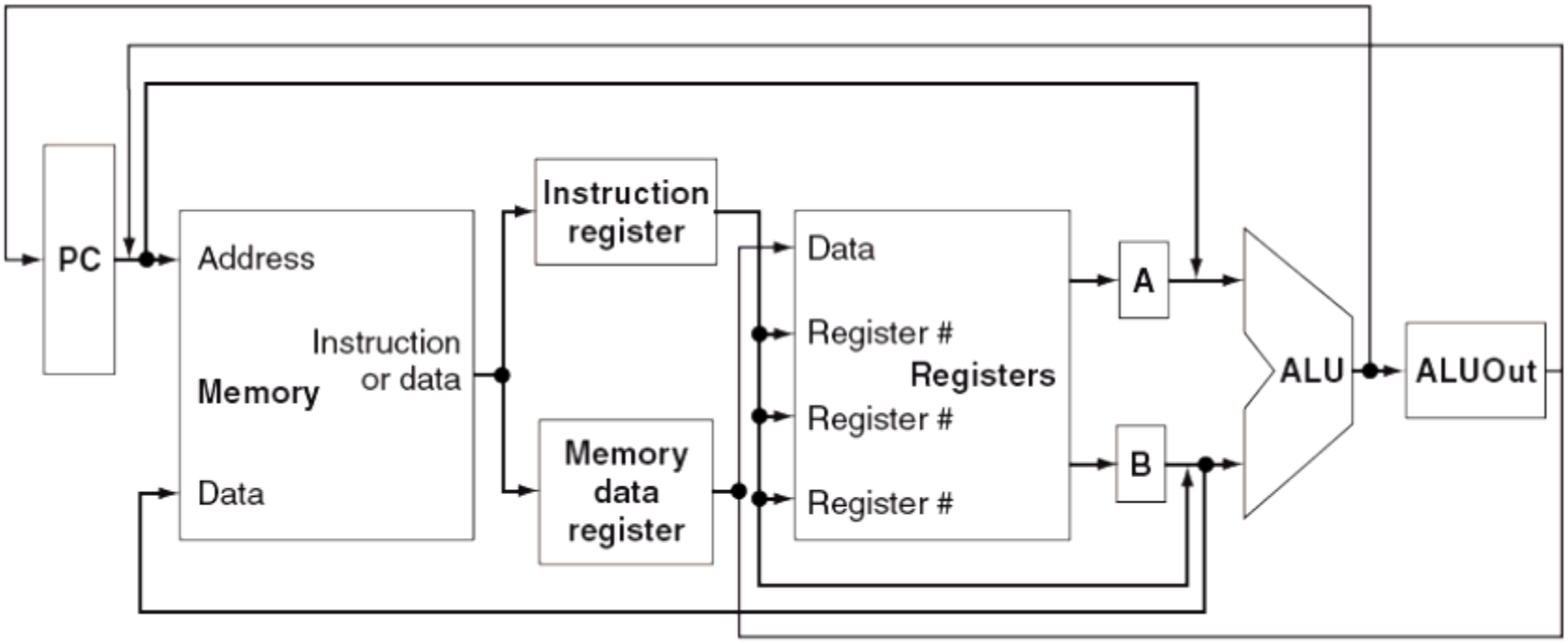
IMPLEMENTAREA CU MAI MULTE CICLURI

Într-o implementare cu mai multe cicluri, fiecare pas al execuției va necesita o perioadă de ceas.

Unitatea funcțională este utilizată mai mult decât o singură dată pe instrucțiune – utilizată evident în cicluri de ceas diferite

Acestea sunt avantajele majore ale implementării cu mai multe cicluri de ceas



La finalul unui ciclu de ceas toate datele care sunt folosite în ciclurile următoare trebuie memorate în elemente de stare

Datele folosite de instrucțiunile următoare într-un ciclu ulterior de ceas vor fi memorate în elementele de stare vizibile programatorului.

Datele folosite de aceeași instrucțiune într-un ciclu ulterior de ceas trebuie memorate în registrele suplimentare

SE PRESUPUNE

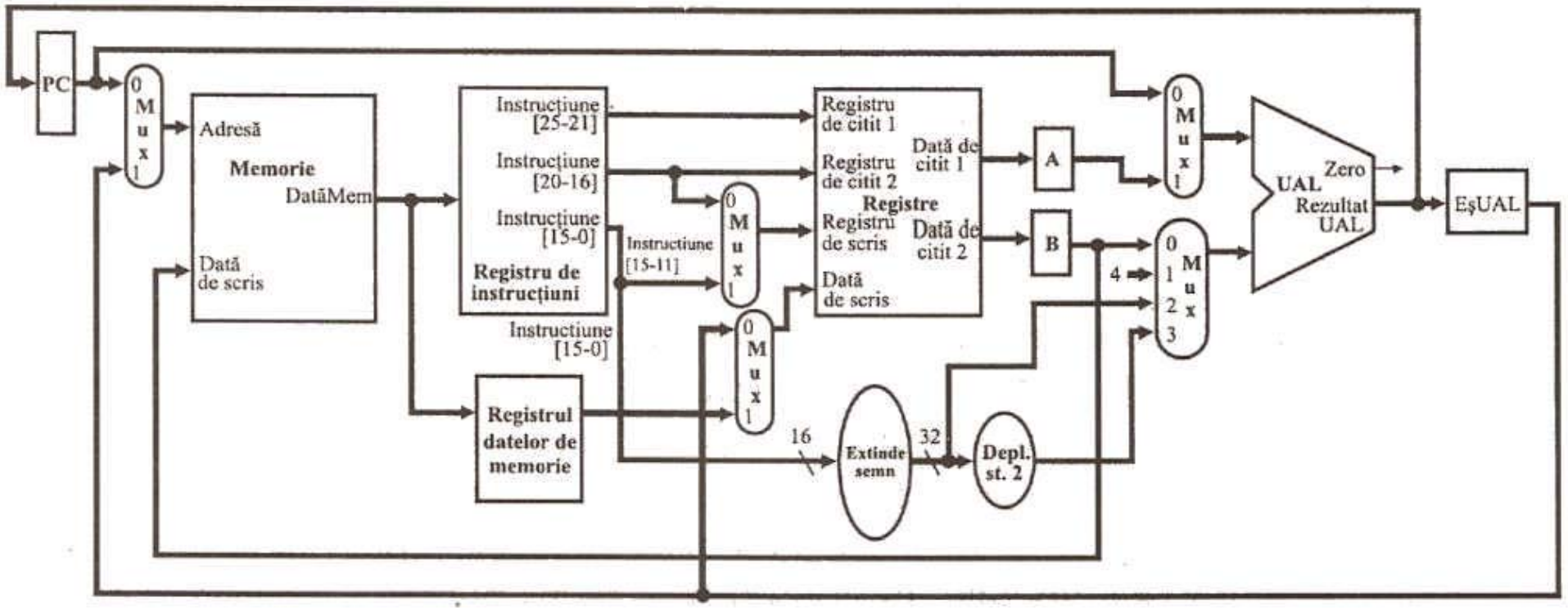
Durata ciclului de ceaspoate deservi cel mult:

- un acces la memorie
- un acces la fișierul de registre – 2 citiri și o scriere
- o operație UAL

Registrul de instructiuni (RI) și Registrul datelor de memorie (MDR)

Registrele A și B

Registrul EșUAL



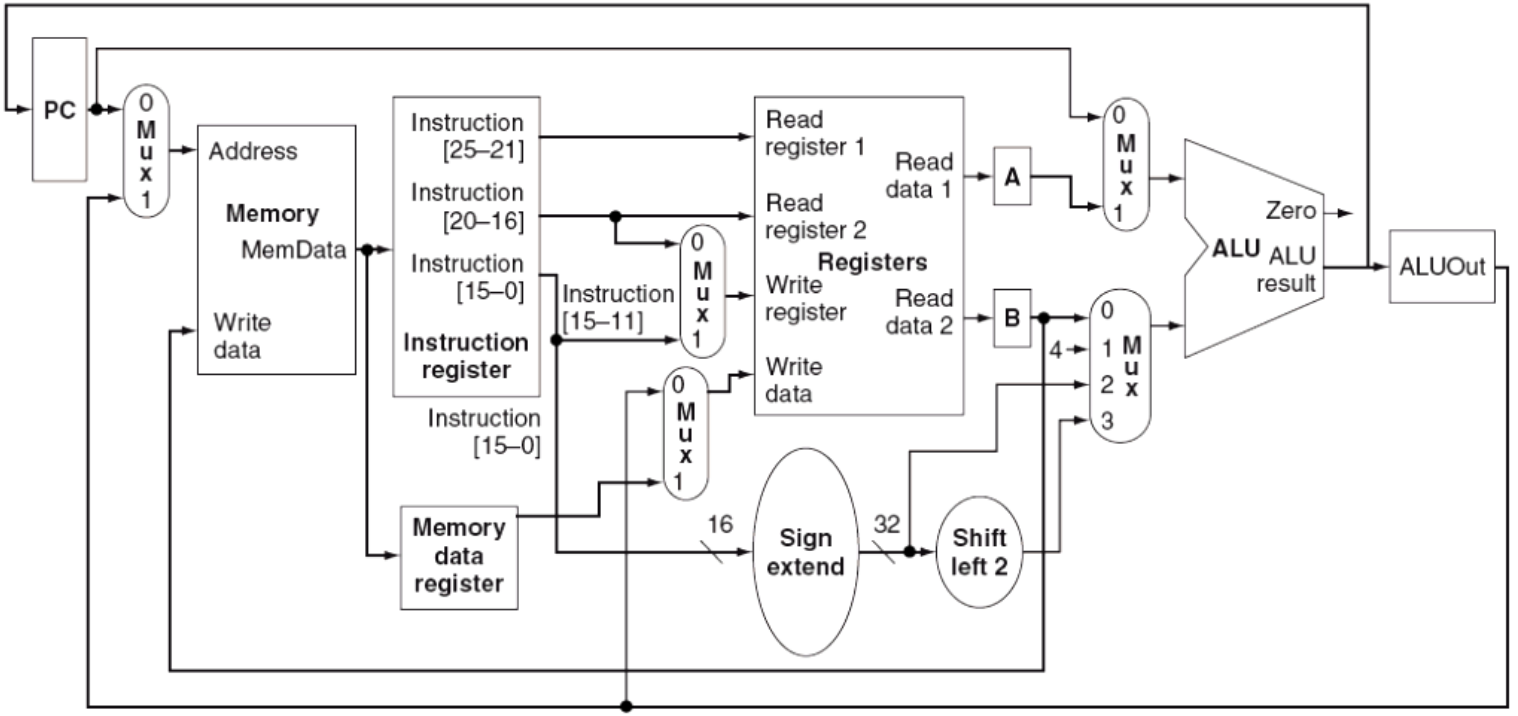
Folosim mai multe unități funcționale în comun, deci trebuie adăugate multiplexoare

Exp: Avem o singură memorie atât pentru date cât și pentru instrucțiuni – PC și OpUAL

Folosim un singur UAL în loc de 3 ca în cazul implementării cu un singur ciclu de ceas, deci vom avea următoarele schimbări

• un multiplexor suplimentar adăugat primei intrări în UAL

• multiplexorul celei de-a doua intrări în UAL din MUX2 se transformă în MUX4



CE AM OBȚINUT ?

- Reducerea numărului de un itați de memorie de la 2 la 1
- Eliminarea a două sumatoare

AM OBȚINUT O REDUCERE SEMNIFICATIVĂ A COSTULUI HARDWARE-ULUI

CE NU AM IMPLEMENTAT ?

- ramificațiile
- salturile

Având în vedere salturile precum și ramificațiile există 3 surse posibile pentru PC

- ieșirea UAL – PC + 4

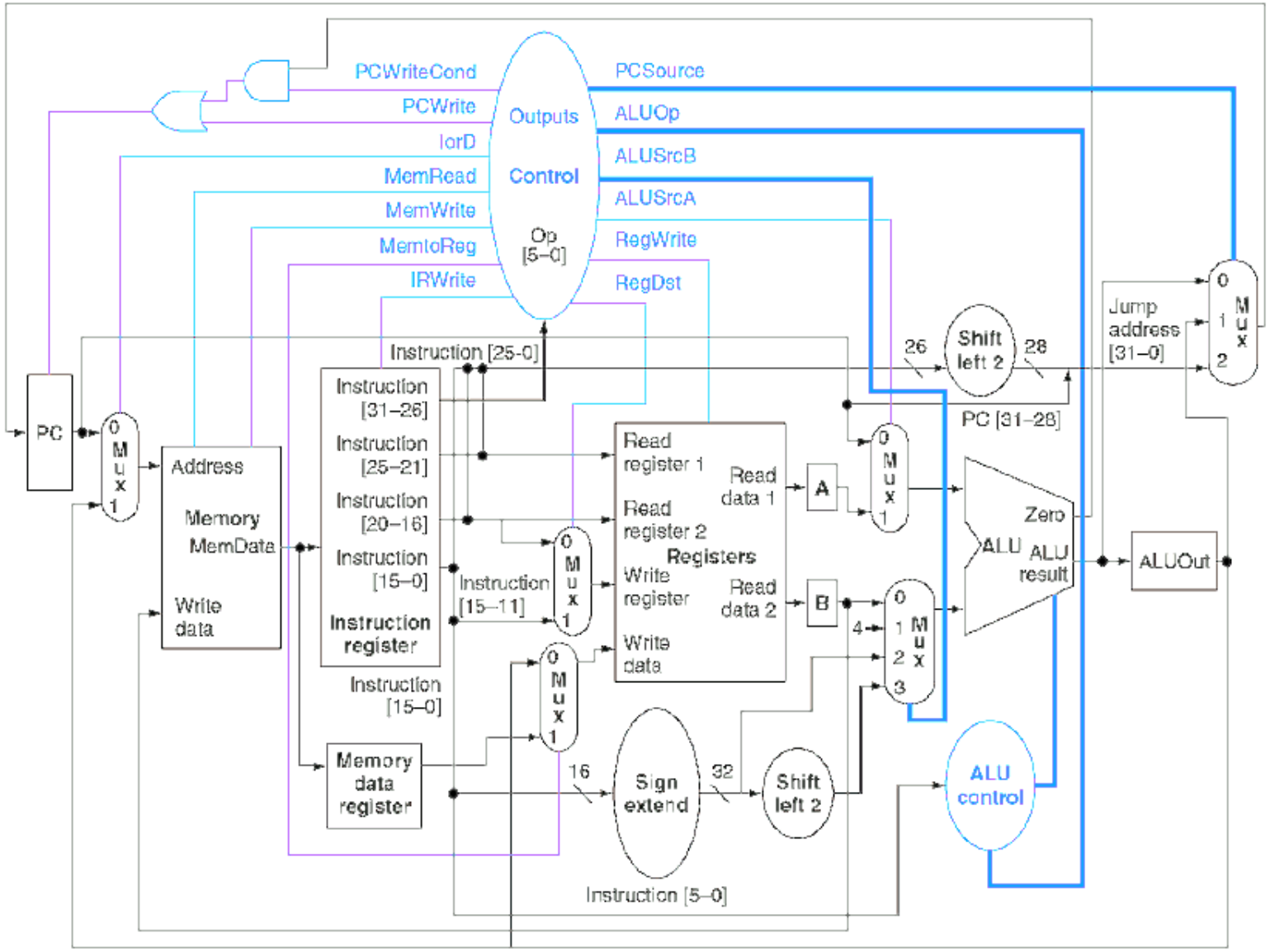
- registrul EșUAL cel care memorează adresa obiectiv pentru ramificație evident după determinarea sa

- cei 26 de biți inferiori ai lui IR deplasați spre stânga cu 2 poziții și concatenați cu cei 4 biți superiori ai PC-ului incrementat – sursa pentru instrucțiunea de salt

CONCLUZIE – PC-ul va trebui scris atât condiționat cât și necondiționat !!!

Vom folosi magistrale partajate

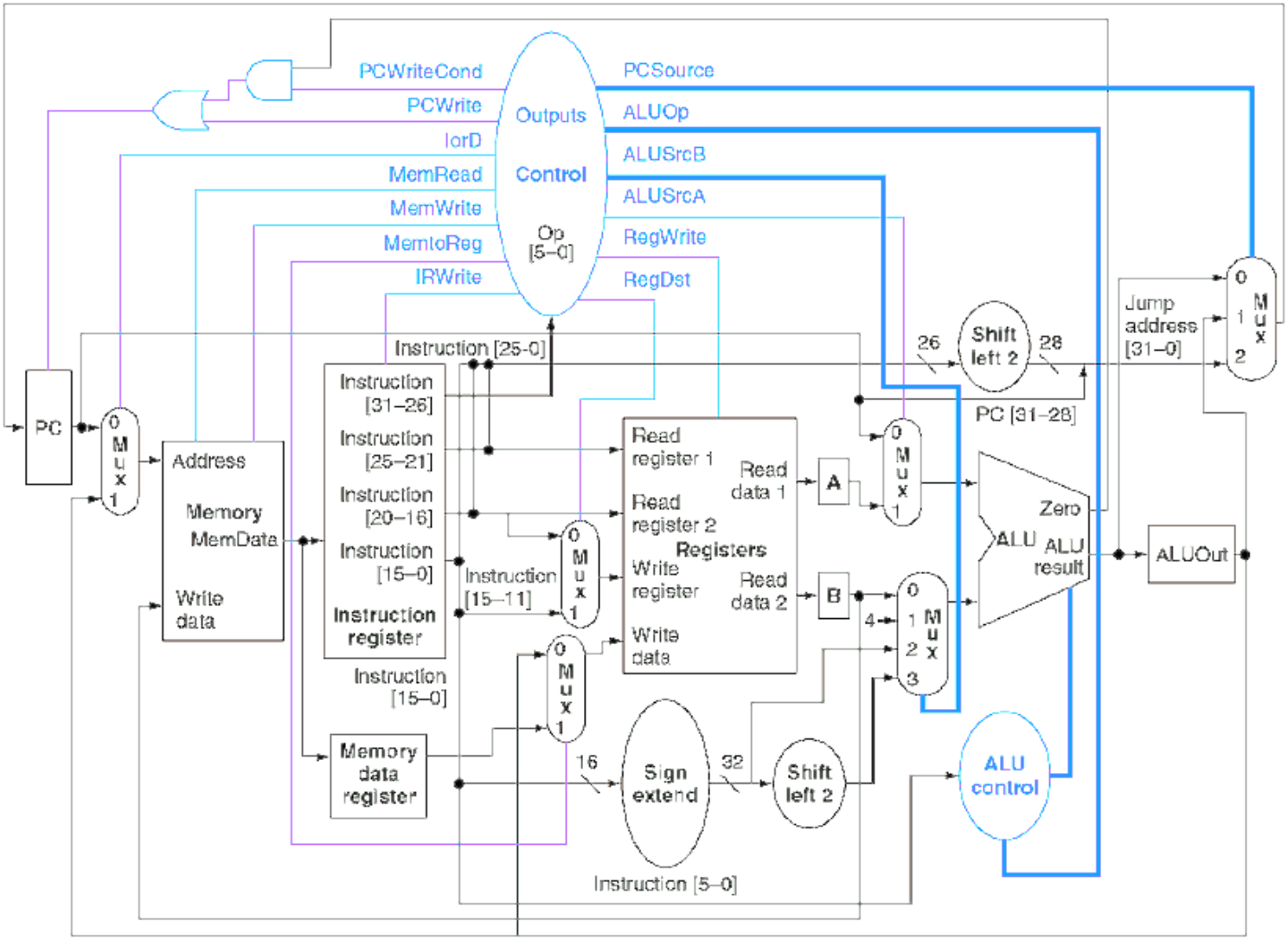
Calea de date completă cu liniile de control necesare



Pașii de execuție a instrucțiunii

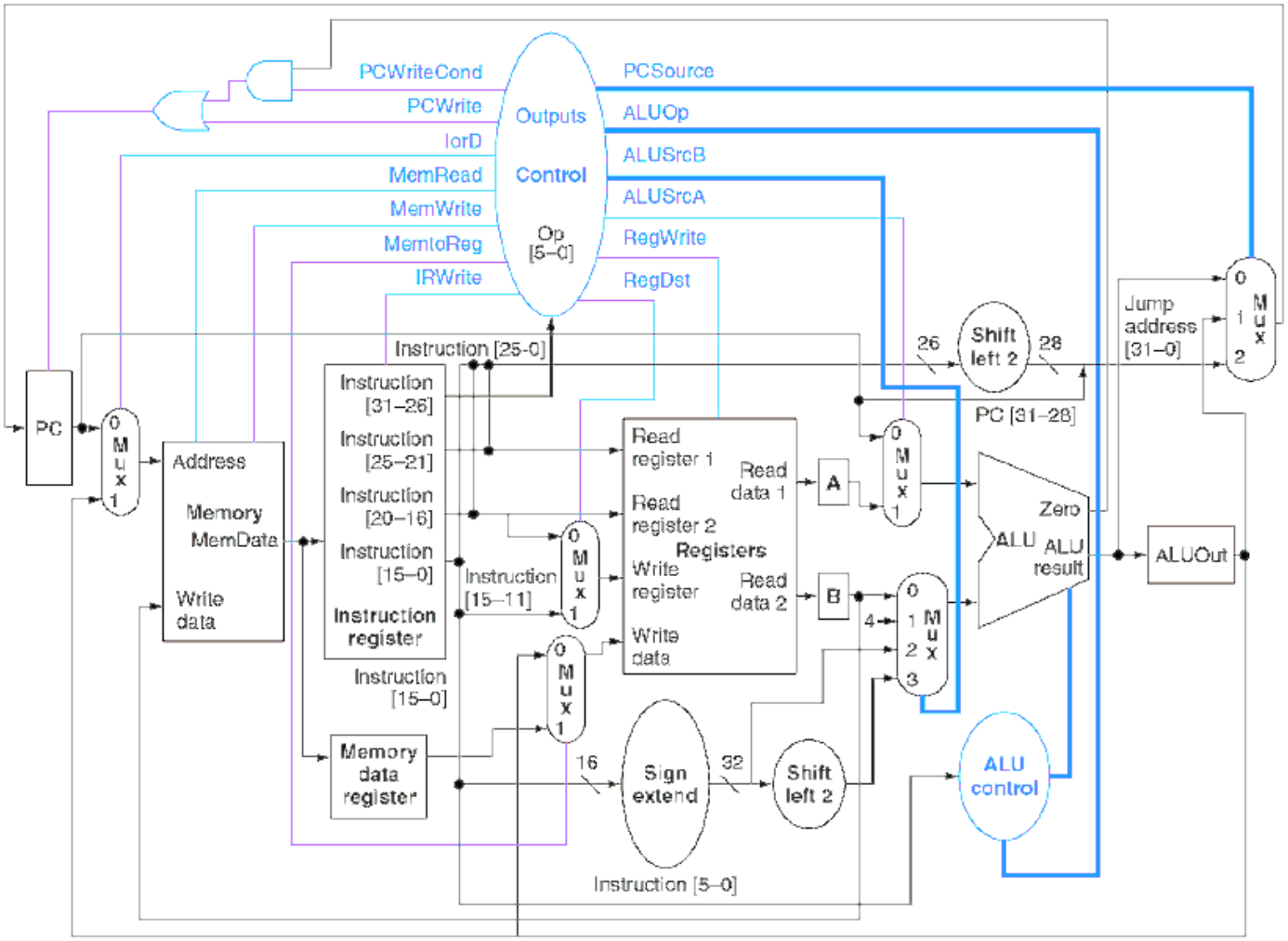
EXTRAGEREA INSTRUCȚIUNII

IR = Memorie(PC);
PC = PC+4



DECODIFICAREA INSTRUCŢIUNII ŞI EXTRAGEREA REGISTRELOR

A = Reg(IR(25-21))
B = Reg (IR(20-16))
EşUAL = PC + (semn-extins(IR(15-0)) << 2);



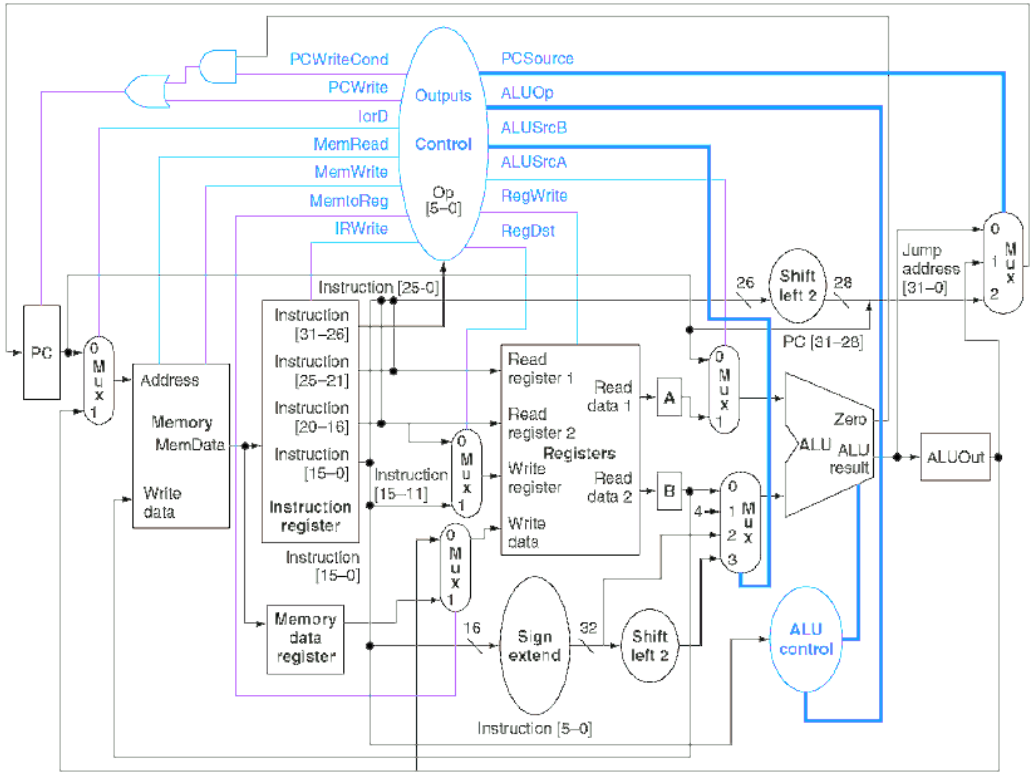
Execuția, calculul adresei de memorie sau terminarea ramificației

Referință la memorie – IeșireUAL = A + semn-extins (IR(15-0))

Instrucțiune tip R – IeșireUAL = A op B

Ramificație – dacă (A==B) atunci PC=EșUAL

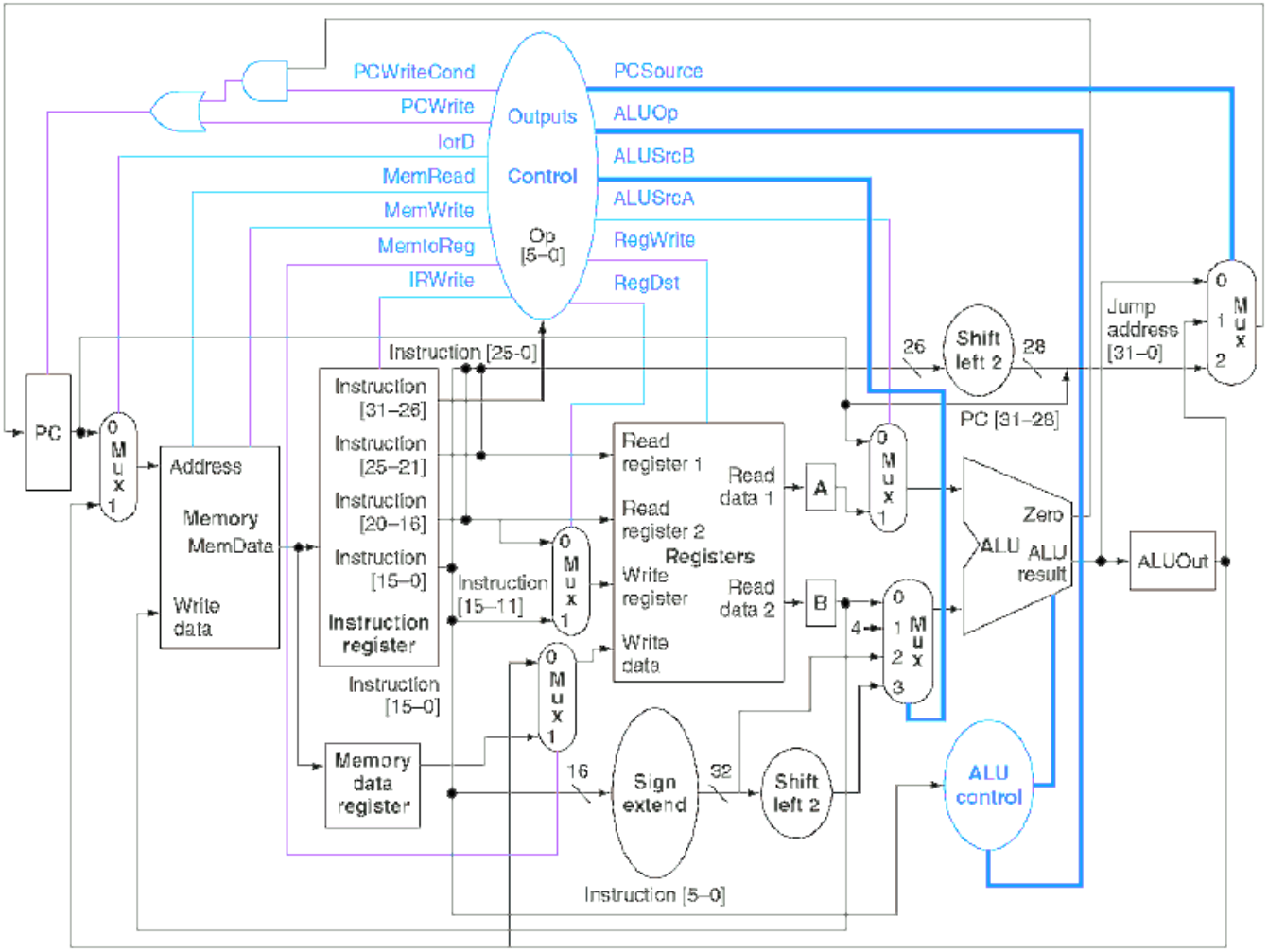
Salt – PC = PC(31-28) || IR(25-0) << 2



Pasul de acces la memorie sau de terminare a instrucțiunii de tip R

O instrucțiune de încărcare/memorare accesează memoria, iar o instrucțiune aritmetică-logică își scrie rezultatul.

Valoarea citită din memorie este scrisă în MDR de unde va fi folosită în ciclul următor

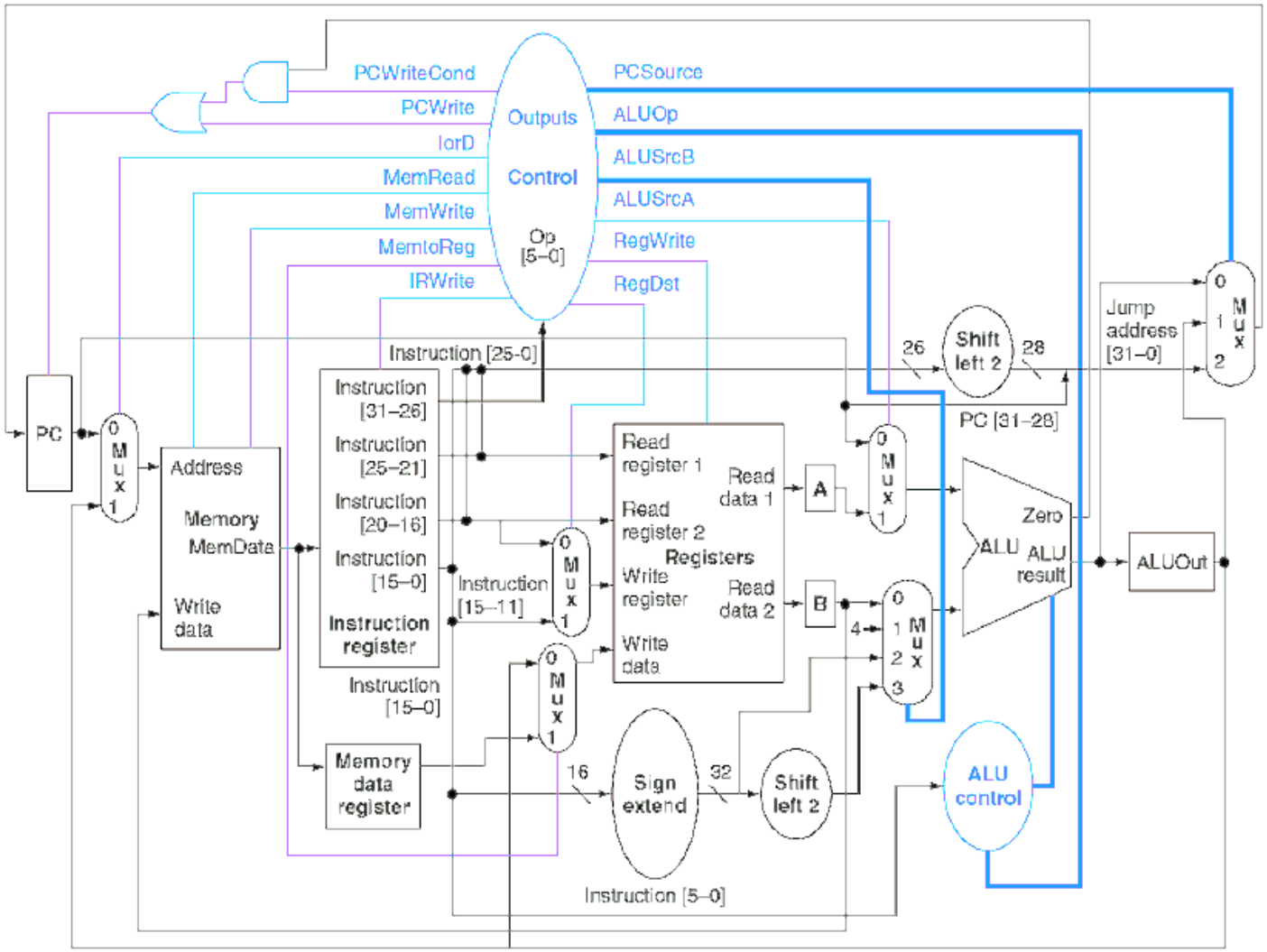


MDR = Memorie (EșUAL)

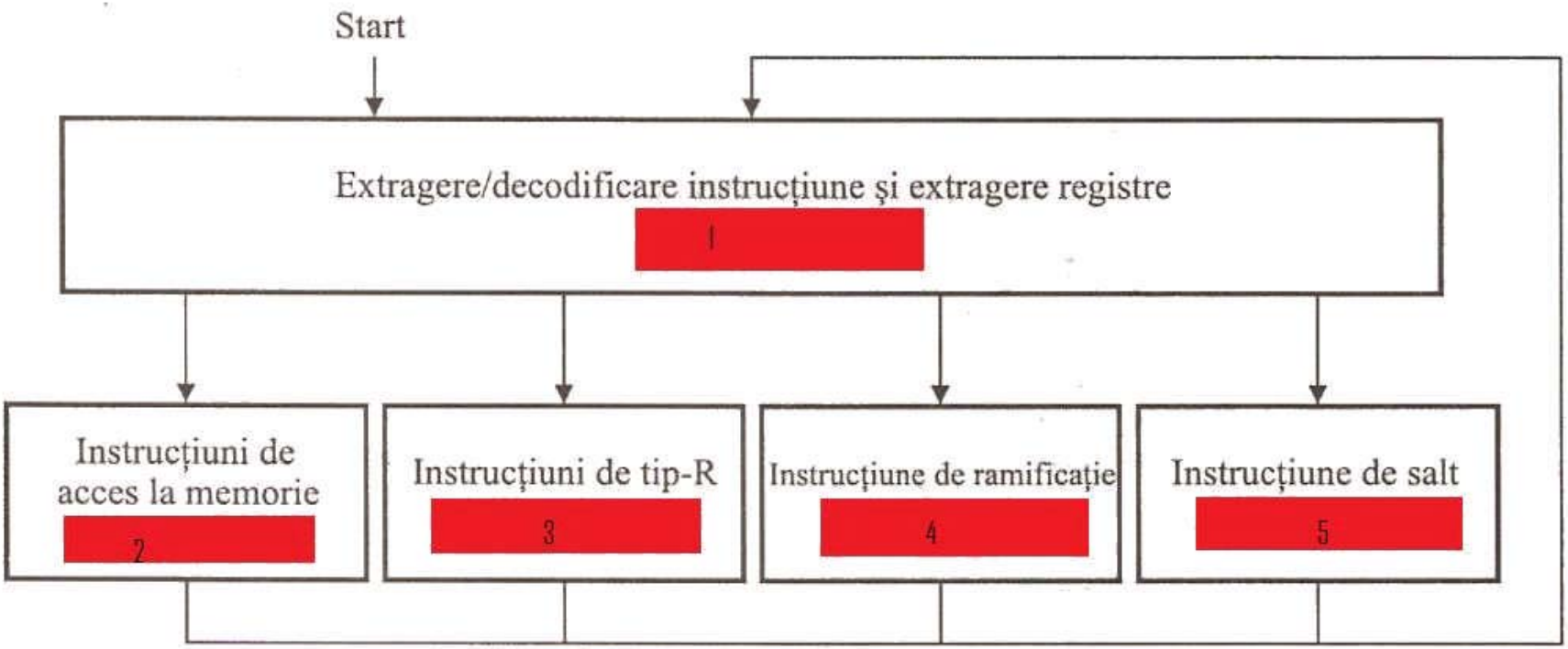
Reg(IR(15 -11)) = EșUAL

Pasul de terminare a citirii memoriei

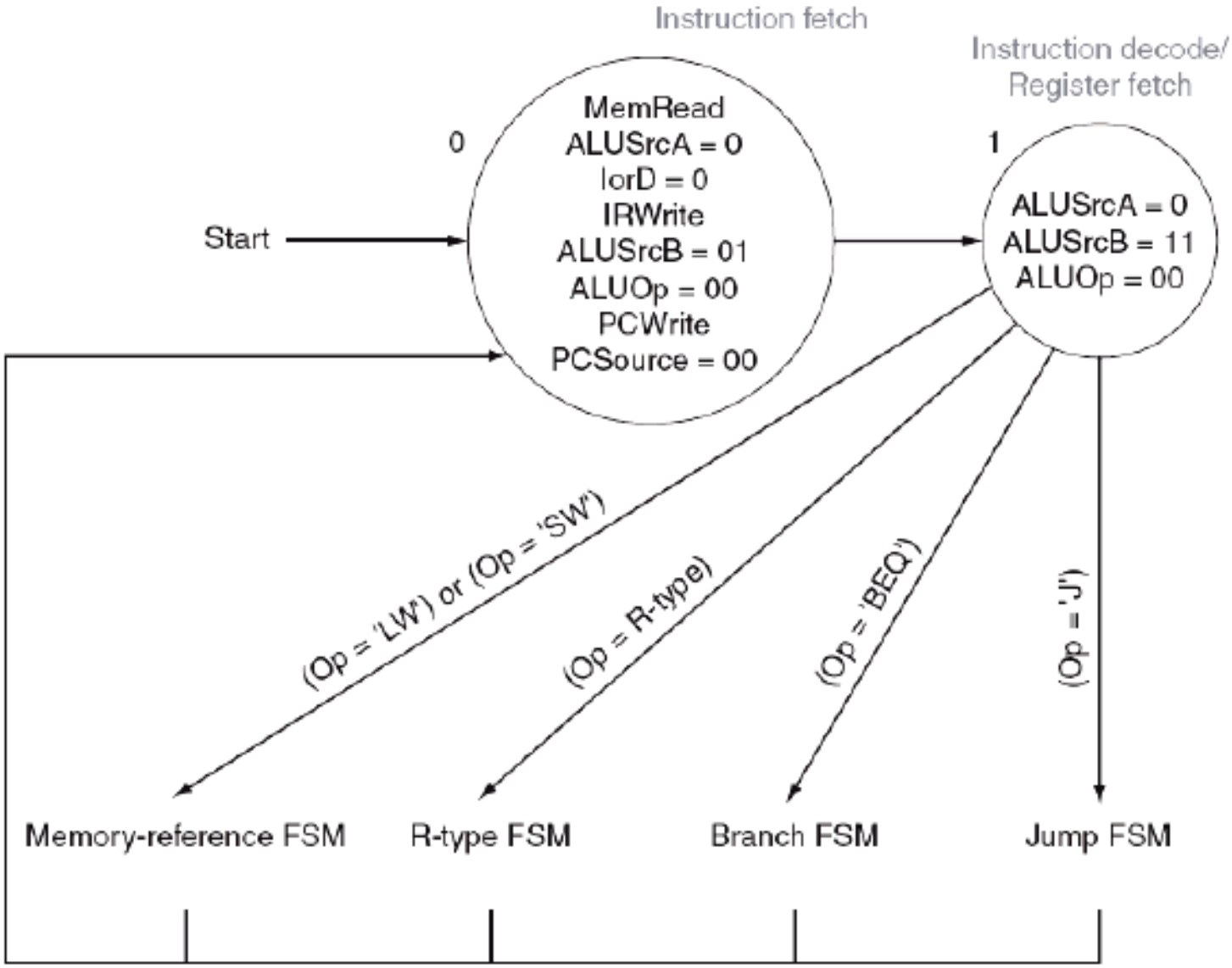
Reg(IR(20-16)) = MDR



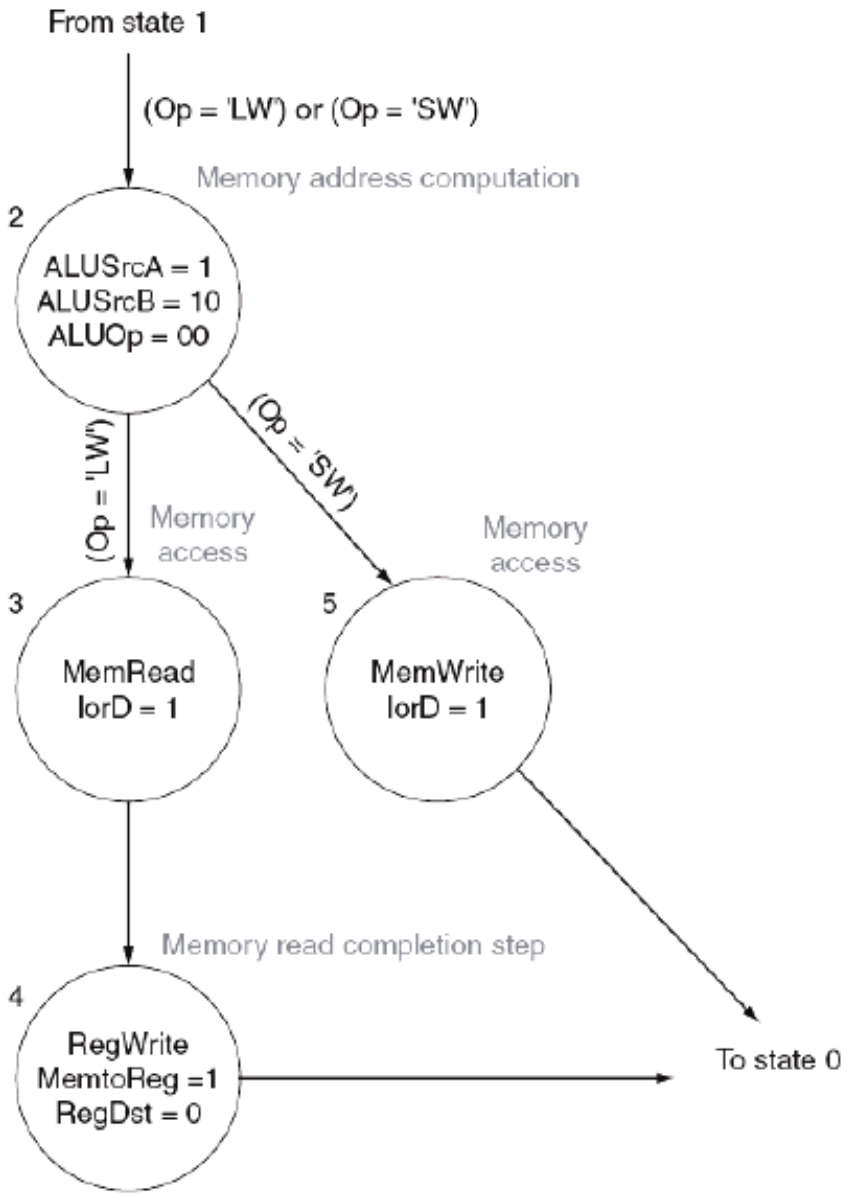
Definirea controlului



Instrucțiunea este extrasă și decodificată



Controlul instrucțiunilor de referire a memoriei



Instrucțiunile de tip R

Instrucțiunile de salt și ramificație și de salt