiunilor cu format variabil
- Există 2 tipuri de structuri în cazul microinstrucțiunilor cu format variabil:



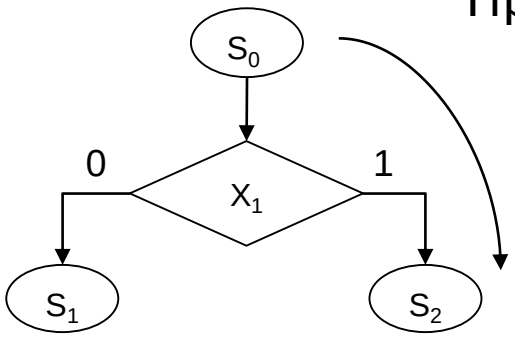


Tip 1

adr S1 = adr S0 +1

Se generează doar ieșiri
Saltul este necondiționat

Tip 0



adr S_2 = adr S_0 + 1

Doar decizie
Nu avem ieșiri

Microinstrucțiuni cu format variabil

- Dacă avem o structură de **Tip 1**, avem nevoie doar de adresa următoare și de ieșiri, nu de 2 adrese + inpcod + ieșiri
  - corespunde saltului necondiționat
- Dacă avem o structură de **Tip 0**, atunci avem nevoie de 2 adrese și de inpcod, dar nu mai avem ieșiri
  - corespunde saltului condiționat de valoare unei intrări
- În plus, dacă pentru RA al memoriei folosim un numărător, atunci putem implementa trecerea la adresa următoare consecutivă prin incrementare:
  - la Tip 1: adr S_1 = adr S_0 + 1 și ne interesează doar ieșirile,
  - la Tip 0: adr S_3 = adr S_1 + 1 și ne interesează doar intrarea (inpcod) și ADR_0
- Convenție: la microinstrucțiuni de Tip 0, când intrarea este 1, adresa următoare este adresa curentă + 1

Microinstrucțiuni cu format variabil

Structura microinstrucțiunii este diferită pentru cele 2 tipuri
Pentru identificarea acestora se introduce un bit care indică tipul:

- pentru salt necondiționat = 1 (tip 1)
- pentru salt condiționat = 0 (tip 0)

1	ieșiri

- $L_{\mu i\ Tip\ 1} = 1 + n_{out}$
- 1 este bitul pentru tipul instrucțiunii

0	INPCOD	ADR0

- $L_{\mu i\ Tip\ 0} = 1 + n_{ci} + n_{adr0}$
- De cele mai multe ori microinstrucțiunile de Tip 1 și Tip 0 vor avea lungimi diferite pentru că depind de logica programului (număr de stări, număr de ieșiri, număr de variabile de intrare)
- Pentru a determina $L_{\mu i}$ se va considera maximul dintre cele 2 valori:

$$L_{\mu i} = \max\left(L_{\mu i\ Tip\ 1}, L_{\mu i\ Tip\ 0}\right)$$

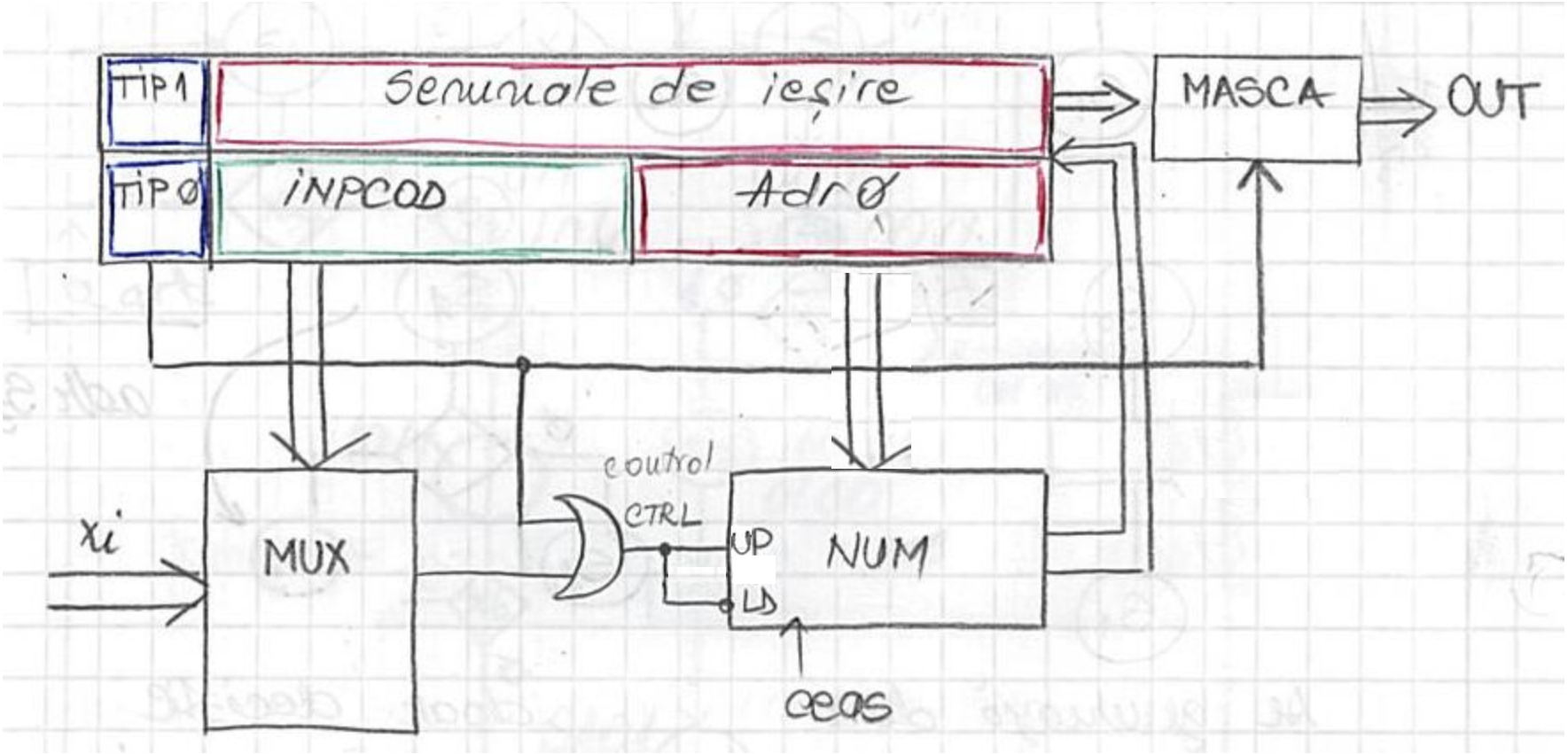
Microinstrucțiuni cu format variabil

Exemplu

- Dacă avem: adrese pe 4 biți, 4 biți pentru codul variabilelor de intrare și 30 ieșiri:
  - $L_{\mu i\ Tip\ 1} = 1 + 30 = 31$
  - $L_{\mu i\ Tip\ 0} = 1 + 4 + 4 = 9$
  - $L_{\mu i} = \max(L_{\mu i\ Tip\ 1}, L_{\mu i\ Tip\ 0}) = 31$

- Pentru a implementa dispozitivele de comandă cu microinstrucțiuni cu format variabil:
  1. Memoria trebuie să știe să diferențieze microinstrucțiunile de tip 1 și 0 între ele (se modifică unitatea de comandă) și
  2. Trebuie modificată organigrama pentru a avea numai instrucțiuni de tip 1 și 0 prin adăugare de stări.

Structura de comandă microprogramată



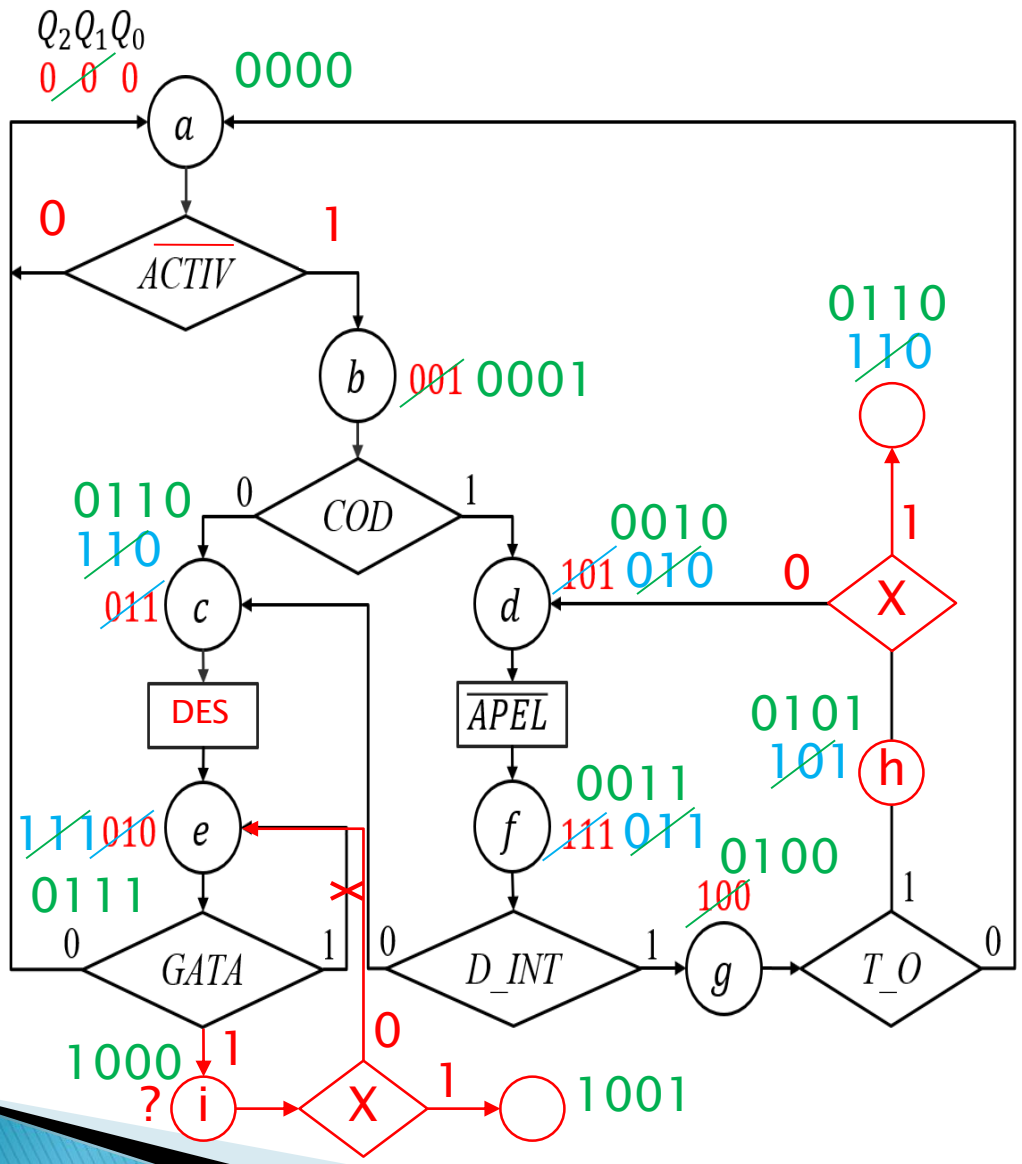
Structura de comandă microprogramată

- Trebuie să ne asigurăm că putem trece de la adresa curentă la adresa incrementată, dar și că se poate face salt la o altă adresă dacă avem instrucțiune de tip 0 și intrarea este 0 → se folosește un numărător universal pentru că permite incrementarea, precum și presetarea (folosind LOAD) unei anumite valori și continuarea incrementării de la acel punct
- Acest numărător joacă rolul RA

Structura de comandă microprogramată

- Dacă Tip = 1, Ctrl = 1
  - → adresa curentă se incrementează (Up = 1)
  - → se activează masca pentru a putea citi ieșirile (acestea se citesc direct)
- Dacă tip = 0
  - → se dezactivează masca pentru a nu putea citi ieșirile
  - Dacă intrarea este = 1, Ctrl = 1 și adresa curentă se incrementează (Up = 1)
  - Dacă intrarea este = 0, Ctrl = 0, Load se activează și se încarcă ADR0 în numărător

Modificarea organigramei



- *Pas 1*: **transformăm organigrama** astfel încât să avem doar instrucțiuni de tip 1 sau 0
- *Pas 2*: **recodificăm stările** a.î.
  - la μi-Tip 1:

adresa următoare = adresa curentă+1

  - la μi-Tip 0:

adresa următoare = adresa curentă+1 când variabila de intrare = 1

X=0: variabilă de intrare suplimentară cu valoare = 0 poate duce la creșterea n_{ci}

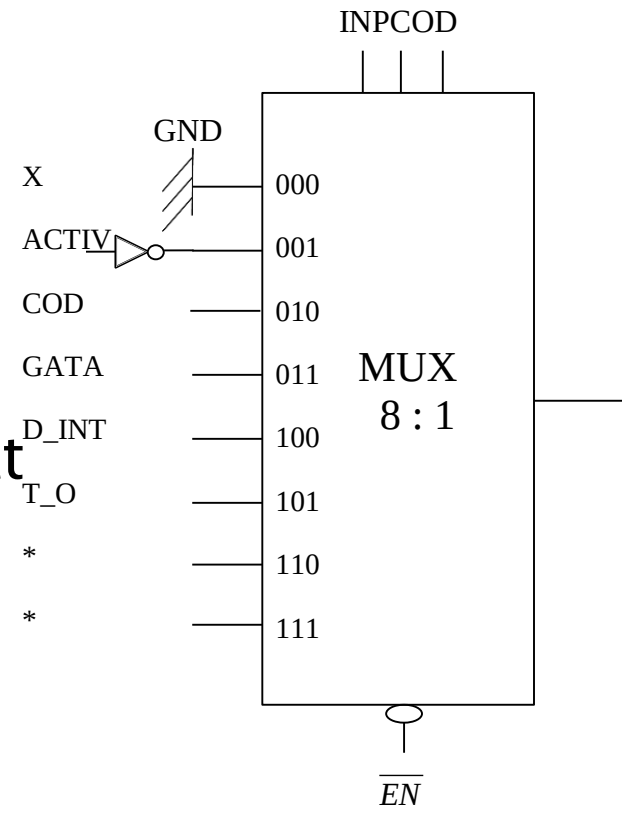
Modificarea organigramei (2)

- Din starea g nu poate efectua o tranziție pe 1 și atunci se introduce o stare suplimentară și o variabilă de intrare legată permanent la masă (=0).
  - astfel, tranziția pe 0 asigură tranziția la adresa dorită
  - tranziția pe 1 duce către o stare unde nu se ajunge niciodată.
- Observații:
  1. codul stărilor unde nu se ajunge niciodată poate fi refolosit
  2. variabila de intrare legată la masă poate și ea fi refolosită de oricâte ori este nevoie pentru astfel de tranziții

Modificarea organigramei

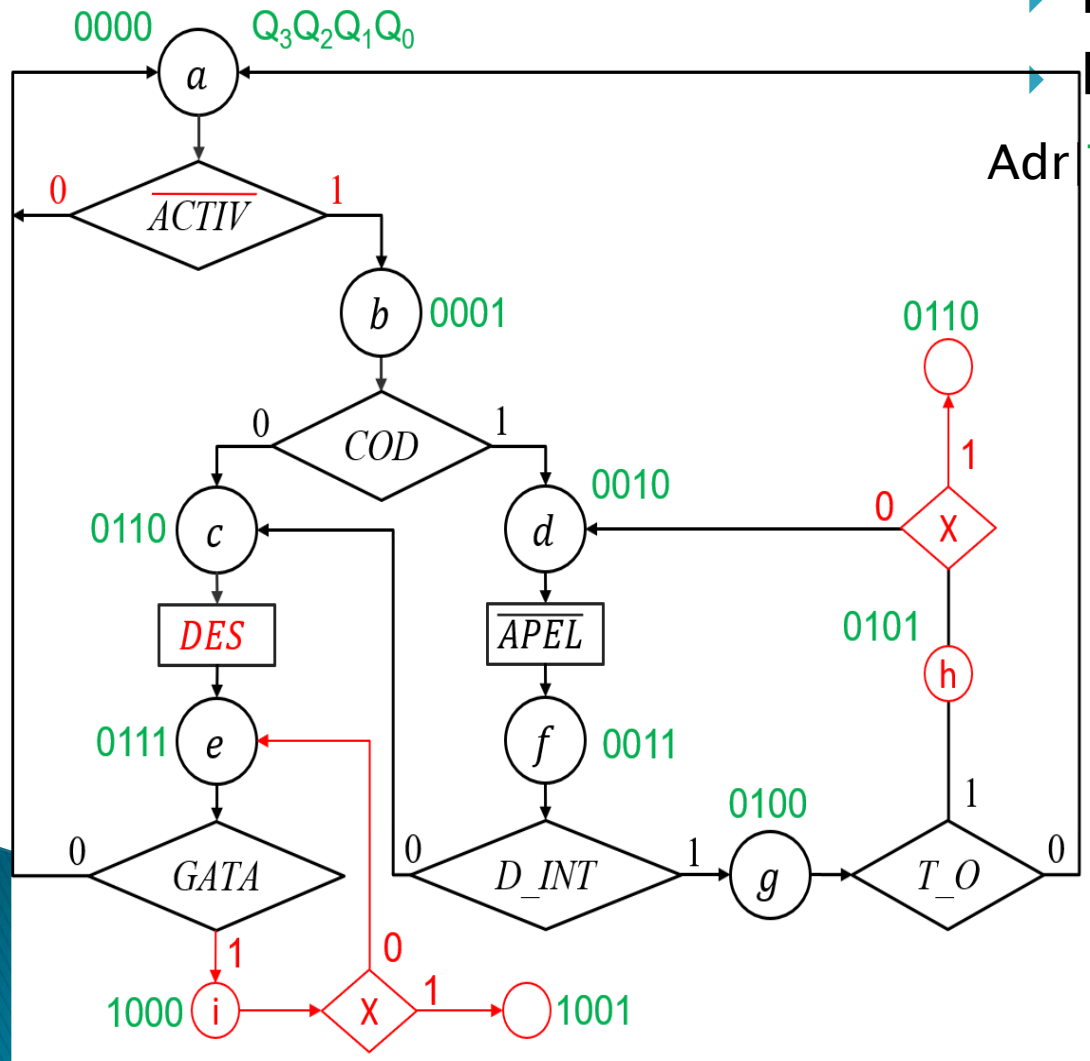
Ce înseamnă modificarea ACTIV în !ACTIV?

- În unitatea de comandă, pe intrarea MUX se conectează !ACTIV în loc de ACTIV
- Pentru !ACTIV se poate folosi același cod desemnat inițial pentru ACTIV doar dacă în organigramă nu apar atât ACTIV cât și !ACTIV, în diferite stări.
- ACTIV și !ACTIV negat sunt tratate ca variabile independente
- Daca în organigamă avem nevoie atât de ACTIV cât și de !ACTIV se introduce o nouă variabilă de intrare Z=!ACTIV căreia i se alocă un cod diferit



Memoria de microprogram

ACTIV – 001, COD – 010, GATA – 011,
D-INT – 100, T_0 –101



- $L_{\mu i \text{ tip } 1} = 1 + 2 = 3$
- $L_{\mu i \text{ tip } 0} = 1 + 3 + 4 = 8$
- $L_{\mu i} = \max(L_{\mu i \text{ tip } 1}, L_{\mu i \text{ tip } 0}) = 8$

Adr | TIP | DES + !APEL (COD + ADR0)

a – 0000	0 | 001 0000
b – 0001	0 | 010 0110
d – 0010	1 | 00x xxxx
f – 0011	0 | 100 0110
g – 0100	0 | 101 0000
h – 0101	0 | 000 0010
c – 0110	1 | 11x xxxx
e – 0111	0 | 011 0000
i – 1000	0 | 000 0111
* – 1001	x | xxx xxxx

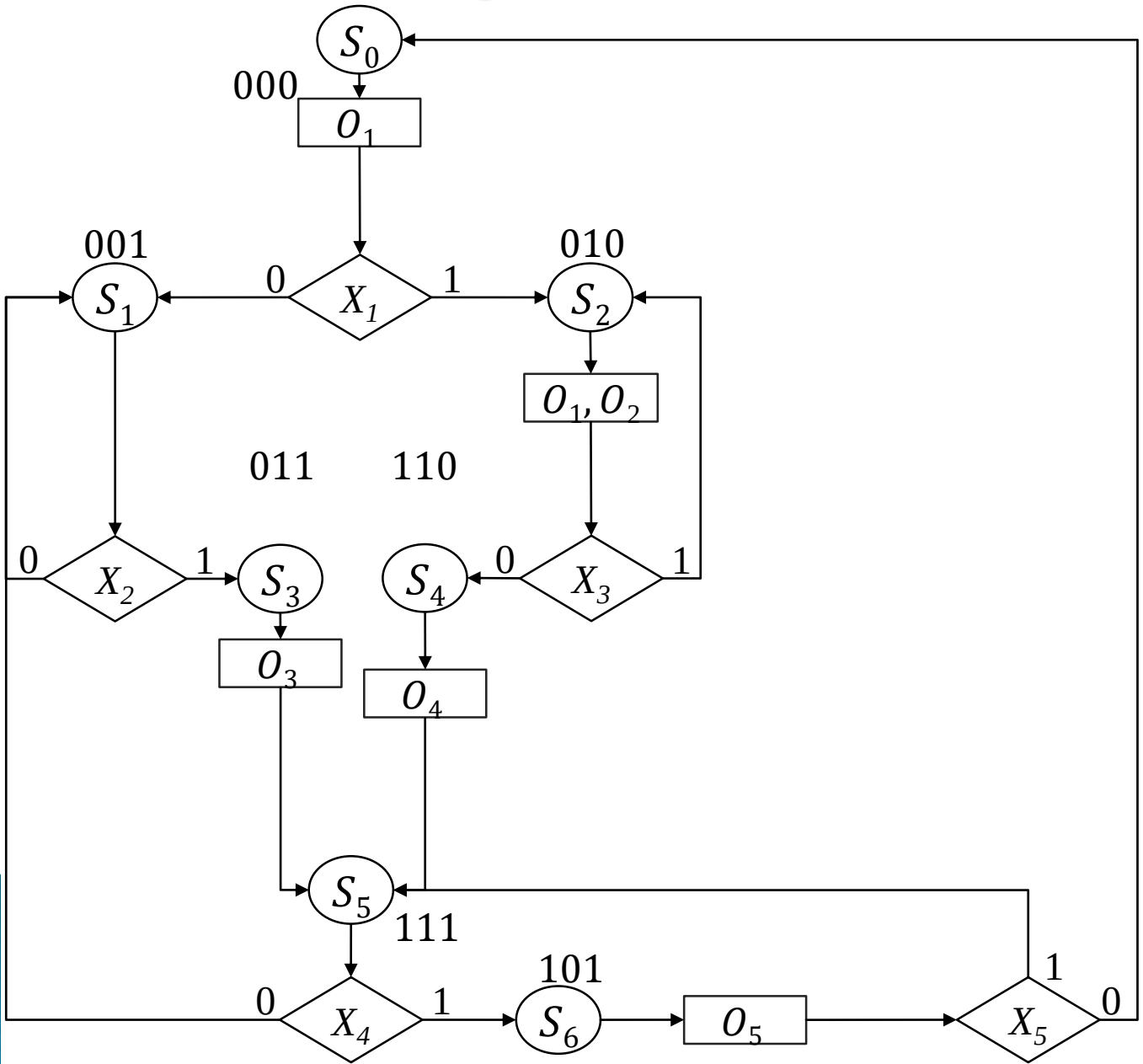
Memoria de microprogram

- Având în vedere că de cele mai multe ori cele 2 tipuri de microinstrucțiuni au dimensiuni diferite, trebuie ca una dintre ele să se completeze cu biți cu valoare indiferentă (x). Acest lucru se poate face fie la MSB, fie la LSB
  Memorie consumată:
  - Format fix: 7 stări *11 biți
  - Format variabil: 9 stări * 8 biți

  Tipul de microinstrucțiune folosit va influența alegerea tipului de memorie utilizat pentru implementare

Alt exemplu

- Care e lungimea microinstrucțiunii cu format fix? $L_{\mu i} = 3 + 2 * 3 + 5 = 14$ biți



- Nr. intrări = 5
- n_{ci} =3
- Nr. stări = 7
- Nr. variabile de stare = 3
- Nr. ieșiri = 5
- n_{OUT} = 5

Codificare intrări

X_1 – 001

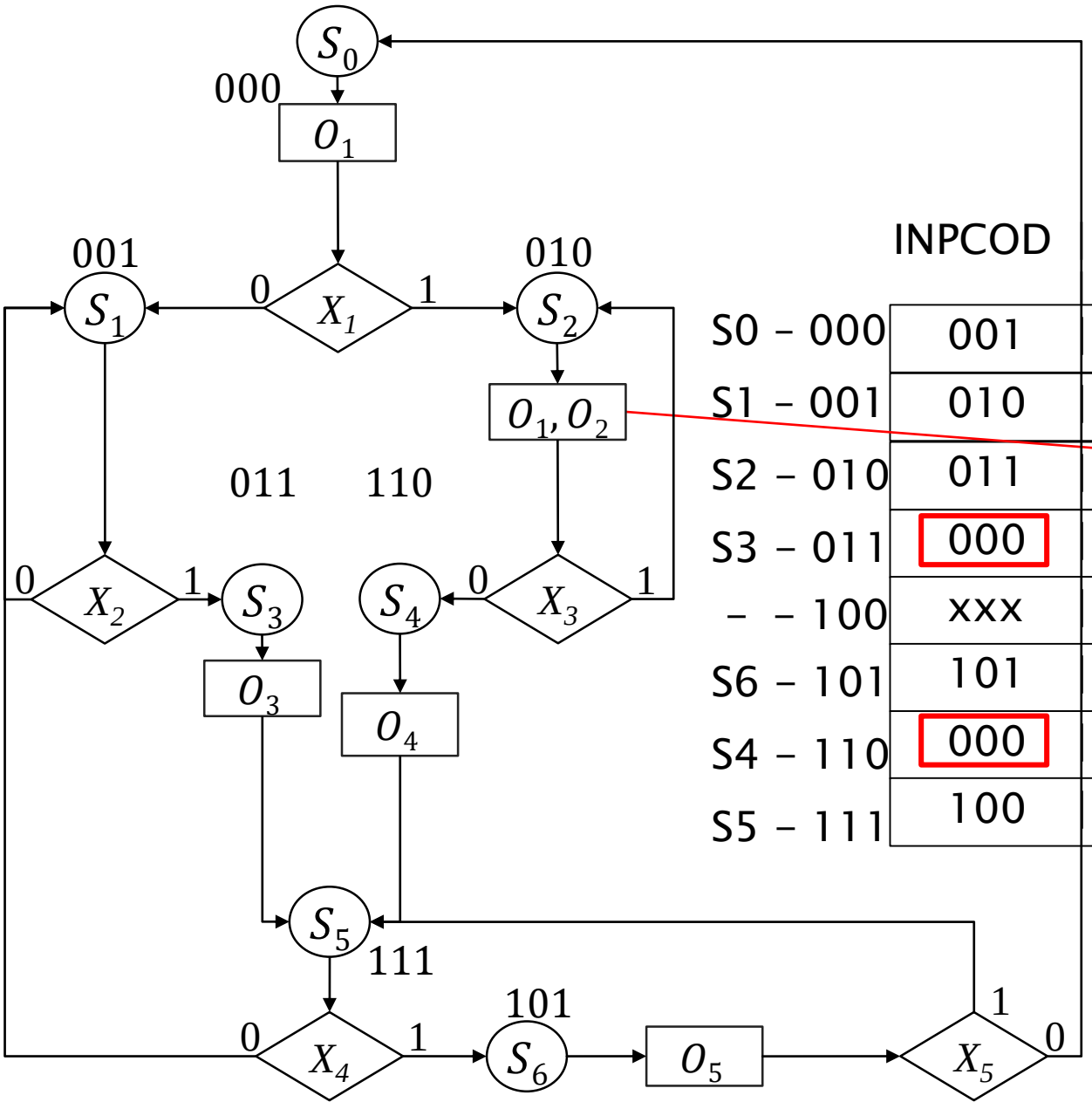
X_2 – 010

X_3 – 011

X_4 – 100

X_5 – 101

Format Fix – Memoria de microprogram

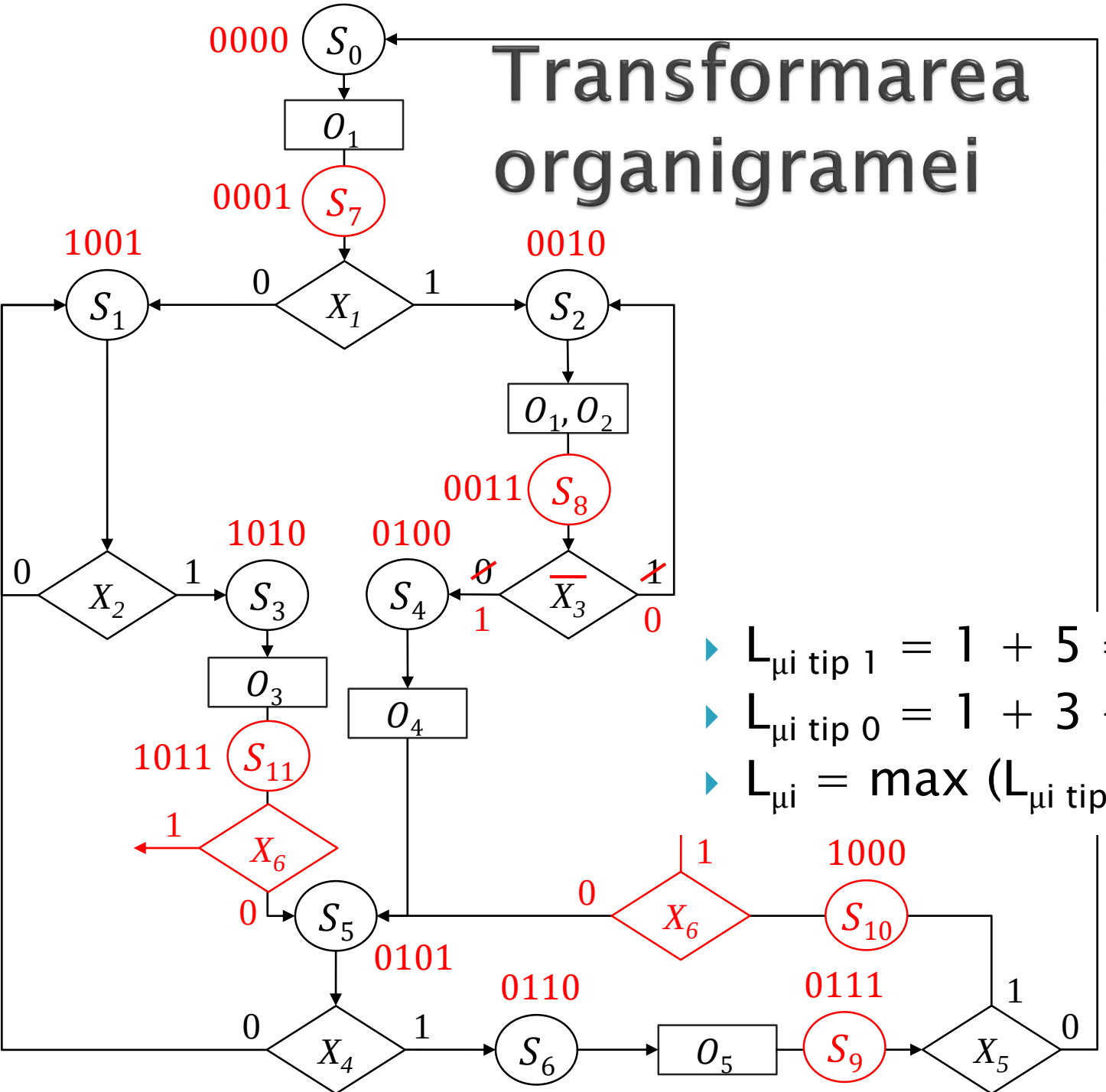


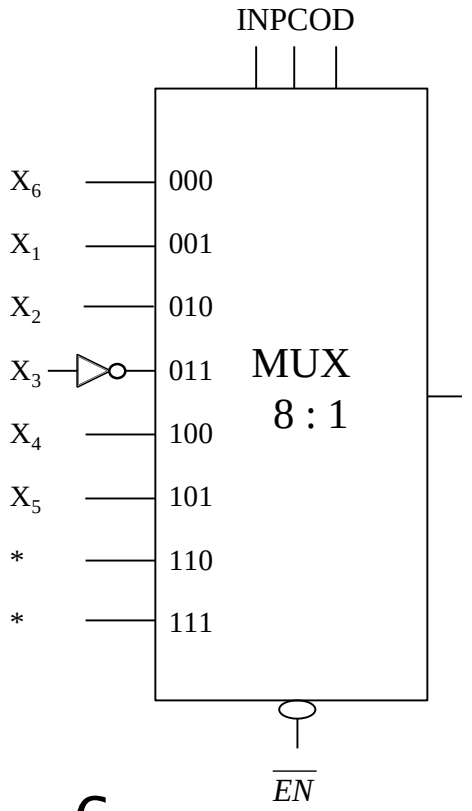
Codificare intrări
X_1 – 001, X_2 – 010,
X_3 – 011, X_4 – 100,
X_5 – 101

	INPCOD	ADR0	ADR1	O5	O4	O3	O2	O1
S0 – 000	001	001	010	0	0	0	0	1
S1 – 001	010	001	011	0	0	0	0	0
S2 – 010	011	110	010	0	0	0	1	1
S3 – 011	000	111	111	0	0	1	0	0
– – 100	xxx	xxx	xxx	x	x	x	x	x
S6 – 101	101	000	111	1	0	0	0	0
S4 – 110	000	111	111	0	1	0	0	0
S5 – 111	100	001	101	0	0	0	0	0

Folosim chiar adresele din organigramă → avem nevoie de memorie de 8 cuvinte

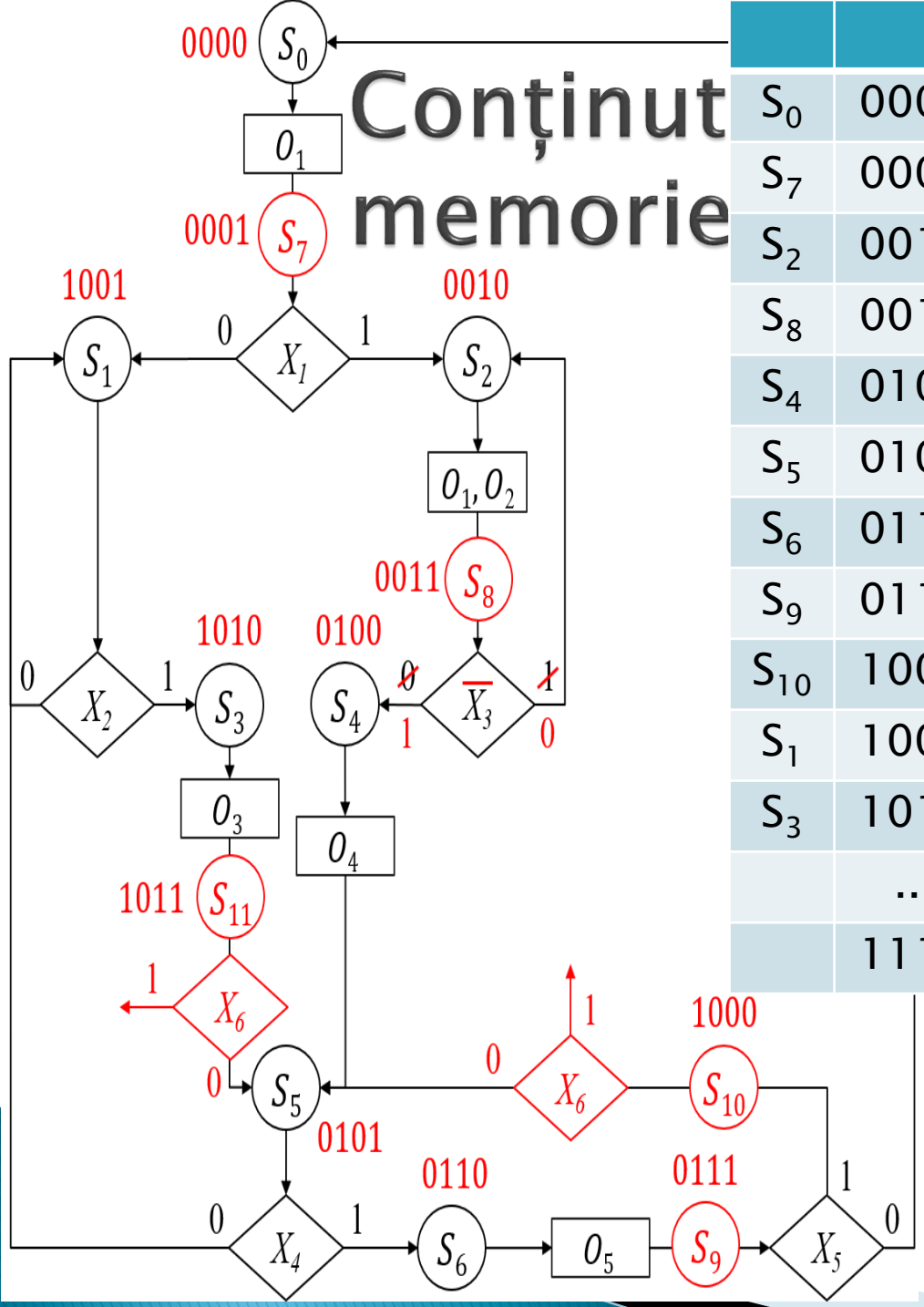
Transformarea organigramei





- $L_{\mu i \text{ tip } 1} = 1 + 5 = 6$
- $L_{\mu i \text{ tip } 0} = 1 + 3 + 4 = 8$
- $L_{\mu i} = \max(L_{\mu i \text{ tip } 1}, L_{\mu i \text{ tip } 0}) = 8$

Conținut memorie



		Tip	O_5	O_4	O_3	O_2	O_1	–	–
S_0	0000								
S_7	0001								
S_2	0010								
S_8	0011								
S_4	0100								
S_5	0101								
S_6	0110								
S_9	0111								
S_{10}	1000								
S_1	1001								
S_3	1010								
	...								
	1111								

Codificare intrări:

X_1 – 001, X_2 – 010,

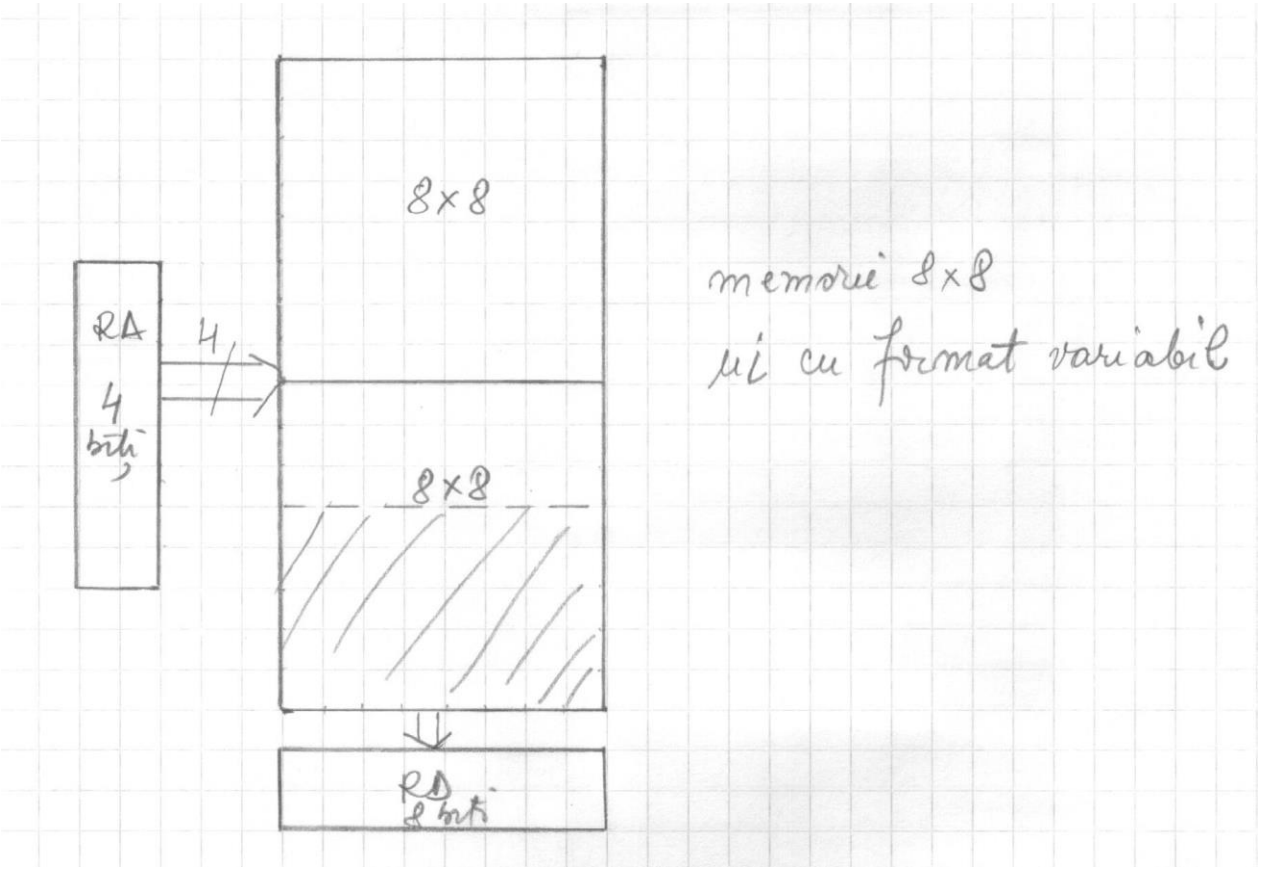
!X_3 – 011, X_4 – 100,

X_5 – 101, X_6 – 000

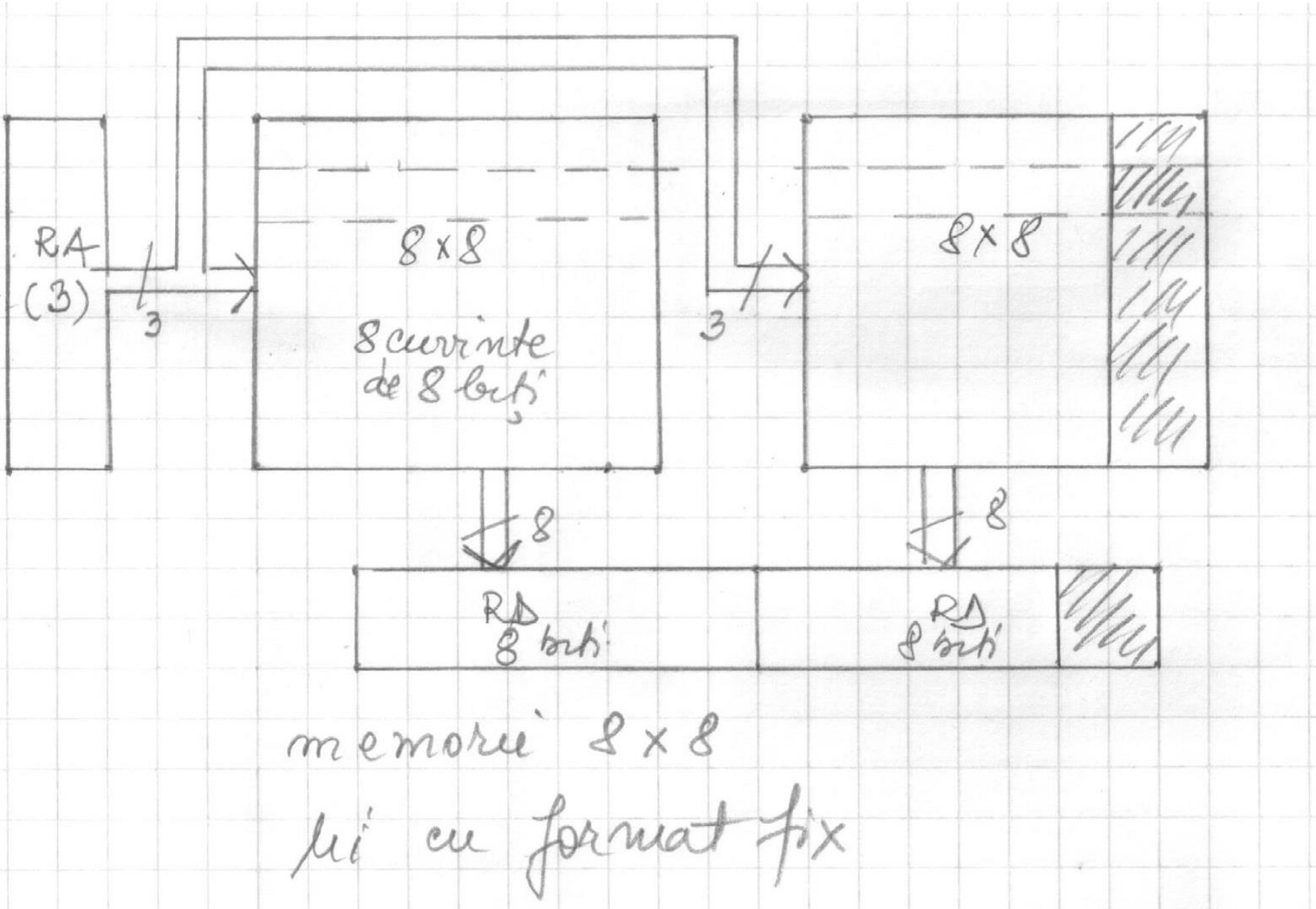
Memoria de microprogram

- Format fix: 7 stări * 14 biți
- Format variabil: 11 stări * 8 biți

Tip memorie	Număr chip-uri pentru µi cu format fix	Număr chip-uri pentru µi cu format variabil
8 x 8	2	2
8 x 16	1	2
16 x 8	2	1
16 x 16	1	1
…		

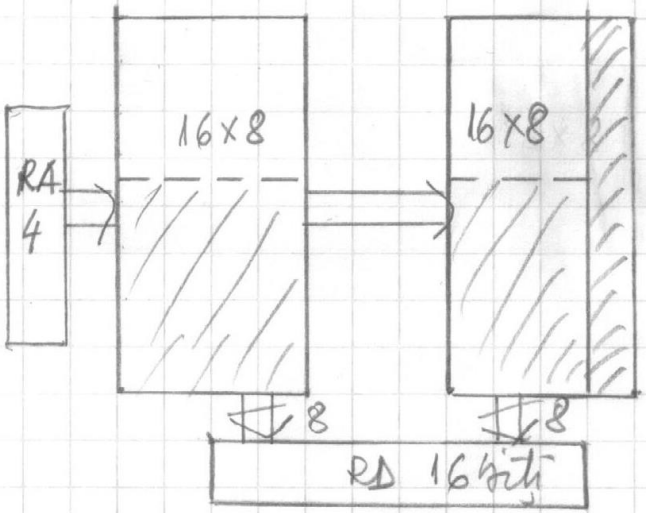


memorii 8×8
ii cu format variabil

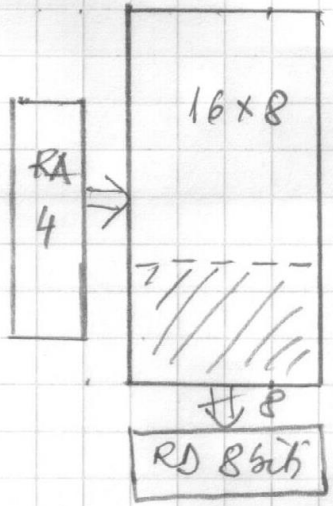


memorii 8 x 8

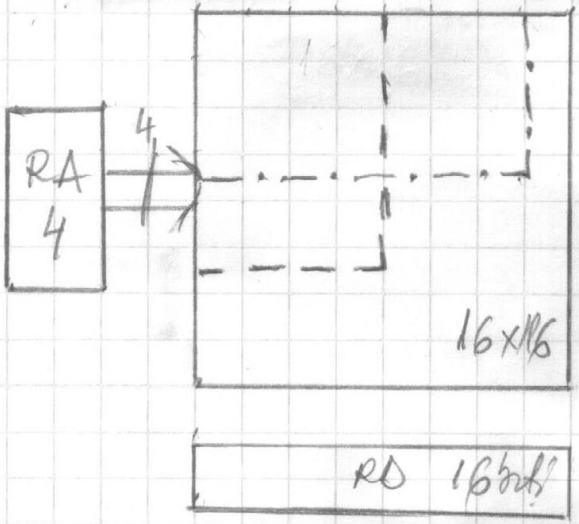
lți cu format fix



cu format fix



cu format variabil



ÎNTREBĂRI?