

# Sisteme logice secventiale complexe

Unitate de executie

Unitate de comandă

Microprogramare

S.l. Dr. Ing. Vlad-Cristian Miclea

Universitatea Tehnica din Cluj-Napoca

Departamentul Calculatoare



# CUPRINS

- 1) Introducere
- 2) Definirea problemei
- 3) Unitatea de executie
  - Arhitectura circuitului
  - Designul componentelor
- 4) Unitatea de comanda
  - Semnalele de control
  - UC cablata
  - UC microprogramata
- 5) Microprogramare
  - Microoperatii
  - Generarea urmatoarei instructiuni - microinstructiuni
  - Programul de control
- 6) Concluzii



# PLAN CURS

- Partea 1 – VHDL
  1. FPGA
  2. Limbajul VHDL – 1
  3. Limbajul VHDL – 2
  4. Limbajul VHDL – 3
- Partea 2 – Implementarea sistemelor numerice
  - 5. Realizarea unui sistem numeric complex; Unitate de executie**
  - 6. Unitate de comanda; Microprogramare**
- Partea 3 – Automate
  7. Automate finite
  8. Stari
  9. Automate sincrone
  10. Automate asincrone
  11. Identificarea automatelor
  12. Automate fara pierderi
  13. Automate liniare
- Partea 4 – Probleme si discutii

# CONTEXT

## Sisteme numerice sincrone complexe

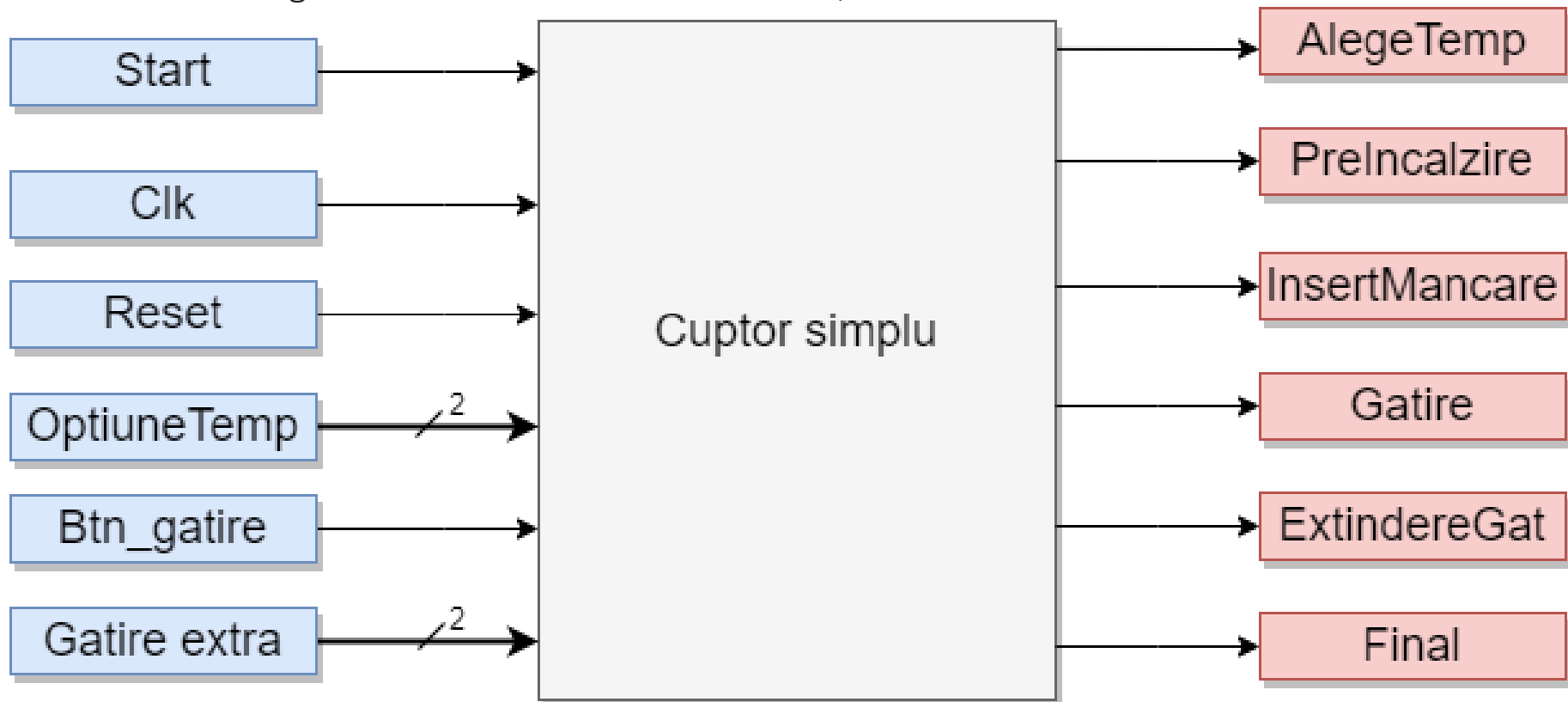
- Realizarea unor sisteme digitale complexe
- Se va discuta abordarea, definirea, analiza, implementarea si testarea unor sisteme hardware complexe
- Exemplu: un cuptor de gatit
- Se va discuta realizarea componentelor principale: UC si UE
- Unitate de executie
  - Numaratoare, registre, MUX-uri, ALU
  - Trebuie accesate la momentul potrivit
- Unitate de comanda/control
  - Generare semnale de control
  - Generarea urmatoarei stari
  - Unitate cablata (hardwired)
  - Microprogramare

# Definirea problemei

- Cuptor de gatit – exemplul de la laborator (putin mai complicat)
  - Gateste la 4 temperaturi predefinite: 180, 200, 220, 240 grade
    - In fiecare secunda, temperatura creste cu 10 grade;
  - Gateste 25 de minute
  - La sfarsit, se poate extinde perioada cu o valoare de baza de 50 de minute, la care se poate aduna sau reduce 5, 8 sau 12 min fata de perioada de gatire extra
- Detalii:
  - Se asteapta pana cand se apasa buton “Start”
  - Daca Start, se asteapta alege temp – se asteapta introducerea temp (V0,V1,V2,V3);
  - Se preincalzeste cuptorul; Apare un led “Preincalzire”
  - Apoi se stinge “Preincalzire”, se aprinde “InsertMancare” (se sta maxim 5 min)
  - Daca se introduce mancarea, se apasa “Buton\_gatire” si se asteapta 25 min (se fiseaza “Gatire”)
  - Dupa finalizare gatire, se asteapta extindere gatire
  - Daca se doreste extra-gatire, se alege optiunea dorita
  - Se calculeaza timpul de extra-gatire
  - Se asteapta timpul de gatire suplimentare; Apare un led “Extra\_gatire”
  - La final, se stinge ledul “Gatire” si Extra-gatire si se asteapta un nou proces

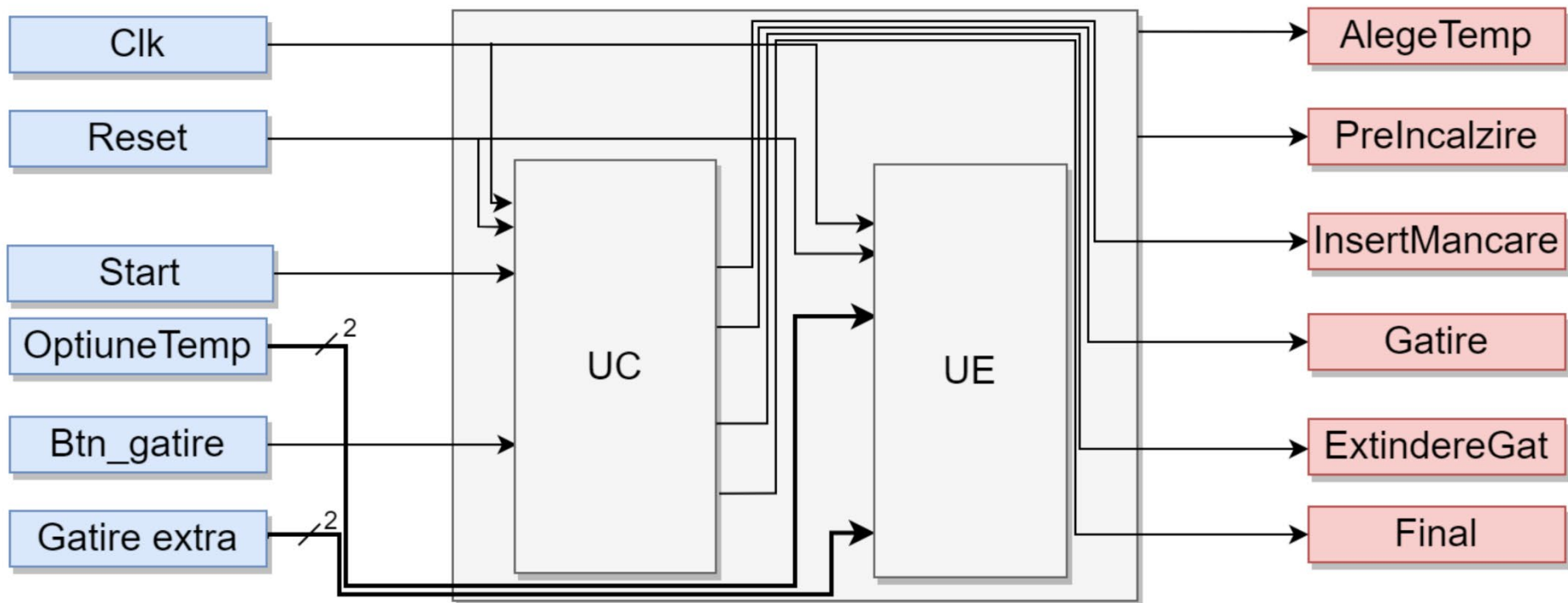
# Schema bloc

- Evidentiaza intrarile si iesirile sistemului
- Pot exista intrari/iesiri care sa nu fie evidente (ex buton pt validare date)
- Se identifica use-case-urile sistemului
  - Se identifica pas-cu-pas fiecare etapa prin care trece sistemul
  - Se vor descoperi eventualele actiuni ascunse
  - Pot fi adaugate eventuale semnale de intrare/iesire



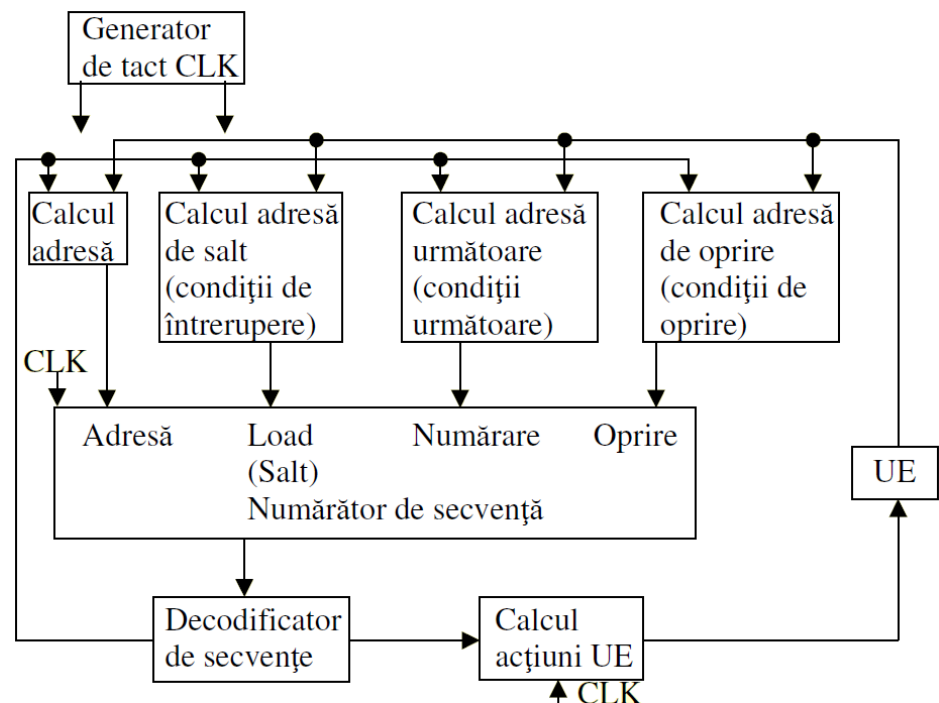
# Schema bloc - componente

- Prima divizare conceptuala a sistemului: UC vs UE
- Unitate de comanda/control
  - Logica de control din sistem
- Unitate de executie
  - Resursele necesare pentru system
  - Componentele (alcatuite eventual din sub-componente)



# Unitate cablata (hardwired)

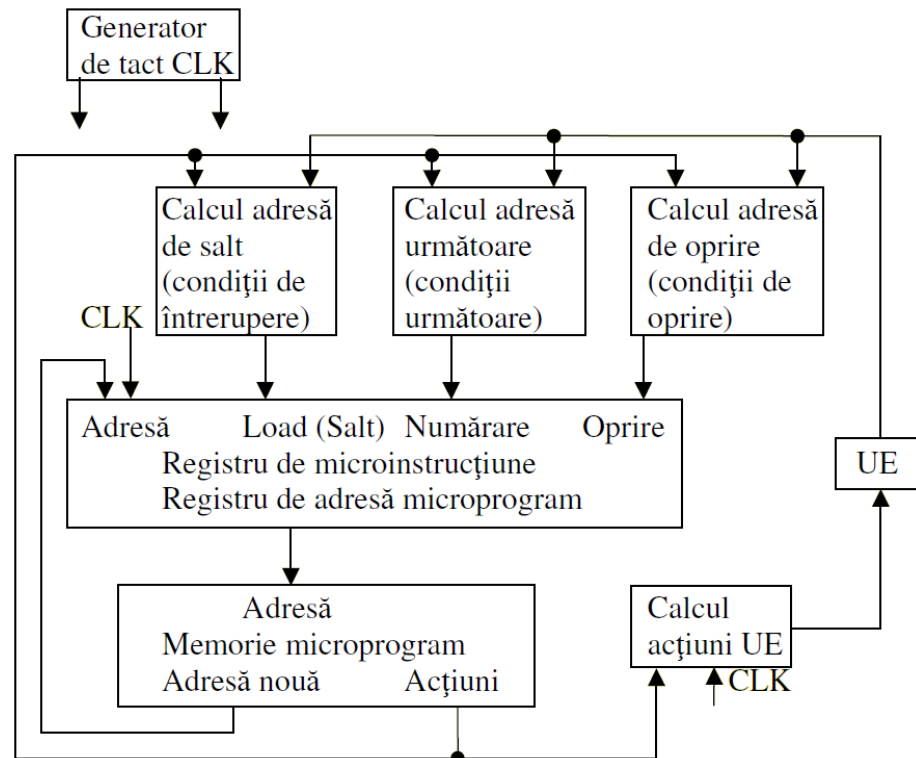
- Folosește un FSM (finite state machine) -
  - Contine o parte combinatională – generează următoarea stare
  - Contine un registru/numarator – ține starea curentă
  - Fiecare stare corespunde la un set de semnale de control
- Metoda rapidă, toate informațiile se generează direct din codificarea stării
- Utilizare limitată (circuite simple)
- Greu de extins cu semnale noi
- Greu de extins cu stări noi
  - Trebuie redefinit tot sistemul
- Exemplu





# Unitate microprogramata

- Semnalele de control nu mai sunt codificate intr-o anumita stare
- Secventa de instructiuni va fi generata folosind o **memorie**
- Registrul va incara o microinstruțiune, care va contine o **adresa**
- Datele de la adresa respectiva vor contine informatii pentru a genera:
  - Semnalele de control
  - Adresa urmatoarei microinstruțiuni
- Obiectivul de fapt este





# Microprogramare

- apariție - în jurul anilor '50
- concept introdus de Maurice Wilkes - Universitatea din Cambridge
- motivație: elaborarea unor **metode și mijloace tehnice noi** pentru **proiectarea și construcția unităților de comandă** a sistemelor numerice
- microprogramarea presupune înregistrarea (memorarea) semnalelor de comandă în memorii, ca alternativă la generarea lor de către o unitate de comandă cablată (sistem secvențial)
- **semnalele de comandă** sunt grupate în **microinstrucțiuni**

# Model





# Structura de control microprogramata

## Model

- matricea A - conține **linii de control** - dau semnalele de comandă → necesară pentru execuția instrucțiunilor
- matricea B - conține **linii de adresă** → necesară pentru aflarea instrucțiunii următoare
- o linie orizontală reprezintă o microinstrucțiune
- microinstrucțiunea se selectează conform adresei din registrul de adrese
- **bistabili de condiție** → pentru decizie și salt



# Structura de control microprogramata

## Definiție

- **microprogramarea** = controlul unui sistem numeric prin intermediul unor cuvinte (microinstrucțiuni) citite secvențial (pas cu pas) din memorie
  - din microinstrucțiuni se generează semnalele de control pentru funcționarea corectă a sistemului numeric
  - proiectarea - sistematică și flexibilă
-

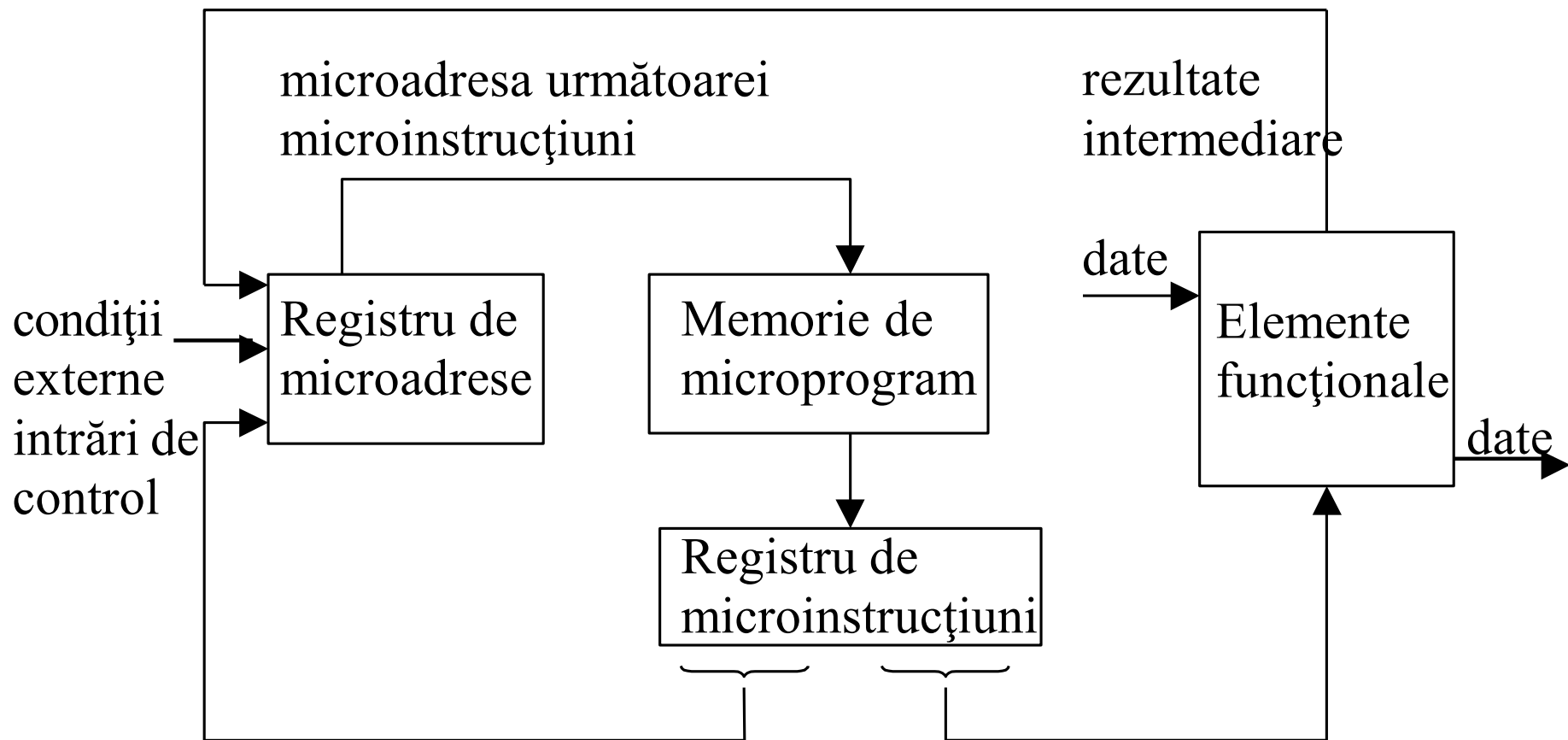


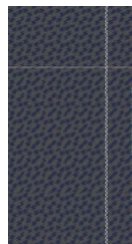
# Structura de control microprogramata

## Observație

- **microprogramarea** este o **tehnică de proiectare** a structurilor numerice - se referă la o modalitate de implementare, la o arhitectură
- a nu se confunda cu programarea microprocesoarelor!!!

# Structura de control microprogramata





# Structura de control microprogramata

## Funcții

- **1. funcția de control** - definește și controlează toate microoperațiile care trebuie executate
- **2. funcția de secvențiere** - determină și controlează adresa microinstrucțiunii următoare
- funcțiile de regăsesc în cuvântul de microinstrucțiune





# Structura de control microprogramata

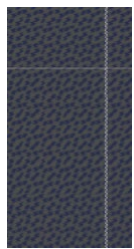
## Model

- instrucțiunea - apelează o operație
- operația implică secvențe de pași = **microoperații**
- exemple de microoperații:
  - transfer registru - registru
  - transfer registru - memorie
  - operații aritmetice
  - operații logice etc.

# Structura de control microprogramata

## Microoperații - exemple

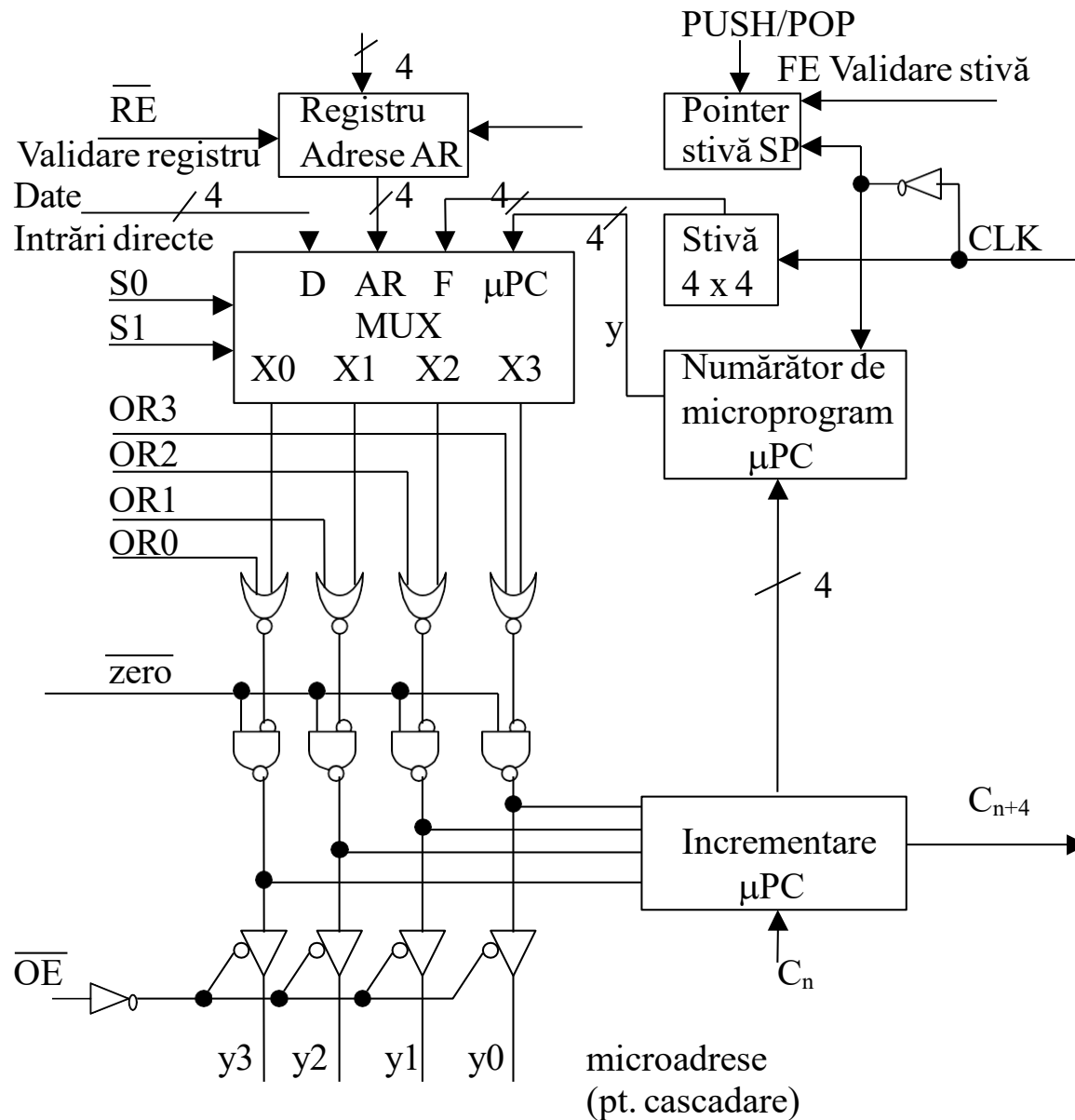
- selecție operand pentru unitatea aritmetico-logică (ALU) – **enable** pentru memoria ROM (eventual)
- funcția executată de ALU – transmite **cod operatie**
- destinația rezultatului ALU – **load si enable** pt numparator
- control transport (carry sau borrow)
- control întreruperi - **enable**
- control intrări / ieșiri – **enable** Buffer
- Control resurse hardware – **selectii MUX**



# Memoria de microprograme

- tipuri: ROM sau RAM
- Obiectiv: incapsularea semnalelor de control + starea urmatoare
- **Organizare – mai multe metode:**
  - 1. fiecărui cuvânt de memorie îi corespunde o microinstrucțiune → pentru citire se folosește un singur ciclu de acces la memorie
  - 2. un cuvânt conține mai multe microinstrucțiuni → se reduce numărul de accese la memorie → crește viteza de lucru
  - 3. memorie împărțită în blocuri → există 2 tipuri de microadrese → se micșorează lungimea cuvântului
  - 4. memorie divizată → sunt 2 tipuri de microinstrucțiuni:
    - simple - număr mic de biți
    - complexe - lungime mare, controlează simultan mai multe resurse
  - 5. memorie structurată pe 2 nivele:
    - nanoprograme - microprograme de nivel scăzut
    - nanoinstrucțiuni – microinstrucțiuni de nivel scăzut

# Generarea $\mu$ -instrucțiunii următoare



# Generarea $\mu$ -instrucțiunii următoare

## Generare adresă următoare

- adresa următoare selectată prin multiplexare:
  - intrări directe (exterioare)
  - intrări de la registrul de adrese intern
  - intrări de la stivă
  - intrări de la numărătorul de microprogram (program counter PC)
- MUX selectat (comandat) cu S1 și S0
- registrul de adrese validat cu RE

# Generarea $\mu$ -instrucțiunii următoare

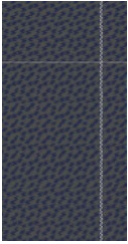
## Generare adresă următoare

- numărătorul de microprogram - registru pe 4 biți + CLC de incrementare cu 1 ( $C_n = 1 \Rightarrow \text{adresa curentă} + 1$ )
  - stiva 4 x 4 memorează adrese de revenire din microsubrutine
    - pointerul de stivă SP indică ultima locație scrisă în stivă
    - combinațiile între FE și PUSH/POP asigură memorarea în, respectiv citirea din stivă
  - ieșirile y indică microadresa următoare
-

# Generarea $\mu$ -instrucțiunii următoare

## Generare adresă următoare

- intrările OR pot forța anumite linii de adresă la 1 logic permițând astfel realizarea instrucțiunilor de salt
- zero face inițializarea adresei de memorie
- OE permite trecerea ieșirilor de adresă în regim three-state



# Formatul $\mu$ -instrucțiunii

- **Definiție:** formatul microinstrucțiunii = împărțirea microinstrucțiunii în zone de control numite **câmpuri**
  - clasificarea microinstrucțiunilor:
    - verticale
    - orizontale
-

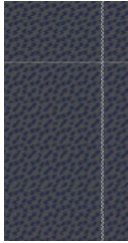


# Formatul $\mu$ -instrucțiunii

## ■ microinstrucțiuni verticale:

- operații simple ( $\mu$ op)
- până la 24 biți
- compromis între lungimea cuvântului și lungimea microprogramului
- exemplu:

cod	$\mu$ op. cu UC
cod	$\mu$ op cu MEM
cod	$\mu$ op. cu I/O
cod	$\mu$ op. salt



# Formatul $\mu$ -instrucțiunii

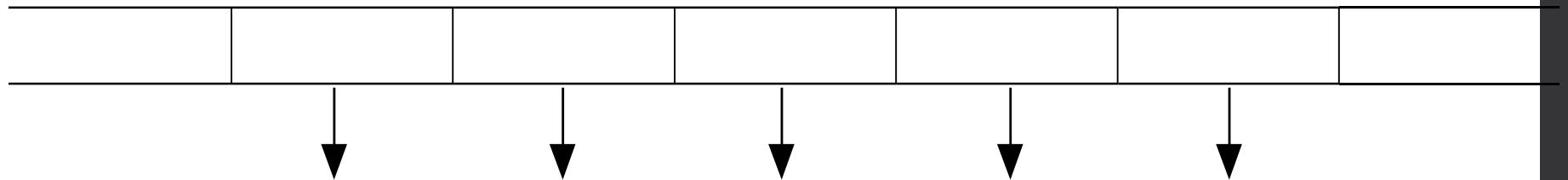
## ■ microinstrucțiuni orizontale

- controlează mai multe resurse în paralel
- lungime de 64 de biți
- codificarea microinstrucțiunii

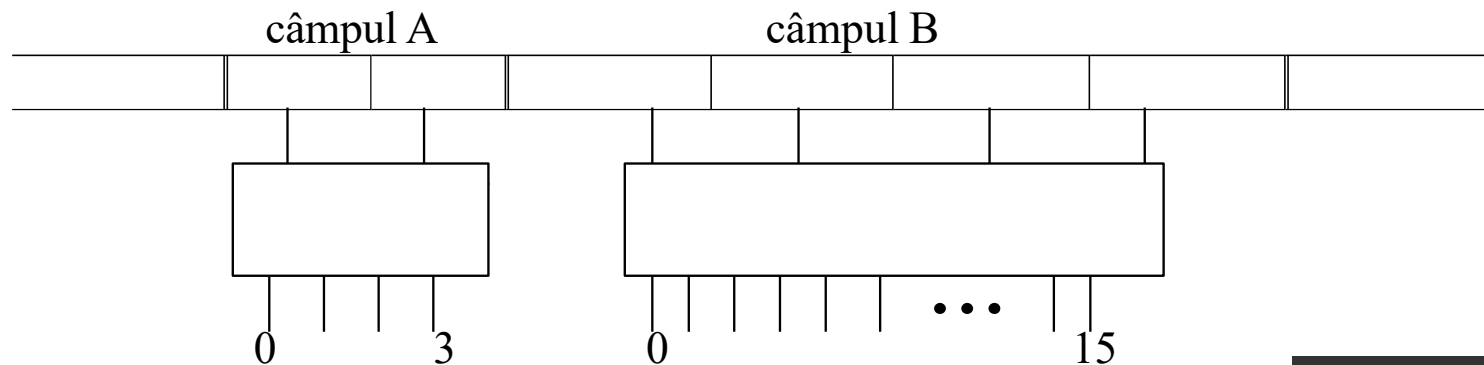
Urmatoarea instrucțiune	Control ALU	Control UC	Control MEM	Control I/O	Control ceas
----------------------------	----------------	------------	----------------	-------------	-----------------

# Formatul $\mu$ -instrucțiunii

- microinstrucțiuni orizontale - **codificarea**
  - a. fără codificare - fiecare bit = o microcomandă (controlează o microoperație)

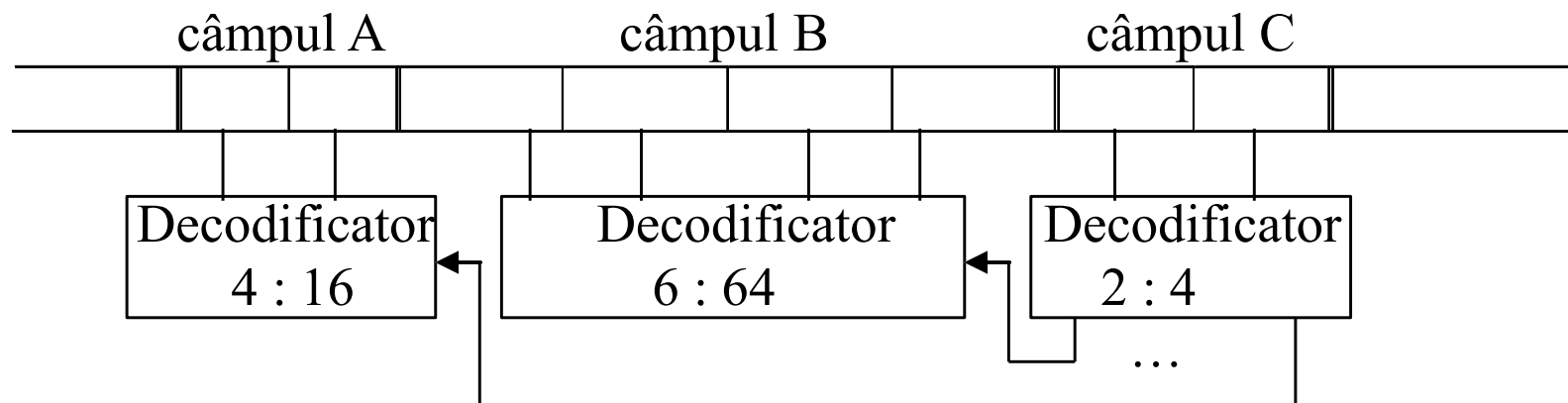


- b. codificare pe un nivel - câmpuri diferite



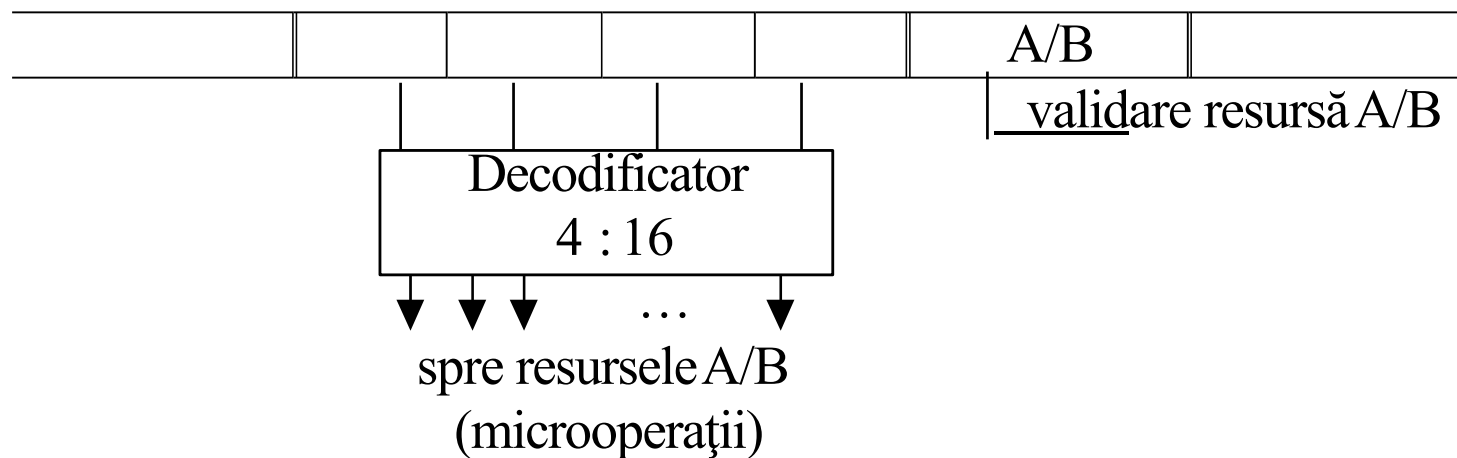
# Formatul $\mu$ -instrucțiunii

- microinstrucțiuni orizontale - **codificarea**
  - c. codificare pe 2 niveluri - semnificația unui câmp depinde de valoarea altui câmp de control



# Formatul $\mu$ -instrucțiunii

- microinstrucțiuni orizontale - **codificarea**
  - d. același câmp controlează resurse hardware diferite, care au funcționare separată în timp; dirijarea câmpului se face cu ajutorul unui câmp separat de control





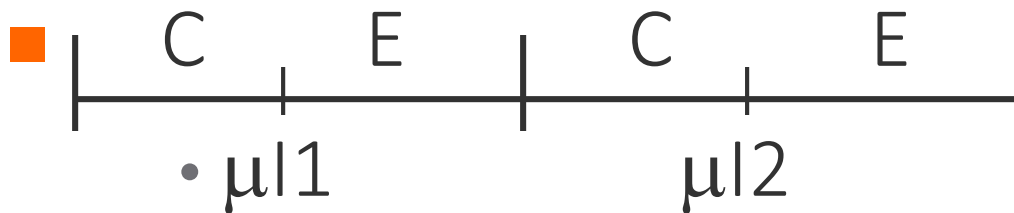
# Implementarea $\mu$ -instrucțiunii

## Execuția microinstrucțiunii

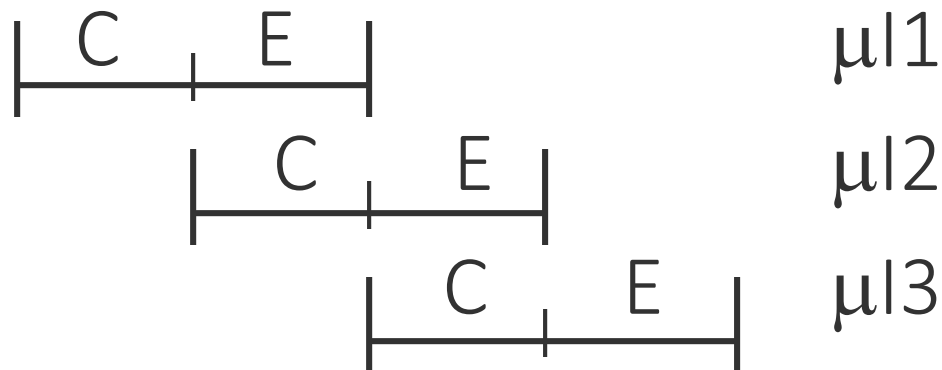
- 1. citirea **C**
  - determinarea adresei următoare
  - obținerea datelor și încărcarea în registrul de microinstrucțiuni
- 2. decodificarea
- 3. execuția propriu-zisă **E**

# Implementarea $\mu$ -instrucțiunii

- Implementarea serie



- pentru creșterea vitezei  $\rightarrow$  suprapunerea execuției microinstrucțiunii curente cu citirea microinstrucțiunii următoare (probleme la salt)



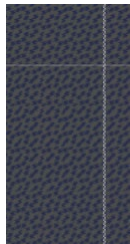


# Implementarea $\mu$ -instrucțiunii

## Caracteristica monofază - polifază

- un ciclu principal de execuție a microinstrucțiunii poate avea mai multe faze utilizate
  - **implementare monofază:**
    - microinstrucțiunea durează doar o perioadă de tact (clock)
    - semnalele de control - generate simultan cu execuția microinstrucțiunii
  - **implementare polifază:**
    - fiecare ciclu principal conține cicluri secundare
    - semnalele de comandă se generează secvențial
-





# Microprogramare

## Avantaje și dezavantaje

### ■ avantaje:

- flexibilitatea modului de control
- adaptabilitate
- ușurință în dezvoltare și întreținere
- preț mai scăzut

### ■ dezavantaje:

- putere limitată a microinstrucțiunii - pierdere de performanță
  - volum și putere consumată mai mari
-



# Concluzii

- Circuit = Unitate Control + Unitate Executie
  - Unitate Control – Cablata sau microprogramata
  - Unitatea microprogramata
    - Foloseste o memorie
    - Tine semnalele de control si microinstructiunea urmatoare
    - Microoperatii – genereaza semnalele de control pt UE
    - Se pot introduce microinstructiuni si cu rol de salt – ordine nu e obligatoriu secventiala
    - Mai multe formate pt microinstructiuni
    - Mai multe moduri de implementare – pe eficientizare
  - Data viitoare: implementare pas-cu-pas a unei UC microprogramate
-