

# MICROPROGRAMARE

## **Noțiuni și concepte de bază**

Organizarea memoriei de control

Suportul fizic pentru păstrarea microprogramelor

Organizarea logică a instrucțiunilor

Codificare verticală

Codificarea orizontală

Codificarea minimală

Codificarea cu control rezidual

Codificarea cu control prin adrese

Codificare mixtă

Implementarea microinstrucțiunilor

# NOTIUNI SI CONCEPTE DE BAZĂ



Din punct de vedere structural unitățile de comandă sunt de două tipuri:

- convenționale în sensul propus de Von Newmann ;
- microprogramate conform conceptului introdus de M.Wilkes.

Microprogramarea este o tehnică de proiectare și implementare a funcțiilor de control a sistemelor de prelucrare a datelor numerice, ca o secvență de semnale de control ce interpretează static sau dinamic funcțiile de prelucrare a datelor. Semnalele de comandă, necesare la un moment dat pentru controlul primitivelor funcționale sunt organizate într-un cuvânt de control, memorat într-o memorie PROM sau RAM. Structura cuvântului de control este influențată de semnificația atribuită noțiunii de microoperație .

Microoperația (  $\mu O$  ) este o primitivă a funcțiilor de prelucrare a datelor, care reprezintă o operație elementară asupra unei primitive funcționale (transfer, acțiuni de înscriere sau de incrementare, activare pe magistrală, etc. ), ce se desfășoară de obicei într-o perioadă de timp (perioada de tact sau de ceas).

Microoperația implică existența unui operator, care corespunde unei unități funcționale și a operanzilor asupra cărora acționează.


Cuvântul de control este pus în corespondență cu noțiunea de microinstrucțiune.

Microinstrucțiunea ( $\mu I$ ) este un set de microoperații independente de date, fără conflict de resurse, care se pot executa simultan (pe perioada unei perioade de sincronizare).



# TIPURI DE MICROUNSTRUCTIUNI

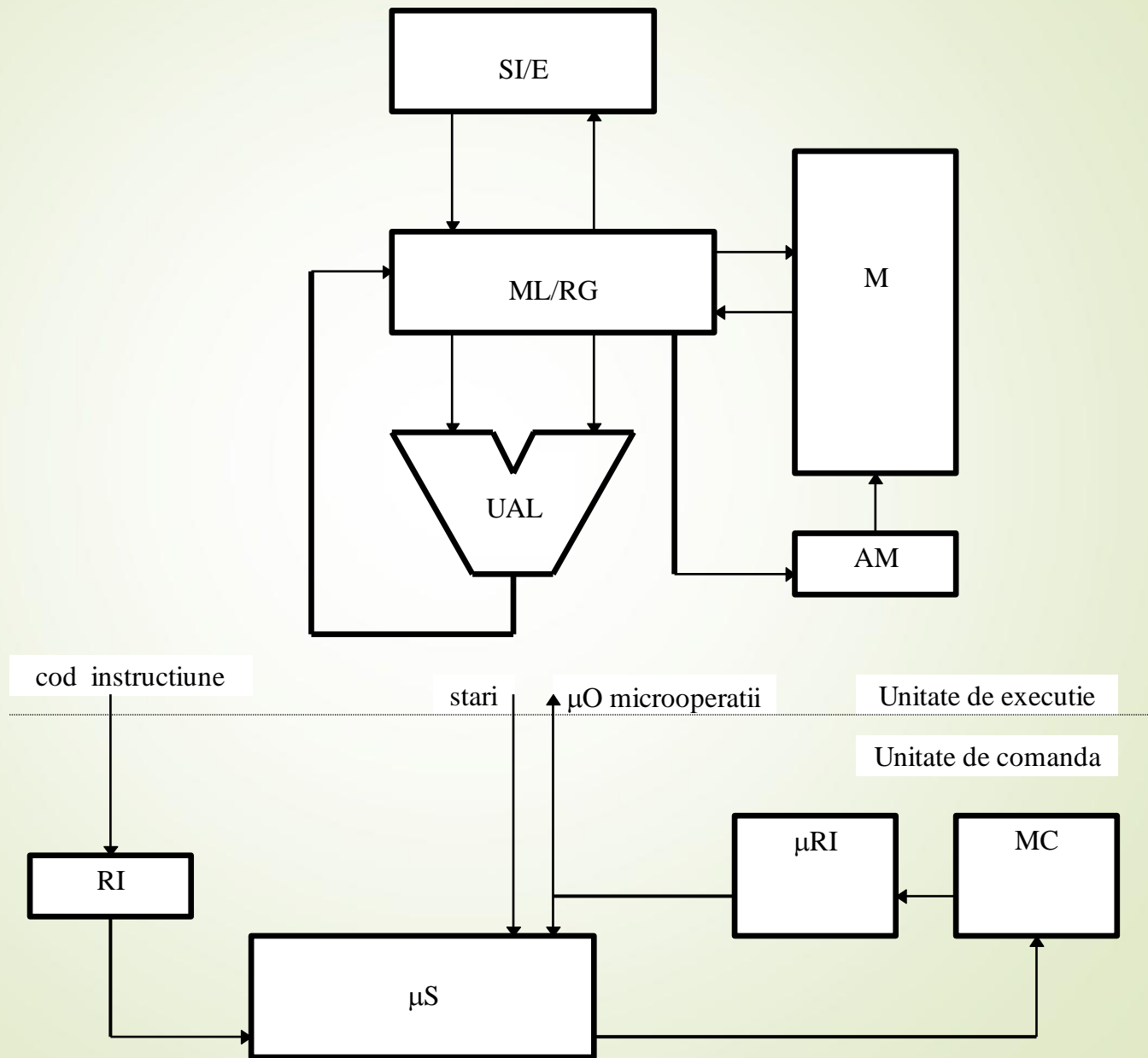
- microinstrucțiuni operaționale care controlează primitivele funcționale ale unității de execuție a sistemului numeric, asigurând fluxul de informație și acțiunile asupra resurselor;
- microinstrucțiune de ramificație (de salt) care inspectează starea primitivelor funcționale și asigură ramificația în algoritmul de control, constituind suportul pentru implementarea deciziilor.



Prin microprogram se înțelege o secvență de microinstrucțiuni ce implementează un algoritm care descrie :

- citirea interpretarea și execuția unui set de instrucțiuni mașină;
- primitive ale sistemului de operare;
- primitive ale limbajelor de programare;
- etc.

# Structura generală a unui sistem de calcul cu unitate de comandă microprogramată



# PRIMITIVE FUNCTIONALE



- **M** - memoria principală a sistemului, în care se păstrează programele ca secvență de instrucțiuni mașină și datele care se prelucrează ;
- **ML** - memoria locală a sistemului, reprezentată de registrele generale de lucru accesibile sau nu utilizatorului;
- **UAL** - unitatea aritmetică logică ;
- **S I/E** - subsistemul de intrări / ieșiri ;
- **MC** - memoria de control în care se păstrează microprogramul ca secvență de microinstrucțiuni ;
- **μRI** - registrul de microinstrucțiuni, care păstrează microinstrucțiunea curentă ce se execută. Conținutul său specifică toate microoperațiile care se execută în acel moment în unitatea de execuție ;
- **μS** - microsecvențiatorul, unitatea de comandă convențională, elementară, care asigură citirea interpretarea și execuția microinstrucțiunilor din memoria de control precum și înlănțuirea acestora pe baza registrului de instrucțiuni mașină RI și a stării primitivelor funcționale.



O structură microprogramată se caracterizează prin:

- organizarea memoriei de control ;
- suportul fizic pentru păstrarea microprogramelor
- organizarea logică a microinstrucțiunilor ;
- implementarea microinstrucțiunilor



# 9 1.1.1. ORGANIZAREA MEMORIEI DE CONTROL

Dupa criteriu de analiză relația MC față de M:

- memorie de control separată de memoria principală, atât din punct de vedere fizic cât și din punct de vedere al adresării logice,
- Raportul dintre viteza de lucru a memoriei de control și cea a memoriei principale este de circa 10 în favoarea memoriei de control,  $V_{mc} \gg V_m$
- memoria de control este implementată în același spațiu fizic și de adresare cu cel al memoriei principale. Ambele au același ciclu de memorie și sunt adresate prin aceeași logică. Divizarea în memorie de control și memorie principală se poate face la nivel fizic sau la nivel logic.

Acest tip de organizare cere ca memoria să fie suficient de rapidă pentru a fi folosită ca MC și destul de ieftină pentru a fi folosită ca memorie principală.

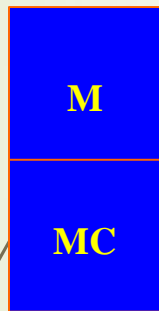
-memoria de control este implementată separat de memoria principală însă este încărcată din aceasta. Se utilizează un sistem de memorie ierarhică

Memoria MC este încărcată din memoria principală prin intermediul memoriei tampon MT.

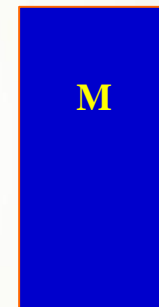




## Organizarea memoriei de control



## Organizarea MC în cadrul MP

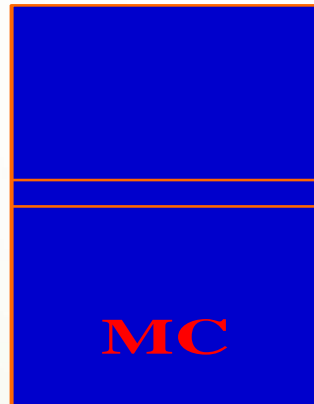


## Organizarea MC separată de MP dar incarcabilă din aceasta



# CLASIFICAREA MEMORIEI DE CONTROL MC

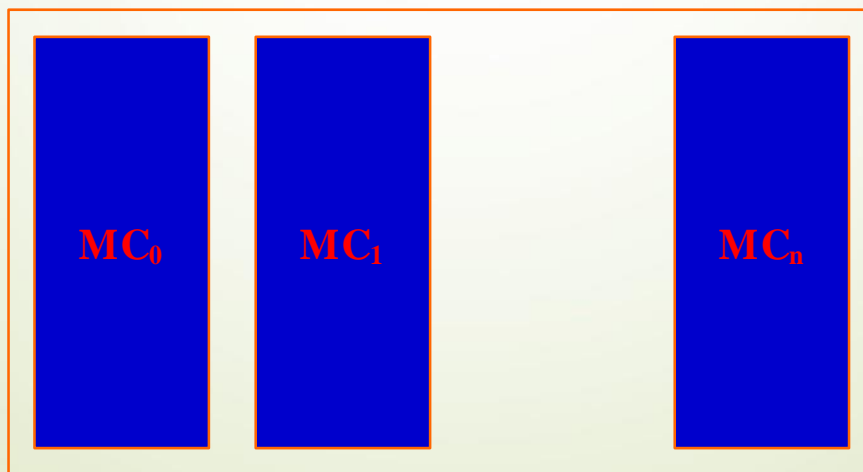
1. **memorie de control cu o microinstrucțiune pe cuvânt**, în care fiecărui cuvânt de memorie îi corespunde o singură microinstrucțiune. Citirea unei microinstrucțiuni presupune un singur acces la MC.



## 2. *memorie de control cu organizare pe pagini*

Unei adrese din memoria de control i se asociază mai multe microinstrucțiuni, din pagini diferite. În acest fel se asigură la nivel de microprogram execuția unei microinstrucțiuni de tip CASE, care introduce facilitatea de decizii multiple.

O adresă de MC va adresa aceeași locație în toate paginile memoriei de control iar vectorul de condiții de test va activa pagina ce specifică microinstrucțiunea următoare care se va executa.

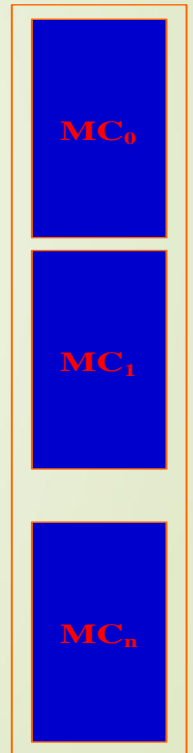


### 3. *memorie de control cu organizare pe blocuri*

Pentru acest tip de organizare există două feluri de adrese:

- adrese de microinstrucțiuni din același bloc cu microinstrucțiunea curentă ;
- adrese de blocuri.

Împărțirea memoriei de microprograme în blocuri se face ținând seama atât de structura microprogramului cât și de resursele fizice disponibile. O organizare de acest fel conduce, în general, la micșorarea lungimii microinstrucțiunii însă introduce timp suplimentar cu comutarea adreselor de blocuri.



#### 4. *memorie de control divizată*

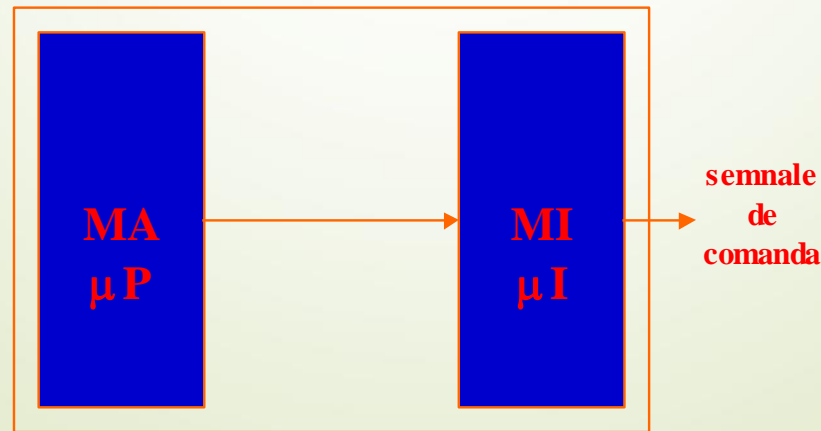
Memoria de microprograme divizată este alcătuită din două unități de memorie distincte :

MI - memorie de microinstrucțiuni, care păstrează toate microinstrucțiunile distincte posibile necesare pentru controlul resurselor fizice ;

MA -memorie de adrese de microinstrucțiuni, care păstrează programul specificat, nu prin microinstrucțiuni ce controlează resursele fizice, ci prin adrese de microinstrucțiuni (adrese pentru memoria MI).

În general numărul de tipuri de microinstrucțiuni distincte este mult mai redus decât microprogramul în sine, ceea ce implică ca numărul de biți necesari pentru adresarea memoriei MI să fie și el redus. În acest fel, lungimea cuvântului din memoria MA este mult mai mic decât al memoriei MI, ceea ce conduce la o reducere substanțială a memoriei de control.

În schimb, pentru a executa o microinstrucțiune trebuie făcute două adresări, una la memoria MA și una la memoria MI ceea ce conduce la un ciclu mai mare de microinstrucțiune.



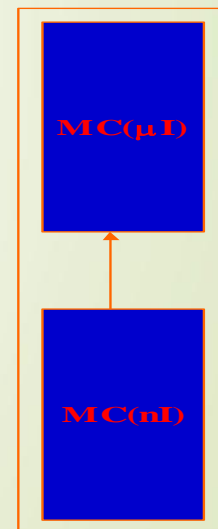
## 5. *memorie de control structurată pe două niveluri*

La o astfel de organizare mecanismul citirii interpretării și execuției unei instrucțiuni mașină este următorul :

- o instrucțiune mașină este interpretată de un set de microinstrucțiuni rezident în memoria de control MC ( $\mu I$ ) ;
- la rândul ei, o microinstrucțiune este interpretată de o secvență de nanoinstrucțiuni rezidentă în memoria MC (nI) .

Această tehnică a nanoprogramării este conceptual echivalentă cu microprogramarea, structurarea pe două niveluri oferă o flexibilitate mai mare și posibilitatea implementării unor structuri de control foarte complexe.

O astfel de organizare întâlnim la calculatorul NANODATA QM1.





# SUPORTUL FIZIC PENTRU PASTRAREA MICROPROGRAMELOR

**Memoria de microprograme MC este o unitate de memorie de mare viteză care păstrează microprogramele ce se execută.**

**În ceea ce privește suportul fizic, pentru păstrarea microprogramelor, acesta poate fi realizat cu memorii PROM sau cu memorii RAM.**

**Realizarea memoriei de control :**

- cu componente de tip RAM conferă o caracteristică statică**
- componentelor de tip RAM oferă o caracteristică dinamică ce permite utilizatorului ca printr-un ansamblu de mijloace software să aibă acces la microprogramul din memoria de control..**

**Este foarte important în a face deosebire între :**

- mașini microprogramate, și**
- mașini microprogramabile**

**Prima categorie se referă la modalitatea de implementare a unității de comandă în sensul conceptului introdus de Wilkes fără a oferi resursele hardware și suportul de programe pentru accesul utilizatorului la nivelul microprogramului.**

**Cea de a doua categorie oferă atât resursele hardware cât și facilitățile software pentru accesul utilizatorului la nivelul microinstrucțiunilor.**



# ORGANIZAREA LOGICA A INSTRUCTIUNILOR

Un cuvânt din memoria de control specifică un set de microoperații ce

constituie componentele primitive ale controlului resurselor sistemului.

Organizarea logică a microinstrucțiunilor este influențată de :

- gradul de paralelism între microoperații, ce se dorește realizat;
- structura mașinii de bază ;
- gradul de codificare sau de flexibilitate dorit ;
- gradul de optimizare al lungimii cuvântului de control .

Presupunând că microprogramele sunt specificate ca o secvență de

seturi disjuncte de microoperații, se pot distinge mai multe modalități de codificare a acestora și anume:

- codificare verticală sau maximală ;
- codificare orizontală sau cu control direct ;
- codificare minimală ;
- codificare cu control rezidual ;
- codificare cu control prin adrese ;
- codificare mixtă.





# CODIFICARE VERTICALA

Fiecare microinstrucțiune operațională specifică o singură microoperație. Setul de microoperații (MO) necesar pentru controlul primitivelor funcționale se codifică în  $\lceil \log_2 |MO| \rceil$  biți, ce constituie lungimea cuvântului din MC.

Această codificare reprezintă un caz extrem, deoarece elimină orice posibilitate

de desfășurare paralelă a operațiilor elementare.

Din punctul de vedere al minimizării cuvântului de control, codificarea verticală

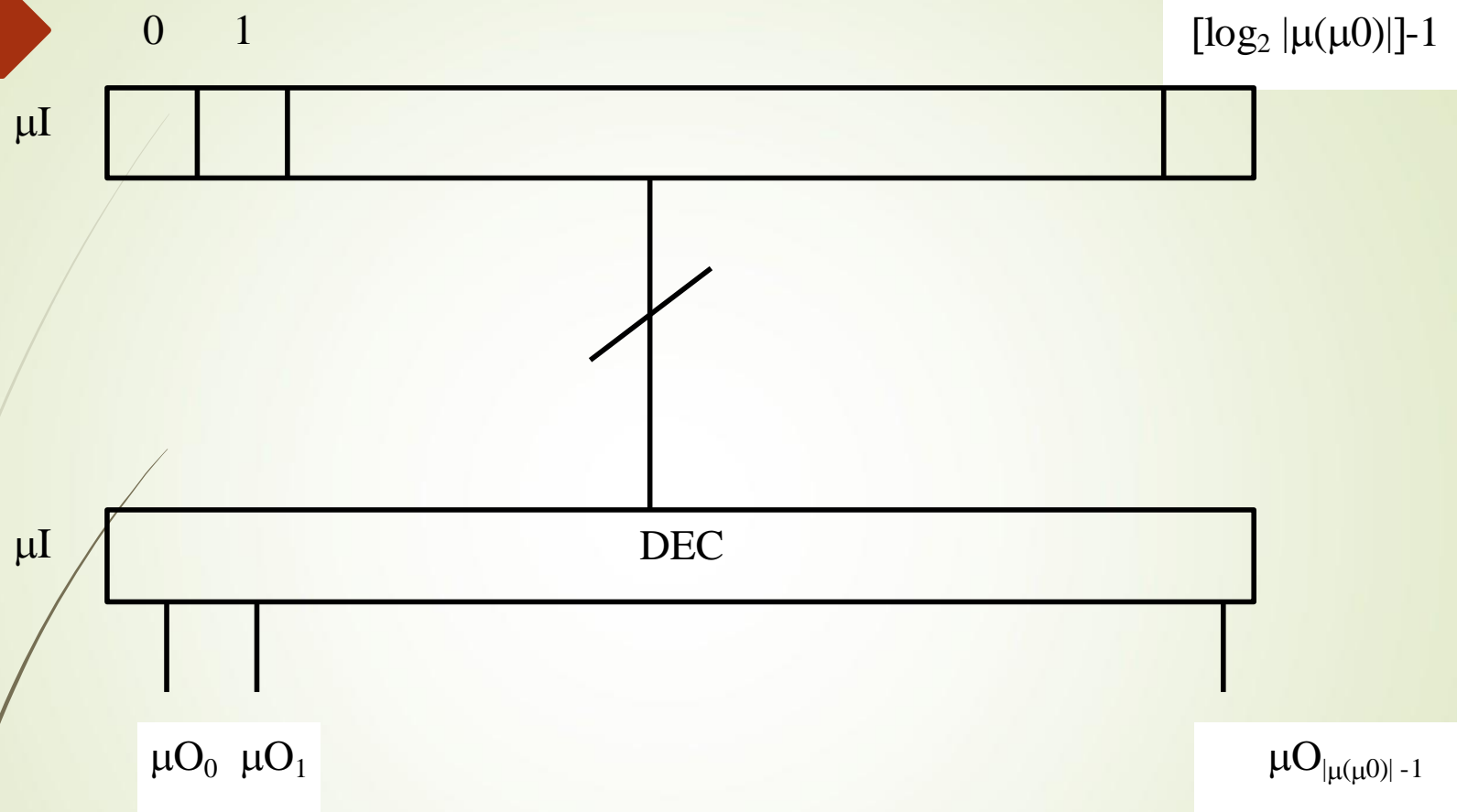
implică numărul cel mai mic de biți. Dimensiunea mare a decodicatorului face ca

realizarea fizică a acestuia să aibe loc pe mai multe niveluri, ceea ce conduce la

introducerea de întârzieri

Un dezavantaj major al codificării maxime îl reprezintă eliminarea controlului

paralel asupra resurselor precum și inflexibilitatea dezvoltării sau completării sistemului în ceea ce privește introducerea de noi microoperații. Este aplicabilă numai în sisteme dedicate care au o structură specifică.

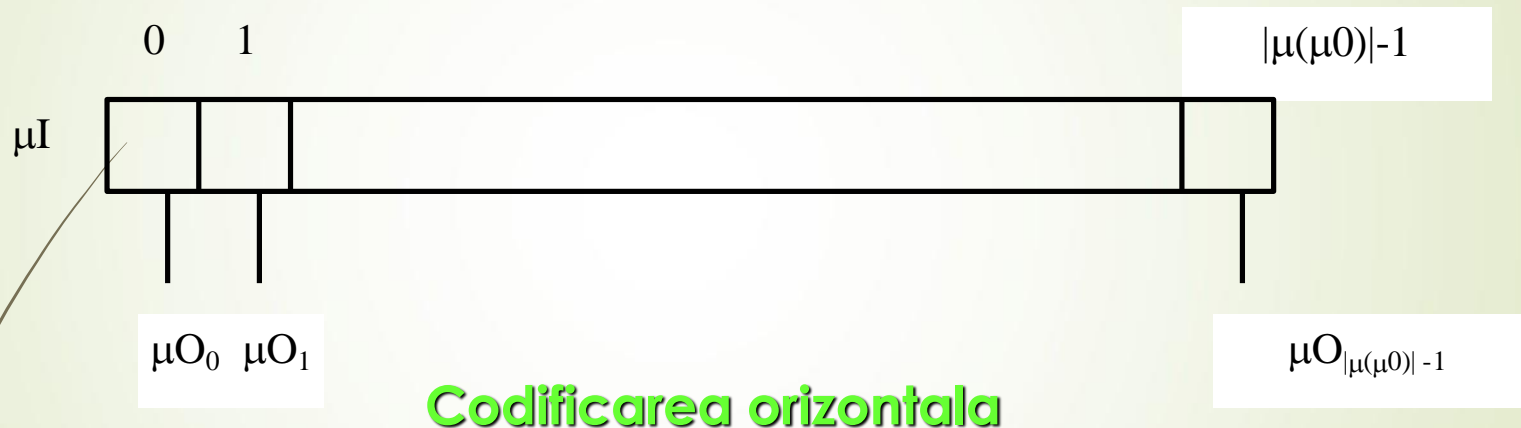


**Codificarea verticală a microoperațiilor**



# CODIFICAREA ORIZONTALA

În cadrul acestei codificări, fiecare microoperație din setul (MO) este pusă în corespondență cu un bit din cadrul cuvântului de control. Controlul microoperațiilor se face în mod direct.



Codificarea orizontală realizează controlul tuturor microoperațiilor paralele, posibile, ce se pot desfășura în sistem.

Deși oferă o flexibilitate mare și asigură paralelismul maxim, utilizarea acestei codificări este un caz extrem datorită folosirii ineficiente a memoriei de control.

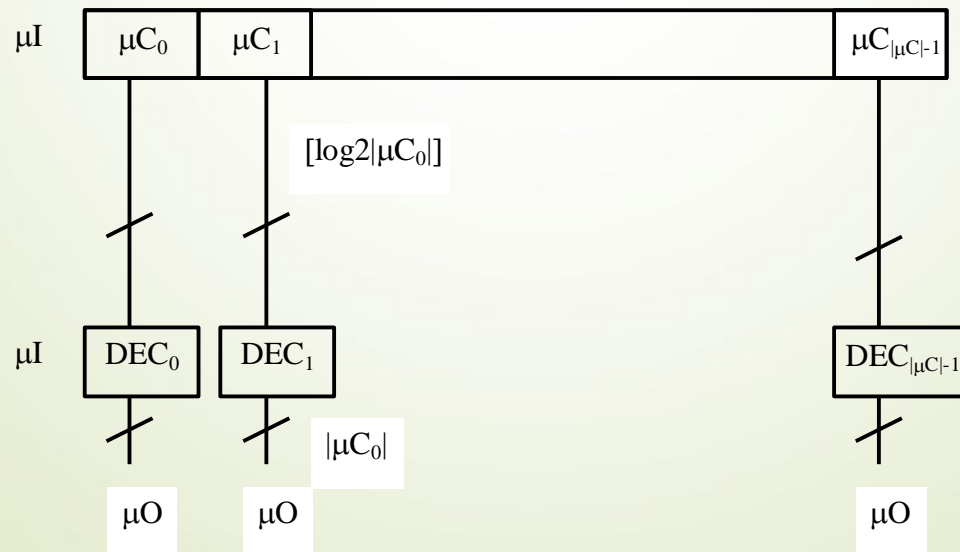


# CODIFICAREA MINIMALA

Combină flexibilitatea și paralelismul potențial oferite de codificarea orizontală cu eficiența codificării verticale.

Ideea de bază este de a grupa în clase de compatibilitate setul de microoperații care se exclud reciproc ( $\mu O$  dintr-o clasă de compatibilitate nu se vor efectua niciodată simultan).

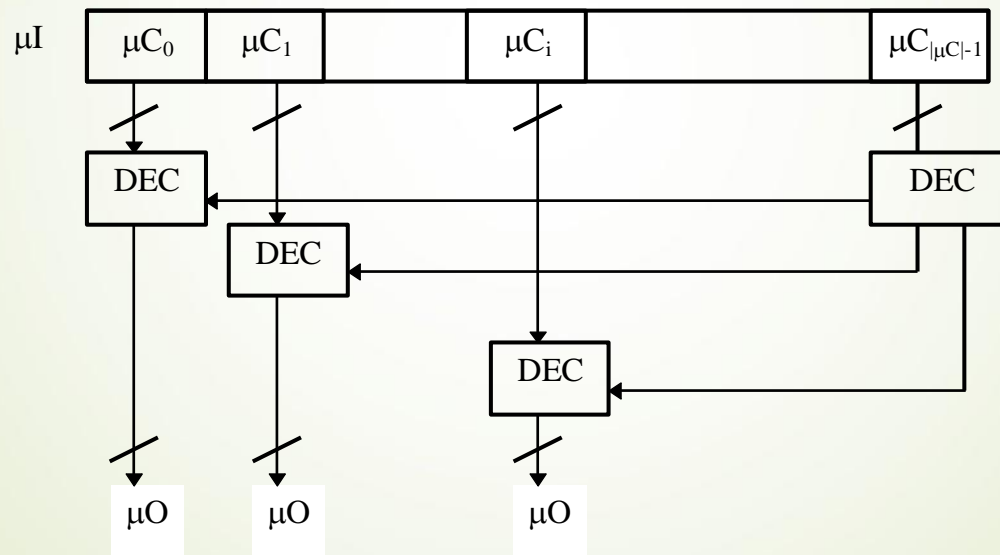
Microinstrucțiunea este împărțită în câmpuri. Un câmp corespunde unei clase de compatibilitate. La nivel de câmpuri se realizează o codificare orizontală iar în cadrul câmpurilor se realizează o codificare verticală.



Pentru un câmp  $\mu C_j$  care codifică  $|\mu C_j|$  microoperații sunt necesari  $\lceil \log_2(|\mu C_j| + 1) \rceil$  biți, deoarece trebuie să se prevadă și posibilitatea de a nu specifica nici o microoperație din cadrul câmpului.

O variantă a acestei codificări o reprezintă codificarea pe două niveluri sau indirectă.

În codificarea pe două niveluri, validarea unor câmpuri depinde de valoarea altui câmp de control din microinstrucțiune.

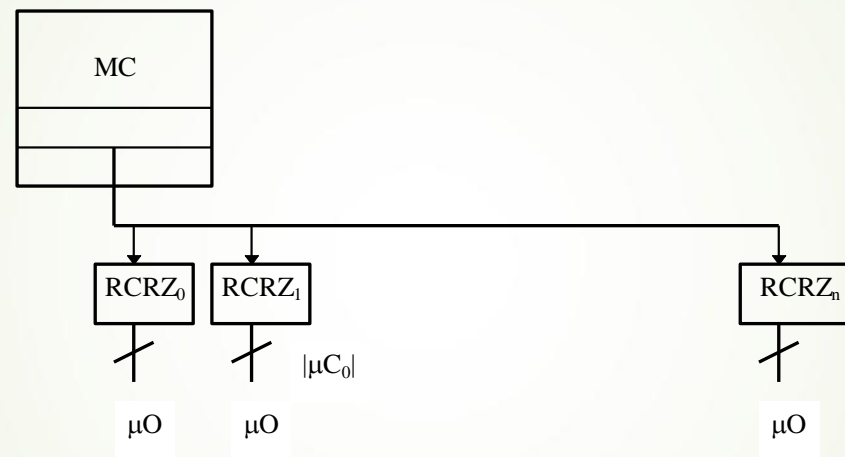


**Codificare minimală pe două niveluri**



# CODIFICAREA CU CONTROL REZIDUAL

Această metodă de codificare folosește pentru controlul primitivelor funcționale registre de control rezidual. Cuvântul de control nu controlează direct resursele ci prin intermediul registrelor de control rezidual încărcate sub acțiunea microinstrucțiunilor



## Codificare cu control rezidual

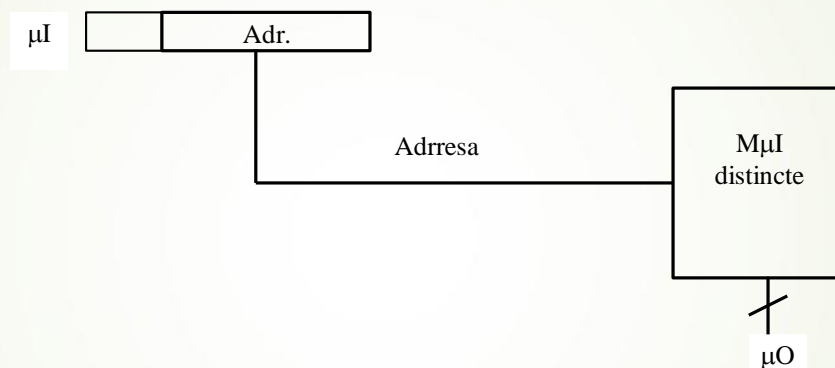
Microinstrucțiunile pot înlocui sau modifica valoarea unuia sau mai multor registre de control. Astfel, se asigură o economie de memorie de control atunci când unele primitive funcționale realizează aceeași operație în mod repetat sau când un set de microoperații este activ o perioadă mare de timp, iar alte seturi de microoperații se modifică.

Registrele de control rezidual  $RCRZ_j$ , care specifică microoperațiile de control al resurselor hardware pot fi manevrate cu ajutorul unor microinstrucțiuni de dimensiuni reduse.



# CODIFICAREA CU CONTROL PRIN ADRESE

O modalitate de implementare a microinstrucțiunilor operaționale este aceea în care se specifică o adresă în cadrul unei memorii, unde sunt memorate toate microinstrucțiunile distincte posibile ce controlează sistemul.



## Codificare cu control prin adrese

Numărul de microinstrucțiuni distincte nu depinde de numărul de resurse controlate ci de numărul de  $\mu O$  distincte, de mărimea  $\mu P$  și de numărul de variabile de stare testate.

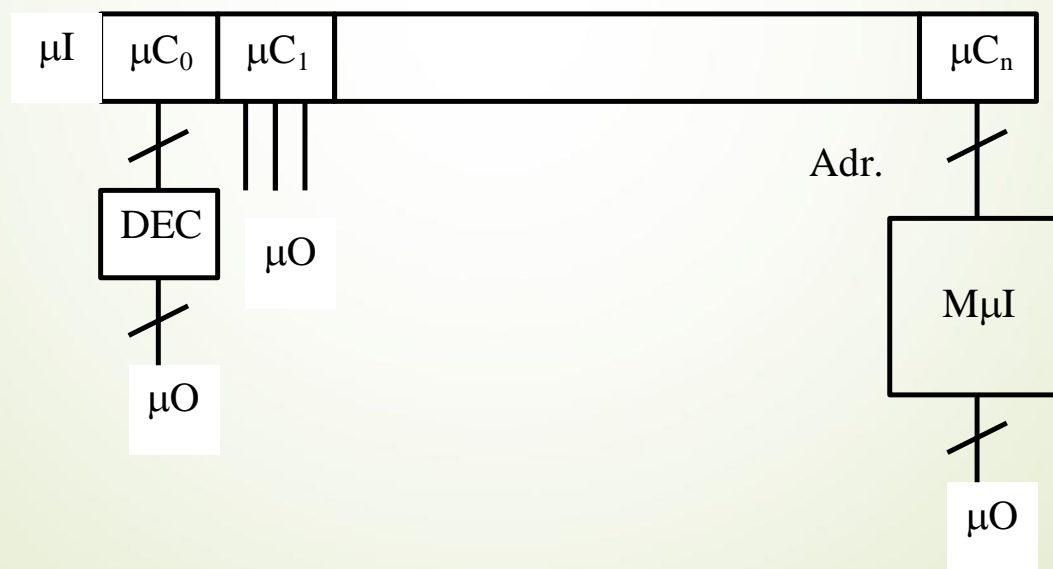
Memoria care păstrează microinstrucțiunile distincte va avea lungimea cuvântului suficient de mare pentru a controla toate microoperațiile care se pot efectua simultan. Trebuie notat că fiecare  $\mu I$  este memorată o singură dată.

O astfel de implementare, face ca microprogramul să fie format dintr-o secvență de adrese care apelează  $\mu I$  păstrate în memoria de  $\mu I$ .



# CODIFICAREA MIXTA

O variantă utilizată mult în practică este aceea în care microinstrucțiunea este împărțită în câmpuri. Unele câmpuri controlează direct microoperațiile (sub formă codificată sau directă) iar altele specifică adrese de memorie ce conține un subset de microinstrucțiuni distincte.







# IMPLEMENTAREA INSTRUCȚIUNILOR

Microinstrucțiunile sunt citite, interpretate și executate de către microsecventiator ( $\mu S$ ) în același fel în care o unitate de comandă convențională execută instrucțiuni mașină.

Se poate defini o caracteristică serie-paralel care măsoară cantitatea de suprapunere între faza de execuție a  $\mu l$  curenți și citirea, interpretarea  $\mu l$  următoare.

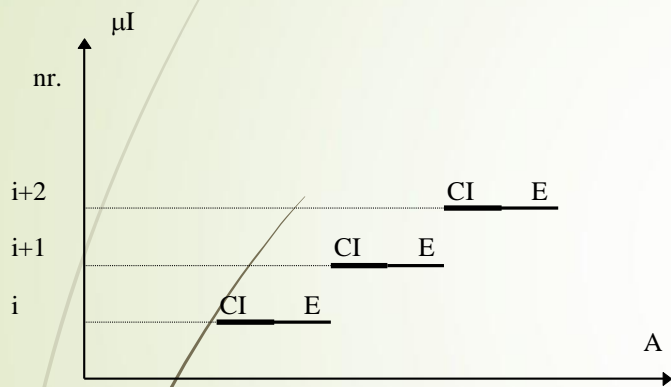
Din punctul de vedere al caracteristicii serie-paralel distingem trei tipuri de implementări :

- implementare serie;
- implementare serie-paralelă ;
- implementare paralelă.

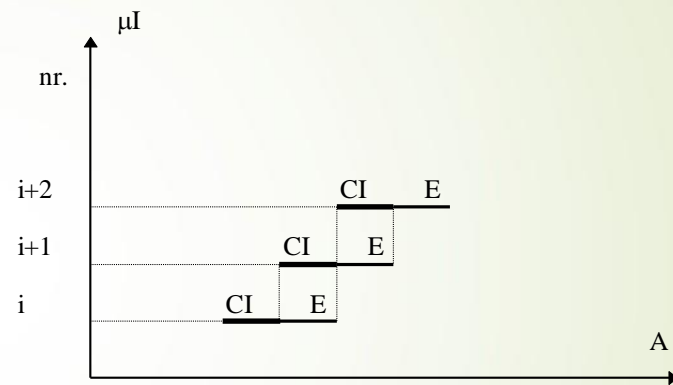
Fie CI-faza de citire interpretare și E-faza de execuție a unei  $\mu l$ .

În implementarea serie, citirea microinstrucțiunii următoare nu începe decât după ce s-a terminat execuția microinstrucțiunii curenți.

În implementarea paralelă, Fig. 9.17, faza de citire a  $\mu l$  următoare se desfășoară în același timp cu execuția  $\mu l$  curenți.

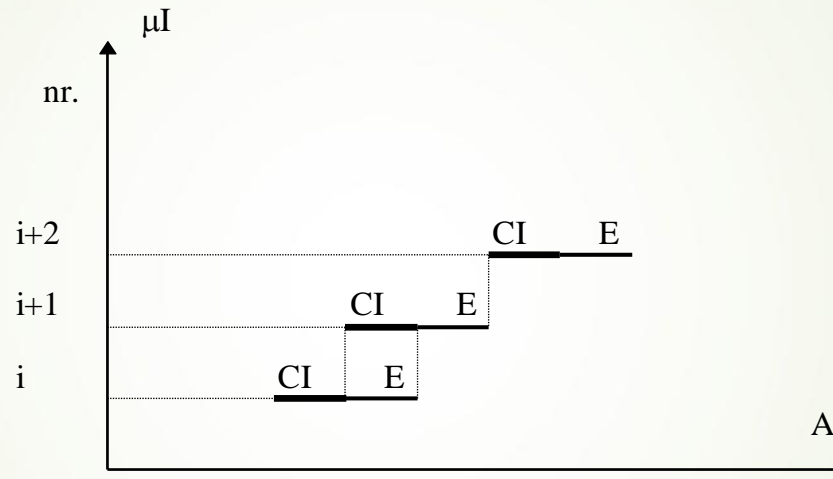


Implementare serie



Implementare paralel

O îmbinare a performanțelor implementării paralele și a costului redus al implementării serie este realizată de implementarea serie-paralelă a microinstrucțiunilor din figura următoare.



**Implementare serie-paralel**



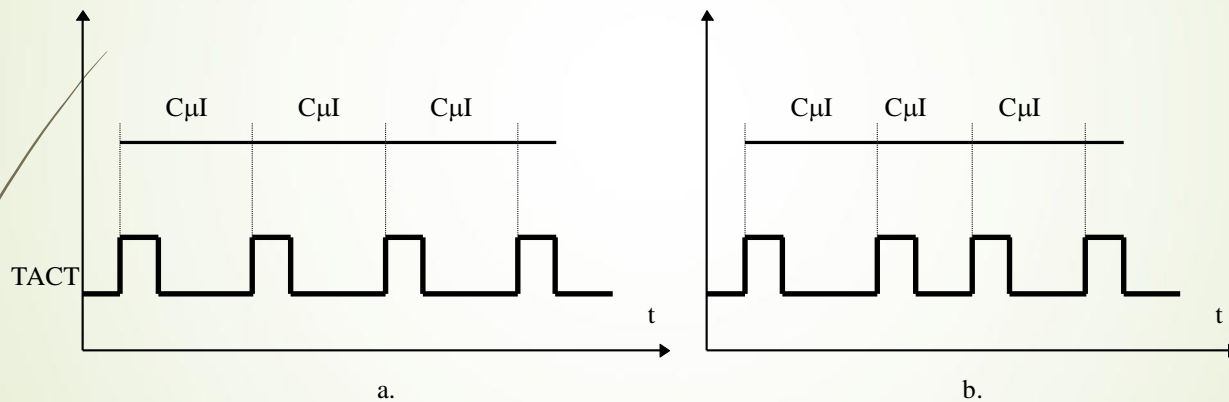
Faza de citire a  $\mu$ l următoare se desfășoară fie în timpul execuției  $\mu$ l curente, fie după terminarea ei, în funcție de tipul microinstrucțiunii curente.

Sucesiunea de microinstrucțiuni operaționale se desfășoară prin suprapunerea fazelor de citire și execuție, iar cele imediat următoare unor ramificații se citesc după terminarea execuției  $\mu$ l curente (de ramificație).

Implementarea serie-paralelă este caracteristică arhitecturilor "pipe line", și este cea mai folosită.

Referitor la implementarea microinstrucțiunilor se poate defini și o caracteristică monofază-polifază care se referă la numărul de faze utilizate într-un ciclu de microinstrucțiune.

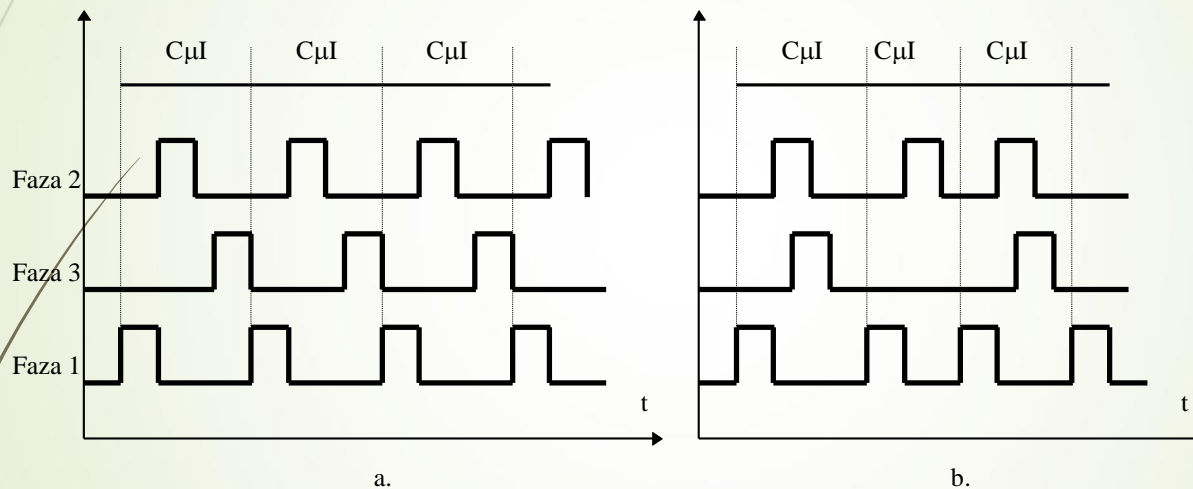
Într-o implementare monofază, microoperațiile sunt generate simultan, toate semnalele de control specificate de microinstrucțiune fiind active în același timp (pentru cele de tip impuls se ține seama de front).



Implementare monofază

a. ciclul macroinstrucțiunii C<sub>μI</sub> constant

b. ciclul macroinstrucțiunii C<sub>μI</sub> variabil



## Implementare polifază

a. ciclul microinstrucțiunii constant

b. ciclul microinstrucțiunii variabil