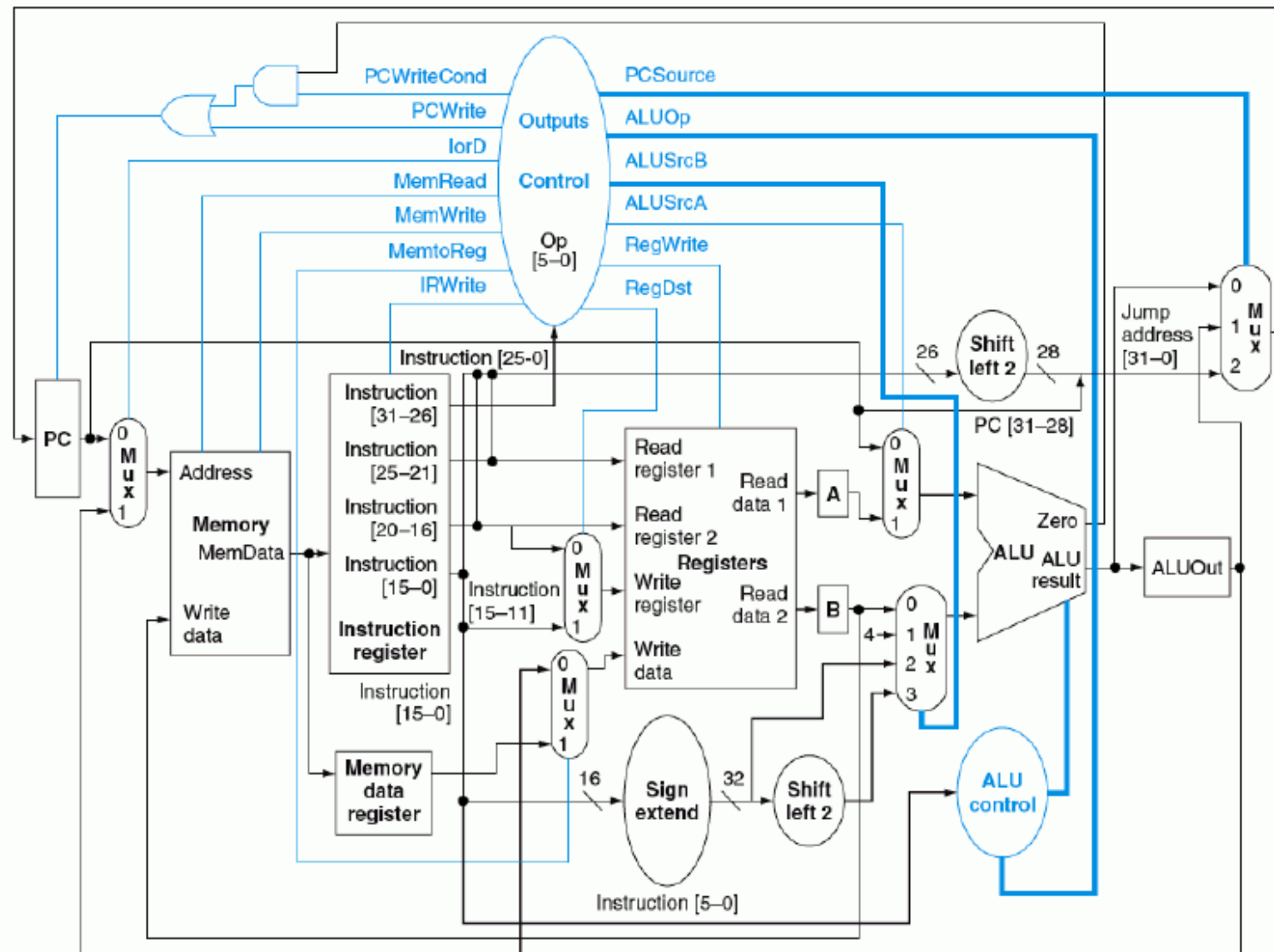


**Procesorul MIPS care operează  
în mai multe cicluri de ceas –definirea controlului**

*– Curs 10 –*

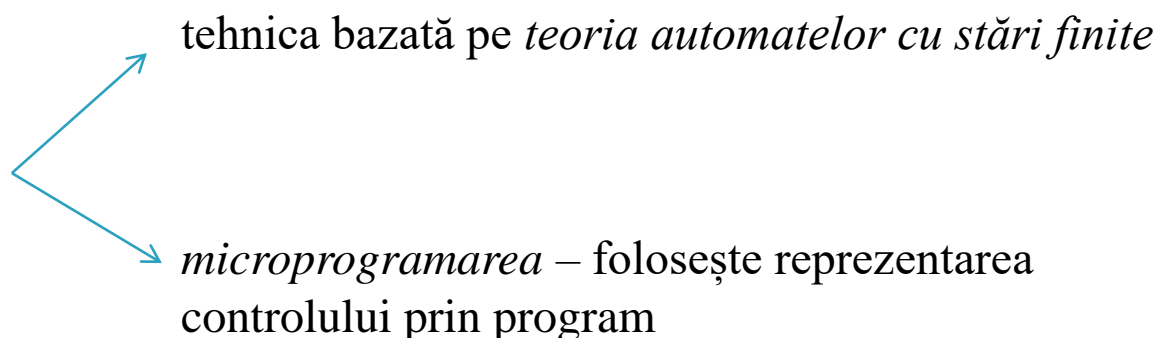


## Definirea controlului

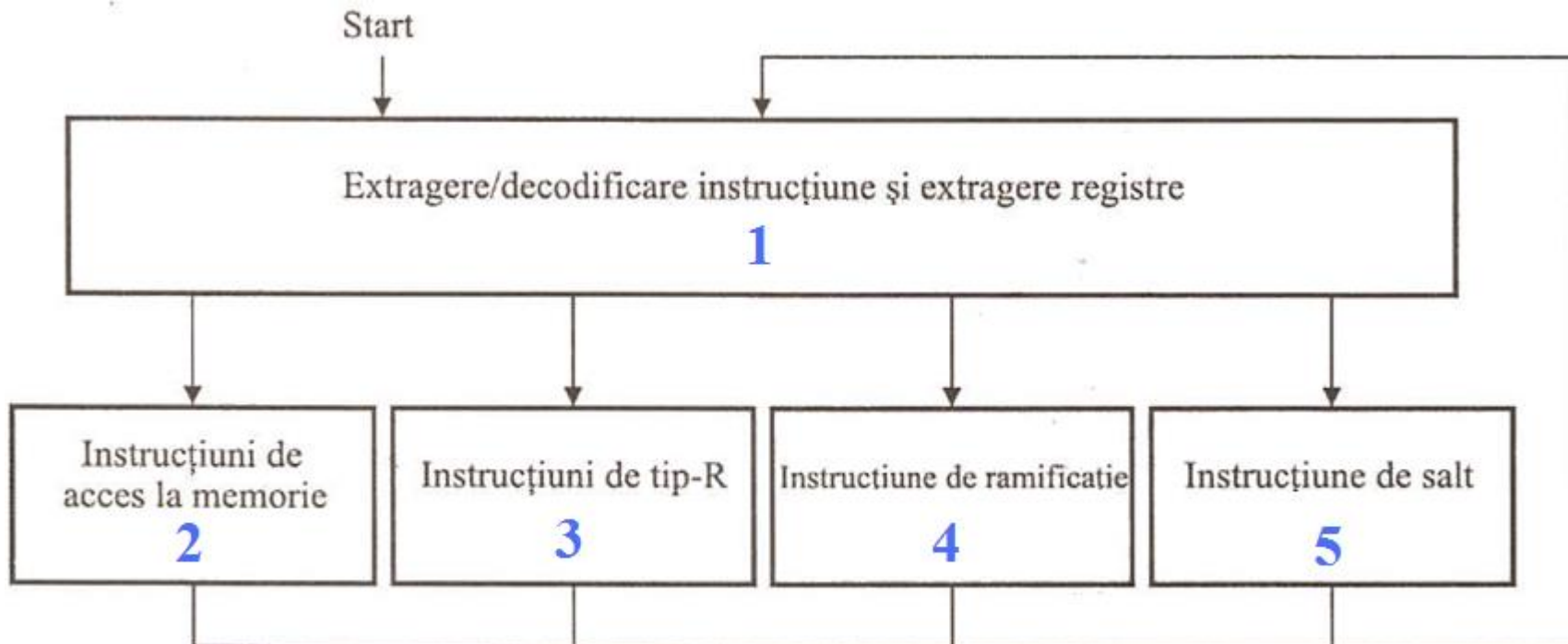
---

Controlul pentru calea de date cu mai multe cicluri trebuie să specifice atât semnalul care trebuie stabilit în fiecare pas, cât și următorul pas din secvență.

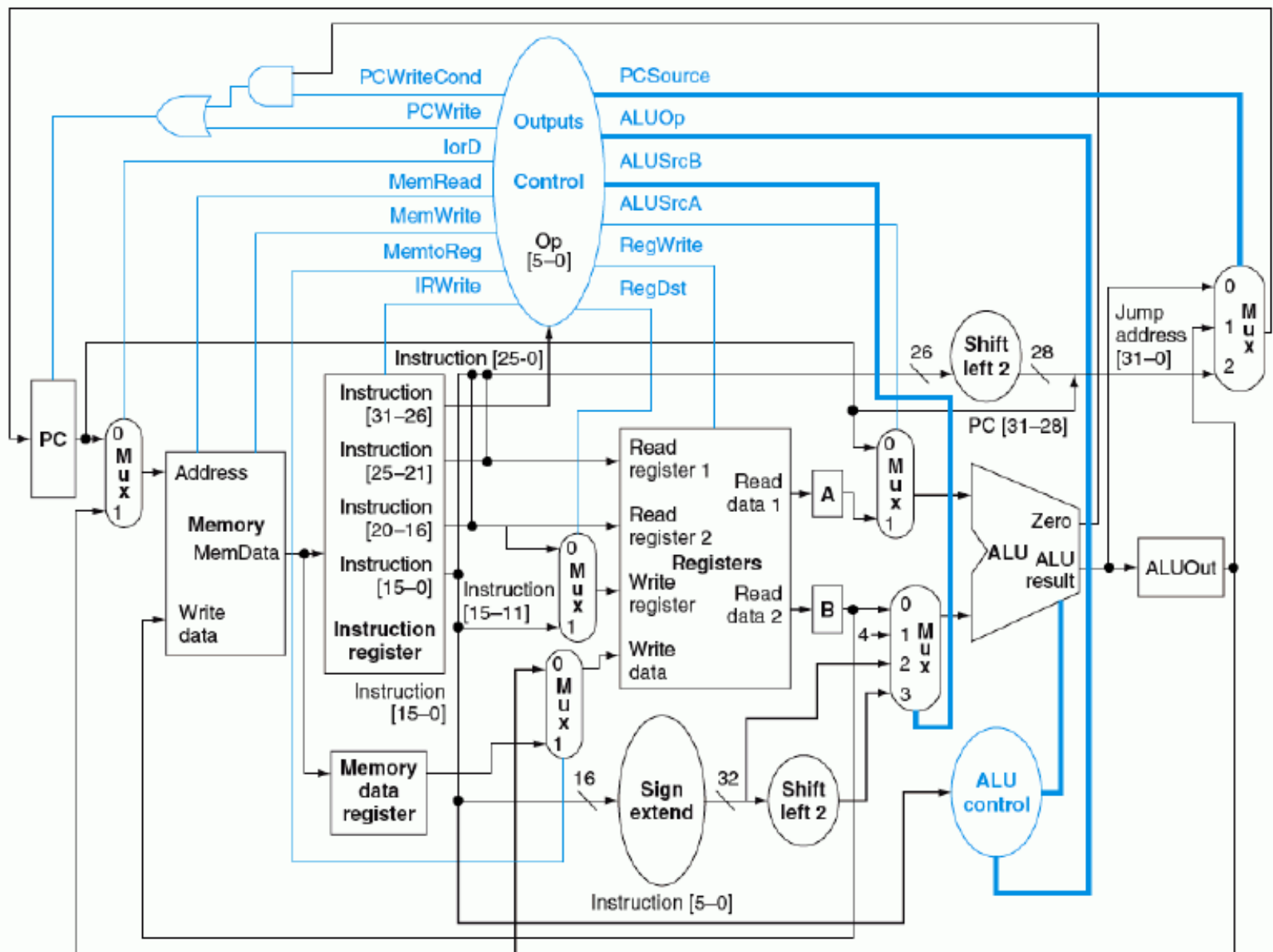
### **Tehnici pentru specificarea controlului**



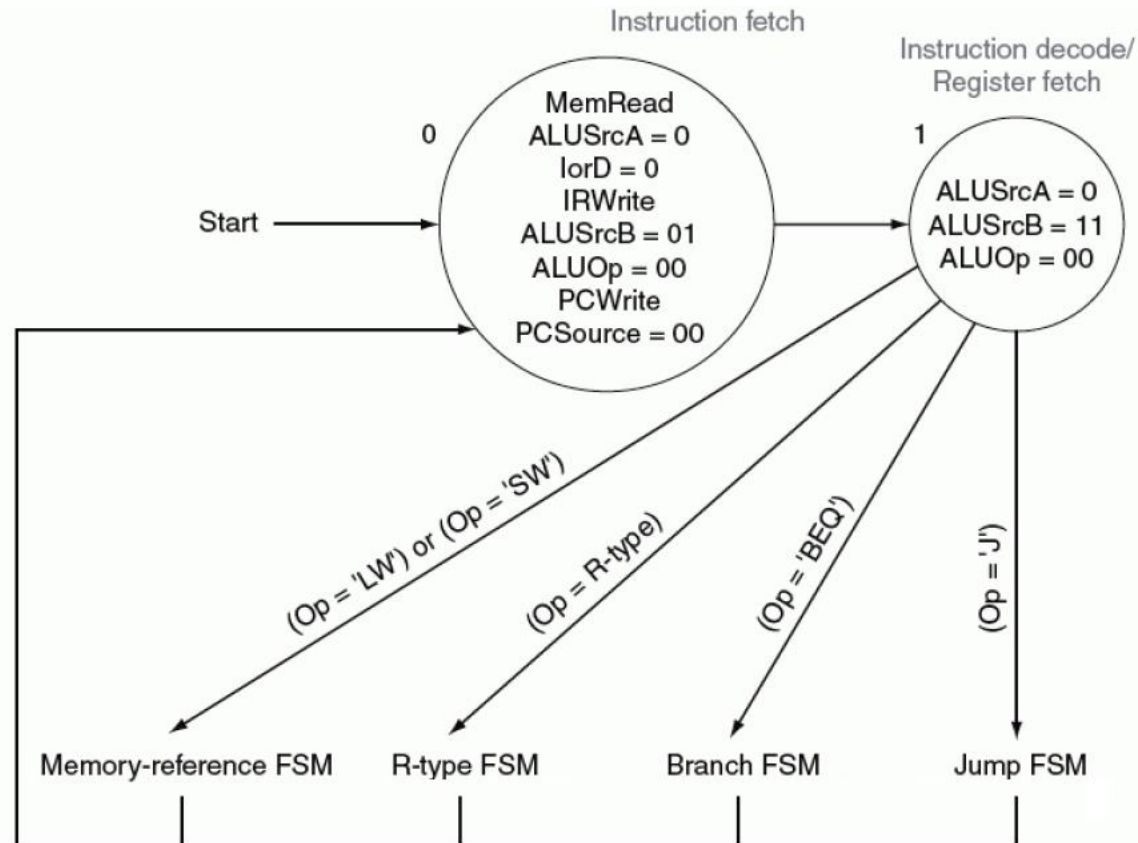
Un automat cu stări finite constă dintr-un set de stări și de directive pentru schimbarea stărilor. Directivele sunt definite de funcția stării următoare care determină o stare nouă din starea curentă și intrările ei.



- Primii doi pași sunt identici pentru fiecare instrucțiune și sunt independenți de clasa instrucțiunii.
- Se execută apoi o serie de secvențe care depind de codul de operație al instrucțiunii. După completarea acțiunilor pentru clasa respectivă de instrucțiuni, controlul revine la extragerea următoarei instrucțiuni.



# 1. Instrucțiunea este extrasă și decodificată



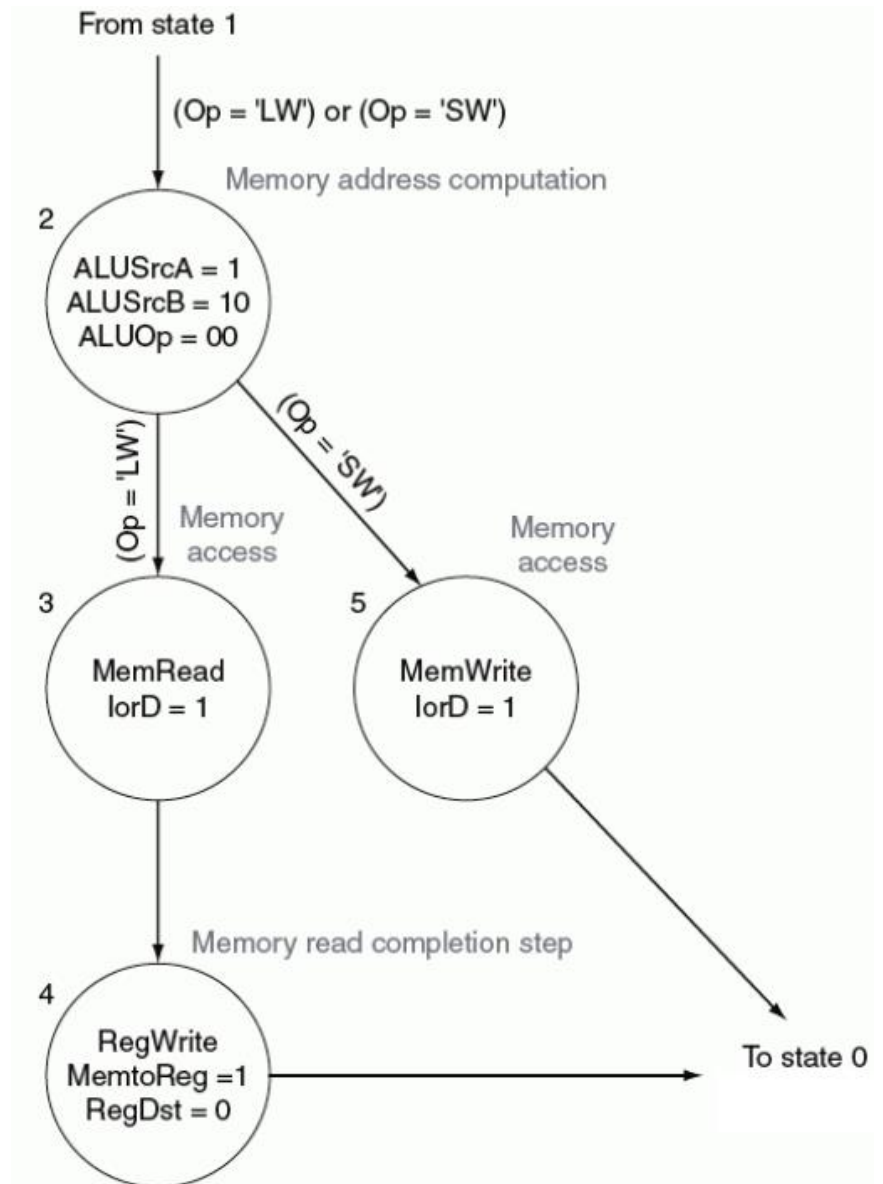
- În prima stare se activează două semnale pentru a provoca citirea unei instrucțiuni din memorie și scrierea ei în Registrul de Instrucțiuni și se stabilește  $IorD=0$  pentru a se alege PC ca sursă pentru adresă.

- Semnalele din starea 0 sunt stabilite pentru a calcula  $PC+4$  și pentru memorarea rezultatului în PC. El va fi memorat și în  $ALUOut$  dar nu va fi niciodată utilizat de acolo.

- În starea următoare se calculează adresa obiectiv pentru ramificație iar apoi se memorează rezultatul în registrul  $ALUOut$  care este scris la fiecare ciclu de ceas.
  - Există 4 stări care pot urma în funcție de clasa instrucțiunii sale care în această stare este cunoscută.

## 2. Controlul instrucțiunilor de referire a memoriei

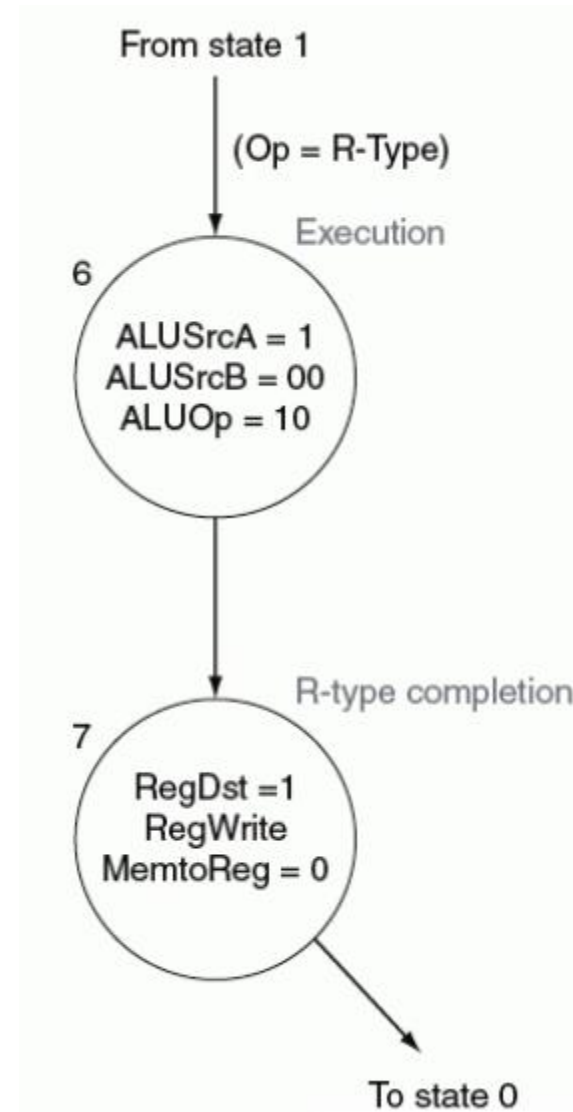
- După efectuarea calculului adresei de memorie sunt necesare secvențe separate pentru instrucțiunile de încărcare și de memorare.
- Este folosită stabilirea semnalelor de control *ALUSrcA*, *ALUSrcB* și *UALOp* pentru a determina ca în starea 2 să se calculeze adresa de memorie.
- Încărcarea necesită o stare suplimentară pentru a scrie rezultatul din MDR (starea 4) în fișierul de registre.





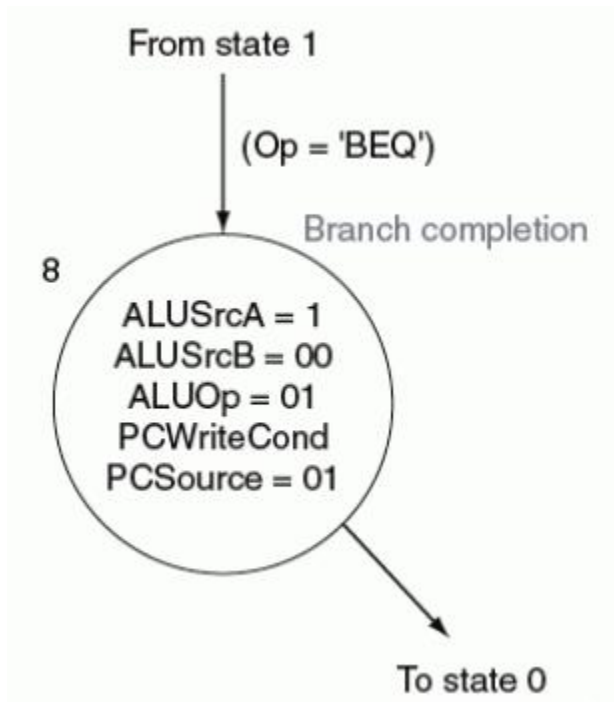
### 3. Instrucțiunile de tip R

- Prima stare determină efectuarea operației UAL.
- A doua stare face ca rezultatul UAL-ului să fie scris în fișierul de registre.
- Cele 3 semnale active în starea 7 determină ca valoarea din *ALUOut* să fie scrisă în fișierul de registre și anume în registrul specificat de câmpul *rd* al registrului de instrucțiuni.



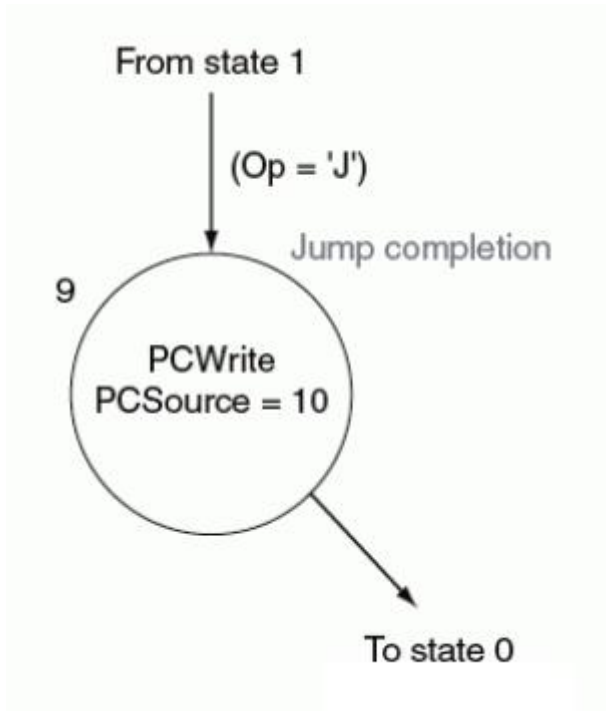


## 4. Instrucțiunea de ramificație



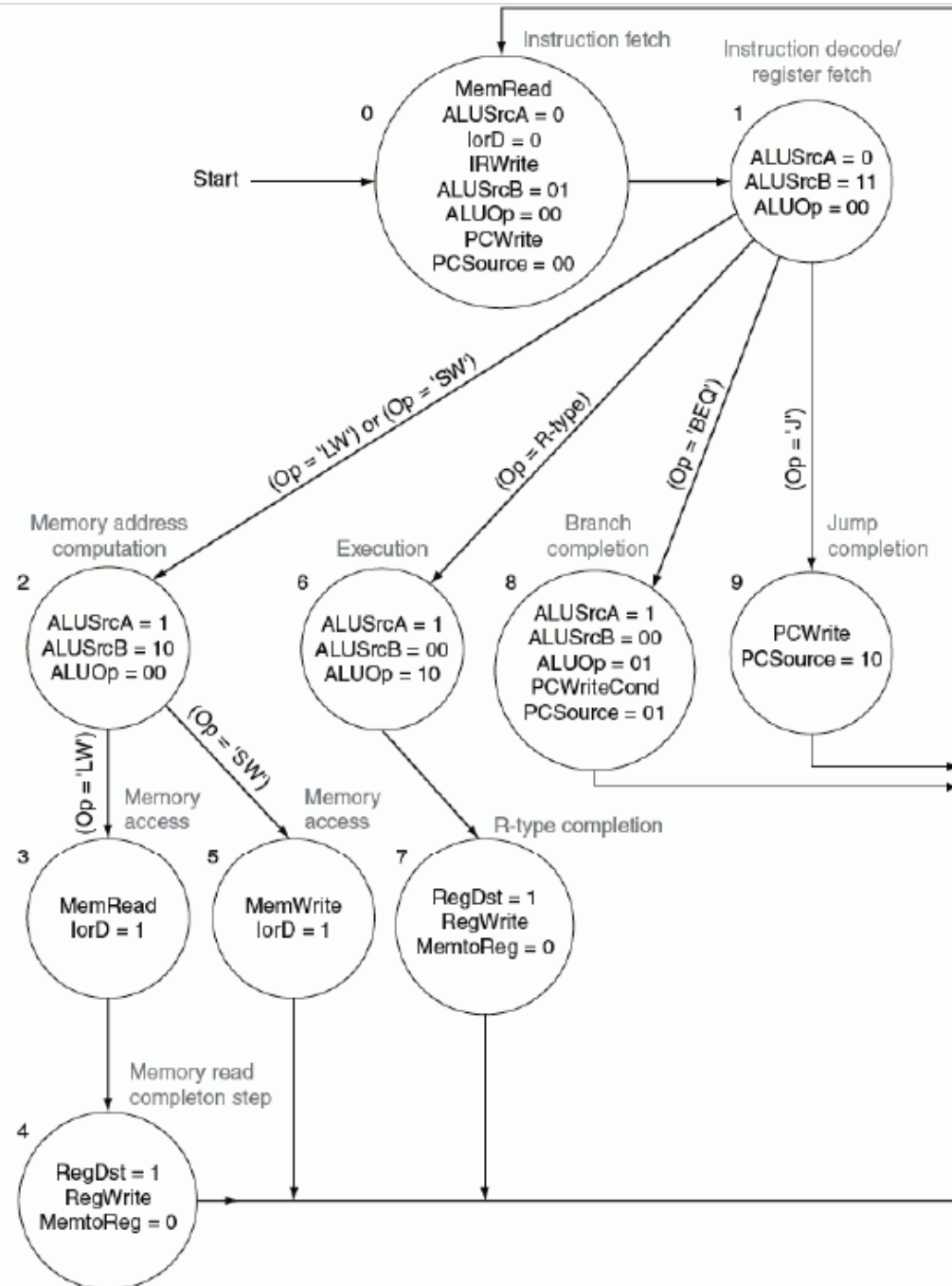
- Primele 3 ieșiri care sunt activate determină ca UAL să compare registrele în timp ce semnalele *PCSource* și *PCWriteCond* efectuează scrierea condiționată în cazul în care condiția de ramificație este adevărată.
- Adresa obiectiv pentru ramificație este citită din ALUOut.

## 5. Instrucțiunea de salt



- Activează 2 semnale de control pentru a scrie în PC cei 26 de biți inferiori ai registrului de instrucțiuni deplasați spre stânga cu 2 biți și concatenați cu cei 4 biți superiori ai PC-ului acestei instrucțiuni.

## Automatul complet

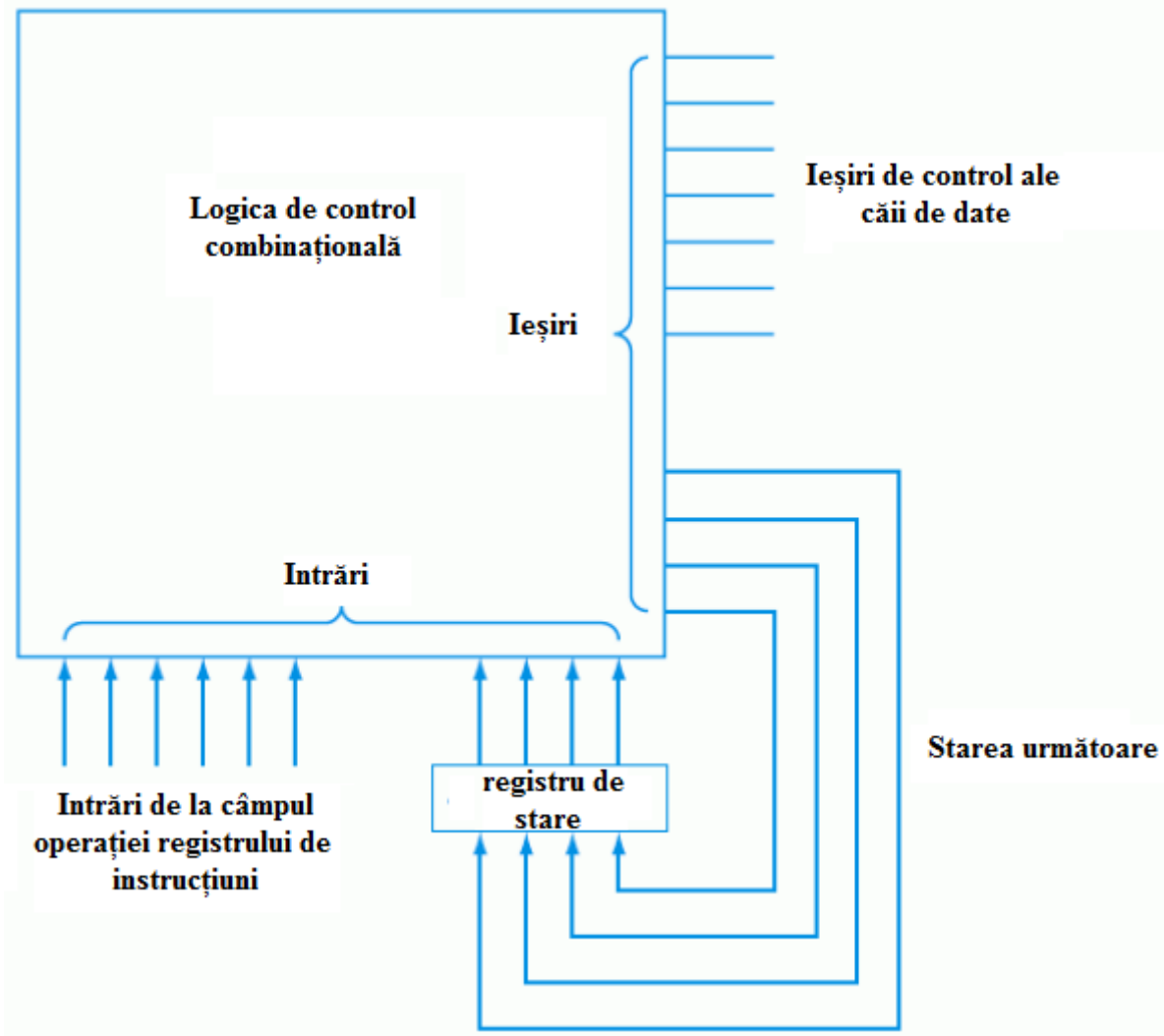


Controlul cu automatul cu stări finite este în mod obișnuit implementat utilizând un bloc de logică combinațională și un registru care să păstreze starea curentă.


### Mașina Moore



Caracteristica: ieșirile depind numai de starea curentă.



- ❑ O **microinstrucțiune** definește setul de semnale de control ale căii de date care trebuie activate într-o stare dată.
- ❑ Executarea unei microinstrucțiuni are efectul de a activa semnalele de control specificate de microinstrucțiune.
- ❑ De asemenea, este importantă specificarea succesiunii microinstrucțiunilor.
- ❑ În descrierea controlului unui program:
  - microinstrucțiunile scrise secvențial sunt executate în secvență

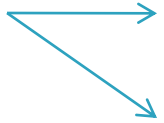


se poate implementa cu o structura ca mașina Moore


  - ramificațiile trebuie indicate explicit.


*Proiectarea controlului sub forma unui program care implementează instrucțiunile mașinii în termenii microinstrucțiunilor se numește **microprogramare**.*

Microprogramul este o reprezentare simbolică a controlului care va fi translatată de către un program în logică de control.

Trebuie să alegem:  câte câmpuri trebuie să aibă o microinstrucțiune  
ce semnale de control sunt afectate de fiecare câmp

**Exp.:** este util să existe un câmp care să controleze UAL-ul și destinația rezultatului UAL-ului.

**!!!** Formatul microinstrucțiunii trebuie să facă dificilă scrierea **microinstrucțiunilor inconsistente**  dacă ea cere să se atribue unui anumit semnal de control două valori diferite.

 Fiecare câmp al microinstrucțiunii trebuie făcut responsabil cu specificarea unui set de semnale de control care să nu se suprapună în timp.

**Fiecare microinstrucțiune conține șapte câmpuri (6 pentru controlul căii de date și ultimul pentru selectarea microinstrucțiunii următoare).**

Control UAL	specifică operația efectuată de UAL în timpul acestui ciclu. Rezultatul este scris în ALUOut.
SRC1	specifică sursa pentru primul operand UAL
SRC2	specifică sursa pentru al doilea operand UAL
Control de registre	specifică scrierea sau citirea pentru fișierul de registre și sursa valorii pentru scriere
Memoria	specifică scrierea sau citirea și sursa pentru memorie. Pentru citire specifică registrul de destinație.
Control ScribePC	specifică scrierea PC-ului
Secvență	specifică modul de alegere a următoarei microinstrucțiuni



### *Moduri de alegere a următoarei microinstrucțiuni:*

1. Se incrementează adresa microinstrucțiunii curente: indicatorul *Seq* în câmpul *secvență*.
2. Se face transferul controlului la microinstrucțiunea care începe execuția următoarei instrucțiuni MIPS: indicatorul *Fetch* în câmpul *secvență*. (microinstrucțiune inițială)
3. Se alege următoarea microinstrucțiune pe baza intrării în unitatea de control: *distribuție*.



Se creează un *tabel de distribuție* care conține adresele microinstrucțiunilor obiectiv cărora le este transferat controlul.

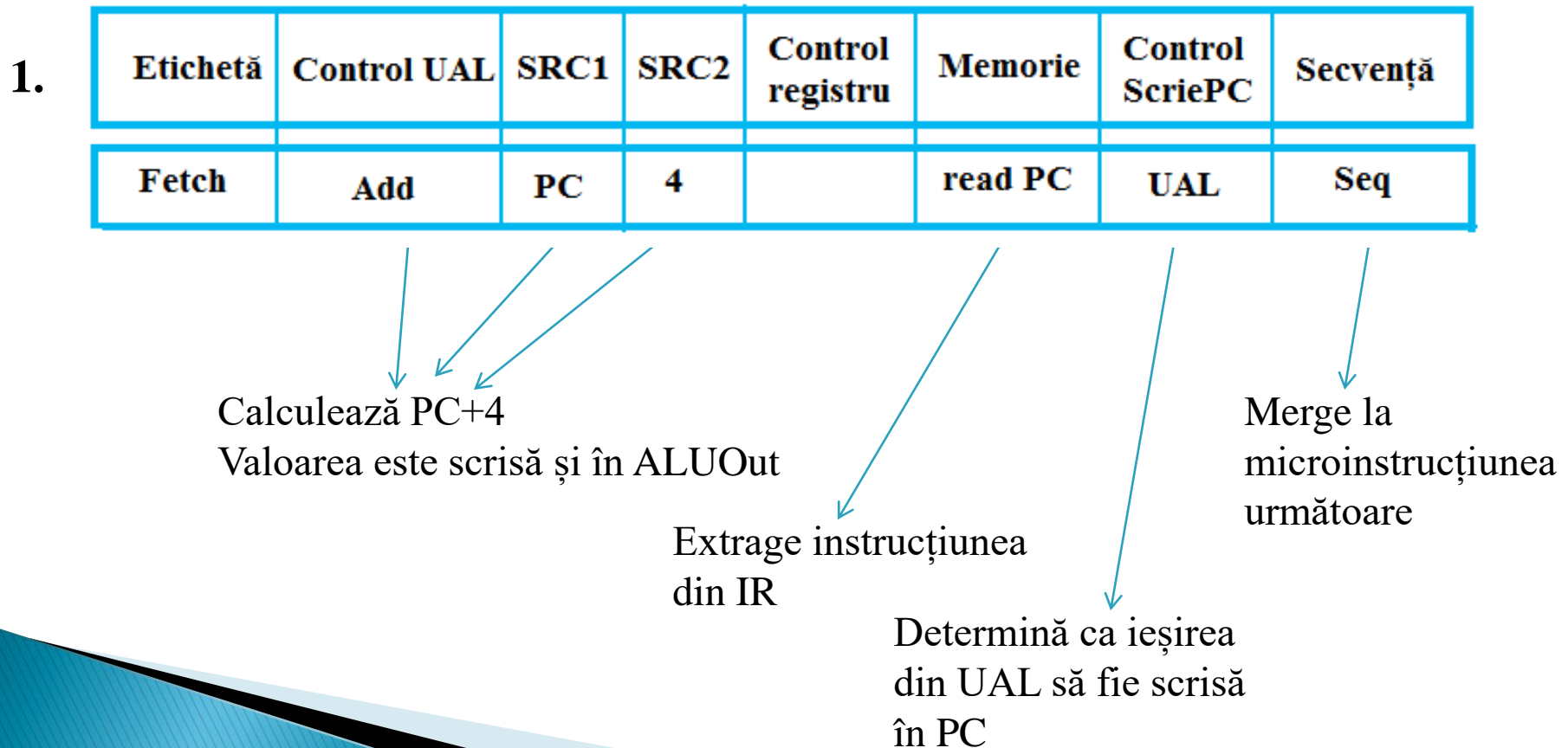


*Dispatch i* în câmpul *secvență*, unde *i* este numărul tabelului de distribuție.

## Descompunerea microinstrucțiunii

Prima componentă a execuției unei instrucțiuni este extragerea instrucțiunilor, decodificarea lor și calcularea PC-ului secvențial și cel obiectiv, pentru ramificație.

Microinstrucțiunile pentru primii doi pași:



2.

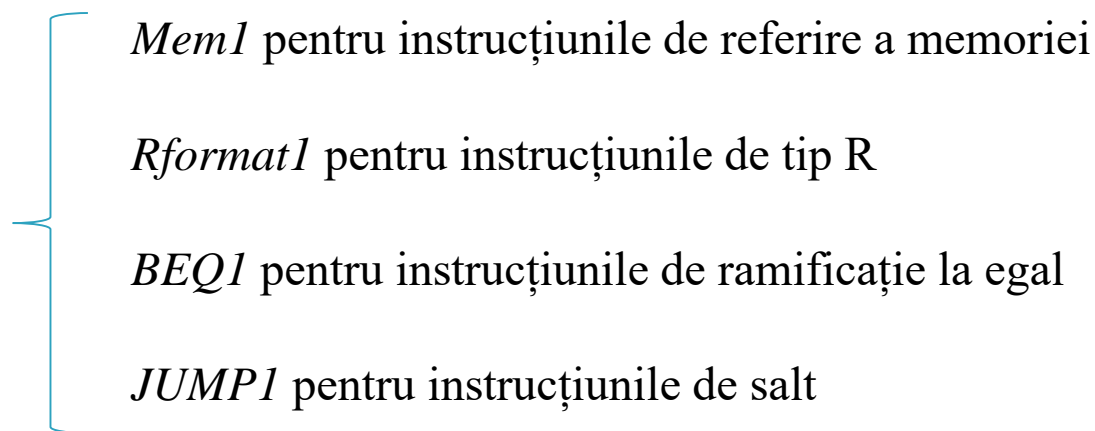
Etichetă	Control UAL	SRC1	SRC2	Control registru	Memorie	Control SriePC	Secvență
	Add	PC	Extshft	read			Dispatch 1

Memorează  
PC+semn extins ( $IR[15-0] \ll 2$ )  
în ALUOut

Folosește câmpurile *rs* și *rt* pentru citirea  
registrelor, punând datele în A și B

Folosește tabelul de distribuție 1 pentru alegerea  
adresei următoarei microinstrucțiuni

Operația de distribuție trebuie privită ca un *switch*, iar tabelul de distribuție selectează cele 4 secvențe de microinstrucțiuni diferite:



Eticheta este folosită pentru a determina obiectivele pentru operațiile de distribuire a secvențelor execuției.

## Microinstrucțiunile pentru referire la memorie

Prima calculează adresa, două efectuează încărcare și una scrie în fișierul de registre.

Etichetă	Control UAL	SRC1	SRC2	Control registru	Memorie	Control SriePC	Secvență
Mem1	Add	A	Extend				Dispatch 2
Lw2					Read ALU		Seq
				Write MDR			Fetch
Sw2					Write ALU		Fetch

Se calculează adresa: registru (rs)+semn extins ( $IR[15-0] \ll 2$ ) și se scrie în ALUOut  
Se folosește al 2-lea tablou de distribuție ca să se sară la LW2 și SW2

→ Citește memoria folosind pentru adresă ieșirea UAL și scrie data în MDR.  
Apoi merge la instrucțiunea următoare.

→ Scrie conținutul MDR în *rt*. Merge la microinstrucțiunea etichetată *Fetch*.

→ Scrie memoria folosind ca adresă conținutul lui ALUOut și ca valoare  
conținutul lui B. Merge la microinstrucțiunea etichetată  
*Fetch*.

## Microinstrucțiunile pentru instrucțiunile de tip R

Etichetă	Control UAL	SRC1	SRC2	Control registru	Memorie	Control ScribePC	Secvență
Rformat1	Func code	A	B				Seq
				Write ALU			Fetch

UAL operează cu conținutul registrelor A și B folosind câmpul funcțiunii pentru specificarea operației UAL. Apoi se merge la microinstrucțiunea următoare.

Valoarea ALUOut este scrisă în componenta specificată de rd. Merge la microinstrucțiunea etichetată *Fetch*.

## Microinstrucțiunile pentru ramificație

Etichetă	Control UAL	SRC1	SRC2	Control registru	Memorie	Control SriePC	Secvență
Beq1	Subt	A	B			ALUOut cond	Fetch

↓ ↓ ↓  
UAL scade operanzii din A și B pentru a genera ieșirea Zero.

↓  
Determină ca PC să fie scris folosind valoarea care este deja în UAL, cu condiția ca ieșirea Zero a UAL-ului să fie adevărată.

↓  
Merge la microinstrucțiunea etichetată *Fetch*.



## Microinstrucțiunile pentru salt

Etichetă	Control UAL	SRC1	SRC2	Control registru	Memorie	Control SriePC	Secvență
Jump1						Jump address	Fetch

Determină ca PC-ul să fie scris folosind adresa obiectiv pentru salt.

Merge la microinstrucțiunea etichetată *Fetch*.

## Programul întreg

Etichetă	Control UAL	SRC1	SRC2	Control registru	Memorie	Control ScribePC	Secvență
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC		Read			Dispatch 1
Mem1	Add	A	Extend				Dispatch 2
Lw2					Read ALU		Seq
				Write MDR			Fetch
Sw2					Write ALU		Fetch
Rformat1	Func code	A	B				Seq
				Write ALU			Fetch
Beq1	Subt	A	B			ALUOut cond	Fetch
Jump1						Jump address	Fetch

Pentru implementarea lui se poate folosi un ROM sau un PLA.

## Excepții

---

Controlul reprezintă aspectul cel mai delicat al proiectării unui procesor.

Cea mai dificilă parte a controlului o reprezintă implementarea:

- ***Excepțiilor***      ➡      evenimente neașteptate din interiorul procesorului  
(de ex. depășirea aritmetică)
- ***Întreruperilor***      ➡      evenimente ce produc schimbări neașteptate în fluxul  
controlului, care sosesc însă din exteriorul procesorului.  
*Ex.* - sunt folosite de dispozitivele de I/E pentru  
comunicarea cu calculatorul.

### ***Alte exemple:***

- Invocarea, din programul utilizatorului, a sistemului de operare
- Folosirea unei instrucțiuni nedefinite
- Funcționare hardware defectuoasă

❑ Detectarea condițiilor excepționale și efectuarea acțiunilor corespunzătoare se situează adesea pe calea de *timp critic* al mașinii, care determină durata ciclului de ceas și deci – performanța.

❑ La apariția unei excepții, mașina efectuează salvarea, în *contorul de program pentru excepții (EPC)*, a adresei instrucțiunii care a produs perturbarea și apoi transferul controlului către sistemul de operare, la o adresă specificată.

❑ Sistemul de operare poate să hotărască în continuare:

- ✓ efectuarea acțiunii corespunzătoare
- ✓ efectuarea unor acțiuni predefinite ca răspuns la depășire
- ✓ oprirea execuției programului și raportarea unei erori.

Arhitectura MIPS stabilește *cauza* excepției prin folosirea unui *Registru de Cauză*, care conține un câmp ce indică motivul care a provocat excepția.

Altă metodă este utilizarea *întreruperilor cu vector*.

*Exemplu:*

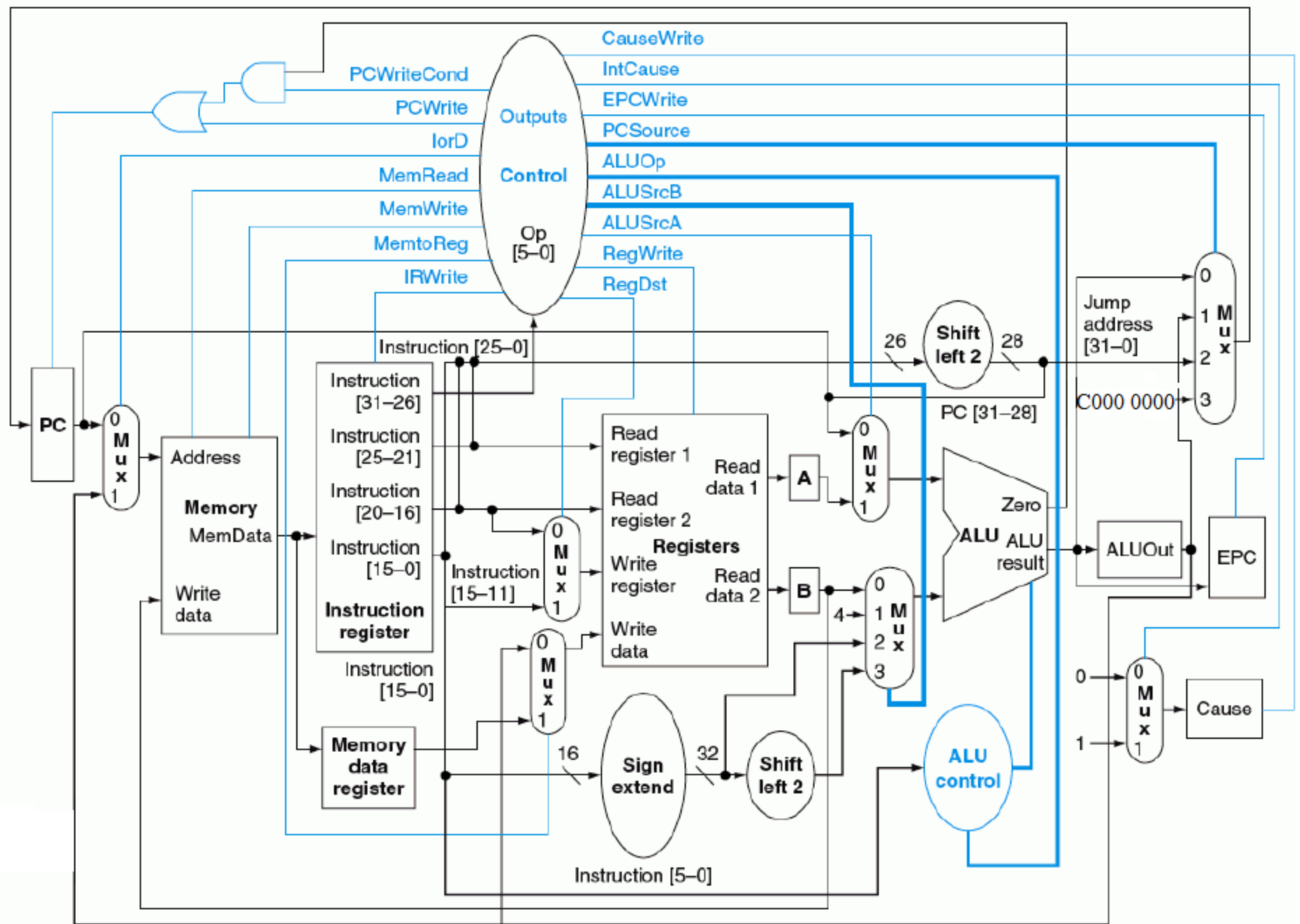
- Instrucțiune nedefinită ➡ adresa vectorului de excepție în hex este:  
 $C0\ 00\ 00\ 00_{hex}$
- Depășire aritmetică ➡ adresa vectorului de excepție în hex este:  
 $C0\ 00\ 00\ 20_{hex}$

## Arhitectura MIPS adaugă:

- ✓ **EPC** = un registru de 32 biți pentru păstrarea adresei instrucțiunii afectate de excepție.
- ✓ **Cauză** = un registru de 32 biți pentru înregistrarea cauzei.  
Bitul inferior codifică cele două excepții posibile:

Instrucțiune nedefinită = 0

Depășire aritmetică = 1




## Semnale de control pentru tratarea excepțiilor

---

**EPCWrite** = pentru scrierea în registrul EPC

**CauseWrite** = pentru scrierea în registrul Cauză

**IntCause** = semnal de un bit pentru a seta corespunzător bitul inferior al registrului Cauză

În PC trebuie scrisă adresa excepției  multiplexorul va avea 4 căi, intrarea suplimentară fiind setată la valoarea constantă C000 0000 hex.

Pentru a selecta ca această valoare să fie scrisă în PC, semnalul PCSource trebuie setat pe 11.