9. Sisteme microprogramate

9.1 Noţiuni şi concepte de bază

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

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

Conceptul de microprogramare a fost introdus de Maurice Wilkes, de la Universitatea din Cambridge, în anul 1951, ca o alternativă sistematică de proiectare a unităţilor de comandă ale calculatoarelor numerice sau, în general, a procedurilor de control asupra primitivelor funcţionale ale unui sistem numeric.

Microprogramarea poate fi considerată o tehnică de proiectare şi implementare a funcţiilor de control a sistemelor de prelucrare a datelor numerice, ca o secvenţă de semnale de control ce interpretează static sau dinamic funcţiile de prelucrare a datelor. Semnalele de comandă, necesare la un moment dat pentru controlul primitivelor funcţionale sunt organizate într-un cuvânt de control, memorat într-o memorie PROM sau RAM. Structura cuvântului de control este influenţată de semnificaţia atribuită noţiunii de microoperaţie .

Microoperaţia (μO) este o primitivă a funcţiilor de prelucrare a datelor, care reprezintă o operaţie elementară asupra unei primitive funcţionale (transfer, acţiune de înscriere sau de incrementare, activare pe magistrală, etc.), ce se desfaşoară de obicei într-o perioadă de timp (perioada de tact sau de ceas).

Microoperaţia implică existenţa unui operator, care corespunde unei unităţi funcţionale şi a operanzilor asupra cărora acţionează.

Cuvântul de control este pus în corespondenţă cu noţiunea de microinstrucţiune.

Microinstrucţiunea (μI) este un set de microoperaţii independente de date, fără conflict de resurse, care se pot executa simultan (pe perioada unei perioade de sincronizare).

În general, microinstrucţiunile sunt de două tipuri:

- microinstrucţiuni operaţionale care controlează primitivele funcţionale ale unităţii de execuţie a sistemului numeric, asigurând fluxul de informaţie şi acţiunile asupra resurselor;
- microinstrucţiune de ramificaţie (de salt) care inspectează starea primitivelor funcţionale şi asigură ramificaţia în algoritmul de control, constituind suportul pentru implementarea deciziilor.

Prin microprogram se întelege o secvenţă de microinstrucţiuni ce implementează un algoritm care descrie :

- citirea interpretarea şi execuţia unui set de instrucţiuni maşină;
- primitive ale sistemului de operare ;
- primitive ale limbajelor de programare ;
- etc.

Structura generală a unui sistem de calcul cu unitate de comandă microprogramată este prezentată în Fig. 9.1.

Primitivele funcţionale au următoarele semificaţii :

M - memoria principală a sistemului, în care se păstrează programele ca secvenţă de instrucţiuni maşină şi 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 / ieşiri ;

MC - memoria de control în care se păstrează microprogramul ca secvenţă de microinstrucţiuni ;

µRI - registrul de microinstrucţiuni, care păstrează microinstrucţiunea curentă ce se execută. Conţinutul său specifică toate microoperaţiile care se execută în acel moment în unitatea de execuţie ;

µS - microsecvenţiatorul, unitatea de comandă convenţională, elementară, care asigură citirea interpretarea şi execuţia microinstrucţinilor din memoria de control precum şi înlănţuirea acestora pe baza registrului de instrucţiuni maşină RI şi a stării primitivelor funcţionale.

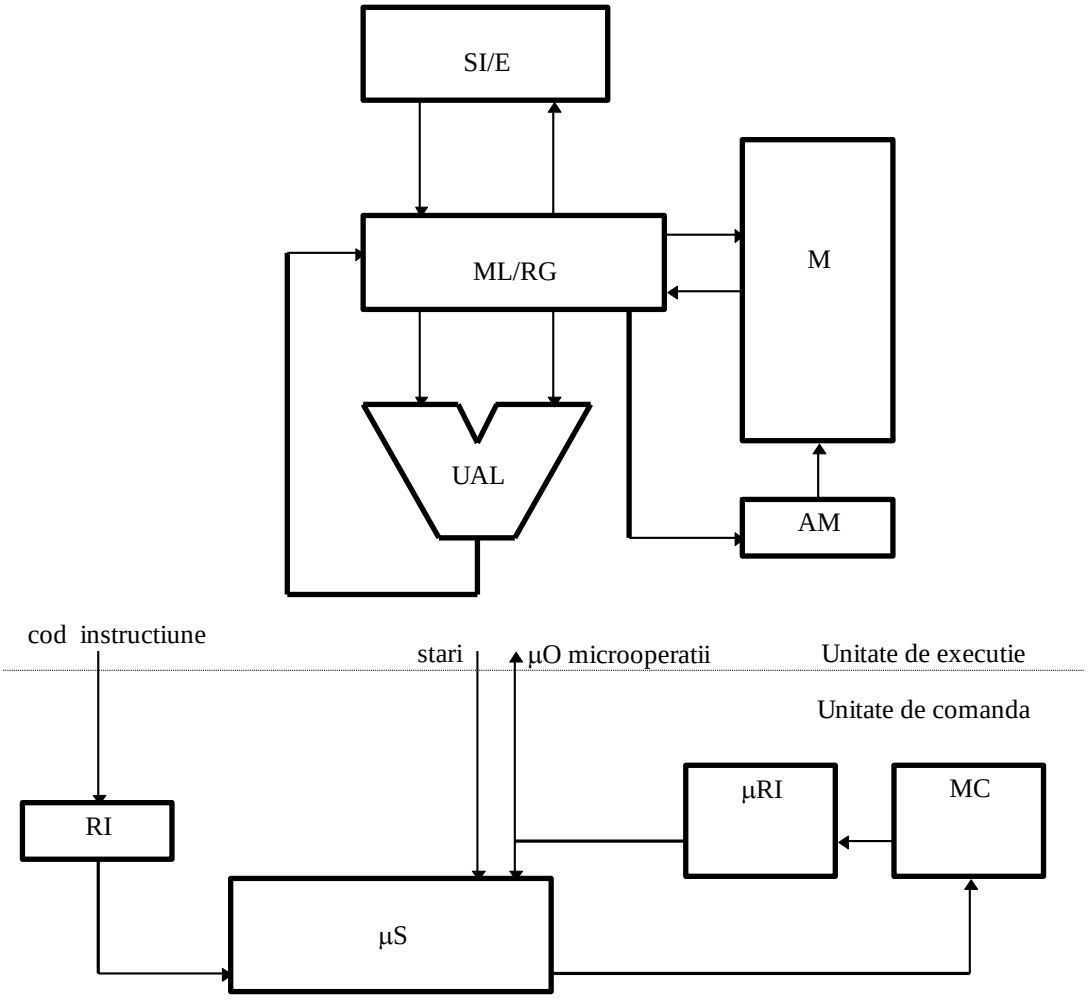

Figura ..19.19.2

Structura generala a unui sistem microprogramat

Din punctul de vedere al proiectantului de sistem, unitatea de comandă microprogramată reprezintă un calculator primar capabil să interpreteze (emuleze) orice alt pseudo calculator (calculator virtual) aflat la un nivel superior celui al calculatorului primar.

Structura simplificată prezentată în Fig. 9.1 are rolul de a scoate în evidenţă faptul că microprogramul conţine informaţiile (semnalele) de control a primitivelor funcţionale şi că acestea sunt memorate în memoria de control în acelaşi fel cum instrucţiunile maşină sunt memorate în memoria principală a sistemului.

O instrucţiune maşină este citită interpretată şi executată de o secvenţă de microinstrucţiuni care controlează şi comandă acţiunile asupra primitivelor funcţionale .

Implementarea sub formă microprogramată a unităţilor de comandă a permis dezvoltarea conceptului de emulare, care a apărut ca urmare a încercărilor de a soluţiona într-un mod eficient problema transportabilităţii programelor, adică de a executa programele de bază şi de aplicaţii ale unui sistem pe alt sistem. Astfel, folosind aceleaşi resurse fizice, este posibil, prin diferite microprograme să se realizeze diverse structuri, care din punct de vedere funcţional, să se comporte diferit sau să corespundă unor calculatoare existente (dintr-o anumită clasă).

În general, o structură microprogramată se caracterizează prin:

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

9.1.1 Organizarea memoriei de control

Organizarea memoriei de control poate fi privită din două puncte de vedere şi anume:

- relaţiei poziţionale între memoria de control MC şi memoria principală M ;
- structurii memoriei de control MC .

Folosind drept criteriu de analiză relaţia MC faţă de M se întâlnesc trei tipuri de organizare a memoriei de control :

1. *memorie de control separată de memoria principală*, atât din punct de vedere fizic cât şi punct de vedere al adresării logice, Fig. 9.2 .
   Raportul dintre viteza de lucru a memoriei de control şi cea a memoriei principale este de circa 10 în favoarea memoriei de control, Vmc >> Vm .
   O asfel de organizare întâlnim la calculatoarele din familia CORAL, I100, PDP 11, IBM 360/50, etc.
2. *memoria de control este implementată în acelaşi spaţiu* fizic şi de adresare cu cel al memoriei principale, Fig. 9.3 . Ambele au acelaşi ciclu de memorie şi sunt adresate prin aceeaşi logică . Divizarea în memorie de control şi 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 şi destul de ieftină pentru a fi folosită ca memorie principală.
   O astfel de organizare întâlnim la calculatoarele IBM 370/145; IBM 360 /25 .
3. *memoria de control este implementată separat de memoria principală* însă este încărcată din aceasta. Se utilizează un sistem de memorie ierarhică, Fig.9.4.
   Memoria MC este încărcată din memoria principală prin intermediul memoriei tampon MT.
   Acest tip de organizare este curent utilizat la calculatoarele mari cum ar fi IBM 370/145. Performanţele acestui tip de organizare depind de structura maşinii de bază şi de algoritmul de lucru cu memoria tampon.

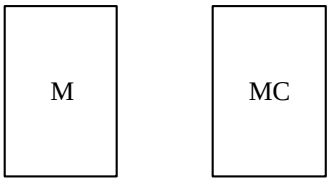

Figura 9.3
Organizarea memoriei de control

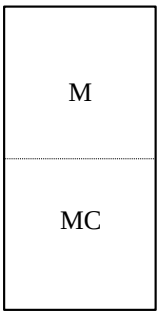

Figura 9.4
Organizarea MC în cadrul MP

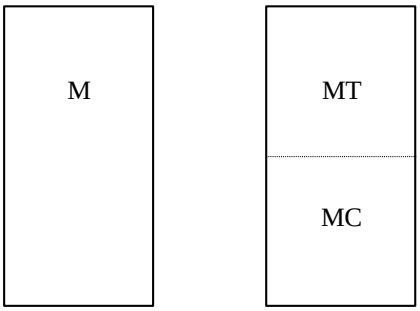

Figura 9.5
Organizarea MC separată de MP
dar incarcabilă din aceasta

Din punctul de vedere al structurii memoriei de control MC aceasta poate fi organizată în mai multe moduri şi anume :

1. *memorie de control cu o microinstrucţiune pe cuvânt*, în care fiecărui cuvânt de memorie îi corespunde o singură microinstrucţiune, Fig. 9.5. Citirea unei microinstrucţiuni presupune un singur acces la MC.

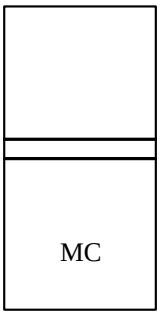

Figura 9.6
MC cu microinstructiune pe cuvânt

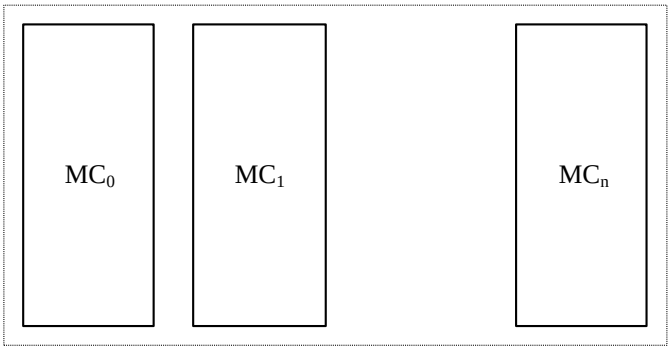

Figura 9.7
MC cu organizare pe pagini

2. *memorie de control cu organizare pe pagini,* Fig. 9.6.
   Unei adrese din memoria de control i se asociază mai multe microinstrucţiuni, din pagini diferite. în acest fel se asigură la nivel de microprogram execuţia unei microinstrucţiuni de tip CASE, care introduce facilitatea de decizii multiple .
   O adresă de MC va adresa aceeaşi locaţie în toate paginile memoriei de control iar vectorul de condiţii de test va activa pagina ce specifică microinstrucţiunea următoare care se va executa.

3. *memorie de control cu organizare pe blocuri*, Fig. 9.7.
   Pentru acest tip de organizare există două feluri de adrese:
   -adrese de microinstrucţiuni din acelasi bloc cu microinstrucţiunea curentă ;
   -adrese de blocuri .
   Împărţirea memoriei de microprograme în blocuri se face ţinând seama atât de structura microprogramului cât şi de resursele fizice disponibile. O organizare de acest fel conduce, în general, la micşorarea lungimii microinstrucţiunii însă introduce timp suplimentar cu comutarea adreselor de blocuri.

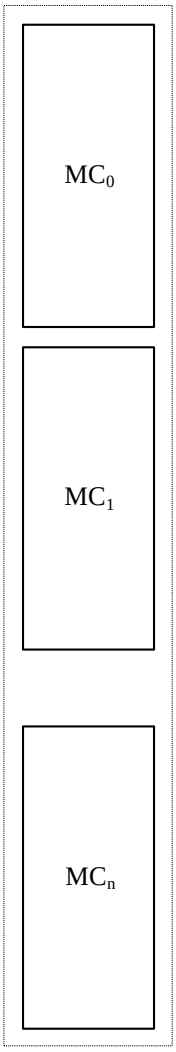

Figura 9.9
MC cu organizare pe blocuri

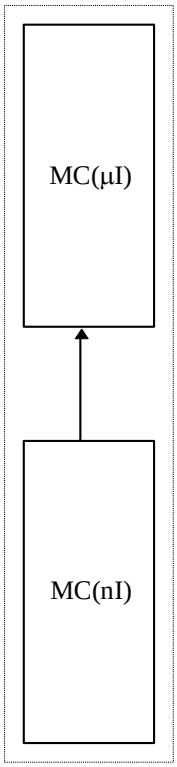

Figura 9.8
MC structurată pe două niveluri

4. *memorie de control divizată*, Fig. 9.8.
   Memoria de microprograme divizată este alcătuită din două unităţi de memorie distincte :
   MI - memorie de microinstrucţiuni, care păstrează toate microinstruciunile distincte posibile necesare pentru controlul resurselor fizice ;
   MA - memorie de adrese de microinstrucţiuni, care păstrează programul specificat, nu prin microinstrucţiuni ce controlează resursele fizice, ci prin adrese de microinstrucţiuni (adrese pentru memoria MI).
   În general numărul de tipuri de microinstrucţiuni distincte este mult mai redus decât microprogramul în sine, ceea ce implică ca numărul de biţi necesari pentru adresarea memoriei MI să fie şi el redus. In acest fel, lungimea cuvântului din memoria MA este mult mai mic decât al memoriei MI, ceea ce conduce la o reducere substanţială a memoriei de control.

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

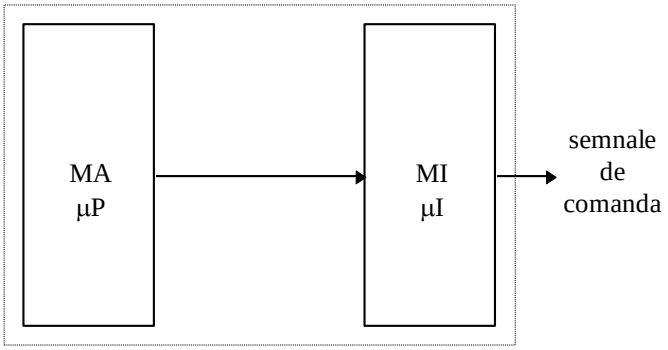

Figura 9.10
MC divizata

5. *memorie de control structurată pe două niveluri*, Fig. 9.9

La o astfel de organizare mecanismul citirii interpretării şi execuţiei unei instrucţiuni maşină este următorul :

- o instrucţiune maşină este interpretată de un set de microinstrucţiuni rezident în memoria de control MC (μI) ;
- la rândul ei, o microinstrucţiune este interpretată de o secvenţă de nanoinstrucţiuni rezidentă în memoria MC (nI) .

Această tehnică a nanoprogramării este conceptual echivalentă cu microprogramarea, structurarea pe două niveluri oferă o flexibilitate mai mare şi posibiltatea implementării unor structuri de control foarte complexe.
O astfel de organizare întâlnim la calculatorul NANODATA QM1.

9.1.2 Suportul fizic pentru păstrarea microprogramelor

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

În ceea ce priveşte 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 structurii microprogramate, în timp ce utilizarea componentelor de tip RAM oferă o caracteristică dinamică .

Caracteristica dinamică, permite utilizatorului ca printr-un ansamblu de mijloace software să aibă acces la microprogramul din memoria de control. Accesul poate consta în modificarea parţială, în completarea microprogramului pentru a crea noi primitive software sau chiar în modificarea totală a microprogramului, ceea ce înseamnă schimbarea arhitecturii maşinii de bază.

Este foarte important în a face deosebire între :

- maşini microprogramate, şi
- maşini microprogramabile

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

Cea de a două categorie oferă atât resursele hardware cât şi facilităţile software pentru accesul utilizatorului la nivelul microinstrucţiunilor.

Dintre calculatoarele cu memorie de control inscriptibilă amintim BURROUGH 1700, MICRODATA 1600, I102F, etc.

9.1.3 Organizarea logică a instrucţiunilor

Un cuvânt din memoria de control specifică un set de microoperaţii ce constituie componentele primitive ale controlului resurselor sistemului. Organizarea logică a microinstrucţiunilor este influenţată de :

- gradul de paralelism între microoperaţii, ce se doreşte realizat;
- structura maşinii de bază ;
- gradul de codificare sau de flexibilitate dorit ;
- gradul de optimizare al lungimii cuvântului de control .

Presupunând că microprogramele sunt specificate ca o secvenţă de seturi disjuncte de microoperaţii, se pot distinge mai multe modalităţi de codificare a acestora şi anume:

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

9.1.3.1 Codificare verticală

În cadrul codificării verticale, fiecare microinstrucţiune operaţională specifică o singură microoperaţie. Setul de microoperaţii (MO) necesar pentru controlul primitivelor funcţionale se codifică în $[\log_2 | (MO)]$ biţi, ce constituie lungimea cuvântului din MC.

Pentru identificarea microoperaţiei specificate de μI se utilizează un decodificator, Fig. 9.9.

Această codificare reprezintă un caz extrem, deoarece elimină orice posibilitate de desfăşurare paralelă a operaţiilor elementare.

Din punctul de vedere al minimizării cuvântului de control, codificarea verticală implică numărul cel mai mic de biţi. Dimensiunea mare a decodificatorului 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 maximale îl reprezintă eliminarea controlului paralel asupra resurselor precum şi inflexibilitatea dezvoltării sau completării sistemului în ceea ce priveşste introducerea de noi microoperaţii. Este aplicabilă numai în sisteme dedicate care au o structură specifică.

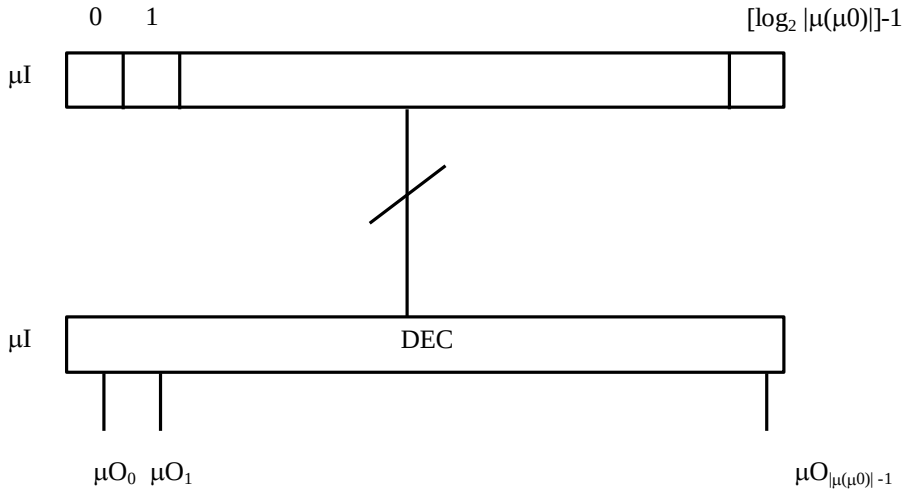

Figura 9.11
Codificarea verticală a microoperaţiilor

9.1.3.2 Codificarea orizontală

În cadrul acestei codificări, fiecare microoperaţie din setul (MO) este pusă în corespondenţă cu un bit din cadrul cuvântului de control. Controlul microoperaţiilor se face în mod direct, Fig. 9.11.

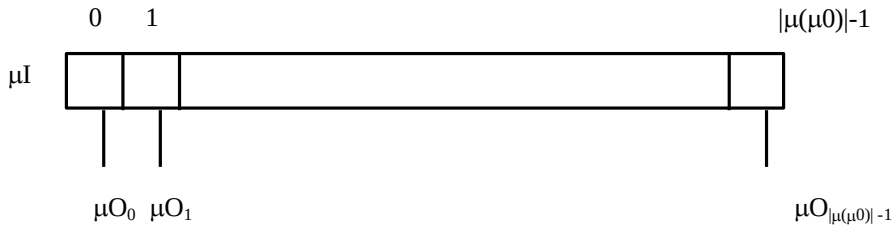

Figura 9.12
Codificarea orizontala

Codificarea orizontală realizează controlul tuturor microoperaţiilor paralele, posibile, ce se pot desfăşura în sistem.

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

9.1.3.3 Codificarea minimală

Combină flexibilitatea şi paralelismul potenţial oferite de codificarea orizontală cu eficienţa codificării verticale.

Ideea de bază este de a grupa în clase de compatibilitate setul de microoperaţii care se exclud reciproc (μO dintr-o clasă de compatibilitate nu se vor efectua niciodată simultan).

Microinstrucţiunea este împărţită 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, Fig 9.12 a).

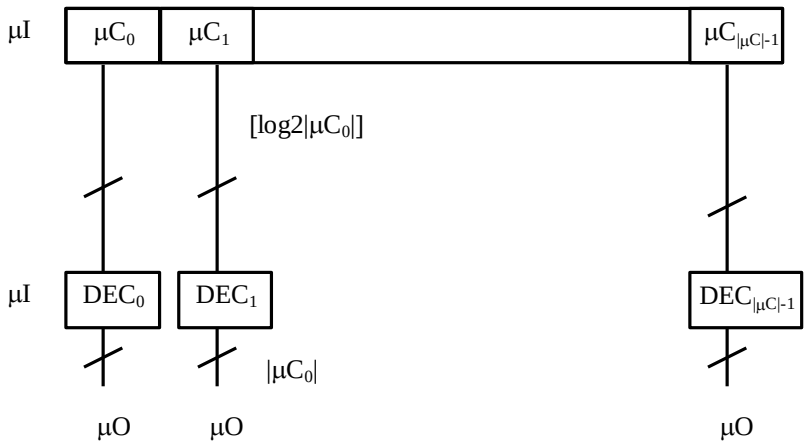

Figura 9.13 a)
Codificare minimală

Pentru un câmp μC_j care codifică $|\mu C_j|$ microoperaţii sunt necesari $|\log_2(|\mu C_j|+1)|$ biţi, deoarece trebuie să se prevadă şi posibilitatea de a nu specifica nici o microoperaţie 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 microinstrucţiune, Fig 9.12 b).

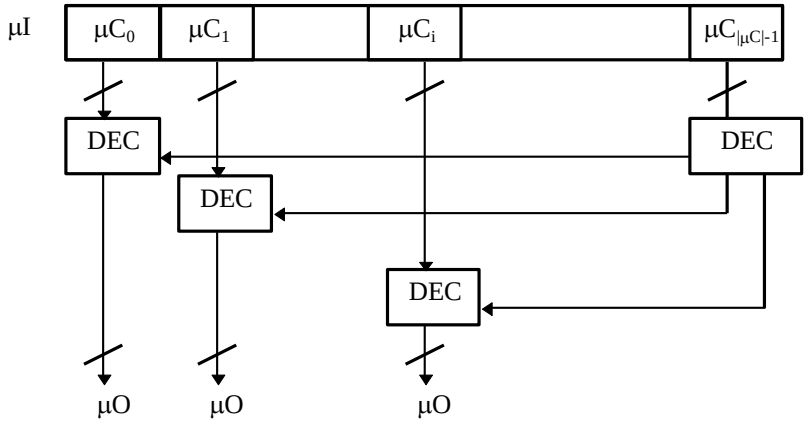

Fig. 9.12 b) Codificare minimală pe două niveluri

9.1.3.4 Codificarea cu control rezidual

Această metodă de codificare foloseşte pentru controlul primitivelor funcţionale registre de control rezidual. Cuvântul de control nu controlează direct resursele ci prin intermediul registrelor de control rezidual încărcate sub acţiunea microinstrucţiunilor, Fig. 9.13

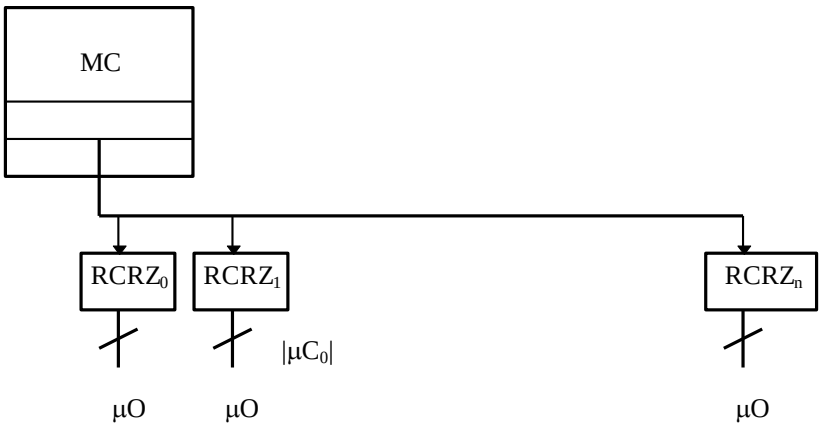

Figura 9.14
Codificare cu control rezidual

Microinstrucţiunile pot să înlocuiască sau să modifice valoarea unuia sau mai multor registre de control. Această tehnică, a controlului rezidual, asigură o economie de memorie de control atunci când unele primitive funcţionale realizează aceeaşi operaţie în mod repetat sau când un set de microoperaţii este activ o perioadă mare de timp, iar alte seturi de microoperaţii se modifică.

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

9.1.3.5 Codificarea cu control prin adrese

O modalitate de implementare a microinstrucţiunilor operaţionale este aceea în care nu se specifică direct microoperaţiile care trebuie să se desfăşoare ci se specifică o adresă în cadrul unei memorii, unde sunt memorate toate microinstrucţiunile distincte posibile ce controlează sistemul, Fig. 9.14.

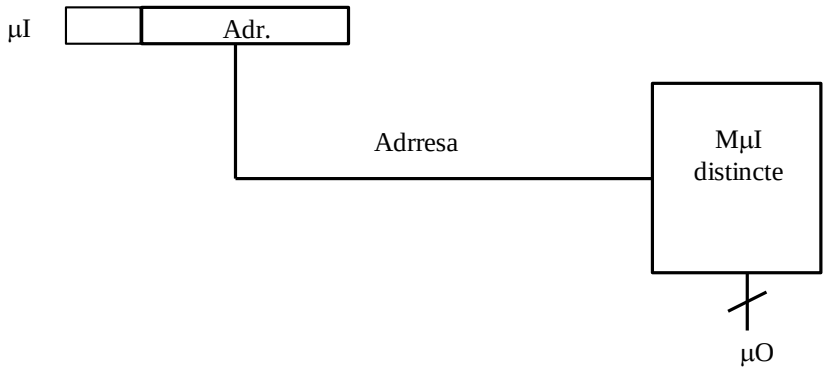

Figura 9.15
Codificare cu control prin adrese

Această modalitate reprezintă o formă simplificată a conceptului de nanoprogramare.

Numărul de microinstrucţiuni distincte nu depinde de numărul de resurse controlate ci de numărul de μO disticte, de mărimea μP şi de numărul de variabile de stare testate .

Memoria care păstrează microinstrucţiunile distincte va avea lungimea cuvântului suficient de mare pentru a controla toate microoperaţiile care se pot efectua simultan. Trebuie notat că fiecare μI este memorată o singură dată.

O astfel de implementare, face ca microprogramul să fie format dintr-o secvenţă de adrese care apelează μI păstrate în memoria de μI.

Un dezavantaj al acestei metode constă în faptul că necesită două accese la memorie în cadrul unui ciclu de microinstrucţiune, dar în schimb se realizează o economie importantă de memorie.

9.1.3.6 Codificare mixtă

O variantă utilizată mult în practică este aceea în care microinstrucţiunea este împărţită în câmpuri. Unele câmpuri controlează direct microoperaţiile (sub formă codificată sau directă) iar altele specifică adrese de memorie ce conţine un subset de microinstrucţiuni distincte, Fig. 9.15.

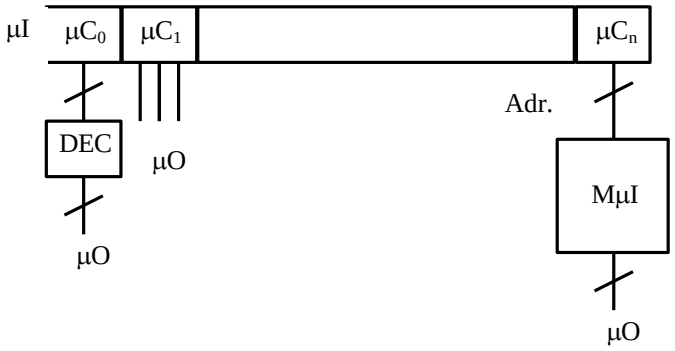

Figura 9.16
Codificare mixtă

9.1.4 Implementarea microinstrucţiunilor

Microinstrucţiunile sunt citite, interpretate şi executate de către microsecventiator (μS) în acelaşi fel în care o unitate de comandă convenţională execută instrucţiuni maşină.

Se poate defini o caracteristică serie-paralel care măsoară cantitatea de suprapunere între faza de execuţie a μI curente şi citirea, interpretarea μI 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 şi E-faza de execuţie a unei μI.

În implementarea serie, citirea microinstrucţiunii următoare nu începe decât după ce s-a terminat execuţia microinstrucţiunii curente, Fig. 9.16.

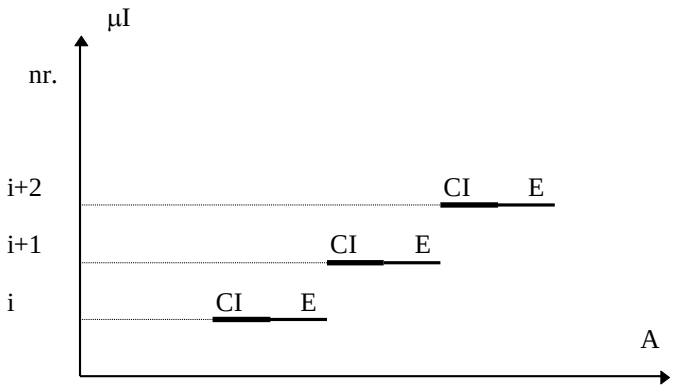

Figura 9.17
Implementare serie

În implementarea paralelă, Fig. 9.17, faza de citire a μI următoare se desfăşoară în acelaşi timp cu execuţia μI curente.

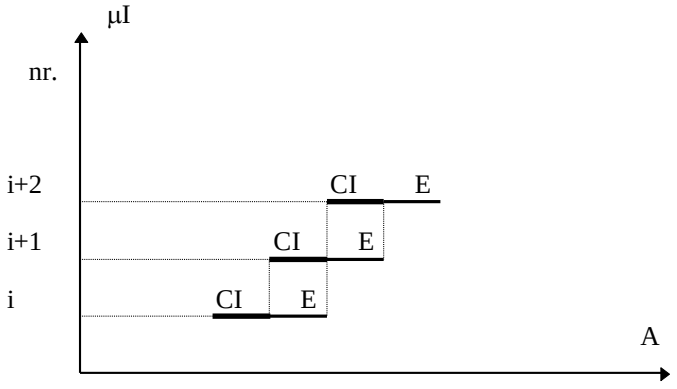

Figura 9.18
Implementare paralel

Se observă că ciclul de execuţie al microprogramului se reduce.

O astfel de implementare impune existenţa unor resurse fizice de comandă (microsecvenţiator, etc.) mai complexe, care să poată anticipa adresa microinstrucţiunii următoare chiar şi în cazul salturilor condiţionate.

O îmbinare a performanţelor implementării paralele şi a costului redus al implementării serie este realizată de implementarea serie-paralelă a microinstrucţiunilor, Fig. 9.18.

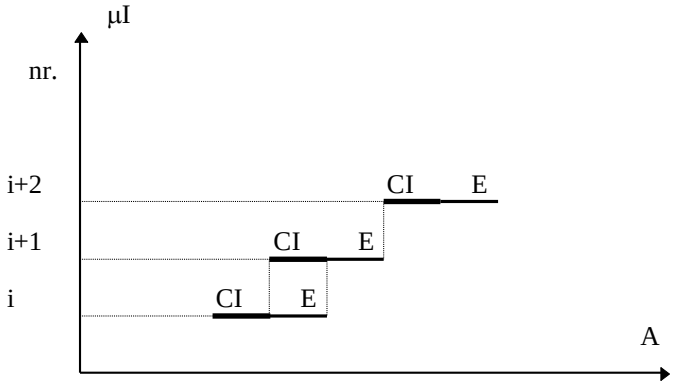

Figura 9.19
Implementare serie-paralel

Faza de citire a μI următoare se desfăşoară fie în timpul execuţiei μI curente, fie după terminarea ei, în funcţie de tipul microinstrucţiunii curente.

Succesiunea de microinstrucţini operaţionale se desfăşoară prin suprapunerea fazelor de citire şi execuţie, iar cele imediat următoare unor ramificaţii se citesc după terminarea execuţiei μI curente (de ramificaţie).

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

Referitor la implementarea microinstrucţiunilor se poate defini şi o caracteristică monofază-polifază care se referă la numărul de faze utilizate într-un ciclu de microinstrucţiune.

Într-o implementare monofază, microoperaţiile sunt generate simultan, toate semnalele de control specificate de microinstrucţiune fiind active în acelaşi timp (pentru cele de tip impuls se ţine seama de front), Fig. 9.19.

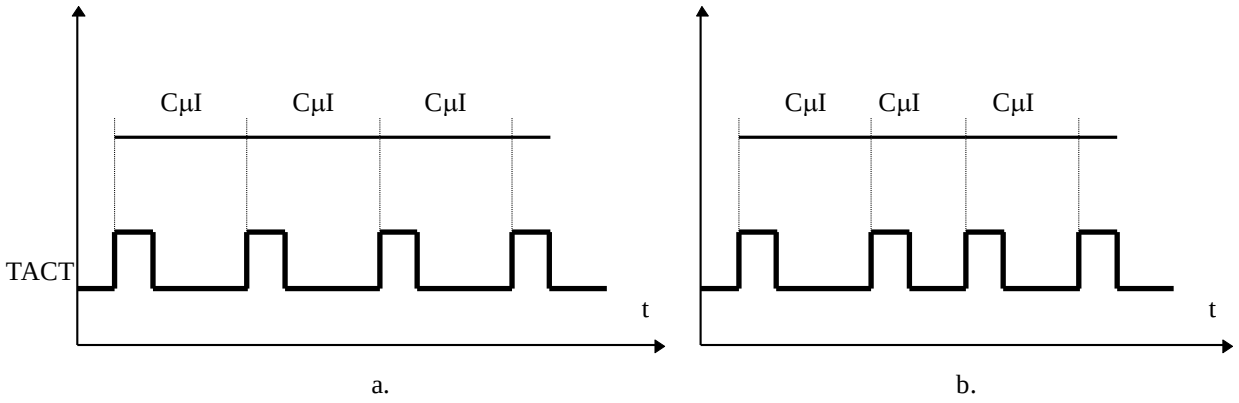

Figura 9.20
Implementare monofază
a. ciclul macroinstrucţiunii CμI constant
b. ciclul macroinstrucţiunii CμI variabil

Într-o implementare polifazică, microinstrucţiunea este activă o perioadă de tact însă semnalele de control sunt repartizate pe faze ale ciclului, Fig. 9.20.

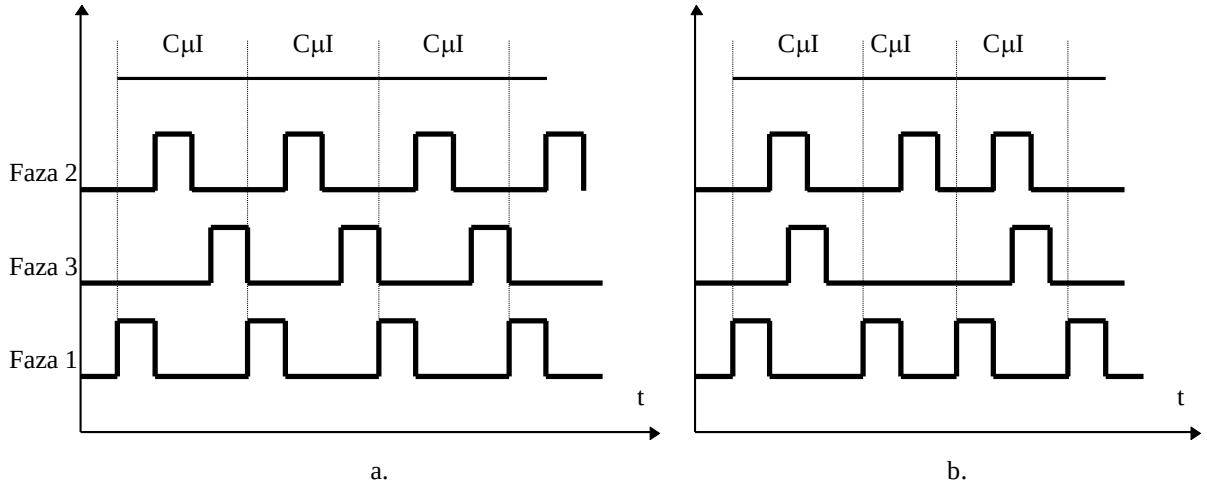

Figura 9.21
Implementare polifază
a. ciclul microinstructiunii constant
b. ciclul microinstructiunii variabil

Ciclul unei microinstrucţiuni poate fi constant sau variabil în sensul că poate avea aceeaşi durată pentru toate μI, respectiv o durată pentru un grup de microinstrucţiuni şi altă durată pentru un alt grup de μI.