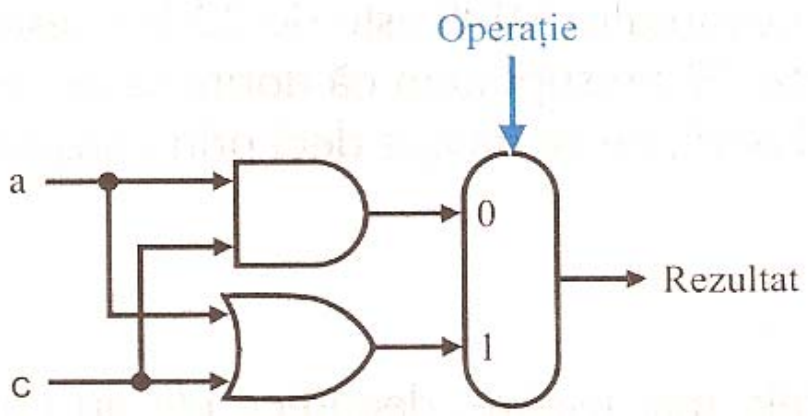
Proiectarea UAL

Operaţii logice cu corespondenţă directă în hardware

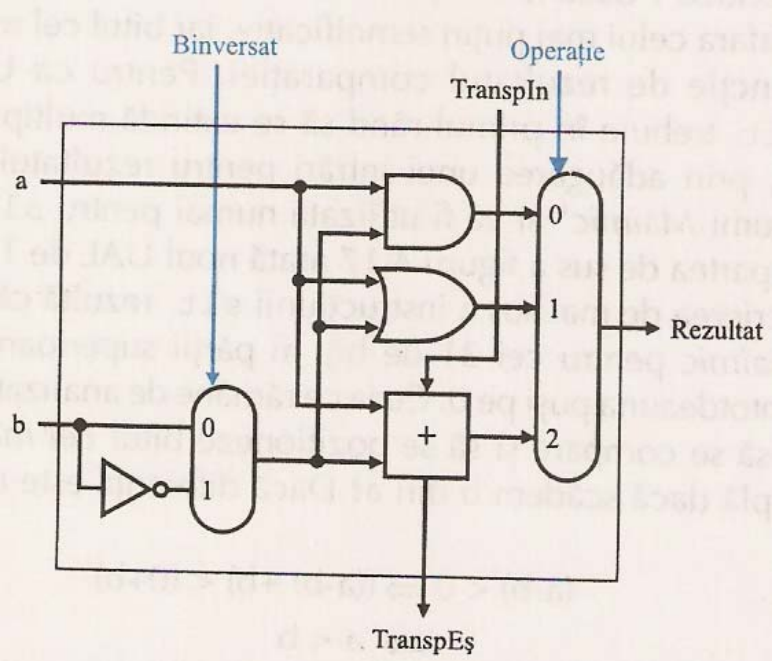
AND	c = a * b
SAU	c = a + b
INV	c = not a
MUX	dacă d==0, c=a altfel c=b

Unitatea logică pentru 1 bit – liniile albastre sunt linii de comandă



Realizarea operaţiei de adunare presupune existenţa unui sumator complet

Realizarea operaţiei de scădere presupune introducerea unui inversor

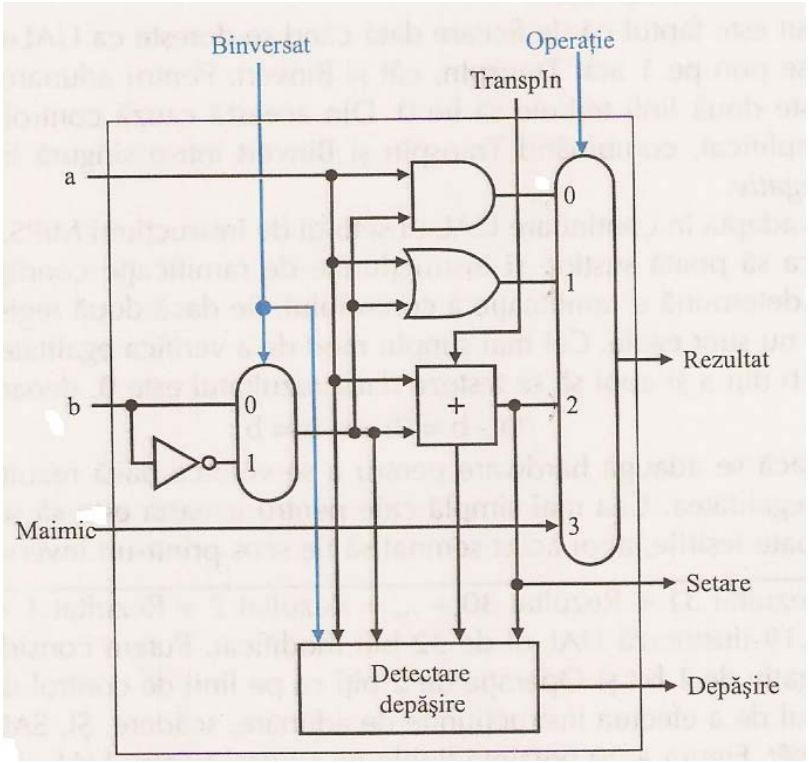


Hardware-ul rezultat este foarte simplu => cc2 standard universal pentru aritmetica numerelor întregi

Ce operaţie încă nu se poate realiza ?

Operaţia ***slt*** produce 1 dacă rs < rt şi 0 altfel

Se observă că este suficient să determinăm bitul de semn al operaţiei a-b



Cum se poate modifica figura alăturată astfel încât să implementăm ramificarea condițională ?

Sumatorul este implementat cu ajutorul unui CLA - Carry Look Ahead prezentat la CN 1

Operațiile de înmulțire/împărțire inclusiv virgula mobilă au prezentate în CN 1

Calea de date și de control

Durata perioadei de ceas precum și numărul de cicluri/instrucțiune sunt date de implementarea procesorului.

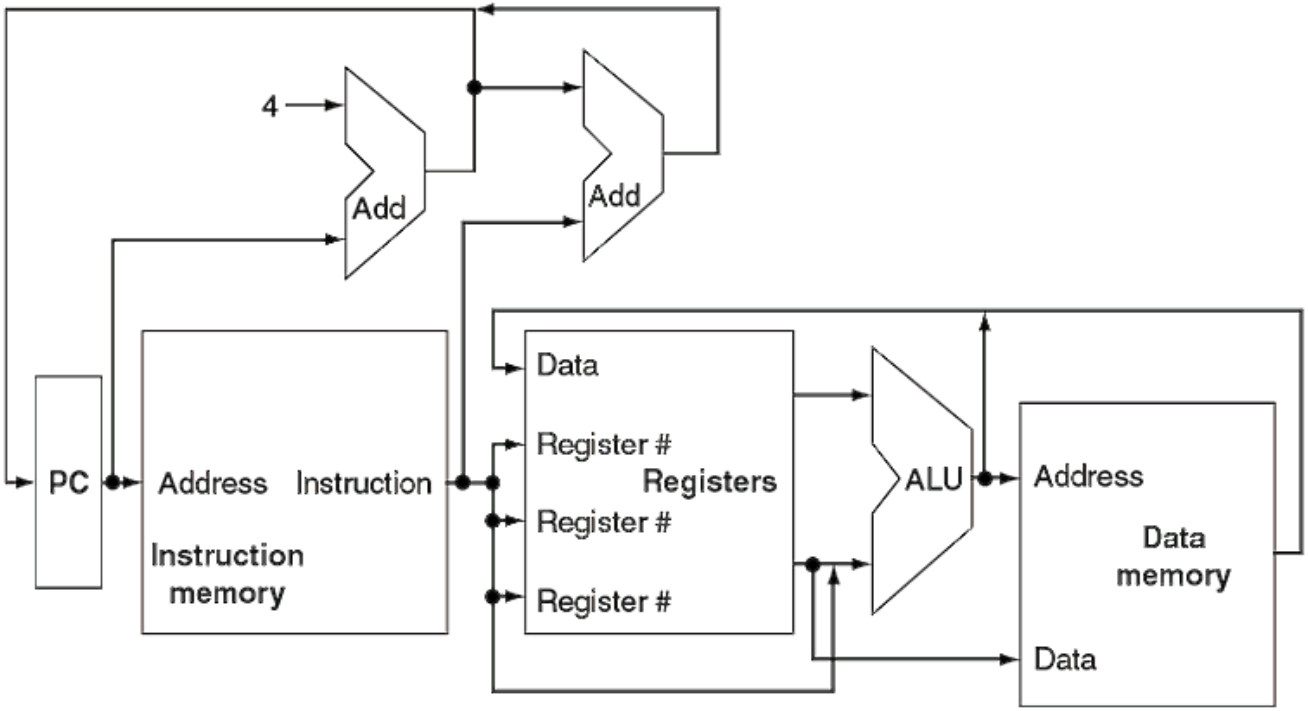
Ne propunem implementarea următoarelor instrucțiuni

- ✓ instrucțiuni de referire a memoriei - încărcare/memorare
- ✓Instrucțiuni aritmetice și logice - add, sub, and, or și slt
- ✓Instrucțiunile de ramificație la egal și de salt

Indiferent de clasa instrucțiunii, primii pași de implementarea aunei instrucțiuni sunt la fel:

- ✓ se va trimite memoriei PC-ul și se va extrage instrucțiunea din memorie
- ✓Se citesc 1 sau 2 registre - vom folosi câmpurile instrucțiunii pentru selectarea registrelor

Unitățile funcționale și legăturile dintre ele

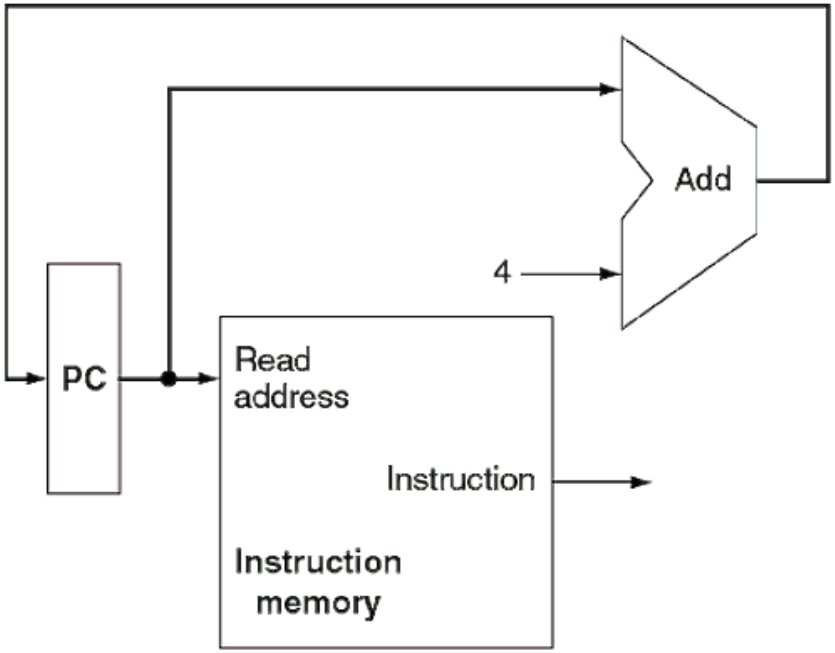


✓ Dacă instrucțiunea este de tip aritmetic sau logic => rezultatul din UAL trebuie scris într-un registru

✓ Dacă operația este de încărcare/memorare => rezultatul UAL va fi o adresă

✓ Pentru ramificații vom folosi ieșirea UAL în determinarea adresei următoarei Instrucțiuni de executat. Este necesar pentru aceasta să introducem o logică de control

Realizarea căii de date - determinarea instrucțiunii curente și trecerea la următoarea instrucțiune

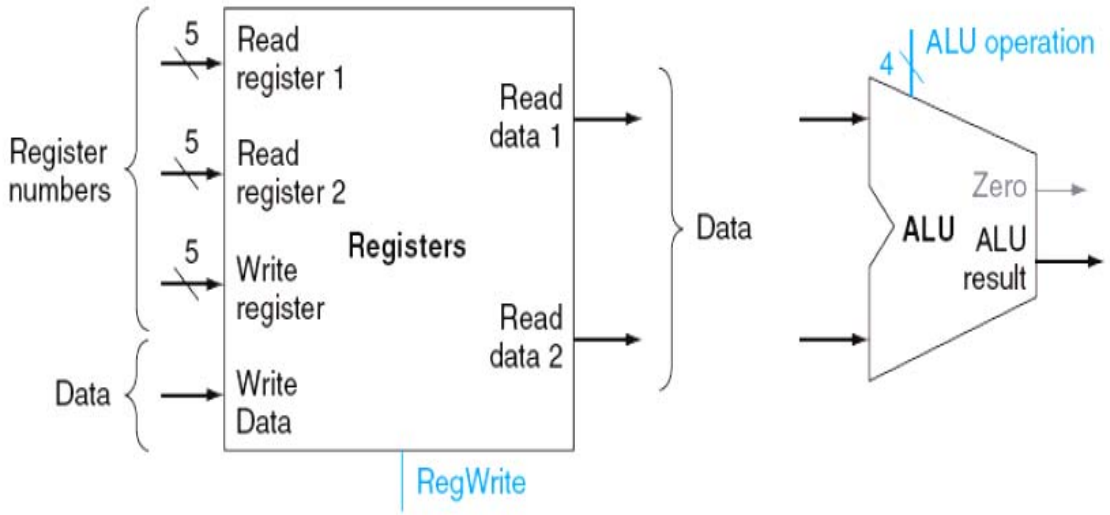


Extragem instrucțiunea din memorie

Incrementăm CP-ul cu 4 pentru Trecerea la instrucțiunea următoare

Instrucțiunea următoare este situată la o distanță de 4 octeți

Realizarea căii de date - instrucțiunile aritmetice și logice

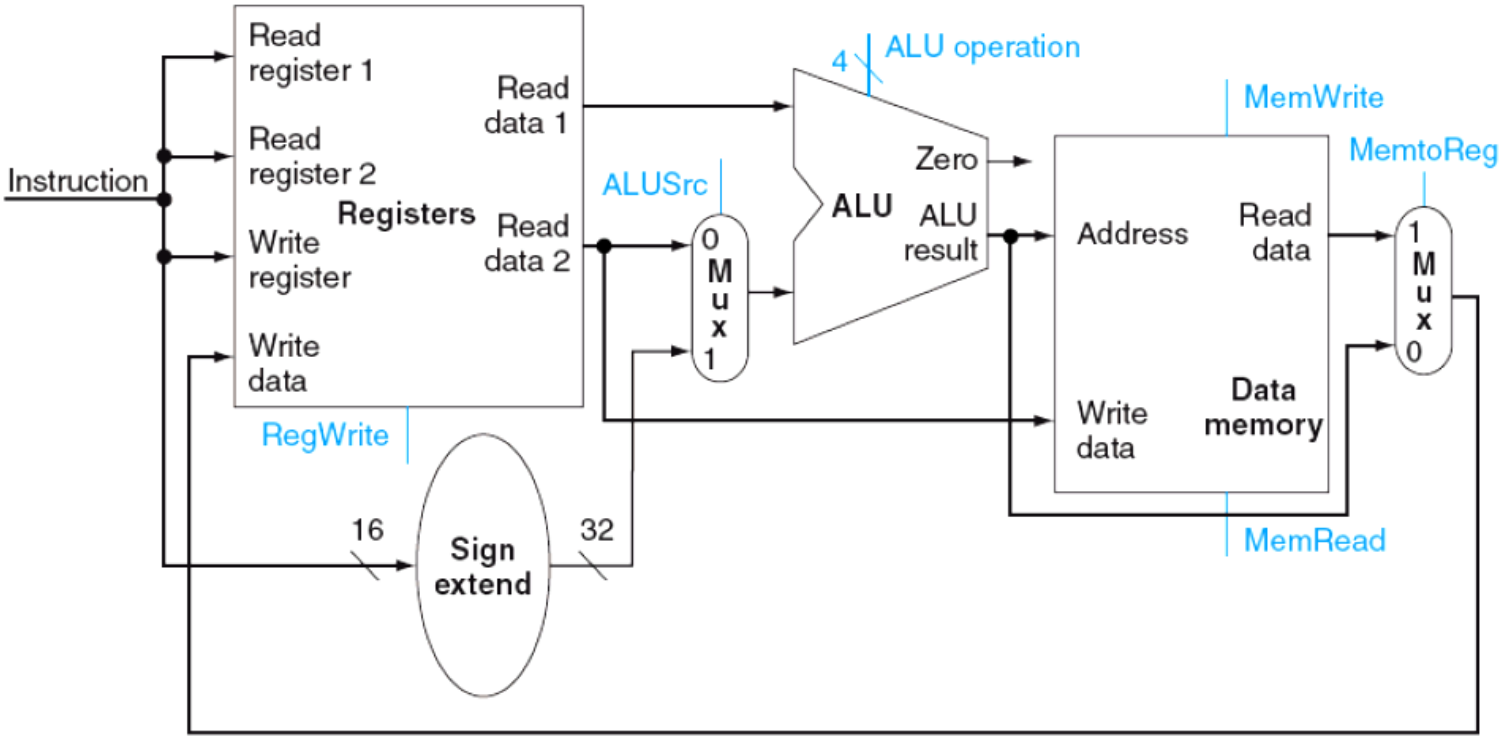


Instrucțiunile de tipul R
add, sub, slt

Registre generale

UAL care operează cu valorile citite din registre

Realizarea căii de date - instrucțiunile de încărcare și memorare



Instrucțiunile sunt:

lw $t1, valoare_deplasare($t2)
sw $t1, valoare_deplasare($t2)

Este necesară o unitate pentru a extinde semnul câmpului de deplasare din instr.

Realizarea căii de date - instrucțiunea beq

Instrucțiunea este formată din 2 registre care sunt comparate pentru egalitate și o deplasare de 16 biți

beq $t1, $t2, offset

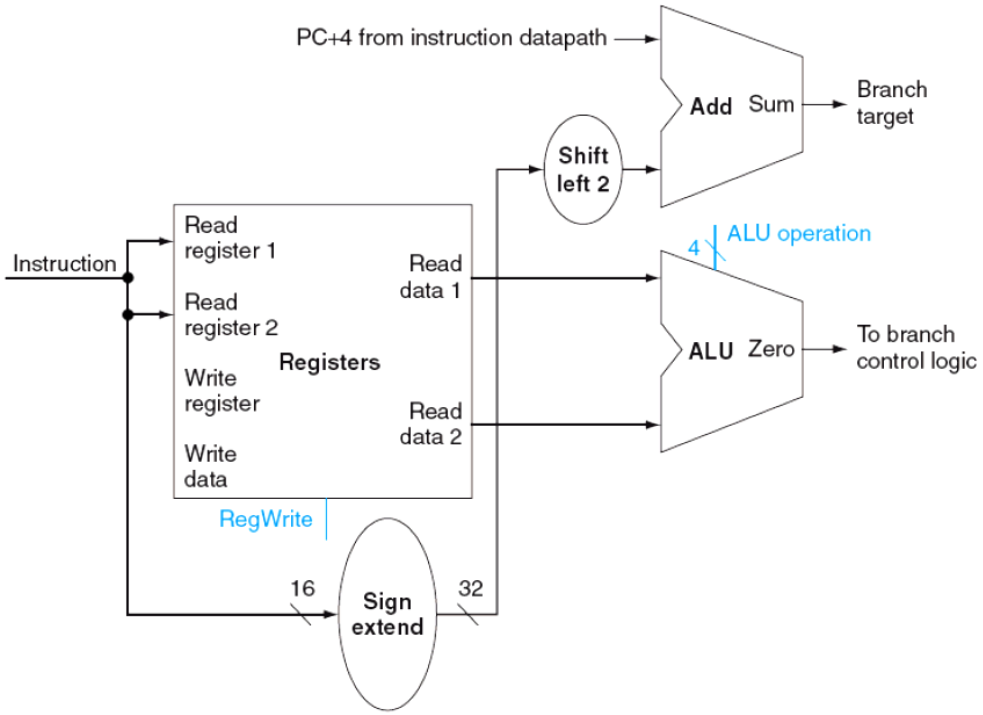
În vederea implementării acestei instrucțiuni trebuie determinată adresa obiectiv pentru ramificație => PC + câmpul de deplasare al instrucțiunii cu semnul extins

Particularități:

1. Baza pentru calculul adresei de ramificație este adresa instrucțiunii care urmează ramificației => PC+4 valoarea ca bază

2. Câmpul deplasării din instrucțiune trebuie deplasat stânga cu 2 biți deoarece deplasarea este la nivel de cuvânt => crește domeniul efectiv al câmpului deplasării cu un factor de 4.

Realizarea căii de date - instrucțiunea beq



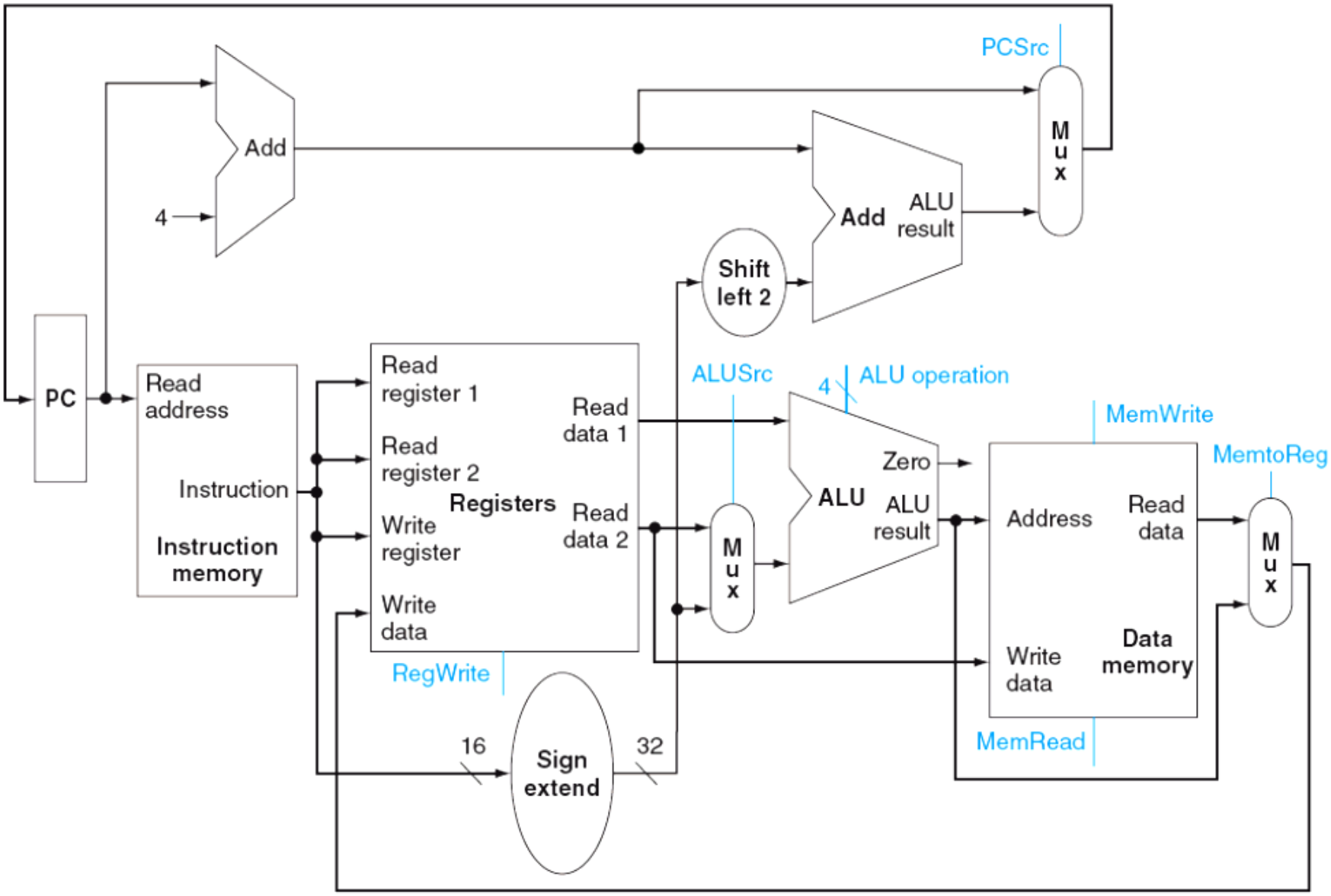
Operanzii sunt egali => adresa obiectiv pentru ramificație devine noul PC

Operanzii sunt diferiți => PC-ul incrementat este noul PC

Trebuie realizate 2 operații:

1. Calcularea adresei obiectiv pentru ramificație
2. Compararea conținutului registrelor

Obiectiv final - calea de date pentru execuția într-un singur ciclu de ceas



Controlul UAL

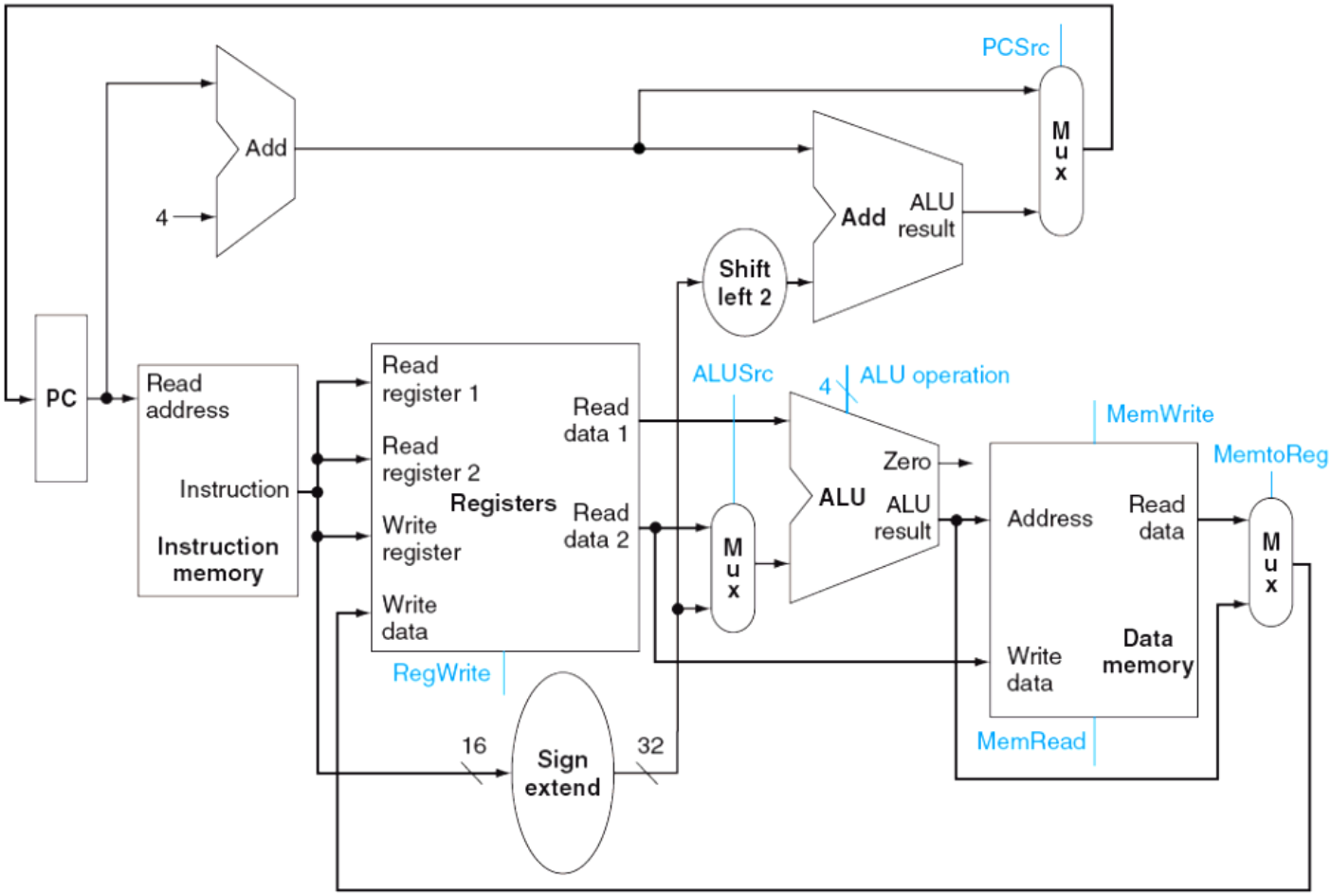
UAL are trei intrări de control din care folosite sunt 5

000	ȘI
001	SAU
010	+
110	-
111	setare la mai mic decât

Modalitatea de implementare - utilizăm 2 biți (OpUAL) și cei 6 biți ai codului funcțiunii

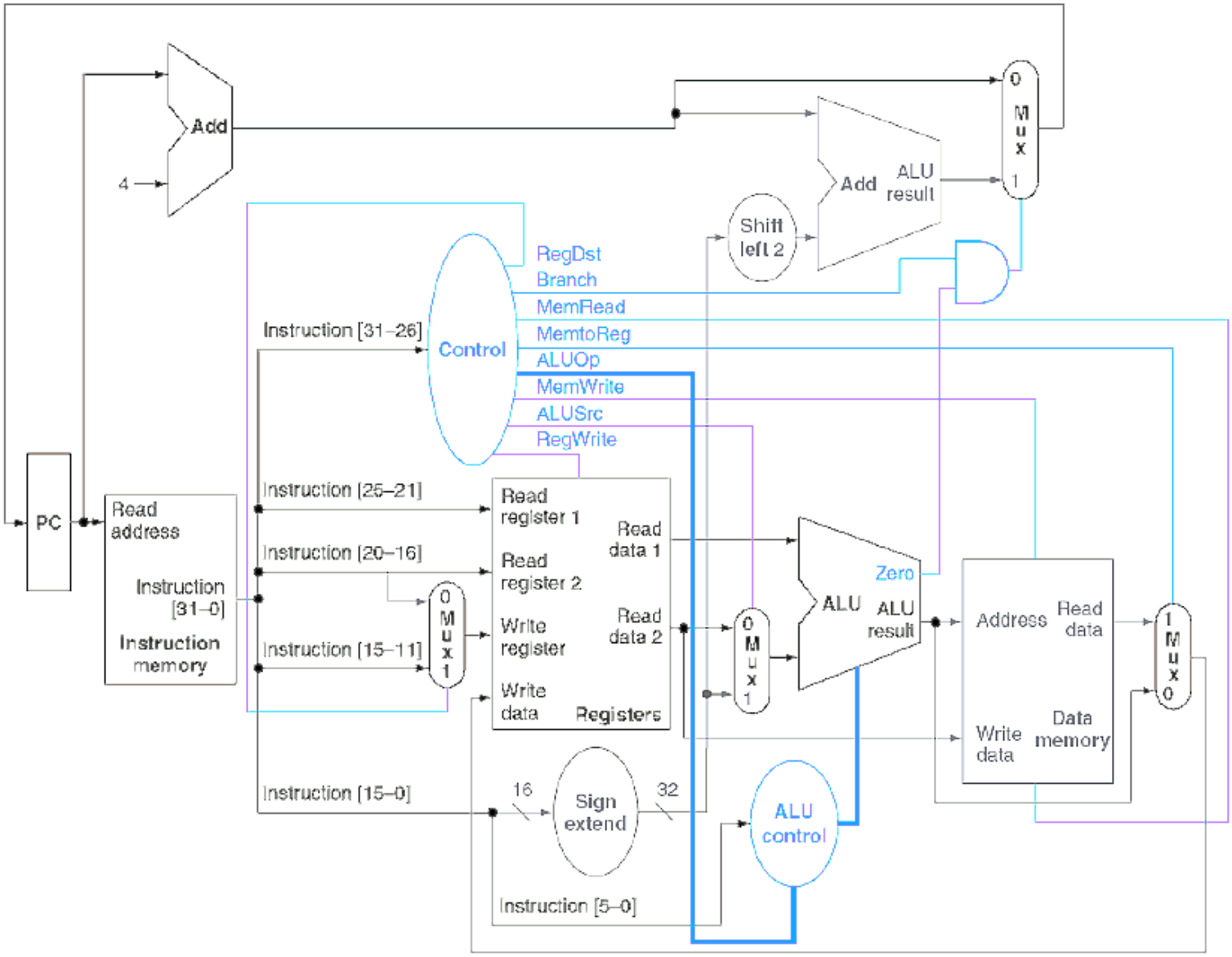
OpUAL	Operația	Câmpul funcțiunii	Acțiunea UAL	Intrare de control UAL
00	Încărcare cuv	xxxxxx	adunare	010
00	Memorare cuv	xxxxxx	adunare	010
01	Ramificație la egal	xxxxxx	scădere	110 etc

Schema controlului UAL completă

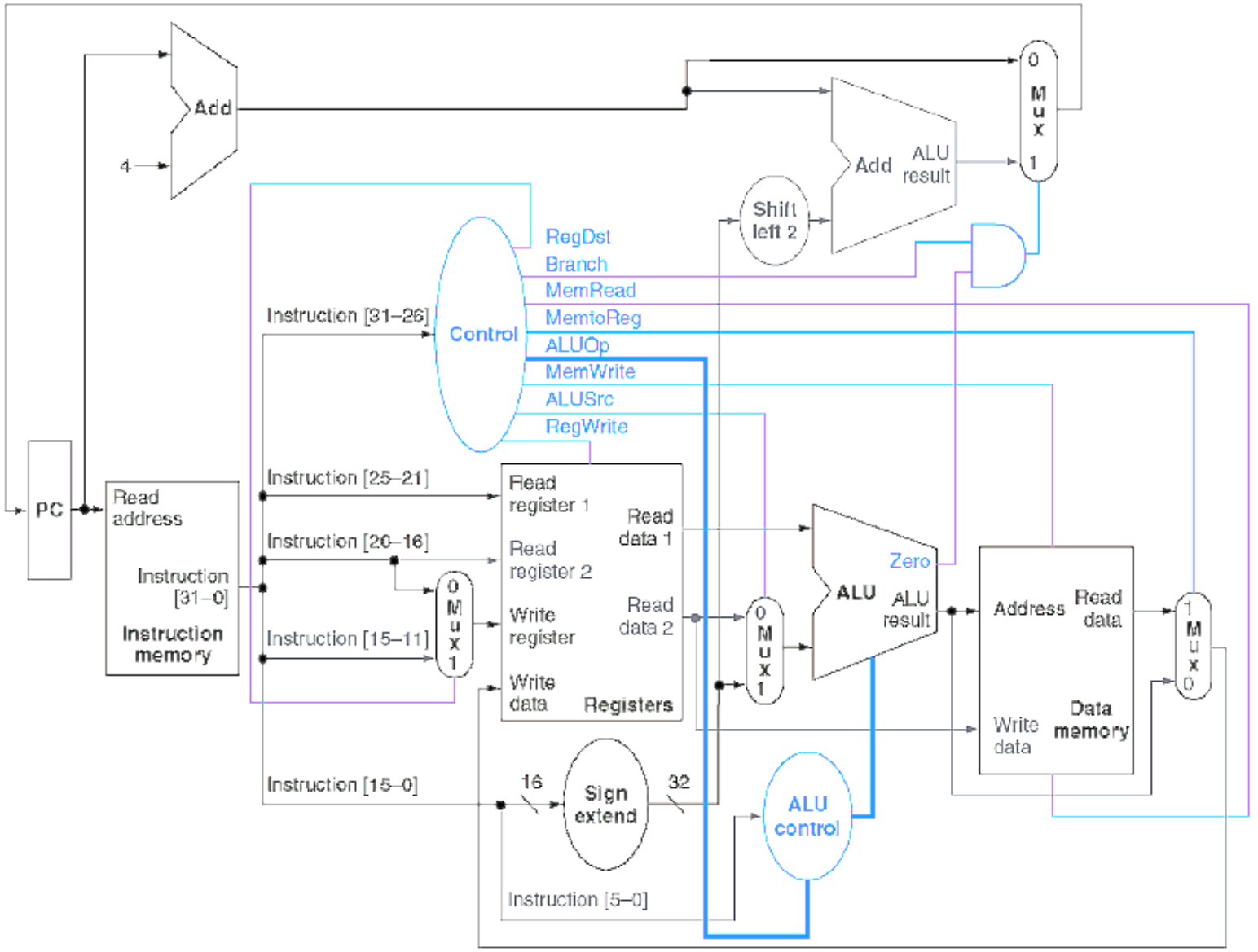


0 (31-26)	rs (25-21)	rt (20-16)	rd (15-11)	shamt(10-6)	funct(5-0)
35 sau 43 (31-26)	rs (25-21)	rt (20-16)	adresa(15-0)		
4 (31-26)	rs (25-21)	rt (20-16)	adresa(15-0)		

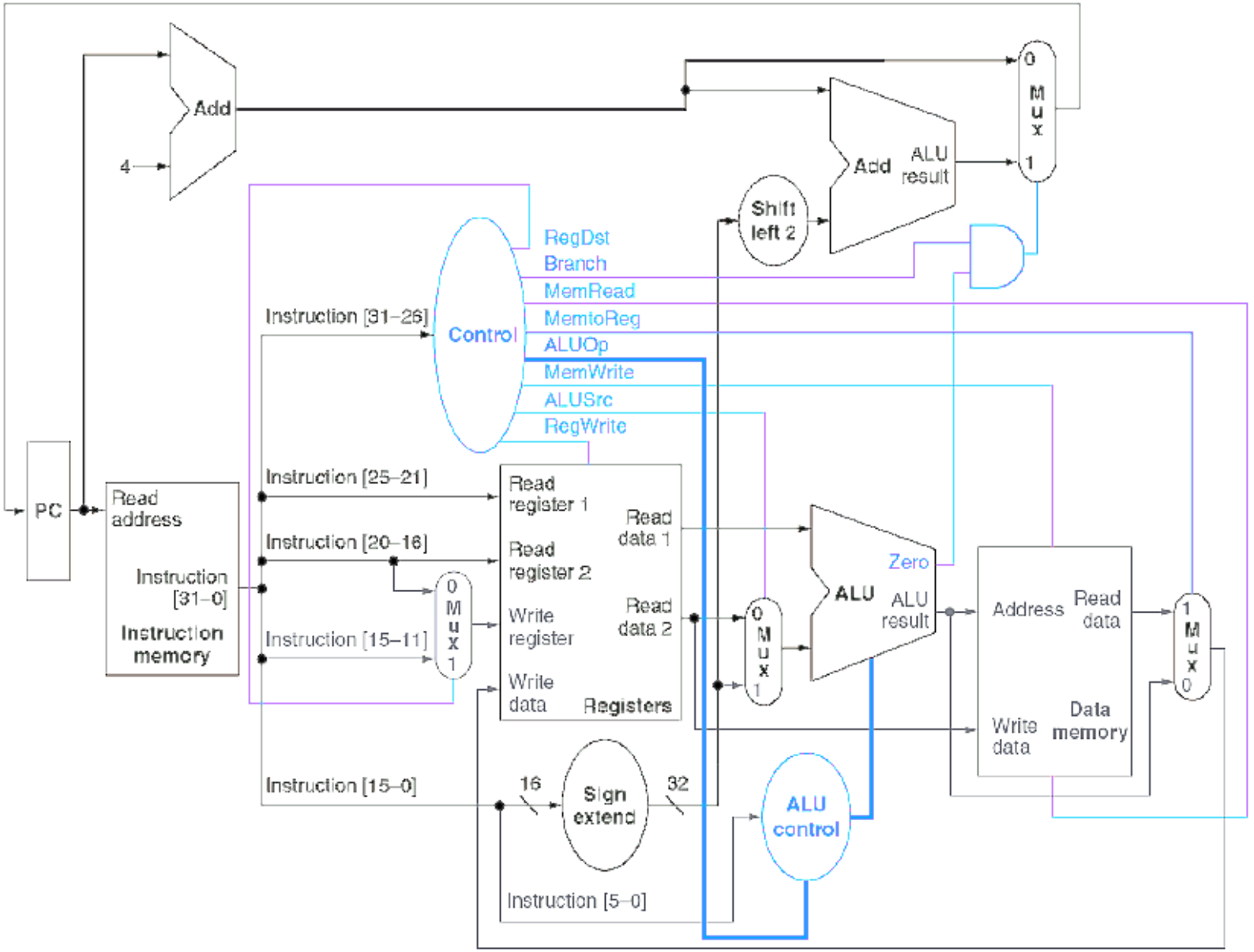
Execuția instrucțiunii de tip R



Execuția instrucțiunii de încărcare



Execuția instrucțiunii de ramificare la egal

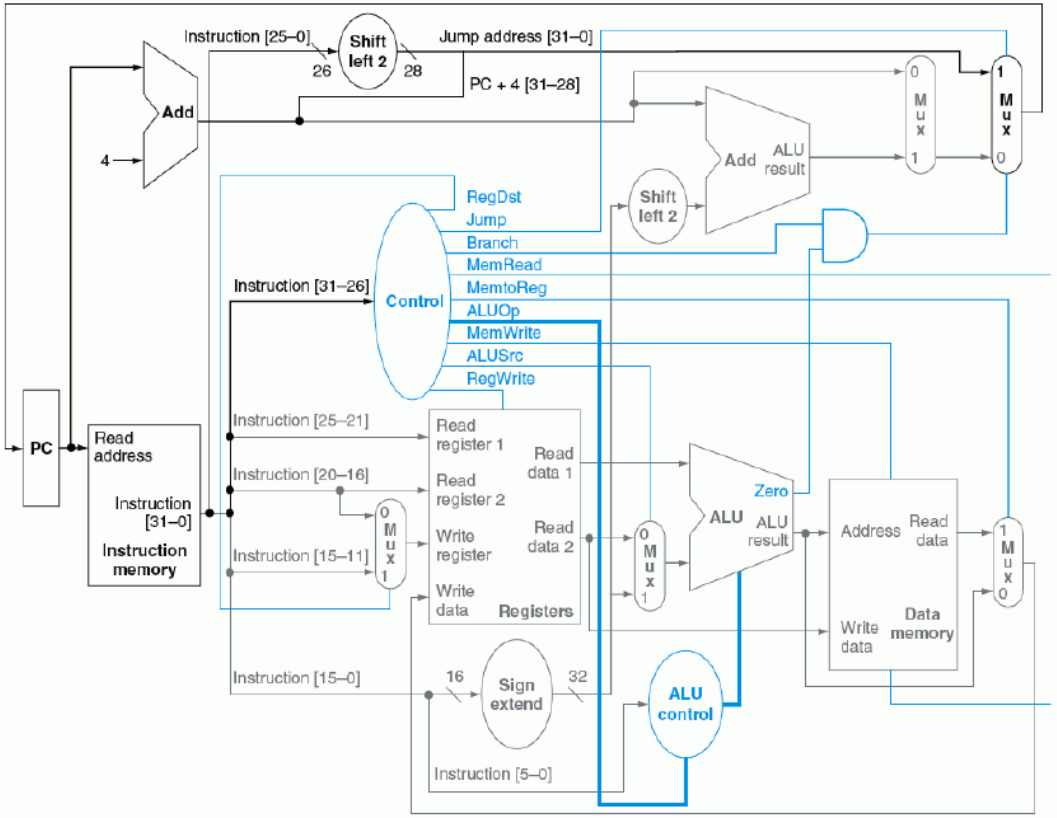


Implementarea instrucțiunii de salt

2 (31-26) adresa (25-0)

Calcularea PC-ului obiectiv

- ✓ cei 4 biți MSB vin de la PC+4;
- ✓ câmpul imediat de 26 biți ai instrucțiunii de salt;
- ✓ cei 2 biți inferiori ai unei adrese de salt sunt întotdeauna 00



Folosim sau nu implementarea cu o singură perioadă de ceas ?

În cazul proiectării cu un singur ciclu de ceas, perioada ceasului trebuie să aibă aceeași durată pentru fiecare instrucțiune => ciclul de ceas este determinat de cea mai lungă cale posibilă prin mașină

Cazul cel mai frecvent nu are o execuție rapidă

Fiecare unitate funcțională poate fi utilizată o singură dată într-un ciclu de ceas => repetarea un ităților funcționale => creșterea prețului implementării

Metode alternative :

implementarea cu mai multe cicluri

PIPELINE

Metodă INEFICIENTĂ, neutilizată în practică