

AC - C1

francis.jacob@cs.pub.ro

Birou: EF 202

Bibliografie:

Obiecte:

10<sup>p</sup> testi de lab (10 x 1<sup>p</sup>)  
 10<sup>p</sup> tema 1  
 15<sup>p</sup> tema 2  
 15<sup>p</sup> calorii final  
 6<sup>p</sup> Bonus

} minimum 25<sup>p</sup>

Examen: 50<sup>p</sup> minimum 25<sup>p</sup>

Acum suntem:

O să avem numai probleme anul acesta (examen online)

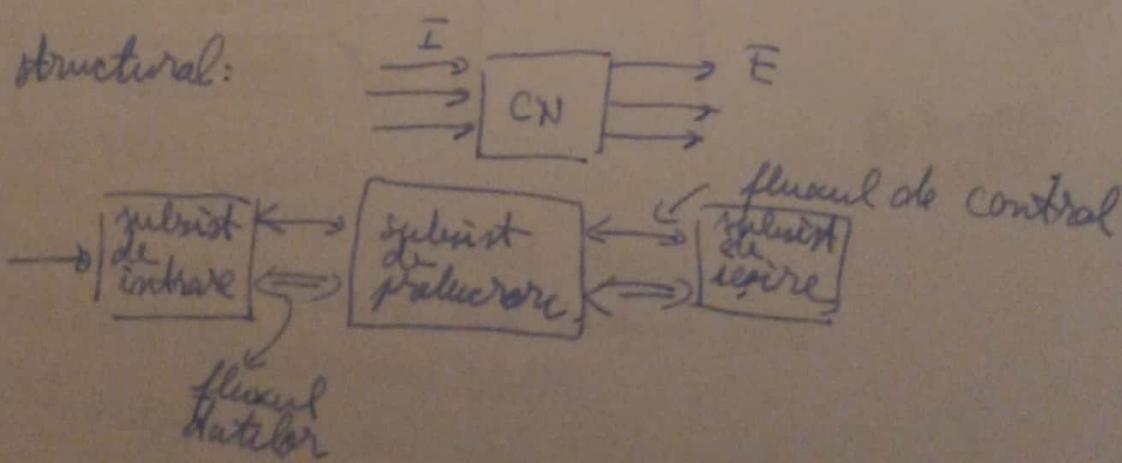
Introducere

Model struktură

Aplicații specifice:

- calcule științifice
- simulare
- instruire vizuală
- conducere de proces

Model structural:



In prezentare găsești modele detaliate

DMA = direct memory access

## Unități funcționale

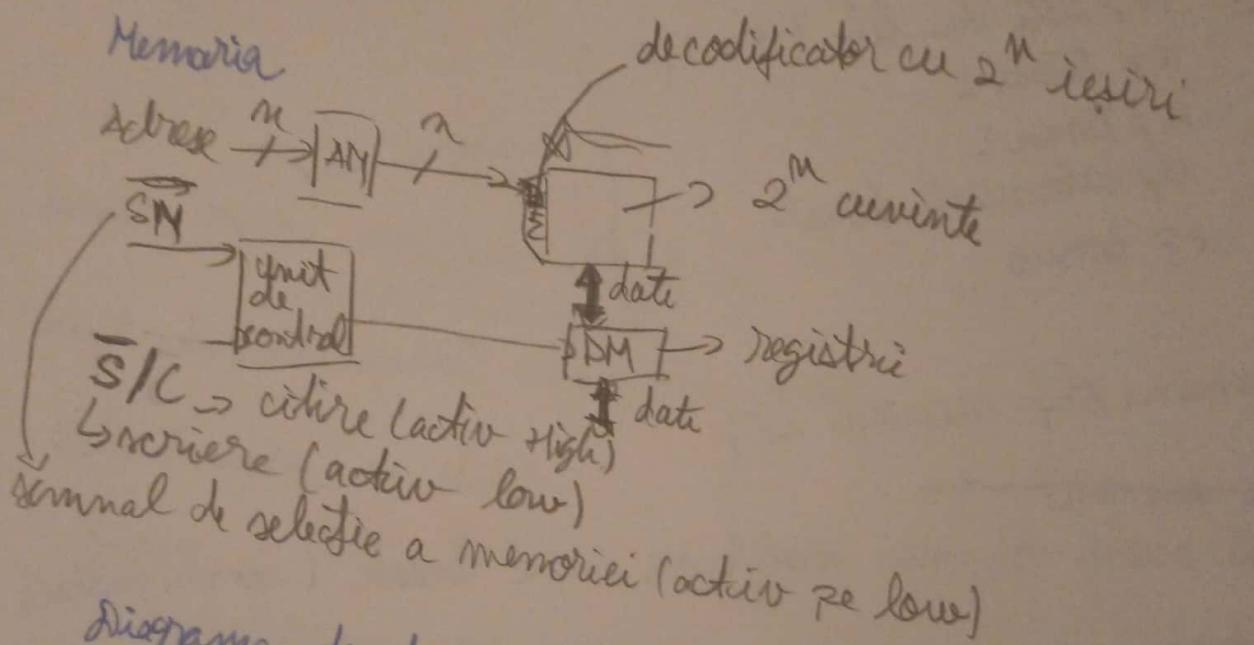


Diagrama de timp (citire)

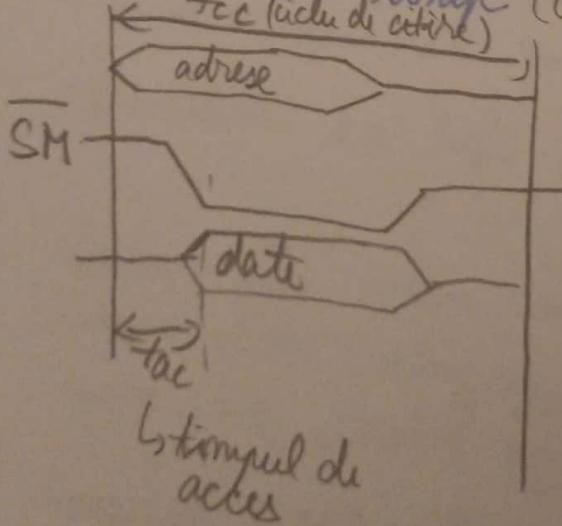
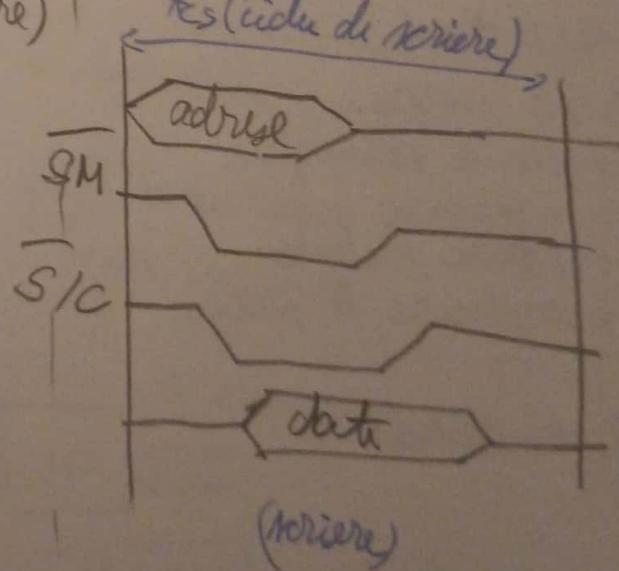


Diagrama de timp (scriere)



# (semnif negativ)

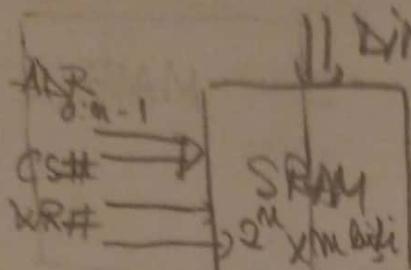
CS - chip select

WR# (write, citire)  
+ = citire

Ecrante de memorie

SRAM (permite read/write)

RAM static



DIN<sub>0:m-1</sub>

m - biti de adresa

DOUT<sub>0:m-1</sub> - datele de intrare pentru leiti

Avantaj SRAM:

- interfata simpla
- sunt mai rapide
- sunt utilizate pentru memoria cache
- ciclu de citire (pe slide)

DRAM

- suprafață ocupată mai mică ⇒ densitate mai mare
- fiind o capacitate reală, se degradă în timp
- furnizare 2<sup>n</sup> biti ⇒  $2^{2n} \times m$  (dim matrice)

nr. curint

RAS# =  $\overline{RAS}$  (row acces)

CAS# =  $\overline{CAS}$  (column acces)

WR# =  $\overline{WR}$  → activ pe 0

Tipuri de DRAM:

FPM DRAM (Fast page mode DRAM)

EDO DRAM (Extended data output DRAM)

SDRAM (Synchroous DRAM)

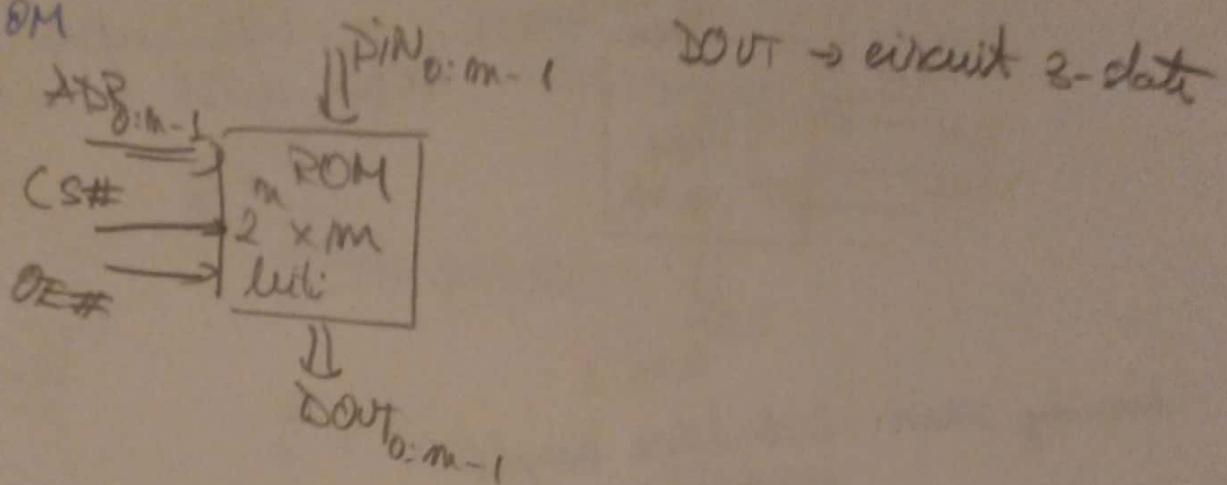
DDR SDRAM (Double data rate SDRAM)

Pracute:

SIMM Single In-line Memory Module

DIMM Dual In-line Memory Module

Memorie ROM

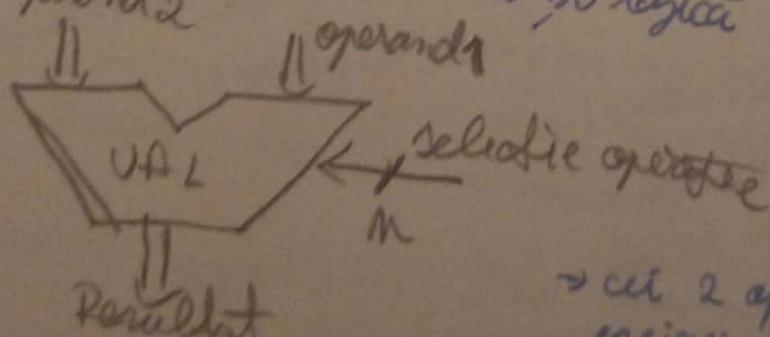


Memoriile ROM sunt de tip combinational

Tipuri de ROM:

ROM  
PROM → programabilă  
EPROM → erasabile PROM  
EAPROM → electrical erasabile PROM  
memoria flash

UAL = unitate aritmetică și logică



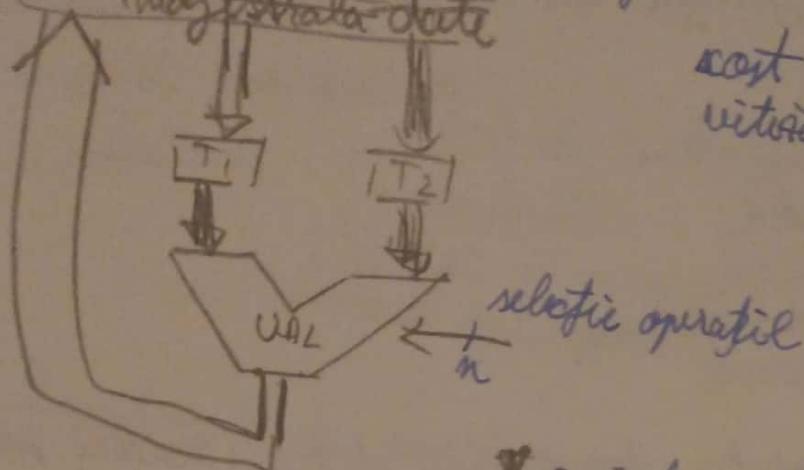
→ cel 2 operațiuni sunt pe același lungime de la baza

Soluții de bază de interconectare

→ UAL conectat cu o singură magistrală de date

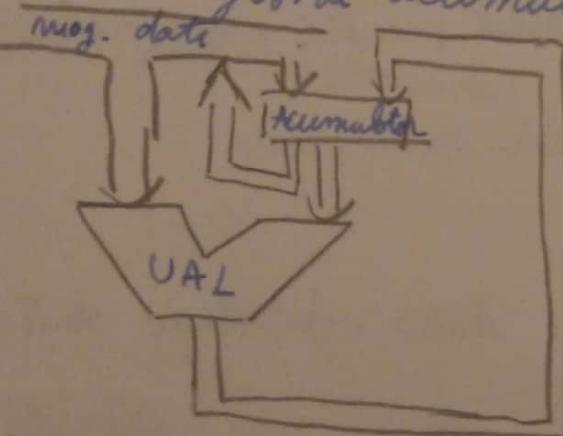
magistrala date

cost redus  
viteză redusă



\* 3 cicluri de memorie

→ UAL cu registru acumulator



→ rez final este furnizat la final din acumulator în magistrală

- Etype:
1. Luăm un operand și prin mag. apunge în reg. acumulator
  2. — " — " din memorie
  3. Dacem rez. în acumulator

→ UAL cu 3 magistrale (ea mai performantă și costisitoare)  
durata medie a unei operații este de 4 ciclu de clasa

- \* Unitatea de comandă
  - conținut de program
  - (IP) CP → indică unde se găsește instrucțiunea următoare
  - de refacere
  - instrucție pointer

Fazele instrucțiunii:

- fetch
- decode
- fetch operand
- execute

Tipuri de instrucțiuni:

- pentru transferul datelor (registrii, porturi)
- aritmétice (operări aritmétice)
- logice și de deplasare
- pentru controlul execuției programului

## 2. Descrierea structurii sistemelor numerice

### Nivelul elementelor de circuit

componente = Z, L, C, transisori

structură: scheme, ecuații

ASM = algorithmic state machine

### Nivelul proiectării logice

a) subnivelul șematelor de combinație

componență: porturi logică, MUX, Bufferi, registre

structură: scheme, ASM

b) subnivelul transf între registre / unit. de execuție

c) subnivelul transf între registre / unit. de comandă

unitate de execuție - componentă: registru, memorie  
structură

unitate de comandă - componente: alg de comandă, ...  
structură

Nivelul interconectării resurselor

componenti: procesor, ...  
structură

nivelul programeelor

→ subnivele:   
    - setul de instrucții  $\hookrightarrow$  struct: IS<sub>7</sub> compl: moduri de adresașe  
    - limbajelor  
    - sist. de operare  
    - al aplicațiilor

Limbajul de descriere a

AHPL - A hardware programming language

Forma generală: procedură)

MODULE: < nume modul >  
        < declaratii >  
        < acrv. de control >  
    END SEQUENCE  
        < conexiuni >  
        < transferuri >

Strucția unei funcții: UNIT: < nume fct > (listă parametrii)  
                        < declaratii >  
                        < conexiuni >  
    END

În curs găsești mai multe detalii

Operanai:

- scalari: litere mici:  $x, y, z$  un bit
- vectori: majuscule:  $X, Y, Z$  strucția liniară pe 8 bits
- Matrice: majuscule valabile:  $X, Y, Z$

Operator de dimensiune vector:  $\rho$  <nume-vector>

$$\rho RA = 16 \quad (RA este pe \times lini)$$

$\rightarrow$  cel mai semnificativ bit

$$RA = (RA_0, RA_1, \dots, RA_{\rho RA - 1})$$

$RA_i$  - bitul de pe poza i a registrului RA

Matrice:  $S_1$ ,  $S_2$

$S_1 M$  - nr de linii

$S_2 M$  - nr de coloane

$$M = \begin{bmatrix} M_0^0 & M_1^0 & \dots & M_{S_2 M - 1}^0 \\ M_0^1 & M_1^1 & \dots & M_{S_2 M - 1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ M_{S_1 M - 1}^0 & M_{S_1 M - 1}^1 & \dots & M_{S_2 M - 1}^{S_1 M - 1} \end{bmatrix}$$

$M^i$  = linia i (cuvântul i)

$M_j^i$  - bitul j din cuvântul i

$M^{i:j}$  → blocul de linii i → j inclusiv

$A^{i:j}$  → listă de la i la j inclusiv

Operări logice:  $\wedge$ ,  $\vee$ ,  $\neg$  sau  $\bullet$ ,  $+$ ,  $-$  ( $\lambda$ , sau, nu)

$$\text{Fie } A = (A_0, A_1, \dots, A_n)$$

$$B = (B_0, B_1, \dots, B_n)$$

și 2 scări a, b

a  $\wedge$  b (a și b)

$$A + B = (A_0 + B_0, A_1 + B_1, \dots, A_n + B_n)$$

-> operatori de reducere:

-  $\wedge$ ,  $\vee$ ,  $\neg$  <sup>produsare</sup>  $\rightarrow$  se aplică întregului vector

$$1/A = A_0 \ 1 \ A_1 \ 1 \ \dots \ 1 \ A_n$$

-> operatori de sincronizare

SYN < nume - semnal - asincron > (meniu se aplică SYN)  
SL < nume - semnal, înainte de SL

SYN  $\rightarrow$  realizează sincronizarea semnalului specificat cu  
semnalul de ceaș al răsturnării

SL  $\rightarrow$  generează un semnal de o durată egală cu  
perioada semnalului de ceas indiferent de durata  
semnalului specificat ca parametru (nume - semnal)

Ex: SL(SYN Horlo)

Exercițiu: Proiectați dispozitive ce implementează  
cei doi operatori

-> operator de selecție

$$\{ i : 1 \}$$

-> operator de selecție prin comprimare

$X/Y$  unde în  $X$  avem 4 selectare din  $Y$

$$A = (0, 1, 1, 1, 0, 1, 0, 0, 1)$$

$$B = (B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7)$$

$$A/B = (B_1, B_2, B_4, B_7)$$

→ operator de selecție prin comprimare și reducere

$M * F \rightarrow$  vector de selecție

↳ selectăm din  $M$  linile ce corespund elementelor din  $F$  cu liniile de 1

$$M = \begin{bmatrix} M^0 \\ M^1 \\ M^2 \end{bmatrix} \quad F = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \Rightarrow M * F = M^2$$

operatori de concatenare

) → pe linii

! - pe coloane

a, b 2 scalari

A, B 2 vectori

M o matrice

$$N = A ! B$$

$$A = a, b$$

operator de codificare ( $T$ )

$n T m$  (repräsentare  $m$  pe  $n$  linii)

$$4T5 = (0, 1, 0, 1)$$

$$8T7 \Rightarrow (0, 0, 0, 1, 0, 0, 0)$$

operatorul de decodificare ( $L$ )

$$L(0, 1, 0, 1) \Rightarrow 5$$

$$DCD(0, 0, 0, 0, 1, 0, 1) \Rightarrow 5$$

operatori de atribuire

a) de conexiune = [realizarea conectarea resurselor]

b) de transfer ← [realizare transf de la surse la destinație]  
(Destinație trebuie să fie în stăt. cu memorare)

$a = b$  conexiune b la a

$z = b$  conexiune restabilire a la b

$b \leftarrow a$  continutul din rezistorul 4 la ieșirea 2 al modulului  
⑩ rezistorul b se transferă în rezistorul 3

operator de dimensionare :  $\varnothing$ ,  $\varnothing_L$ ,  $\varnothing_P$   
operatori relationali (R)  $\times_R$   $\gamma$   $R \in \{ \leq, \geq, =, \neq, \neq \}$

Instructiuni in AHPL

Instructiuni de atribuire

< destinatie> < operator de atrib.> < expresie>

Pentru expresii putem utiliza operatori

IV Indicii trebuie sa fie valori constante  
Putem utiliza paranteze  
Prioritare.

- 1<sup>o</sup>. Negatie și syn
- 2<sup>o</sup> : op. de selecție

Instr. de conexiune

$$\top = (f, g, h) \quad MLCO = (A \mid B \mid C)$$

$$MLCO * F = (A \wedge f) \vee (B \wedge g) \vee (C \wedge h)$$

10. BUS = (A/B) \* (f<sub>1</sub>, f<sub>2</sub>)

## AC - C2

→ Încercați să rezolvați la lab problemele  
contact: Alexandru Petrescu

## Instrucțiunile de transfer

$VD \leftarrow VLCO$  → vector de fit logice  
 $VD \leftarrow MLCO * F$  →  $VLCO$  este încărcat în registrul  $VD$   
 $M = [A \ B \ C]$  → vector de selecție, în general are doar un element egal cu 1  
 $MD * F \leftarrow VLCO$  → se încarcă în linia selectată de  $F$  vectorul  $VLCO$   
Ex:  $MLCO = (A!B!C)$  → concatenare pe coloane  
 $F = (f, g, h)$  → concatenare pe linii  
 $VD = D$

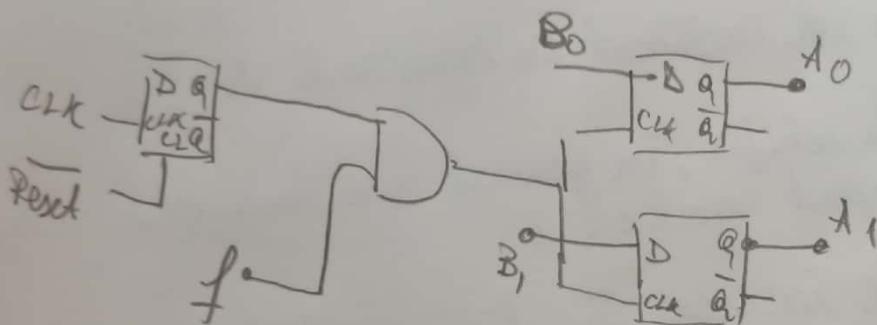
Instrucțiunea:  $D \leftarrow (A!B!C) * (f, g, h)$  se va selecta astfel  
 $D \leftarrow (A_1 f) \vee (B_1 g) \vee (C_1 h)$  → iesire logica  
sau  $(A!B!C) * (f, g, h) \leftarrow D$   
 $(A!B!C!) \leftarrow (((D_1 A) * (f, \bar{f}))! ((D_1 B) * (g, \bar{g}))! ((D_1 C) * (\bar{h}, \bar{h})))$

Exemplu de implementări:

10.  $A * f \leftarrow B$

a) transferul este condiționat pe intrări  
(vezi curs. polf)

b) transferul este condiționat pe tact



Specificarea semnalului de tact

→ unitatea de comandă funcționează pe baza unui semnal de ceară

→ în AHPL putem specifica semnalul de ceară  
 $CK = \text{semnal-de-tact}$

$CK[\text{nume-resursă}] = \text{semnal-de-tact}$

Exemplu:

1) 15.  $A \leftarrow B$  / transferul are loc pe frontul crescător

2) 16.  $CK[A] = TACT;$   
 $A \leftarrow B$  =, transferul are loc pe frontul pozitiv al lui TACT

17.  $CK[A] = \overline{CLOCK}$ ;  $CK[C] = TACT$

$A \leftarrow B$ ; pe  $\overline{CLOCK}$

$C \leftarrow B$ ; pe  $TACT$

$D \leftarrow B$ ; pe  $\overline{CLOCK}$

18.  $CK = \overline{CLOCK}$ ;

$A \leftarrow B$ ;

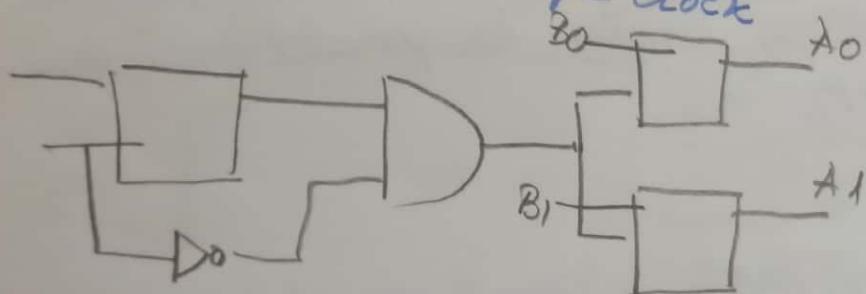
$C \leftarrow B$ ; } pe  $\overline{CLOCK}$

$D \leftarrow B$ ;

Ex implementare:

19.  $CK = \overline{CLOCK}$ ;

$A \leftarrow B \rightarrow$  conectare pe  $\overline{CLOCK}$



Transferuri și conexiuni după END-SEQUENCE

ie: După END SEQUENCE și înainte de END

$$z = (A_1 B) * (f_1, f_2)$$

$f_1, f_2$  simultan active

la destinație se conectează  $(A_1 f_1) \vee (B_1 f_2)$

$f_1 = 1 \Rightarrow$  conectare A  
 $f_2 = 1 \Rightarrow$  conectare B

AHPL = are scop teoretic, nu are o ~~o reprez~~ un echivalent practic

Exemplu:

$$A \leftarrow B$$

Instrucțiuni de salt

1.  $\rightarrow(S)$  salt necondiționat

2.  $\rightarrow(F)/(S)$  Salt condiționat unde  $\rho F = \rho S$  (\*)

3.  $\rightarrow(S)$  salt multiplu necondiționat  $\rho S > 1$

$S \rightarrow$  vector de pești AHPL

$S_i \rightarrow$  reprezentarea pasul AHPL

$F \rightarrow$  vectori de funcție logică (vect de selecție)

$\hookrightarrow$  de preferat doar o componentă 1, abr peste multe și mai multe  $\Rightarrow$  lansăm în paralel mai multe instrucțiuni AHPL

(\*) Pasuri:

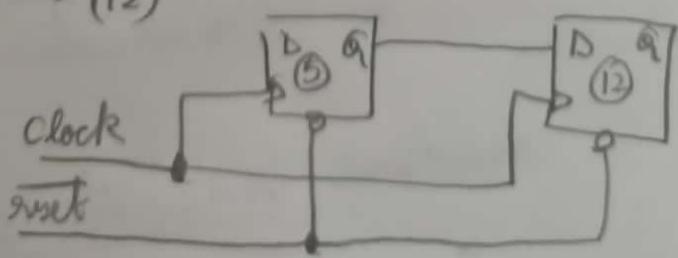
a)  $\rho((F)/(S)) = 0 \Rightarrow$  se trage la pasul următor

b)  $\rho((F)/(S)) = 1 \Rightarrow$  se executa ca și saltul necondiționat

c)  $\rho((F)/(S)) > 1 \Rightarrow$  se inițiază secvențe paralele în pasii AHPL selectați

Ex: salt necondiționat

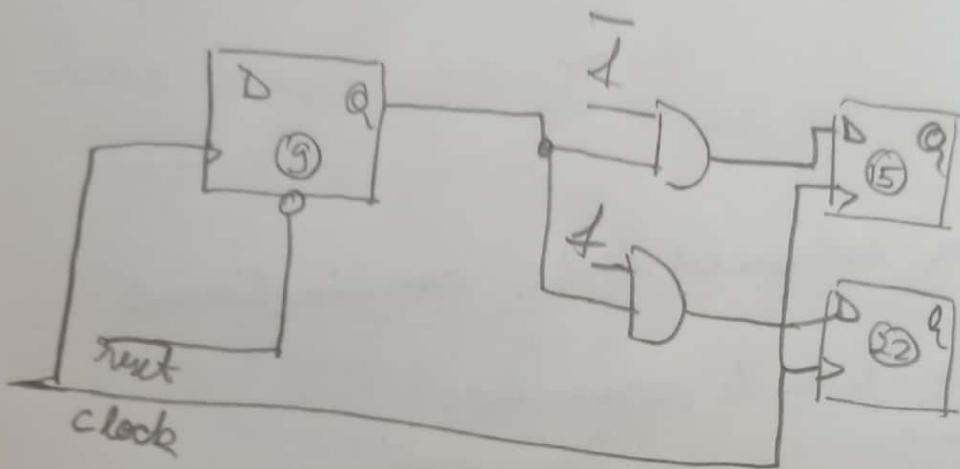
5.  $\rightarrow(12)$



4

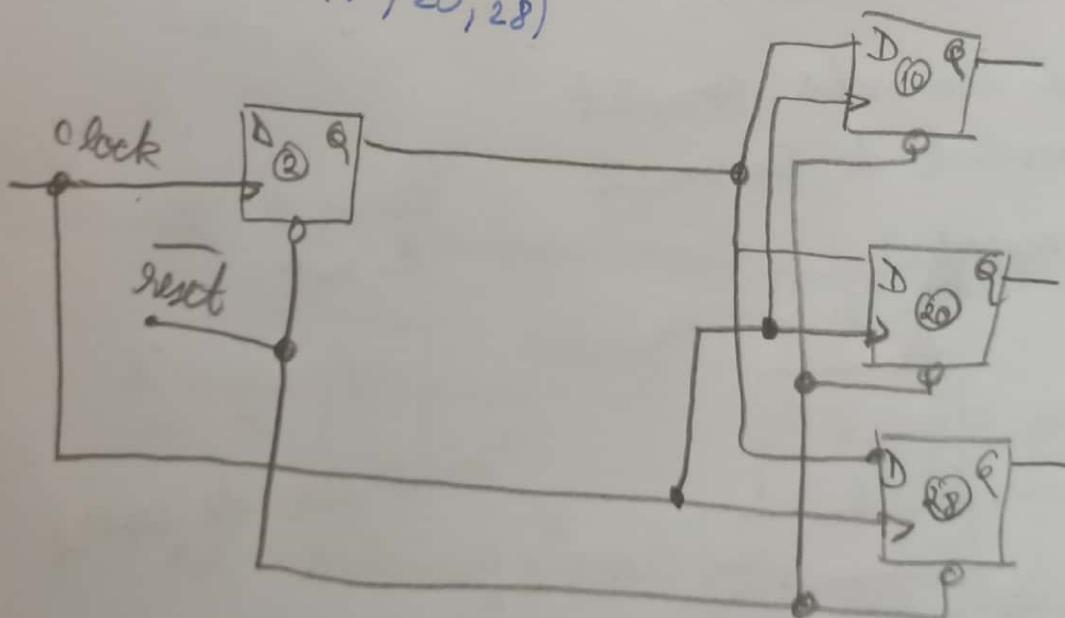
Ex: salt conditionat

9.  $\rightarrow (\bar{f}, f) / (15, 22)$



Ex: salt multiplex reconditionat

2.  $\rightarrow (10, 20, 28)$



DEAD END  $\rightarrow$  se folosește când vrem să opriuim o secvență

Ex:

$$2 \rightarrow (20, 30)$$

$$20 \rightarrow \dots \rightarrow \text{DEAD END}$$

\* NO DELAY

$$30 \rightarrow \dots$$

Descrierea în AHPL a schemelor logice combinatoriale

UNIT: nume-funcție < lista de parametri,  
< declaratii,  
END < conexiuni,  $\rightarrow$  numai conexiuni  
fără transferuri

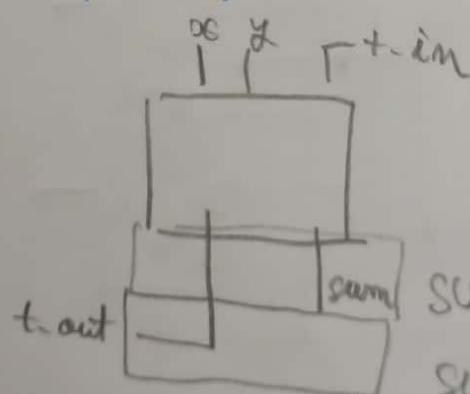
nume-funcție (parametri)

Ex: sumatorul elementar complet

$x, y \rightarrow$  elementele de însumat

t-in - transportul

$\Rightarrow$  sum  
t-out



matricea de ieșire  
 $\leftarrow \text{SUMEC}_0^0$   
 $\text{SUMEC}_0^1$

UNIT: SUMEC ( $x, y, t_{in}$ )

INPUTS ( $x, y, t_{in}$ )

OUTPUTS: SUMEC [2;1]

$$1. a = x \oplus y$$

$$2. b = x \wedge y \quad \} \text{ semnale intermedii}$$

$$3. \text{sum} = a \oplus t_{in}$$

$$4. c = a \wedge t_{in}$$

$$5. t_{out} = b \vee c$$

$$6. \text{SUMEC}_0^0 = \text{sum}$$

$\text{7. } \text{SUMEC}_0^1 = t_{out}$   
END

⑥

Suma paralel pe 16 lini

$$X = (x_0, x_1, \dots, x_{16})$$

$$Y = (y_0, y_1, \dots, y_{16})$$

Suma și transportul din rangul cel mai semnif vor fi

$$\text{ADD} = (\text{ADD}_0, \dots, \text{ADD}_{16})$$

UNIT:  $\text{ADD}(X, Y, t_{16})$

INPUTS  $X[16], Y[16]; t_{16}$

OUTPUTS:  $\text{ADD}[17]$

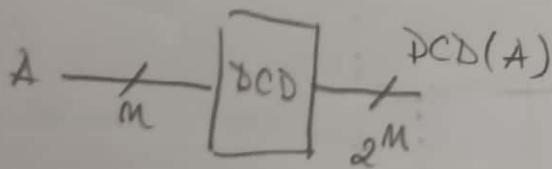
1.  $T = \text{SUMEC}_0^{15} (x_i, y_i, T_{i+1}, t_{16})$

2.  $S = -$

3.  $\text{ADD} = -$

!

Unitate logică combinatorică pentru decodificare



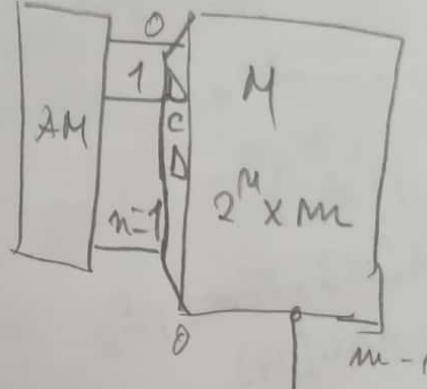
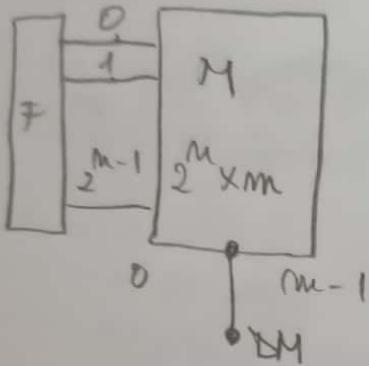
$$\text{DCD}_i = \begin{cases} 0, & \text{dacă } A \neq i \\ 1, & \text{dacă } A = i \end{cases}$$

$A/ \rightarrow$  reducere și codificarea lui  $k$  pe  $m$  lini

$$\text{DE}_i = A / (((\bar{M}T_i)/A), ((\bar{M}\bar{T}_i)/A))$$

Operări de citire scriere în memorie HPL

→ Se consideră o memorie  $M[2^m, m]$  și un vector de selecție  $F$ , cu  $\beta F = 2^m$  și  $\alpha \cdot i + f = 1$  (o singură comp = 1)



Operări de scriere în memorie

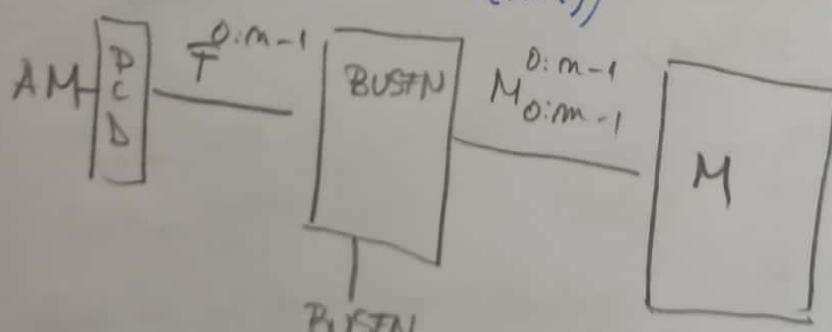
$$M * F \leftarrow DM$$

$DM \rightarrow$  registrul de date

$$\Rightarrow M * DCD(AM) \leftarrow DM$$

Operări de citire

$$DM \leftarrow BUSFN((M; DCD(AM)))$$



1.  $NIVS^{0:m-1} \leftarrow M^{0:m-1} \wedge F^{0:m-1}$
  2.  $NIV^{0:m-1} = NIVS^0 \wedge (NIVS^{1:m-1} \vee NIV^{0:m-2})$
  3.  $BUSFN = NIV^{m-1}$
- o diferență binară diferită de 0

! De refinut că la citire folosim  $BUSIN((M, DCD(AM)))$   
 + stocăm undeva datele

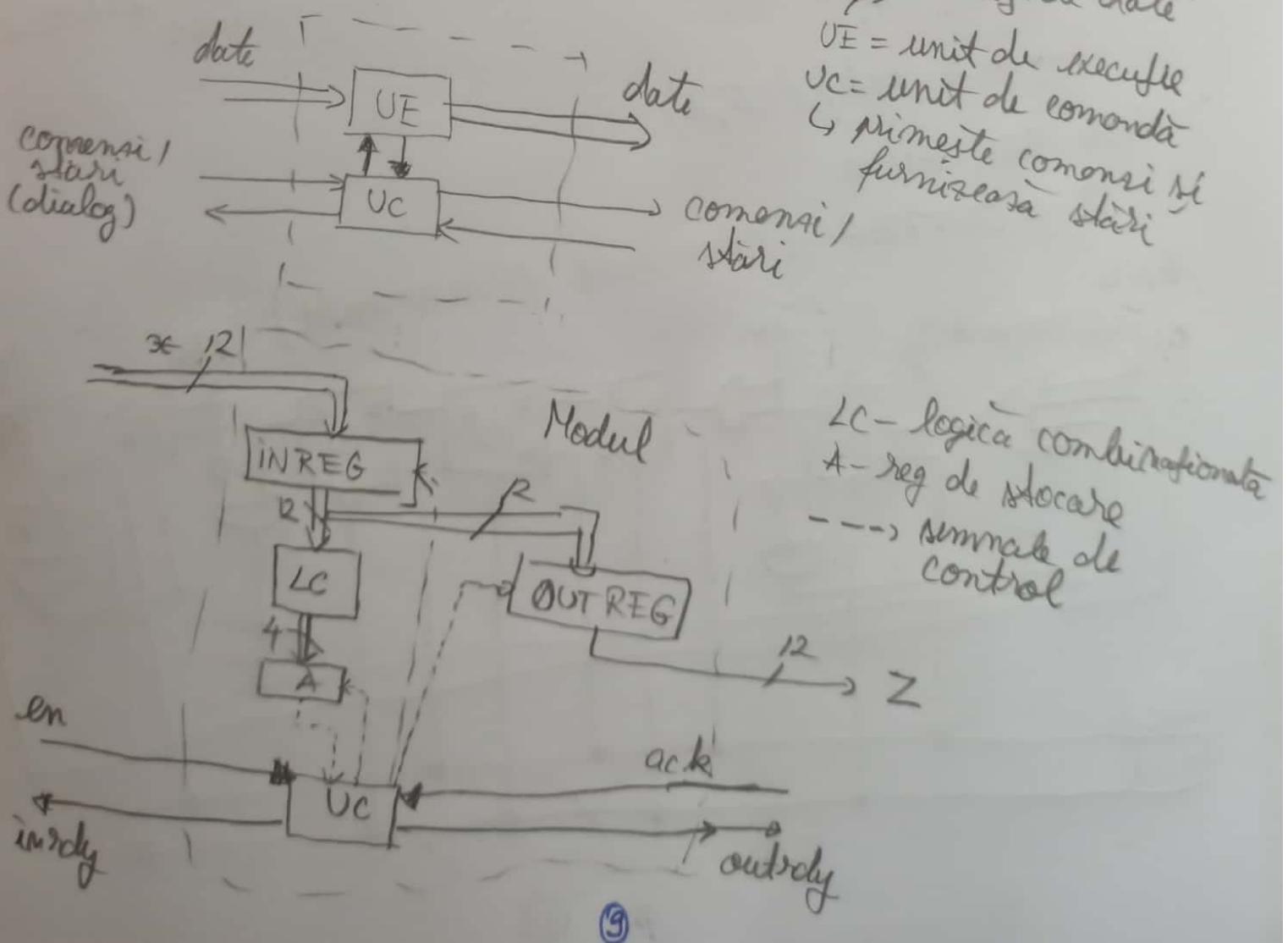
Exemple de proiectare a unui modul numeric în AHPL

1. Să se proiecteze un dispozitiv numeric care preia cinci seturi de 12 biti ( $x_{0:11}$ ) și le transferă mai departe (către alt dispozitiv) cinci seturi ce îndeplinește condiția:  $x_{0:3} \wedge x_{4:7} \wedge x_{8:11} \neq 0000$

Se cere:

- schema bloc
- proiectarea în AHPL
- impl. unit. de comandă
- impl. unit de execuție

Soluție



b)

MODULE : Dispositiv transfer date

MEMORY: INREG[12]; OUTREG[12]; A[4]

INPUTS: X[12]; en; ack

OUTPUTS: Z[12]; i ready; o ready

AM muta plan  
Hări codificate  
→ se poate urca la  
stările complete  
decodificate

1.  $i_{ready} = 1;$   
 $\rightarrow (\bar{en}, en) / (1, 2)$
2.  $INREG \leftarrow X$
3.  $A \leftarrow INREG_{0:3} \wedge INREG_{4:7} \wedge INREG_{8:11};$   
 $OUTREG \leftarrow INREG$
4.  $\rightarrow (\overline{V/A}, V/A) / (4, 5)$
5.  $o_{ready} = 1;$   
 $\rightarrow (ack, ack) / (5, 1)$

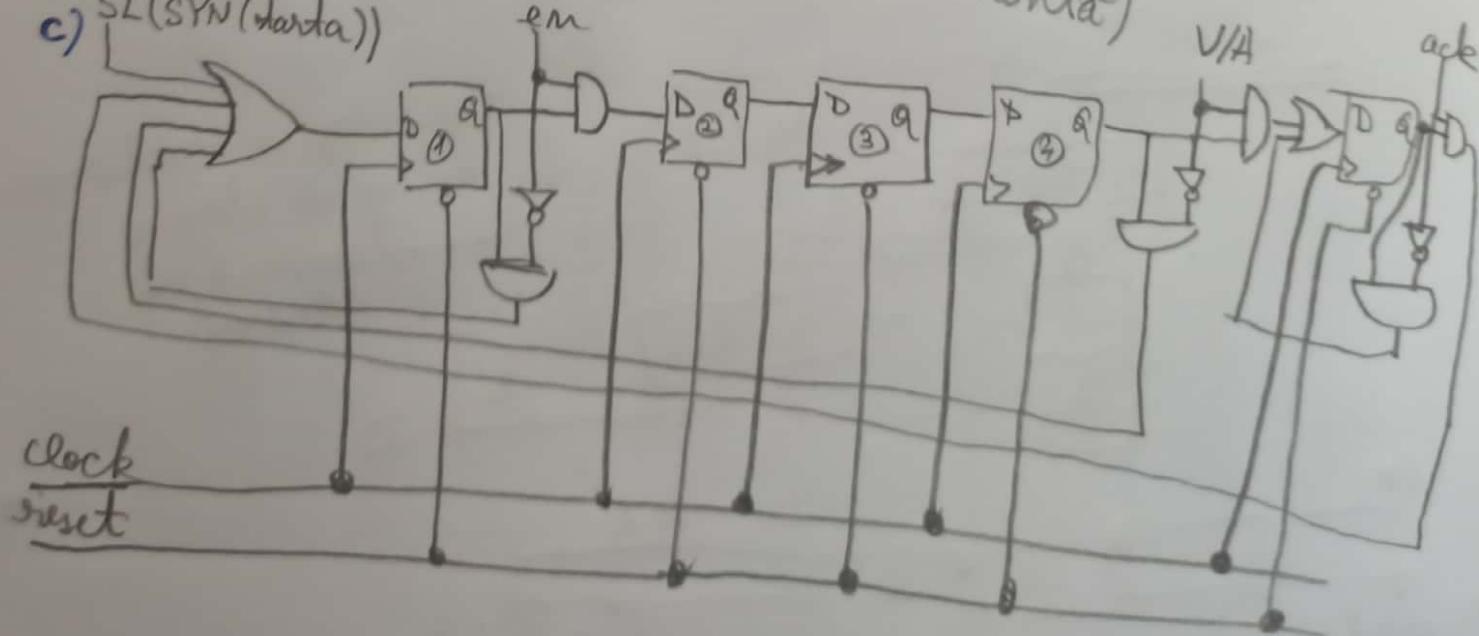
END SEQUENCE

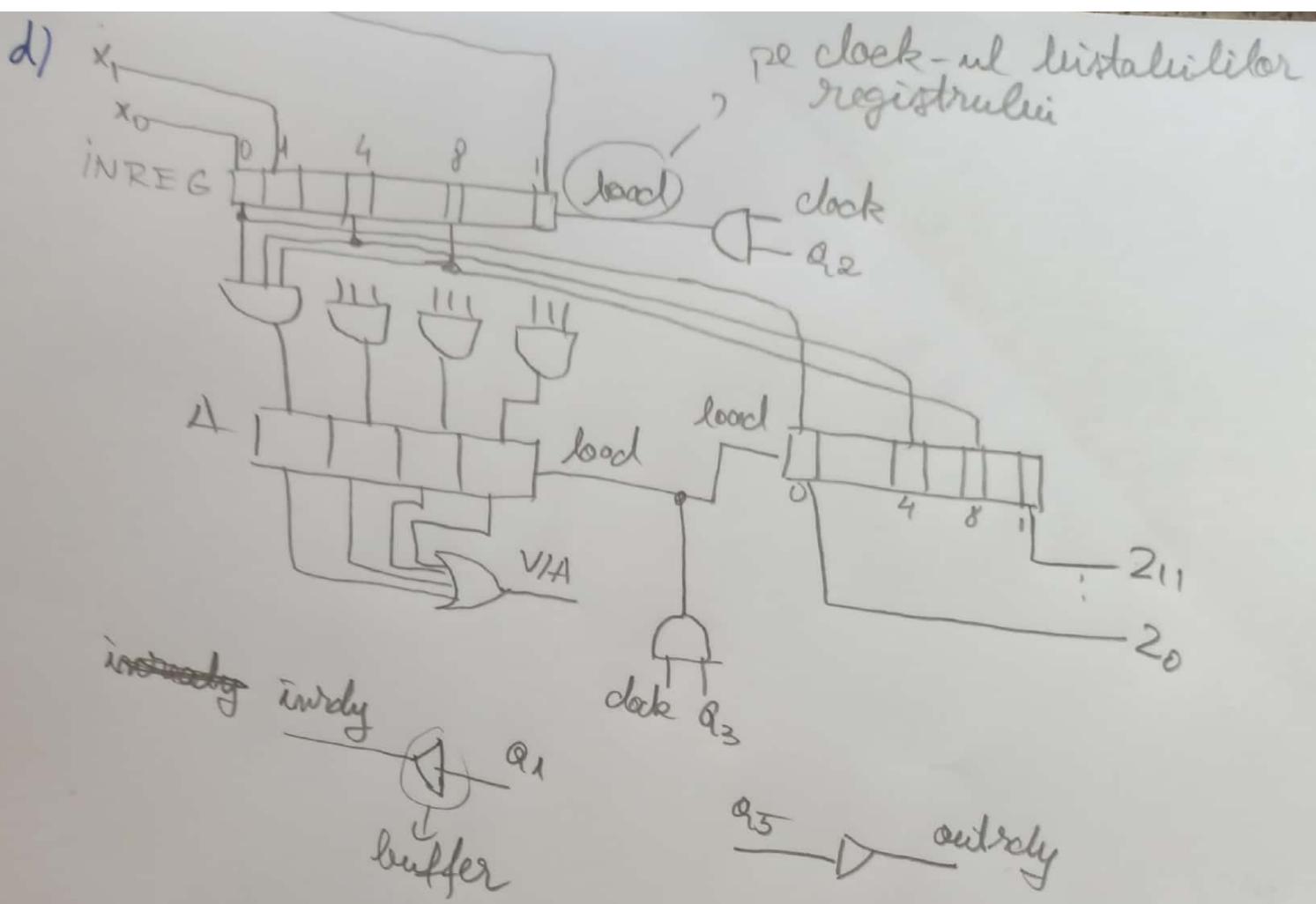
Z = OUTREG

END

c) SL(SYN(harta))

(conexiune permanentă)





# Reprezentarea numerelor în memorie + prelucrarea datelor stocate

→ Reprezentarea în:

- codul binar
- ↳ codul Gray
- ↳ BCD

## Reprezentarea în binar

$$N = x_n x_{n-1} \dots x_0$$

$$V_N = \sum_{i=0}^n x_i * 2^i$$

## Reprezentarea în codul Gray

2 valori succinive diferă printr-un singur bit

Ești  $N_b = b_n b_{n-1} \dots b_0$  (în binar)

$$N_g = g_n g_{n-1} \dots g_0 \text{ (în gray)}$$

$$N_b \rightarrow N_g$$

$$N_g \rightarrow N_b$$

$$\Rightarrow \boxed{\begin{aligned} g_i &= b_i \oplus b_{i+1} \text{ (sau exclsiv)} \\ &\quad b_{n+1} = 0 \\ \rightarrow b_i &= g_i \oplus g_{i+1} \oplus \dots \oplus g_n \end{aligned}}$$

Binar - Gray

$$\begin{aligned}g_0 &= b_0 \oplus b_1 \\g_1 &= b_1 \oplus b_2 \\g_2 &= b_2 \oplus b_3 \\g_3 &= b_3\end{aligned}$$

Gray - Binar

$$\begin{aligned}b_0 &= g_0 \oplus g_1 \oplus g_2 \oplus g_3 \\b_1 &= g_1 \oplus g_2 \oplus g_3 \\b_2 &= g_2 \oplus g_3 \\b_3 &= g_3\end{aligned}$$

→ Codul BCD (binary coded to decimal)

↪ BCD (8-4-2-1)

→ Codul 2-4-2-1 memorată fiecare cifră a nr Pe 4 biți.

→ cod excess 3 → 0 = 0011 excess 3

Eseuri BCD ↔ binar (exemplu în curs)  
→ corecția = adunăm 3 înainte de desplasare  
dacă decada  $\geq 5$

BCD → Binar (cadem 3)  $\checkmark$  când trece un bit  
1 de la o decada la alta (C2)

← adunăm complementul lui 3  
(1101)

\* Putem (trebuie) să facem mai multe corecții  
dacă situația impune

## Reprezentarea în virgulă fixă

Numere întregi:  $x = x_m \dots x_0$ ,

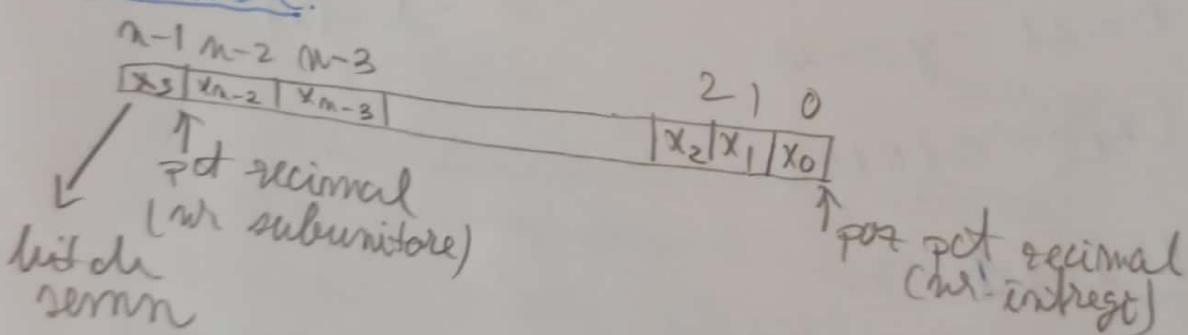
numere subunitare:  $x = x_{-1}x_{-2} \dots x_{m+1}x_m$   
 $= x = \sum_{i=1}^m x_i \cdot 2^{-i}$

numere reale:

$$x = x_k x_{k-1} \dots x_0 x_{-1} x_{-2} \dots x_{-m} \rightarrow \text{nu se reprez. explicit in calculator}$$

$$\Rightarrow x = \sum_{i=0}^k x_i \cdot 2^i + \sum_{i=1}^m x_{-i} \cdot 2^{-i}$$

Reprezentare:



Codul direct:

$$[x]_d = \begin{cases} 0.x_{n-2}x_{n-3} \dots x_0, & \text{dacă } x \geq 0 \\ 1.x_{n-2}x_{n-3} \dots x_0, & \text{dacă } x < 0 \end{cases}$$

representare în semn și modul

$$x = 21 \quad y = -20 \text{ pe 6 lărgi}$$

$$\Rightarrow [x]_d = 010101$$

$$[y]_d = 110100$$

(3)

## Reprezentarea în cod invers (în complement față de 1)

$$[x]_i = \begin{cases} 0\bar{x}_{n-2} \dots \bar{x}_1 \bar{x}_0 & x > 0 \\ 00 \dots 00 \\ 11 \dots 11 \end{cases}^0 \quad x = 0 \\ 1\bar{x}_{n-2} \bar{x}_{n-3} \dots \bar{x}_1 \bar{x}_0 \quad x < 0$$

$$\bar{x}_k = 1 - x_k$$

dacă  $x < 0$

$$|x| + [x]_i = 0\bar{x}_{n-2} \dots \bar{x}_1 \bar{x}_0 + 1\bar{x}_{n-2} \dots \bar{x}_1 \bar{x}_0 = 11\dots 11 = 2^n - 1$$

Un nr. negativ în cod invers este:  $2^n - 1 - |x|$

Exemplu  $x = 24 \quad y = -20$

$$[x]_d = [x]_i^0 = 010101$$

$$[y]_i = 101011$$

## Reprezentarea în cod complementar (complement față de 2)

$$[x]_c = \begin{cases} 0\bar{x}_{n-2} \dots \bar{x}_1 \bar{x}_0 & \text{dacă } x \geq 0 \\ 1\tilde{x}_{n-2} \dots \tilde{x}_1 \tilde{x}_0 & \text{, } x < 0 \end{cases}$$

$$1\tilde{x}_{n-2} \dots \tilde{x}_1 \tilde{x}_0 = 2^n - |x|$$

$$[x]_c = 1\tilde{x}_{n-2} \dots \tilde{x}_1 \tilde{x}_0 = 2^n - |x| = |x| + [x]_i + 1 - |x| = [x]_i + 1$$

$$x = 21 \quad y = -20$$

$$[x]_d = [x]_i = [x]_c$$

$$[y]_c = 101100$$

$$\begin{array}{r} [y]_d = 110100 \\ [y]_i = 101011 \\ \hline [y]_c = 101100 \end{array}$$

$$[y]_d = 110100$$

inversare în sensul opus pătră la fel  
al de sunt 1

## Reprezentarea informației alfumerice

ASCII → pe 7 biți

0 - 1Fh : linie nouă (LF), carriage return

20h - 2Fh : carac. speciale : !, " , #, \$, space

30h - 39h : 0(30h) , 9(39h)

3Ah - 40h : carac speciale

41h - 5Ah : A(41h) - - -

5Bh - 60h : carac. speciale

61h - 7Ah : a(61h)

7Bh - 7Fh : caractere de control

## Operatii aritmetice in calculatoarele numerice

operande      operand<sub>2</sub>  
 $x \leftarrow op \quad y \leftarrow z$   
 operatia

$$op = 0 \text{ (adunare)} \\ op = 1 \text{ (scădere)}$$

$$op_{fin} = x_s \oplus y_s \oplus op$$

$$op_{fin} = 0 \Rightarrow adunare \text{ modulele} \\ op_{fin} = 1 \Rightarrow scădere \text{ modulele}$$

Exemplu:  $x = 11 \quad y = -14 \quad op \in \{0, 1\}$   
 $x - y = z$

$$[x]_d = 001011 \\ [y]_d = 101110$$

$$op_{fin} = 0 \oplus 1 \oplus 1 = 0 \Rightarrow adunarea \text{ modulelor}$$

semnul lui  $z_s = x_s$

$$\begin{array}{r} |x|+ \\ |y| \\ \hline |z| \end{array} \qquad \begin{array}{r} 01011 \\ 01110 \\ \hline 11001 \end{array}$$

$$\Rightarrow z = 011001 \quad (z = 25)$$

↳ rezultatul este corect (nu are transport la c.m.s. B (MSB  $\rightarrow$  cel mai semnificativ bit))

Dacă  $op_{fin} = 1 \Rightarrow scădem \text{ din modulul mai mare pe cel mai mic (în practică avem nevoie de un comparator) semnul este dat de operandul mai mare în modul.}$

Exemplu  $op = 1 \quad |y| > |x|$

$$\text{Ex: } x = -29 \quad y = 17 \quad x + y = 2$$

$$[x]_d = 111101$$

$$[y]_d = 010001$$

$$\text{offim} = 1 \oplus 0 \oplus 0 = 1$$

$$|x| > |y| \Rightarrow z_s = x_s$$

$$\begin{array}{r} |x|- \\ |y| \\ \hline |z| \end{array}$$

$$\begin{array}{r} 11101- \\ 10001 \\ \hline 01100 \end{array}$$

$$\Rightarrow z = 101100 \quad (z = -12)$$

Adunarea și scăderea în cod binar

$$1) x > 0, y > 0, x + y < 2^{n-1} \quad (\text{dă nu avem depășire})$$

$$[x]_i + [y]_i = |x| + |y| = |x+y| = [x+y]_i$$

$$2) x > 0, y < 0, |x| > |y| \Rightarrow \text{rezultat pozitiv}$$

$$[x]_i + [y]_i = |x| + 2^{n-1} - |y| = (2^{n-1} - |y|) \rightarrow \text{transportul } N-1 \text{ se reduce, corecție}$$

$$= |x| - |y| = |x+y| = [x+y]_i$$

$$\text{Exemplu: } x + y = 2$$

$$[x]_i = 011001$$

$$[y]_d = 101001$$

$$[x]_i + [y]_i \leftarrow \begin{array}{l} \text{inclusia} \\ \text{lui } 0 \\ \text{se men} \end{array}$$

$$[z]_i$$

$$x = 25 \quad y = -9$$

$$\Rightarrow [y]_i = 110110$$

$$\begin{array}{r} 011001 \\ 110110 \\ \hline 1001111 \\ \hline 010000 \end{array} \Rightarrow z = 16$$

④

3)  $x > 0, y < 0, |x| < |y|$

$x, y$  au semne  
diferite  
 $|y| > |x|$

$$\begin{aligned} [x]_i + [y]_i &= |x| + 2^m - 1 \rightarrow |y| = 2^m - 1 - (|y| - |x|) = \\ &= 2^m - 1 - |y+x| = [x+y]_i \end{aligned}$$

ex:  $\frac{x}{y}$  nu arem corectie

$$x+y=2 \quad x=5 \quad y=-29$$

$$[x]_i = 000101$$

$$[y]_d = 111101 \Rightarrow [y]_i = 100010$$

$$\Rightarrow [x]_i +$$

$$\begin{array}{r} [y]_i \\ \hline [z]_i \end{array}$$

$$\begin{array}{r} 000101 \\ 100010 \\ \hline 100111 \end{array} = [z]_i$$

$$\Rightarrow [z]_d = 111000 \Rightarrow z = -24$$

4)  $x > 0, y < 0, |x+y| < 2^{m-1}$

$$[x]_i + [y]_i = \cancel{2^m - 1} \quad \cancel{2^m - 1} - |x| + 2^m - 1 - |y| =$$

se redesc  $2^{m-1}$   
corectie

$$= 2^m - 1 - |x| - |y| = 2^m - 1 - |x+y| = [x+y]_i$$

ex:  $x+y=2 \quad x=-11 \quad y=-19$

$$[x]_d = 101011$$

$$[y]_d = 110011$$

$$[x]_i = 110100$$

$$[y]_i = 101100$$

$$\Rightarrow [x]_i +$$

$$\begin{array}{r} [y]_i \\ \hline [z]_i \end{array}$$

$$\begin{array}{r} 110100 \\ 101100 \\ \hline 100000 \end{array}$$

$$[z]_i = \frac{1}{100001} \Rightarrow [z]_d = 111110$$

$$\Rightarrow z = -30$$

## Adunarea și scăderea în cod complementar

! Nici adunarea mereu sau scăderea, se adună și la jumătate de semn

$$1) x > 0 \quad y > 0, |x| + |y| < 2^{n-1}$$

$$[x]_c + [y]_c = |x| + |y| \quad (\text{nu arem depășire})$$

$$2) x > 0 \quad y < 0 \quad |x| > |y|$$

$$[x]_c + [y]_c = |x| + 2^n - |y| \quad \text{apare un transport de la raza} \\ \text{de semn ce se neglijă}$$

$$\text{ex: } x + y = z \quad x = 22 \quad y = -18 \quad |y| = |x| - |y| = |x + y| = [x + y]_c$$

$$[x]_c = 010110$$

$$[y]_d = 110010$$

$$\begin{array}{r} [x]_c + \\ [y]_c \\ \hline [z]_c \end{array} \quad \Rightarrow [y]_c = 101110$$

$$\begin{array}{r} 010110 \\ 101110 \\ \hline 1000100 \end{array}$$

$$3) x > 0 \quad y < 0 \quad |x| < |y| \Rightarrow \text{rez. negativ} \quad \Rightarrow [z]_c = 000100 \quad (z = 4)$$

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

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

$$Ex: \quad x = 20 \quad y = -28 \quad x+y = z$$

$$[x]_c = 010100$$

$$[y]_d = 111100 \rightarrow [y]_c = 100100$$

$$\Rightarrow \begin{array}{r} [x]_c + \\ [y]_c \\ \hline [z]_c \end{array} \quad \begin{array}{r} 010100 \\ 100100 \\ \hline 111000 \end{array} \Rightarrow z = -8$$

$$4) \quad x < 0 \quad y < 0 \quad |x+y| < 2^{m-1} \quad \Rightarrow [z]_d = 101000$$

$$\begin{aligned} [x]_c + [y]_c &= 2^m - |x| + 2^m - |y| \leftarrow \text{se neglig.} \Rightarrow \text{correcte} \\ &= 2^m - |x+y| = [x+y]_c \end{aligned}$$

$$Ex: \quad x = -4, y = -13$$

$$[x]_d = 100100$$

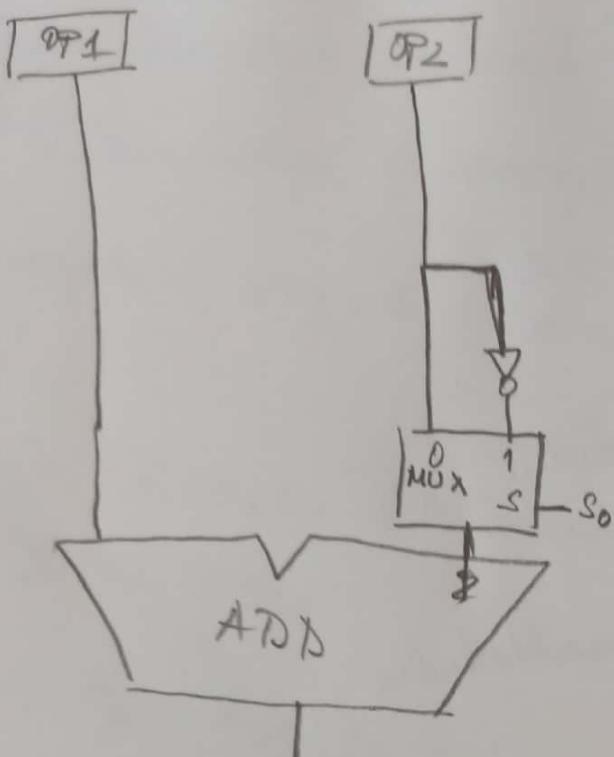
$$[y]_d = 101101 \Rightarrow [x]_c = 111100$$

$$[y]_c = 110011$$

$$\begin{array}{r} [x]_c + \\ [y]_c \\ \hline [z]_c \end{array} \quad \begin{array}{r} 111100 \\ 110011 \\ \hline 101111 \end{array}$$

$\Rightarrow$  negligible

$$\Rightarrow [z]_c = 101111 \Rightarrow [z]_d = 110001 \quad (z = -17)$$



Indicatori de condiții → bîstuișii  
exemplu:

1 semn, memorată în UAL (semnal rezultatului)

2 zero, este 1 dacă rezultatul este 0 (altfel 1)  
 $Z = V_1 \text{ Resultat}$

3 semn (redusere)       $\downarrow$  indică eroare  
depasire, rezultatul obținut nu este corect

$$D = \bar{x}_s 1 \bar{y}_s 1 z_s \vee x_s 1 y_s 1 \bar{z}_s$$

T → în funcție de cod poate să fie o eroare <sup>se indică</sup>  
transport (1 dacă apare un transport de la rangul <sup>curent</sup>)

P paritate (1 în cazul în care rezultatul are un număr par de unități)

# Înmulțirea în virgulă fixă

I. Înmulțirea directă  $x \cdot y = z$

Etape:

1) Dacă se cunoaște rezultatul:  $z_s = x_s + y_s$

2) Calcularea modulu lui rezultatului

a) Numere întregi:

$$|x| \cdot |y| = |x| \cdot \sum_{k=0}^{m-2} y_k \cdot 2^k = \sum_{k=0}^{m-2} |x| \cdot y_k \cdot 2^k$$

$$|x| \cdot y_k \cdot 2^k = \begin{cases} 0 & \text{daca } y_k = 0 \\ |x| \cdot 2^k & \text{daca } y_k = 1 \end{cases}$$

↪ deplasarea lui  $|x|$  spre stânga cu  $k$  pozitii.

b) Numere subunitare

$$|x| \cdot |y| = |x| \cdot \sum_{k=1}^{m-1} y_{-k} \cdot 2^{-k} = \sum_{k=1}^{m-1} |x| \cdot y_{-k} \cdot 2^{-k}$$

$$|x| \cdot y_{-k} \cdot 2^{-k} = \begin{cases} 0 & \text{daca } y_{-k} = 0 \\ |x| \cdot 2^{-k} & \text{daca } y_{-k} = 1 \end{cases}$$

↪ modelul  $x$  deplasat spre dreapta cu  $k$  pozitii.

Pentru numerele subunitare:

3) Trunchiere și rotunjirea rezultatului

Exemplu:  $x \cdot y = z$

$$x = \frac{20}{32} \text{ și } y = -\frac{19}{32}$$

$$[x] = 0.10100 \quad \text{făcem shifturi}$$

$$[y] = -1.10011 \quad (20 \text{ în binar, shiftat cu 5 pozitii})$$

$$1) z_s = x_s \oplus y_s = 1$$

$$2) \text{Modulul} \quad |z| = 1 \times 1 \cdot |y|$$

$$\begin{array}{r} 10100 \\ 10011 \\ \hline 00000 \\ 000010 \\ 000000 \\ 0000000 \\ 010100000 \\ \hline 010111100 \end{array}$$

$$\begin{aligned} & 10100 \cdot 10100 \\ & 10011 \\ & \hline 101 \cdot 9-5 \cdot 2^5 \\ & 101 \cdot 2-4 \cdot 2^4 \\ & : \\ & 101 \cdot 2-1 \cdot 2^{-1} \end{aligned}$$

$$\Rightarrow [z] = 1.0101111100 \quad \Rightarrow z = \frac{-380}{1024}$$

3) Trunchiere și rotunjire

$$\begin{array}{r} 01011 \quad 11100 \\ 01011 \quad \swarrow \quad \uparrow \\ \hline 01100 \quad \text{dacă e 1} \\ \quad \quad \quad \leftarrow \quad \text{adunăm un 1} \end{array}$$

$$\Rightarrow z \approx -\frac{12}{32}$$