7. UNITATEA DE COMANDĂ ŞI CONTROL

Funcțiile principale ale unității de comandă şi control sunt următoarele:

- Decodificarea codului operației;
- Calculul adresei operanzilor care participă la operație şi extragerea lor din memorie;
- Generarea secvenței de comenzi necesare execuției instrucțiunii;
- Generarea secvenței de comenzi necesare memorării rezultatului şi a informațiilor de stare;
- Calculul adresei instrucțiunii următoare şi citirea acesteia din memorie.

7.1. Micro-operații

Prelucrările efectuate de UCP pentru execuția unei singure instrucțiuni reprezintă un *ciclu de instrucțiune*. Fiecare ciclu de instrucțiune constă din următoarele subcicluri:

- Subciclul de *extragere* a instrucțiunii din memorie;
- Subciclul de *execuție*;
- Subciclul de *întrerupere*.

Pe lângă aceste subcicluri prezentate în cadrul capitolului despre UCP, mai există un subciclu care poate apare la execuția unei instrucțiuni. Acesta este subciclul de *indirectare*.

Execuția unei instrucțiuni poate necesita extragerea unuia sau a mai multor operanzi din memorie. După extragerea instrucțiunii, se testează dacă un operand este adresat în modul indirect. În caz afirmativ, se încarcă operandul respectiv utilizând adresarea indirectă.

Ciclul de instrucțiune, completat cu subciclul de indirectare, este reprezentat simplificat în Figura 7.1.

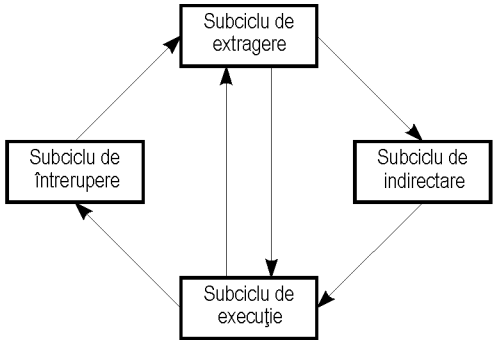

Figura 7.1. Ciclul de instrucțiune.

Fiecare subciclu al unei instrucțiuni poate fi descompus într-o serie de operații elementare. O asemenea operație elementară, care realizează o prelucrare numerică a informației sau un transfer al acesteia, care se efectuează în paralel, pe durata unui singur impuls al generatorului de tact, se numeşte *micro-operație*. Aceste micro-operații sunt operațiile fundamentale, indivizibile ale UCP.

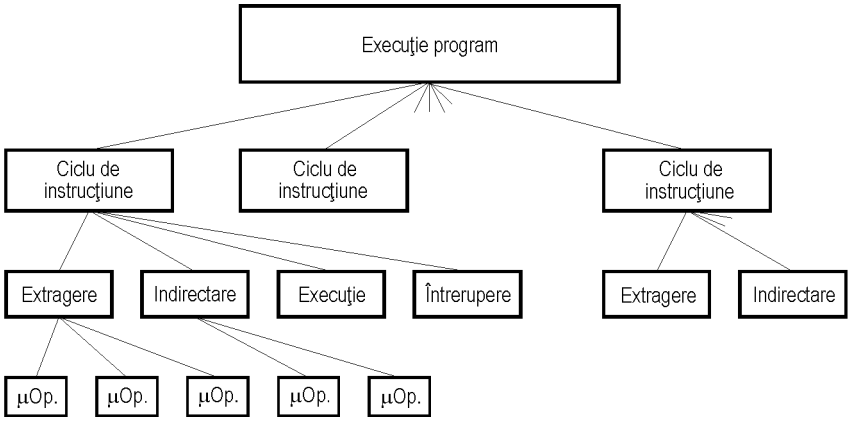

Figura 7.2. Elementele componente ale execuției unui program.

În Figura 7.2 se prezintă execuția unui program, care constă din execuția secvențială a instrucțiunilor. Fiecare instrucțiune se execută în timpul unui ciclu de instrucțiune, format din subcicluri. Pentru fiecare subciclu se execută mai multe micro-operații.

Unitatea de comandă şi control are rolul de a genera succesiunea semnalelor de comandă care asigură secvența corectă de execuție a micro-operațiilor. Această succesiune este specifică fiecărei instrucțiuni, fiind determinată atât de codul operației, cât şi de recepționarea unor semnale de stare de la circuitele controlate, prin care se verifică îndeplinirea unor condiții.

În continuare se analizează subciclurile din care se compune execuţia unei instrucţiuni şi micro-operaţiile din care se compune fiecare ciclu.

7.1.1. Subciclul de extragere

Presupunem că UCP utilizează următoarele registre:

- *Registrul de adrese* al memoriei (RA). Este conectat la liniile de adresă ale magistralei sistem. Conţine adresa de memorie pentru o operaţie de citire sau scriere.
- *Registrul de date* al memoriei (RD). Este conectat la liniile de date ale magistralei sistem. Conţine valoarea care trebuie depusă în memorie sau ultima valoare citită din memorie.
- *Contorul de program* (PC). Conţine adresa următoarei instrucţiuni care se va executa.
- *Registrul de instrucţiuni* (RI). Conţine ultima instrucţiune citită.

Figura 7.3 arată operaţiile executate în subciclul de extragere.

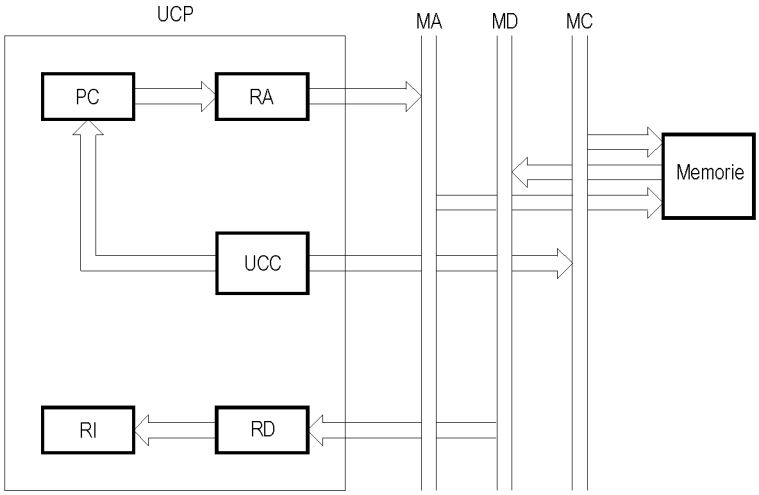

Figura 7.3. Operaţii executate în subciclul de extragere.

La începutul unui subciclu de extragere, adresa următoarei instrucţiuni care se va executa se află în registrul PC. Primul pas este transferul adresei din PC în registrul de adrese RA, care este singurul registru conectat la liniile de adrese ale magistralei sistem. Al doilea pas este citirea instrucţiunii. Adresa din RA este depusă pe magistrala de adrese, unitatea de comandă transmite o comandă de citire pe magistrala de comenzi, iar

rezultatul citirii (codul instrucțiunii) este plasat pe magistrala de date şi copiat în RD. Este necesară de asemenea incrementarea registrului PC cu 1 pentru a pregăti citirea următoarei instrucțiuni. Deoarece aceste două operații (citirea din memorie şi incrementarea PC) sunt independente, se pot efectua în acelaşi timp. Al treilea pas este transferul registrului RD în RI, ceea ce eliberează registrul de date pentru a se putea utiliza eventual în subciclul de indirectare.

Deci, subciclul de extragere constă din trei paşi şi patru micro-operații. Fiecare micro-operație implică transferul datelor în sau dintr-un registru.

Se presupune că există un semnal de tact (ceas), care generează impulsuri la intervale constante. Fiecare impuls de ceas defineşte o unitate de timp. Această unitate de timp este aleasă astfel încât fiecare micro-operație să poată fie executată pe durata unui singur impuls de ceas. Unitățile de timp sunt notate cu t_1, t_2, t_3.

Simbolic, secvența de micro-operații se poate scrie astfel:

t_1: RA ← PC
t_2: RD ← Mem
PC ← PC + 1
t_3: RI ← RD

A treia micro-operație se poate executa şi în unitatea de timp t_3, fără a afecta subciclul de extragere.

7.1.2. Subciclul de indirectare

După extragerea instrucțiunii din memorie, trebuie să se încarce operanzii care participă la operație. Unitatea de comandă examinează conținutul registrului RI, şi dacă se specifică o adresare indirectă, urmează un *subciclu de indirectare* care precede subciclul de execuție.

Operațiile executate în acest subciclu sunt indicate în Figura 7.4.

Câmpul de adresă din registrul de instrucțiuni RI este transferat în RA. Această adresă este utilizată apoi pentru încărcarea adresei operandului în registrul de date RD. Câmpul de adresă din RI este actualizat pentru a conține adresa directă a operandului.

Descrierea simbolică a micro-operațiilor este următoarea:

t_1: RA ← RI (ADR)
t_2: RD ← Mem
t_3: RI (ADR) ← RD (ADR)

Registrul RI este în acest moment în aceeaşi stare ca în cazul în care nu se utiliza adresarea indirectă, şi este pregătit pentru subciclul de execuție.

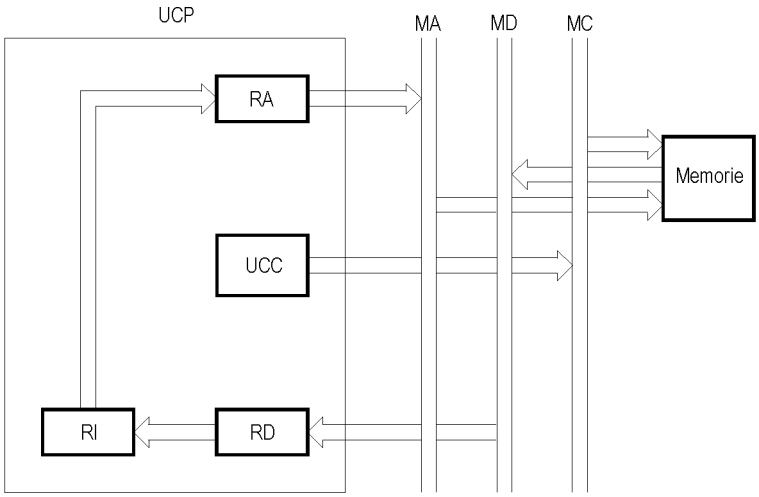

Figura 7.4. Operații executate în subciclul de indirectare.

7.1.3. Subciclul de execuție

Subciclurile de extragere şi de indirectare, ca şi subciclul de întrerupere, implică secvențe fixe de micro-operații. Subciclul de execuție diferă în funcție de instrucțiune, pentru N coduri de operație diferite existând N secvențe diferite de micro-operații care trebuie executate.

Considerăm ca exemplu instrucțiunea:

```
ADD    R1, X
```

care adună conținutul variabilei `X` la registrul `R1`. Execuția acestei instrucțiuni se poate descrie astfel:

t_1: RA ← RI (ADR)
t_2: RD ← Mem
t_3: R1 ← R1 + RD

RI conține codul instrucțiunii ADD. În primul pas, partea de adresă din RI este încărcată în RA. Apoi, se citeşte locația de memorie adresată şi se depune în RD. În final, conținutul registrului R1 şi al RD sunt adunate de UAL.

Acesta este un exemplu simplificat, fiind necesare micro-operații suplimentare pentru adresarea registrului specificat şi pentru a depune operanzii UAL în registre temporare.

7.1.4. Subciclul de întrerupere

La terminarea subciclului de execuţie, în cazul în care întreruperile sunt validate, se testează dacă a apărut o cerere de întrerupere. În caz afirmativ, se execută un subciclu de întrerupere. Secvenţa de operaţii executate variază de la un calculator la altul. Un exemplu este prezentat în continuare.

În primul pas, conţinutul contorului de program PC este transferat în registrul de date RD, pentru a fi salvat în memorie, în scopul reluării operaţiei după terminarea întreruperii. În pasul al doilea, registrul de adrese RA este încărcat cu adresa la care se memorează conţinutul PC. Salvarea se poate face într-o locaţie specială de memorie sau în stivă; în ultimul caz conţinutul indicatorului de stivă SP este transferat în RA. În acelaşi pas poate fi încărcat şi contorul de program cu adresa rutinei de tratare a întreruperii. Deoarece cele mai multe UCP permit mai multe tipuri şi nivele de întrerupere, sunt necesare micro-operaţii suplimentare pentru a obţine adresa de salvare a registrului PC şi adresa rutinei de tratare. Ultimul pas constă în memorarea registrului de date RD. Astfel, în următorul subciclu se va încărca prima instrucţiune a rutinei de tratare.

t_1: RD ← PC
t_2: RA ← Adresa_salvare
PC ← Adresa_rutină
t_3: Mem ← RD

7.1.5. Ciclul de instrucţiune

Se prezintă în Figura 7.5 secvenţa completă de micro-operaţii care compun un ciclu de instrucţiune.

Se introduce un registru de 2 biţi, CCI, care conţine *codul ciclului de instrucţiune*. Acest registru indică starea UCP, în funcţie de subciclul executat:

00: Extragere
01: Indirectare
10: Execuţie
11: Întrerupere

La sfârşitul fiecărui subciclu, registrul CCI este actualizat. Subciclul de indirectare este urmat întotdeauna de subciclul de execuţie. Subciclul de întrerupere este urmat întotdeauna de subciclul de extragere. Pentru subciclul de extragere şi cel de execuţie, următorul subciclu depinde de starea sistemului.

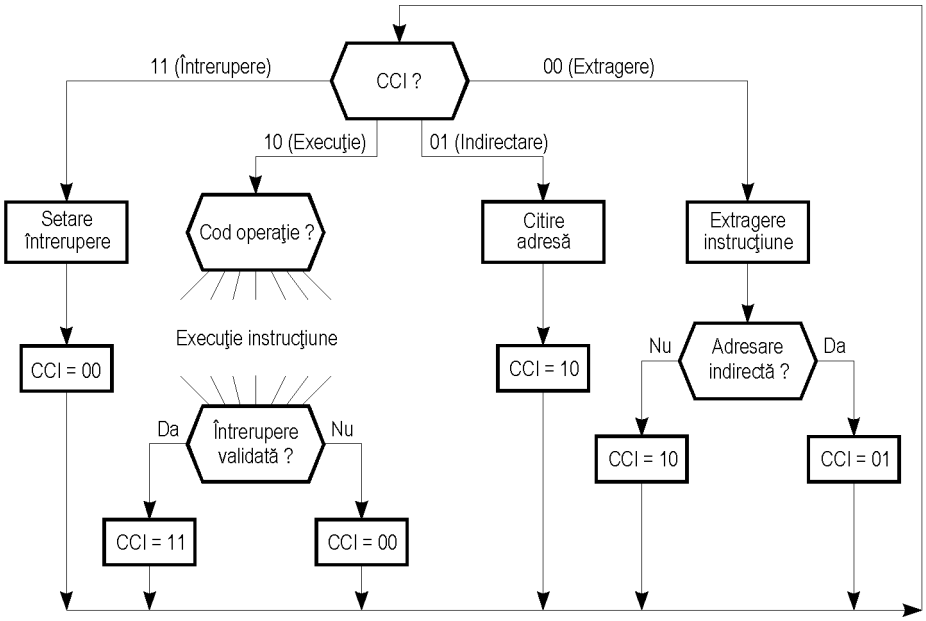

Figura 7.5. Secvența de micro-operații ale unui ciclu de instrucțiune.

7.2. Controlul UCP

7.2.1. Cerințe funcționale pentru unitatea de comandă

Prin definirea operațiilor elementare care trebuie executate de UCP, se pot defini operațiile care trebuie executate de unitatea de comandă. Se pot defini deci cerințele funcționale ale unității de comandă, care se vor utiliza pentru implementarea acestei unități.

Pentru realizarea unității de comandă, sunt necesare următoarele etape:

1. Definirea elementelor de bază ale UCP.
2. Descrierea micro-operațiilor executate de UCP.
3. Determinarea funcțiilor care trebuie executate de unitatea de comandă pentru execuția micro-operațiilor.

Elementele funcționale de bază ale UCP sunt următoarele:

- UAL;
- Registre;
- Căi de date interne;

- Căi de date externe;
- Unitatea de comandă.

Registrele sunt utilizate pentru memorarea datelor interne ale UCP. Anumite registre conţin informaţii de stare necesare pentru secvenţierea instrucţiunilor (de exemplu, cuvântul de stare al programului). Altele conţin date provenite de la UAL, memorie sau module de I/E. *Căile de date interne* sunt utilizate pentru transferul datelor între registre sau între registre şi UAL. *Căile de date externe* efectuează legătura între registre, memorie şi module de I/E, de obicei prin intermediul unei magistrale sistem.

Execuţia unui program constă din operaţii executate cu aceste elemente ale UCP. Micro-operaţiile din care se compun aceste operaţii se încadrează în următoarele *categorii*:

- Transferul datelor între registre;
- Transferul datelor dintr-un registru la o interfaţă externă (de exemplu, magistrala sistem);
- Transferul datelor de la o interfaţă externă într-un registru;
- Execuţia unei operaţii aritmetice sau logice, utilizând registrele pentru operanzi şi rezultat.

Unitatea de comandă are două funcţii de bază:

- *Secvenţiere*: Determină secvenţa corectă a micro-operaţiilor care trebuie executate de UCP.
- *Execuţie*: Determină execuţia fiecărei micro-operaţii.

7.2.2. Semnale de control

Pentru a-şi realiza funcţiile, unitatea de comandă trebuie să aibă intrări care să permită determinarea stării sistemului, şi ieşiri care să permită controlul funcţionării sistemului. Acestea reprezintă specificaţiile externe ale unităţii de comandă şi definesc interacţiunea dintre unitatea de comandă şi alte elemente ale UCP.

Intrările şi ieşirile unei unităţi de comandă şi control se prezintă în Figura 7.6. Intrările sunt următoarele:

- *Ceas*: UCC determină execuţia unei micro-operaţii (sau a unui set de micro-operaţii simultane) la fiecare impuls de ceas. Perioada semnalului de ceas se mai numeşte *ciclu de ceas*.
- *Codul operaţiei*: Se citeşte din registrul de instrucţiuni, care păstrează instrucţiunea curentă, şi se utilizează pentru a determina micro-operaţiile care trebuie executate în timpul ciclului de execuţie.

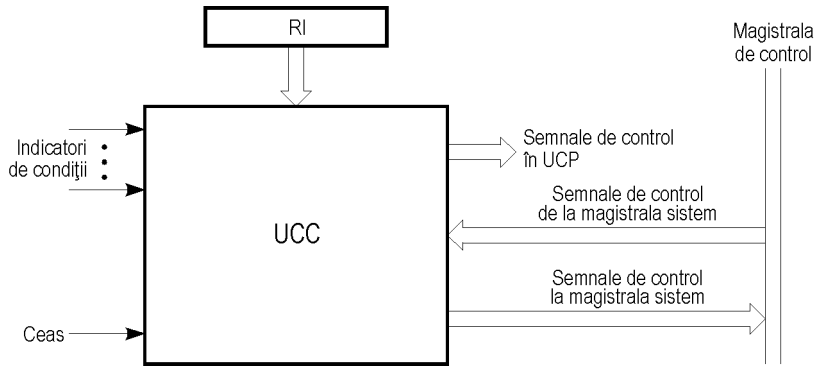

Figura 7.6. Intrările şi ieşirile unei unităţi de comandă şi control.

- *Indicatori de condiţii*: Sunt necesari pentru a determina starea UCP şi rezultatul operaţiei precedente executate de UAL, în scopul executării instrucţiunilor de salt condiţionat.
- *Semnale de control*: Partea care reprezintă magistrala de control din cadrul magistralei sistem furnizează semnale pentru UCC, ca semnale de întrerupere sau de achitare a unei întreruperi.

Ieşirile sunt următoarele:

- *Semnale de control din cadrul* UCP: Acestea sunt de două tipuri: cele care determină transferul datelor dintr-un registru în altul, şi cele care activează funcţii specifice ale UAL.
- *Semnale de control transmise pe magistrala de control*: Acestea sunt tot de două tipuri: semnale de control la memorie şi semnale de control la modulele de I/E.

Elementul care a fost introdus în această figură este *semnalul de control*. Se utilizează trei tipuri de semnale de control:

- Pentru activarea unei funcţii a UAL;
- Pentru activarea unei căi de date;
- Semnale de pe magistrala externă a sistemului sau altă interfaţă externă.

Considerăm din nou ciclul de încărcare al unei instrucţiuni. UCC păstrează evidenţa următorului ciclu care va fi executat. Primul pas constă în transferul:

$$RA \leftarrow PC$$

Acest transfer se efectuează prin activarea unui semnal de control care deschide (activează) porţile dintre cele două registre.

Următorul pas este citirea unui cuvânt din memorie în registrul de date şi incrementarea contorului de program PC:

RD ← Mem
PC ← PC + 1

Aceste operații se efectuează prin activarea simultană a următoarelor semnale de control:

1. Un semnal de control care validează depunerea conținutului RA pe magistrala de adrese.
2. Un semnal de control de pe magistrala de control, pentru citirea memoriei.
3. Un semnal de control care validează memorarea conținutului de pe magistrala de date în registrul de date RD.
4. Semnale de control prin care se incrementează cu 1 conținutul PC şi se depune rezultatul înapoi în PC.

Ultimul pas constă în transferul:

RI ← RD

pentru care se activează un alt semnal de control care validează transferul.

În continuare UCC trebuie să decidă dacă urmează un ciclu de indirectare sau un ciclu de execuție. Pentru aceasta, examinează conținutul registrului de instrucțiuni RI şi testează dacă se utilizează o adresare indirectă.

Ciclurile de indirectare şi de întrerupere se execută similar. Pentru ciclul de execuție, UCC testează codul de operație al instrucțiunii, pe baza căruia decide secvența de micro-operații care va fi executată.

7.2.3. Exemplu de unitate de comandă

Se consideră o unitate centrală simplă cu un singur registru acumulator. Se indică în Figura 7.7 căile de date între elemente şi semnalele de control.

UCC primeşte intrări de la generatorul de ceas, registrul de instrucțiuni şi indicatorii de condiții. La fiecare ciclu de ceas, UCC citeşte toate intrările sale şi generează un set de semnale de control. Semnalele de control au trei destinații separate:

- Căile de date: UCC controlează fluxul intern al datelor. Pentru fiecare cale controlată, există o poartă (indicată printr-un cerc în figură). Un semnal de control deschide (activează) temporar poarta pentru a permite transferul datei.
- UAL: UCC controlează funcționarea UAL printr-un set de semnale de control. Aceste semnale activează diferite circuite din cadrul UAL.
- Magistrala sistem: UCC transmite semnale de control pe liniile de control ale magistralei sistem.

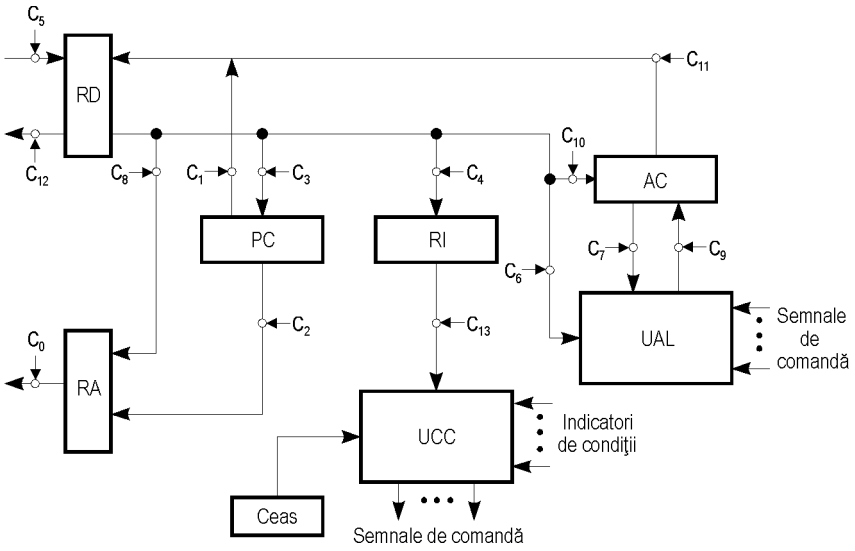

Figura 7.7. Căile de date între elementele unei unităţi centrale şi semnalele de control.

Secvenţierea în timp a operaţiilor este realizată cu ajutorul impulsurilor de ceas, prevăzându-se un timp între diferitele operaţii pentru stabilizarea nivelului semnalelor. În Tabelul 7.1 se indică semnalele de control care sunt necesare pentru o parte a secvenţelor de micro-operaţii descrise anterior. Pentru simplitate, căile de date şi semnalele de control pentru incrementarea PC şi pentru încărcarea adreselor fixe în PC şi MA nu sunt indicate.

Tabelul 7.1. Micro-operaţii şi semnale de control.

Ciclu	Micro-operaţii	Semnale de control active
Extragere:	t_1: RA ← PC	C_2
	t_2: RD ← M	C_5, C_R
	PC ← PC + 1	
	t_3: RI ← RD	C_4
Indirectare:	t_1: RA ← RI (ADR)	C_8
	t_2: RD ← M	C_5, C_R
	t_3: RI (ADR) ← RD (ADR)	C_4
Întrerupere:	t_1: RD ← PC	C_1
	t_2: RA ← Adresa_salvare	
	PC ← Adresa_rutinei	
	t_3: M ← RD	C_{12}, C_W

S-a notat cu: C_R - semnal de citire de pe magistrala sistem;
C_W - semnal de scriere pe magistrala sistem.

7.3. Organizarea internă a UCP

Din cauza numărului mare al căilor de date, structura internă a UCP este complexă. Pentru simplificarea structurii, se utilizează o magistrală internă, la care se conectează UAL şi registrele UCP. Se adaugă porţi şi semnale de control pentru transferul datelor între magistrală şi fiecare registru. Alte semnale de control au rolul de a controla transferul datelor la şi de la magistrala sistem (externă) şi funcţionarea UAL.

Se prezintă în Figura 7.8 structura UCP din exemplul precedent utilizând o magistrală internă.

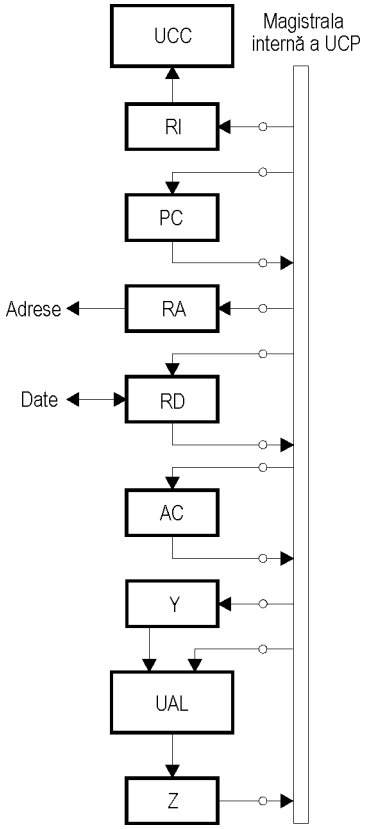

Figura 7.8. UCP utilizând o magistrală internă.

Cu această organizare, o operaţie de adunare a unei valori din memorie la registrul acumulator va avea următoarele etape:

t_1: RA ← RI (ADR)
t_2: RD ← Mem
t_3: Y ← RD

t_4: Z ← AC + Y
t_5: AC ← Z

Sunt posibile şi alte organizări, dar în general se utilizează o magistrală internă sau un set de magistrale interne. Utilizarea căilor de date comune simplifică schema de interconectare şi controlul UCP şi reduce spaţiul necesar.

7.4. Implementarea UCC

Pentru implementarea UCC se utilizează diferite metode, care se pot încadra într-una din două categorii:

- Implementare cablată;
- Implementare microprogramată.

În cazul unei *implementări cablate*, UCC este în principiu un circuit secvenţial care transformă semnalele logice de intrare într-un set de semnale de ieşire, care reprezintă semnalele de control. Modificarea funcţiilor unei astfel de unităţi de comandă necesită modificări ale structurii hardware a circuitului.

Se prezintă câteva tipuri de UCC care utilizează implementarea cablată.

7.4.1. UCC care utilizează un bistabil pe stare

O astfel de unitate de comandă conţine în structura sa un bistabil independent pentru fiecare stare prin care trece circuitul. La un moment dat, un singur bistabil se află în starea "1", cel care determină starea activă.

Această metodă de implementare nu este optimă, deoarece este necesar numărul maxim de bistabile (pentru 16 stări, sunt necesare 16 bistabile, în loc de minimul necesar de 4). Metoda are însă şi unele avantaje:

- Proiectare simplă;
- Partea combinaţională este simplă;
- Depanarea circuitului este mai uşoară.

Structura generală este prezentată în Figura 7.9, unde CC reprezintă un circuit combinaţional.

Trecerea de la starea T_i la starea T_j este determinată atât de semnalele de intrare, cât şi de starea prezentă a circuitului. Dacă circuitul nu necesită intrări, acesta se reduce la un registru de deplasare în care valoarea "1" se deplasează de la un bistabil la altul. Această soluţie se întâlneşte dacă în partea de prelucrare se execută întotdeauna aceeaşi secvenţă de micro-operaţii.

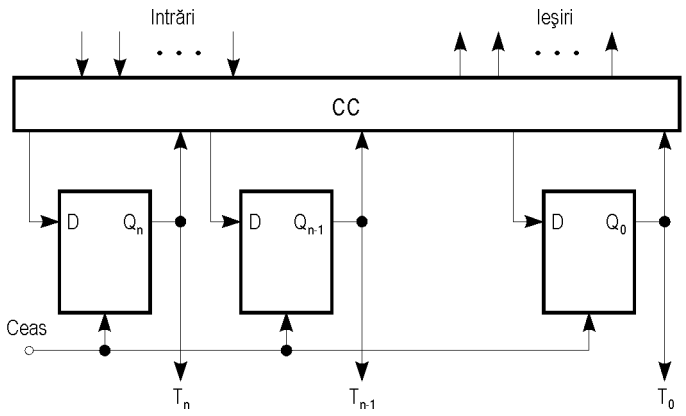

Figura 7.9. Structura generală a unei unităţi de comandă şi control care utilizează un bistabil pe stare.

7.4.2. UCC care utilizează un decodificator

UCC utilizează codul instrucţiunii şi va executa diferite acţiuni pentru fiecare instrucţiune. Codul instrucţiunii va fi decodificat cu ajutorul unui decodificator, care primeşte un cod la intrare şi activează un singur semnal de ieşire. În general, un decodificator are n intrări binare şi 2^n ieşiri binare. Fiecare din cele 2^n coduri de la intrare va activa o singură ieşire care corespunde codului de intrare.

Generatorul de ceas generează o secvenţă de impulsuri. Perioada acestor impulsuri trebuie să fie suficientă pentru a permite propagarea semnalelor prin căile de date şi prin circuitele UCP. UCC generează diferite semnale de control în diferitele unităţi de timp din cadrul aceluiaşi ciclu de instrucţiune. Se utilizează un *generator de faze*, care generează semnalele de intrare T_1, T_2, ..., T_n pentru UCC, pentru fiecare din acestea fiind activate semnale de comandă diferite. La sfârşitul fiecărui ciclu de instrucţiune, UCC reiniţializează generatorul de faze pentru a genera semnalul T_1.

Structura UCC este prezentată în Figura 7.10.

Se va utiliza exemplul anterior pentru a arăta modul în care se deduc ecuaţiile booleene ale semnalelor de comandă în funcţie de intrările UCC. Se va considera un singur semnal de control, C_5. Acest semnal este activat pentru citirea datelor de pe magistrala de date în registrul de date RD. Din tabela cu secvenţele de micro-operaţii, se observă că acest semnal este utilizat de două ori, în două subcicluri diferite. Se definesc două noi semnale de comandă, P şi Q, cu semnificaţia următoare:

$P\,Q = 00$: Subciclu de extragere
$P\,Q = 01$: Subciclu de indirectare
$P\,Q = 10$: Subciclu de execuţie
$P\,Q = 11$: Subciclu de întrerupere

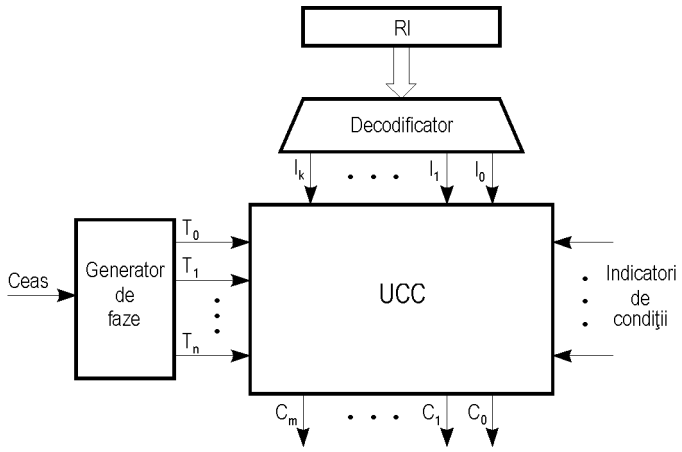

Figura 7.10. Structura generală a unei unități de comandă şi control care utilizează un decodificator.

Semnalul C_5 va fi activat în a doua unitate de timp a subciclului de extragere şi a celui de indirectare, deci se poate defini prin ecuația:

$$C_5 = \overline{P} \cdot \overline{Q} \cdot T_2 + \overline{P} \cdot Q \cdot T_2$$

Acest semnal trebuie însă activat şi în subciclul de execuție. Pentru acest exemplu, se presupune că există numai trei instrucțiuni care execută citirea memoriei: `LDA`, `ADD` şi `AND`. Se notează semnalele activate la ieşirea decodificatorului cu numele instrucțiunii respective. Atunci, C_5 se poate defini ca:

$$C_5 = \overline{P} \cdot \overline{Q} \cdot T_2 + \overline{P} \cdot Q \cdot T_2 + P \cdot \overline{Q} \cdot (LDA + ADD + AND) \cdot T_2$$

În acelaşi mod se scriu ecuațiile tuturor semnalelor de comandă. Rezultatul va fi un set de ecuații booleene care defineşte funcționarea UCC.

UCC trebuie să controleze de asemenea starea ciclului de instrucțiune. La sfârşitul fiecărui subciclu, trebuie să se actualizeze semnalele P şi Q pentru a indica următorul subciclu care se va executa.

7.4.3. Unități de comandă microprogramate

7.4.3.1. Principiul UCC microprogramate

Semnalele de comandă pot fi reunite sub forma unei succesiuni de cifre binare, formând un cuvânt numit *cuvânt de comandă*. Fiecare micro-operație se caracterizează printr-un cuvânt specific de comandă, iar succesiunea cuvintelor de comandă prin care se indică secvența corectă a micro-operațiilor pentru fiecare operație poate fi memorată într-o *memorie de comandă*.

O unitate de comandă în care succesiunea cuvintelor de comandă, adică a valorilor semnalelor de comandă, este memorată, este numită *unitate de comandă microprogramată*. Fiecare cuvânt de comandă memorat în memoria de comandă formează o *microinstrucţiune*, iar secvenţa de microinstrucţiuni formează un *microprogram*.

Deosebirea principală între o unitate de comandă microprogramată şi una cablată constă în modul în care circuitul trece dintr-o stare în alta pentru a genera semnalele de comandă.

- În cazul cablat, o stare corespunde unei *faze*, caracterizată prin activarea unui semnal de fază. Într-o fază sunt generate anumite semnale de comandă necesare execuţiei unei funcţii.
- În cazul microprogramat, o stare corespunde unei *microinstrucţiuni*, care codifică micro-operaţiile care trebuie executate în timpul aceluiaşi semnal de ceas.

O unitate de comandă microprogramată are două funcţii principale:

- *Funcţia de control propriu-zis*, prin care se definesc micro-operaţiile care trebuie executate. Această definire cuprinde de obicei selecţia operanzilor, a operaţiei de executat, selecţia destinaţiei rezultatului etc.
- *Funcţia de secvenţiere*, prin care se defineşte adresa microinstrucţiunii următoare. Această definire se referă la identificarea sursei pentru adresa următoare, la controlul condiţiilor de test sau la generarea directă a valorii adresei.

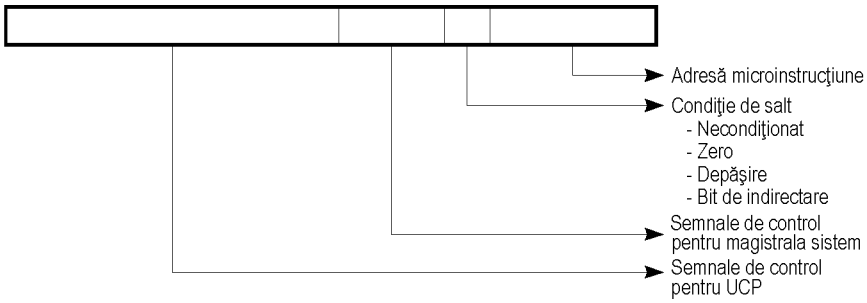

Figura 7.11. Format tipic de microinstrucţiune orizontală.

Conform acestor funcţii, o microinstrucţiune este formată din următoarele câmpuri principale:

- Câmpul semnalelor de comandă generate pentru controlul circuitelor;
- Câmpul de adresă, care conţine adresa următoarei microinstrucţiuni care se va executa, dacă o anumită condiţie este adevărată (de exemplu, dacă bitul de indirectare din codul instrucţiunii este 1). Dacă această condiţie este falsă, se va executa următoarea microinstrucţiune din memoria de comandă.

- Câmpul de condiţii, care indică următoarea microinstrucţiune care va fi executată.

Acest tip de microinstrucţiune se numeşte *microinstrucţiune orizontală* (Figura 7.11). Pe lângă acestea, există şi *microinstrucţiuni verticale*.

Există câte un bit pentru fiecare semnal de comandă intern al UCP şi un bit pentru fiecare semnal de control al magistralei sistem. Câmpul de condiţie arată condiţia în care se execută un salt în microprogram, iar câmpul de adresă conţine adresa la acre se efectuează saltul. O asemenea microinstrucţiune este interpretată în modul descris mai jos.

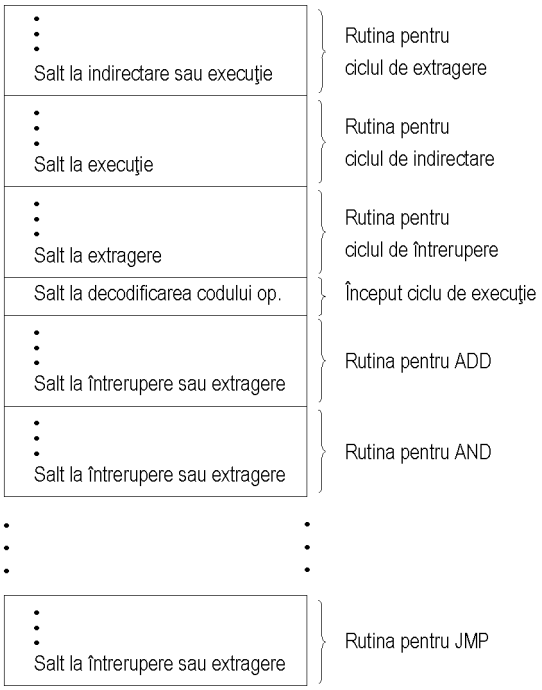

Figura 7.12. Amplasarea microinstrucţiunilor în memoria de comandă.

1. Pentru execuţia microinstrucţiunii, se activează toate semnalele de comandă cărora le corespunde un bit de 1 în câmpul semnalelor de comandă, şi se dezactivează cele cărora le corespunde un bit de 0. Semnalele de comandă care vor fi activate vor determina execuţia uneia sau a mai multor micro-operaţii.
2. În cazul în care condiţia indicată de câmpul de condiţie este falsă, se execută următoarea microinstrucţiune din microprogram.
3. În cazul în care condiţia indicată de câmpul de condiţie este adevărată, se execută microinstrucţiunea indicată de câmpul de adresă.

În Figura 7.12 se prezintă amplasarea microinstrucțiunilor în memoria de comandă.

Microinstrucțiunile din fiecare rutină sunt executate secvențial. Fiecare rutină se termină cu o instrucțiune de salt care indică următoarea rutină care va fi executată. Există o rutină specială pentru începutul unui subciclu de execuție, care indică rutina care va fi executată pentru diferitele instrucțiuni (`ADD`, `AND`, ..., `JMP`), în funcție de codul instrucțiunii.

Memoria de comandă reprezintă o descriere completă a funcționării unității de comandă, deoarece defineşte secvența de micro-operații care trebuie executate în timpul fiecărui subciclu (de extragere, de indirectare, de execuție, de întrerupere), şi specifică secvențierea acestor subcicluri.

7.4.3.2. Structura unei UCC microprogramate

În Figura 7.13 se prezintă elementele principale ale unei unități de comandă microprogramate.

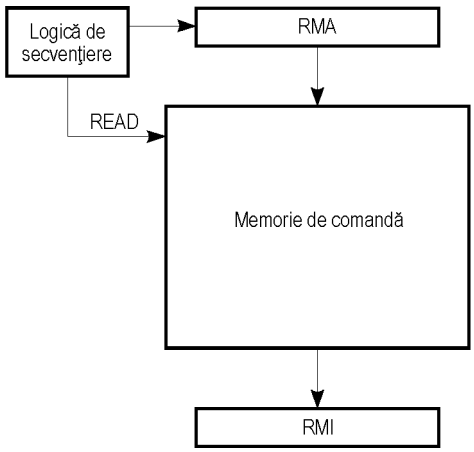

Figura 7.13. Elementele principale ale unei unități de comandă microprogramate.

Registrul de microadrese (RMA) conține adresa următoarei microinstrucțiuni care se va citi din memoria de comandă. După citire, microinstrucțiunea este transferată în *registrul de microinstrucțiuni* (RMI). Prin acest registru se activează semnalele de comandă. Citirea unei microinstrucțiuni din memoria de comandă este echivalentă cu execuția microinstrucțiunii respective. *Logica de secvențiere* încarcă registrul de microinstrucțiuni şi activează comanda se citire.

Structura mai detaliată a unității de comandă este prezentată în Figura 7.14.

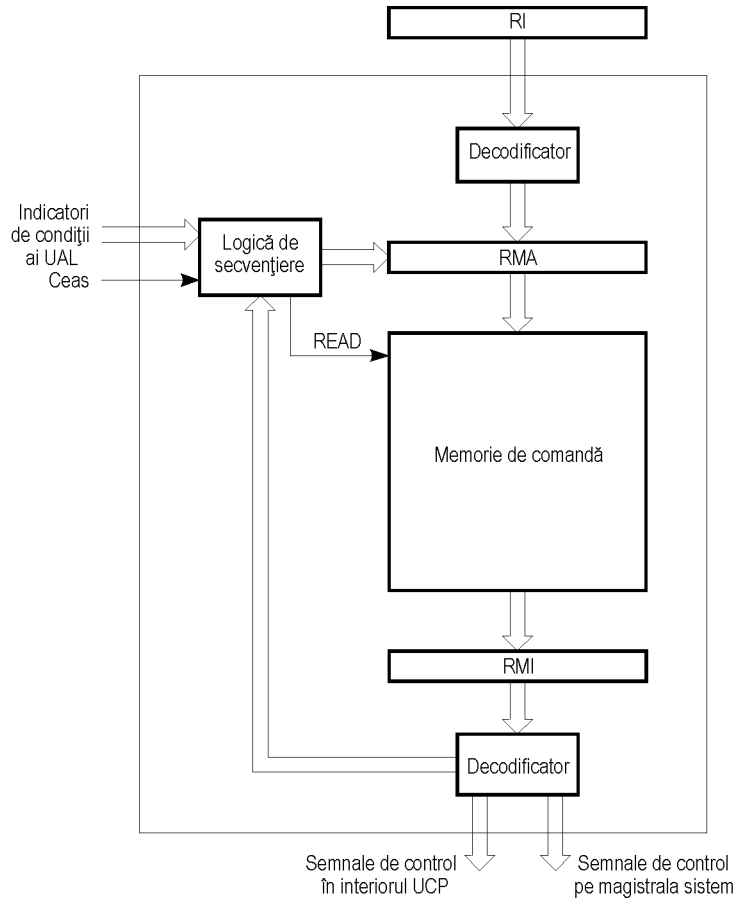

Figura 7.14. Structura detaliată a unei unități de comandă microprogramate.

Comparativ cu o unitate de comandă cablată, unitatea microprogramată are aceleaşi intrări (RI, indicatori de condiții ai UAL, ceas) şi ieşiri (semnale de comandă). Unitatea de comandă funcționează astfel:

1. Pentru execuția unei instrucțiuni, logica de secvențiere activează un semnal de citire a memoriei de comandă.
2. Cuvântul a cărei adresă se specifică în registrul de microadrese este citit în registrul de microinstrucțiuni.
3. Conținutul registrului de microinstrucțiuni activează semnalele de comandă şi generează informații despre adresa următoare pentru logica de secvențiere.

4. Logica de secvențiere încarcă o nouă adresă în registrul de microadrese, pe baza informațiilor despre adresa următoare de la registrul de microinstrucțiuni şi a indicatorilor de condiții ai UAL.

În Figura 7.14 există două decodificatoare. Primul decodificator translatează codul operației din registrul de instrucțiuni RI într-o adresă a memoriei de comandă. Al doilea decodificator nu este utilizat pentru microinstrucțiunile orizontale, ci pentru cele verticale. Într-o microinstrucțiune orizontală, fiecărui bit al câmpului de comandă îi corespunde un semnal de comandă. Într-o microinstrucțiune verticală, se utilizează un cod pentru fiecare operație care trebuie executată, de exemplu RA ← PC, iar decodificatorul translatează acest cod în semnale de comandă individuale.

Avantajul microinstrucțiunilor verticale este că ele sunt mai compacte, utilizând un număr mai mic de biți, cu prețul unei logici suplimentare.

Avantajul principal al utilizării microprogramării pentru implementarea unităților de comandă este simplificarea proiectării acestora. O unitate de comandă cablată trebuie să conțină o logică complexă pentru secvențierea micro-operațiilor din ciclul de instrucțiune. În schimb, decodificatoarele şi logica de secvențiere dintr-o unitate de comandă microprogramată sunt simple.

Dezavantajul principal al unităților microprogramate este că sunt mai lente decât cele cablate realizate într-o tehnologie comparabilă. Cu toate acestea, microprogramarea este tehnica cea mai utilizată pentru implementarea unităților de comandă.