

Proiectarea procesorului MIPS care operează într-un singur ciclu de ceas.

– Curs 8_2 –

Subiecte abordate:

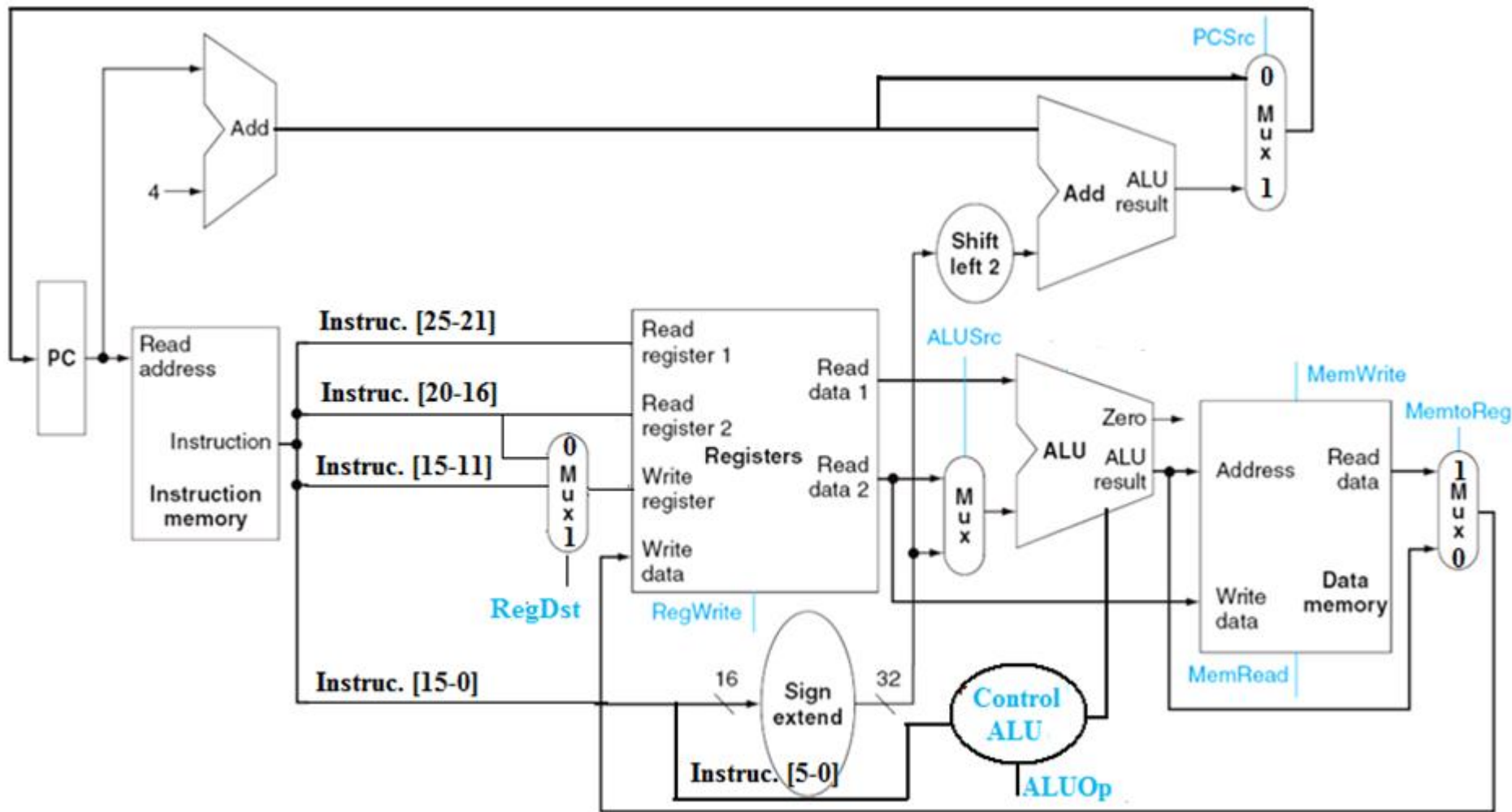
- Unitatea de control pentru procesorul MIPS care operează într-un singur ciclu de ceas

Proiectarea unității principale de control

Instrucțiunea de tip R	0	rs	rt	rd	shamt	funct
	31-26	25-21	20-16	15-11	10-6	5-0
Instrucțiunea de încărcare sau memorare	35 sau 43	rs	rt	adresa		
	31-26	25-21	20-16	15-0		
Instrucțiunea de ramificație	4	rs	rt	adresa		
	31-26	25-21	20-16	15-0		

Obs: Codul operației pentru instrucțiunea de încărcare este 35, iar pentru memorare 43.

Biții 31-26 reprezintă câmpul operației.



Calea de date și liniile de control

UAL are trei intrări de control din care sunt folosite 5:

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

În funcție de clasa instrucțiunii, UAL va trebui să efectueze una din aceste cinci funcții:

- Pentru instrucțiunile de încărcare și memorare a unui cuvânt, UAL este folosit să calculeze prin adunare adresa la memorie
- Pentru ramificație în caz de egalitate, UAL efectuează o scădere.

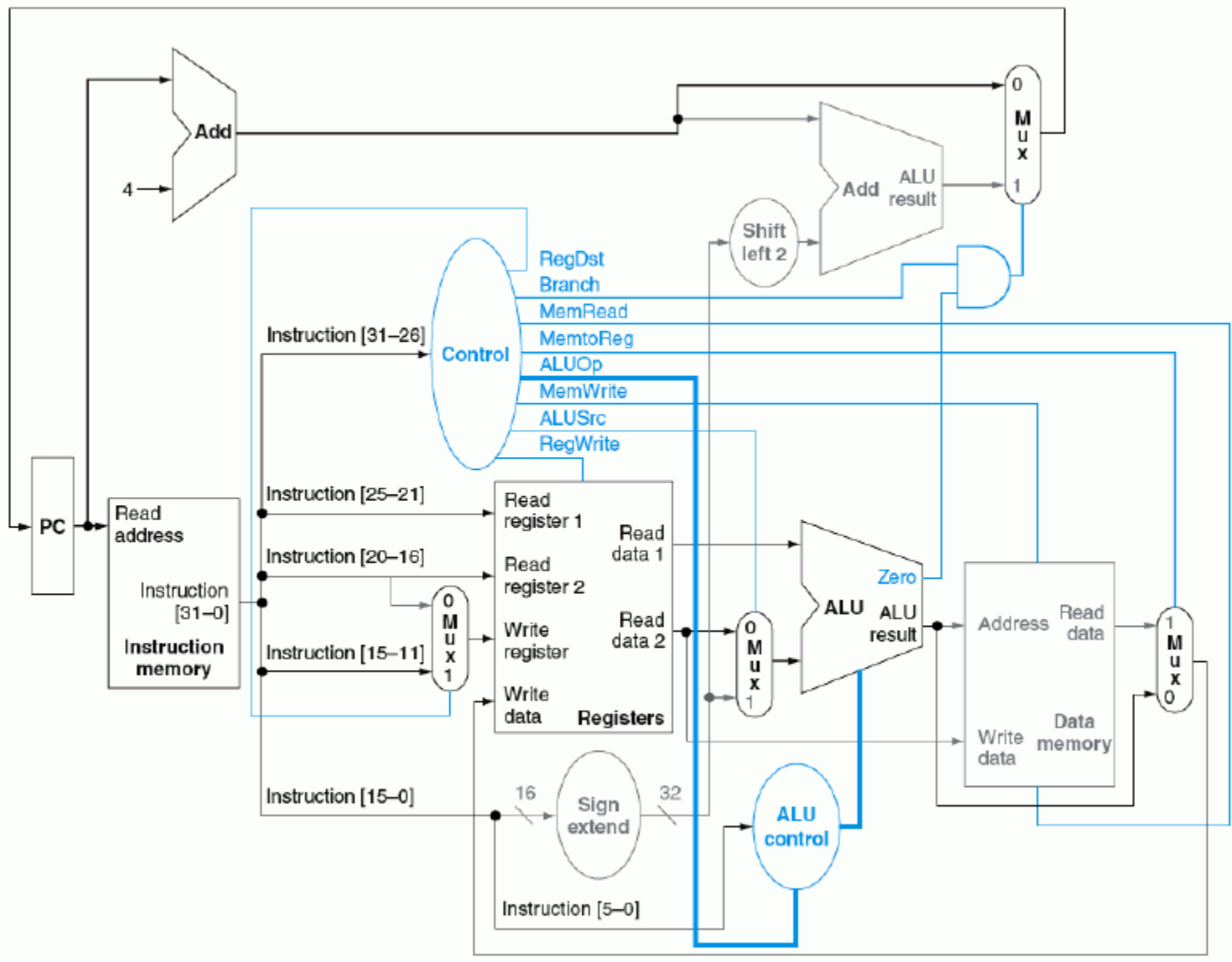
Modalitatea de implementare → se utilizează 2 biți (OpUAL) și cei 6 biți ai codului funcțiunii

Codul operației instrucțiunii	OpUAL	Operația instrucțiunii	Câmpul funcțiunii	Acțiunea UAL dorită	Intrarea de control UAL
LW	00	Încărcare cuvânt	xxxxxx	Adunare	010
SW	00	Memorare cuvânt	xxxxxx	Adunare	010
Ramificație la egal	01	Ramificație la egal	xxxxxx	Scădere	110
Tip R	10	Adunare	100000	Adunare	010
Tip R	10	Scădere	100010	Scădere	110
Tip R	10	ȘI	100100	Și	000
Tip R	10	SAU	100101	Sau	001
Tip R	10	Setare la <	101010	Setare la <	111

Odată determinat acest tabel, el poate fi transformat în hardware.

Efectul activării semnalelor de control:

RegDst	Numărul de destinație al registrului <i>de scris</i> vine din rd (biții 15-11) (altfel vine din rt)
RegWrite	Registrul de la intrarea registrului <i>de scris</i> este scris cu valoarea de la intrarea dată de <i>de scris</i> (<i>Write Register</i>).
ALUScr	Al doilea operand al UAL vine din ieșirea a doua a fișierului de registre (altfel sunt cei 16 biți inferiori ai instr., cu semn extins).
PCScr	PC-ul este înlocuit cu ieșirea sumatorului care calculează obiectivul pentru ramificație (altfel calculează valoarea PC+4)
MemRead	Conținutul memoriei de date desemnate de adresa de intrare este direcționat spre ieșirea <i>Read Data</i>
MemWrite	Conținutul memoriei de date desemnate de adresa de intrare este înlocuit cu valoarea de la intrarea <i>Write Data</i>
MemtoReg	Valoarea furnizată intrării <i>Read Data</i> în registru vine de la memoria de date (altfel vine de la ALU).

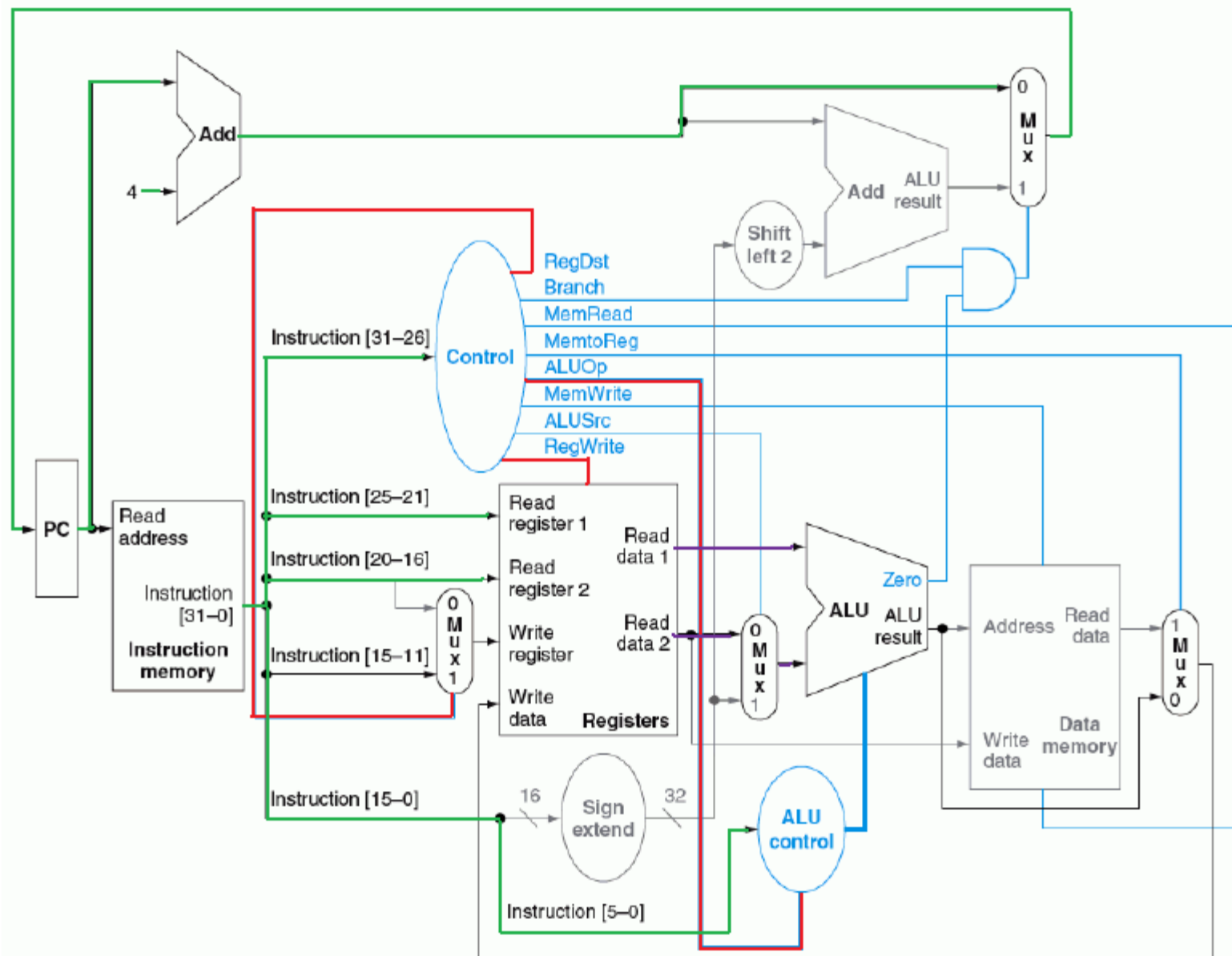


Calea de control, cu unitatea de control

Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Stabilirea semnalelor pe liniile de control este complet determinată de câmpul codului de operație al instrucțiunii.

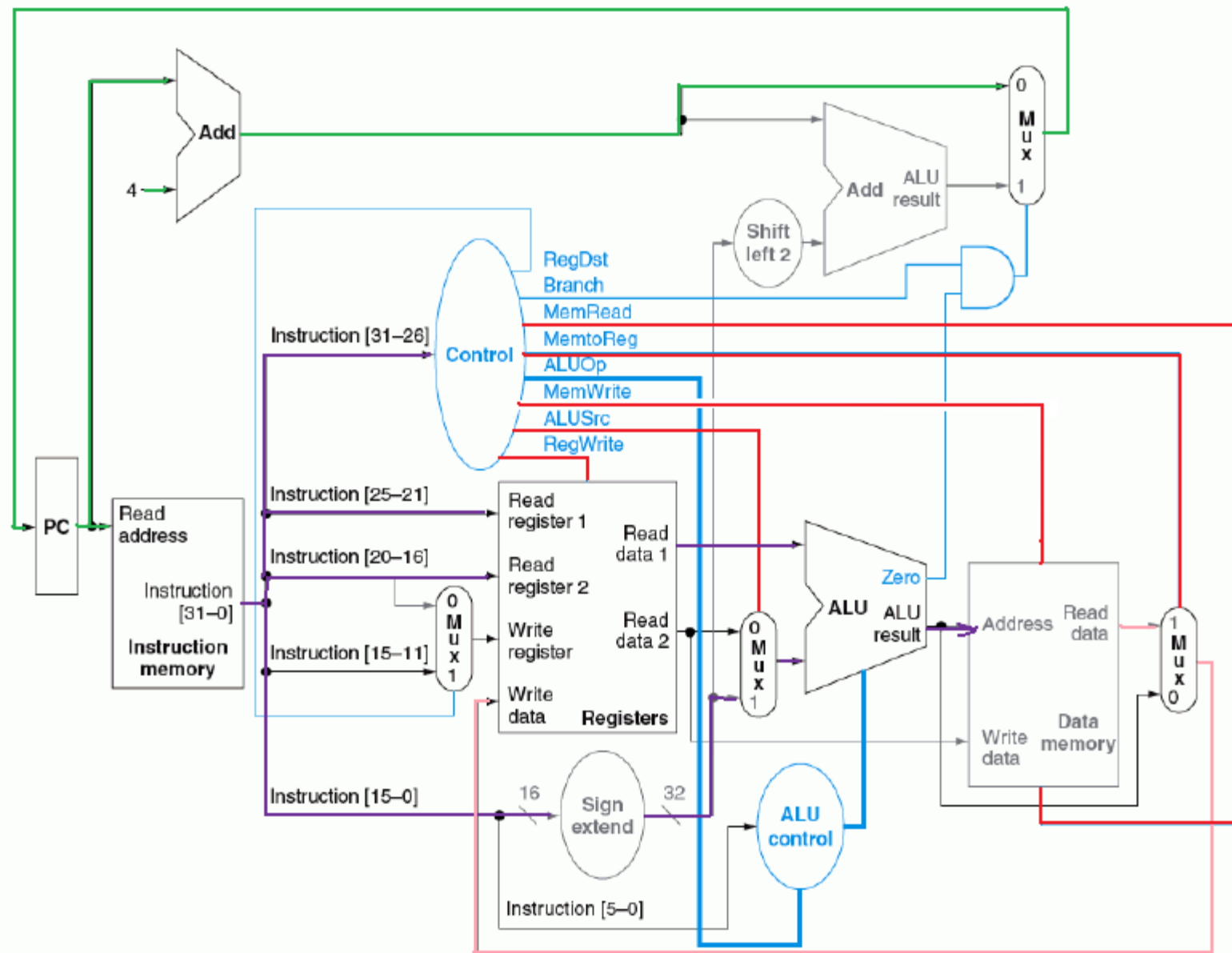
- O instrucțiune de tip R scrie un registru dar niciodată nu citește sau scrie memoria de date.
- câmpurile ALUSrc și UALOp sunt stabilite pentru efectuarea calculului adresei.



Execuția instrucțiunii de tip R

Execuția instrucțiunii de tip R

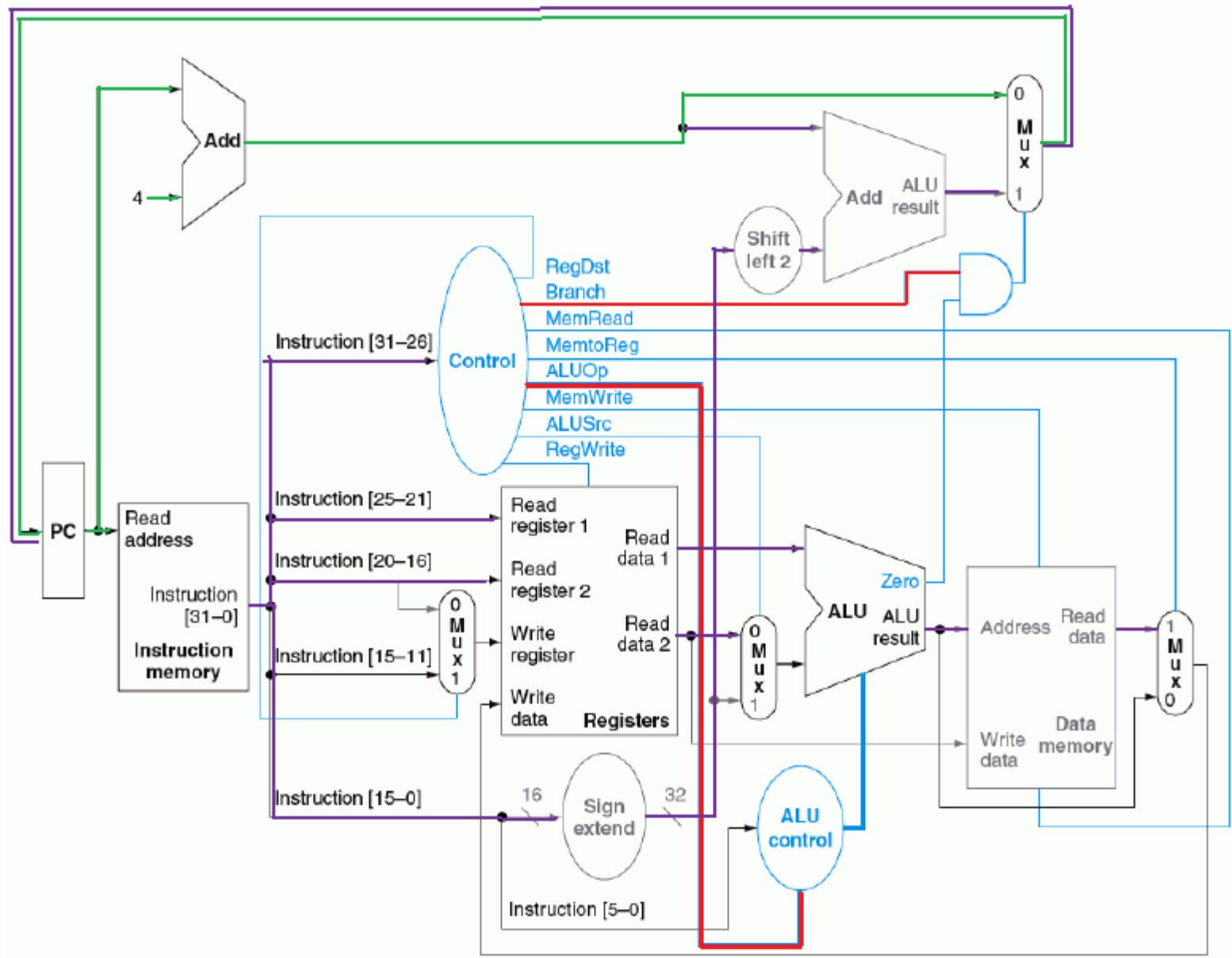
1. O instrucțiune este extrasă din memoria de instrucțiuni iar PC-ul este incrementat (linie verde).
2. Două registre \$t2 și \$t3 sunt citite din fișierul de registre (linie mov). Unitatea principală de control calculează și pe durata acestui pas activarea impulsurilor pe liniile de control.
3. UAL operează cu datele citite din fișierul de registre, utilizând codul funcțiunii pentru generarea funcțiunii UAL.
4. Rezultatul din UAL este scris în fișierul de registre, utilizând biții 15-11 ai instrucțiunii pentru selectarea registrului destinație \$t1.
5. Această implementare este combinațională . Cursul informației urmează această succesiune => stabilizarea semnalelor se va face în această ordine.



Execuția instrucțiunii de încărcare

Execuția instrucțiunii de încărcare

1. O instrucțiune este extrasă din memorie de instrucțiuni, iar PC-ul este incrementat (linie verde).
2. Un registru \$t2 este citit din fișierul de registre (linie mov).
3. UAL calculează suma dintre valoarea citită din fișierul de registre și ultimii 16 biți cu semnul extins ai instrucțiunii (linie mov).
4. Suma din UAL este folosită ca adresă pentru memoria de date (linie mov).
5. Data din unitatea de memorie este scrisă în fișierul de registre; registrul destinație este dat de biții 20-16 ai instrucțiunii (\$t1) (linie roz).

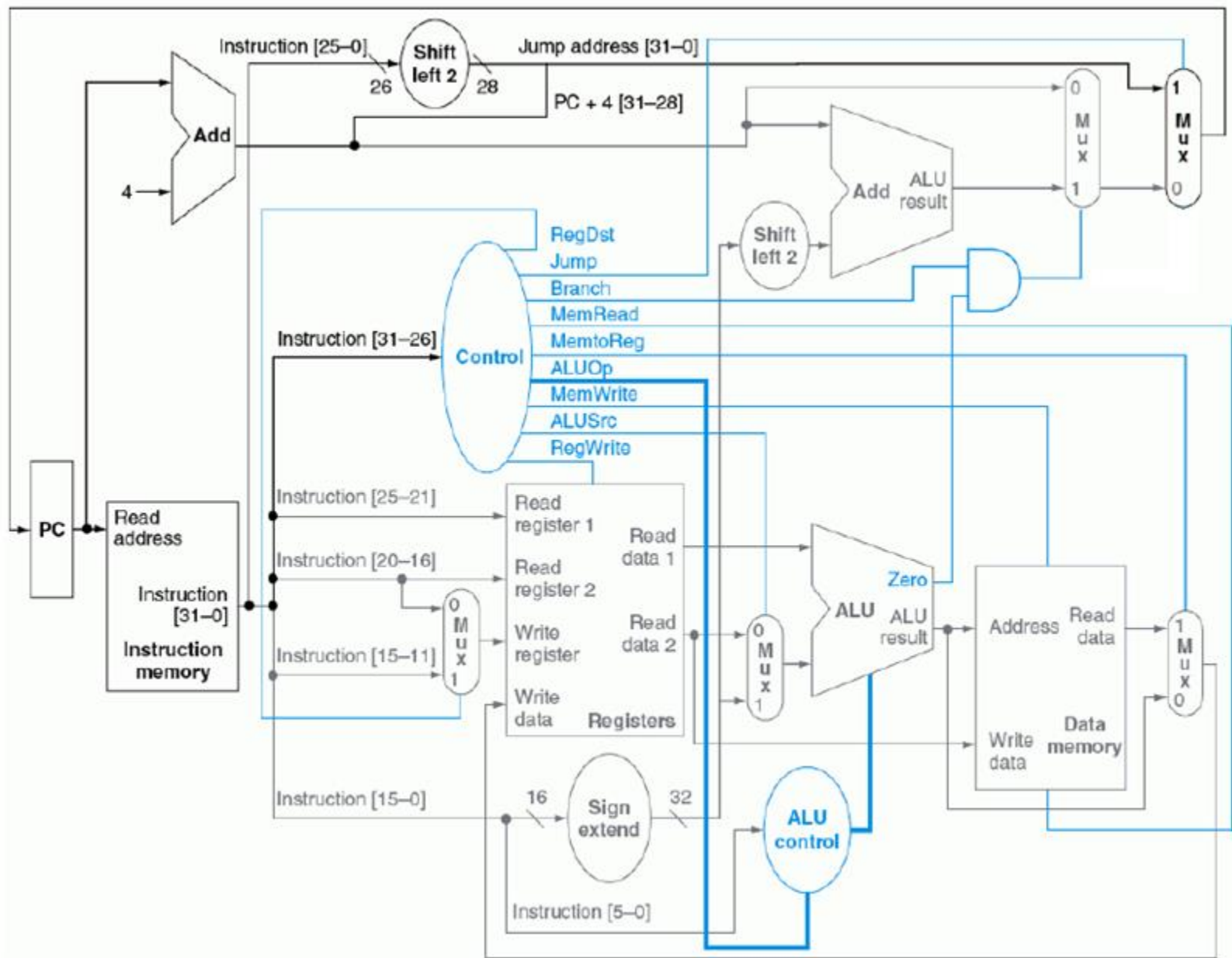


Execuția instrucțiunii de ramificare la egal

Execuția instrucțiunii de ramificare la egal

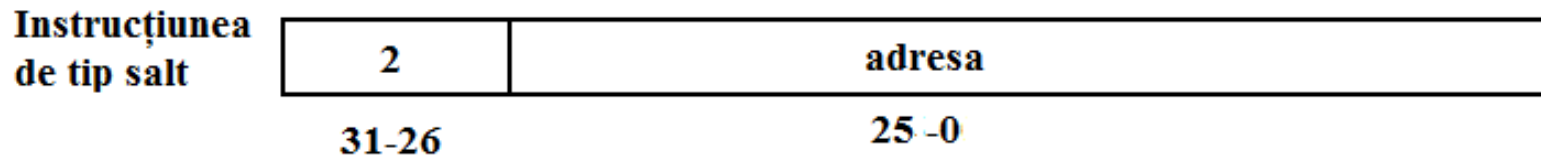
Este asemănătoare cu instrucțiunea R doar că ieșirea UAL-ului este folosită pentru a se stabili dacă în PC se scrie PC+4 sau adresa obiectiv pentru ramificație.

1. O instrucțiune este extrasă din memoria de instrucțiuni, iar PC-ul este incrementat (linie verde).
2. Două registre \$t1 și \$t2 sunt citite din fișierul de registre.
3. UAL-ul efectuează o scădere asupra valorii de date citite din fișierul de registre. Valoarea PC+4 este adunată cu cei mai puțin semnificativi 16 biți, cu semnul extins al instrucțiunii, deplasați cu 2 poziții spre stânga. Rezultatul este adresa obiectiv pentru ramificație (linie mov).
4. Rezultatul ZERO din UAL este folosit pentru a decide care dintre rezultatele sumatoarelor va fi păstrat în PC.



Implementarea instrucțiunii de salt

Codul operației este 2.



Calcularea PC-ului obiectiv:

- ✓ cei mai semnificativi 4 biți ai PC-ului curent +4;
- ✓ câmpul imediat de adresă 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 ?

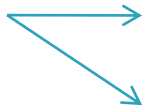
În cazul proiectării cu un singur ciclu de ceas, perioada ceasului trebuie să aibă aceeași durată pentru fiecare instrucțiune .



Un ciclu 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 unităților funcționale => creșterea prețului implementării.

Metode alternative :  implementarea cu mai multe cicluri
PIPELINE

- Metodă ineficientă în multe cazuri, în special ineficiența maximă se dovedește în cazul proiectării unei unități în virgulă mobilă.