Pipeline - Banda de asamblare

– Curs 11_2–

Calea de date în versiunea pipeline și controlul ei

Stagiile de execuție ale unei instrucțiuni

1. IF – citirea următoarei instrucțiuni oferite de PC

2. ID – decodificarea instrucțiunii

3. EX – execuția instrucțiunii

4. MEM – memorarea rezultatului și incrementarea lui PC

5. WB = scriere în registre

Observație – în cazul instrucțiunilor LOAD sau STORE se modifică pasul 3. Pentru BRANCH - în pasul 3 se va seta PC să pointeze către următoarea instrucțiune și pasul 4 nu mai există

- ✓ Toate stagiile unui pipeline trebuie să fie ***echilibrate*** – toate stagiile trebuie să dureze aproximativ același timp.
- ✓ Un task trebuie să folosească toate stagiile și ordinea să fie aceeași pentru toate task-urile.
- ✓ Între stagii trebuie să existe registre deoarece nu toate stagiile au aceeași durată.

Fiecare stagiu trebuie să posede resurse hardware care sunt necesare execuției task-ului.

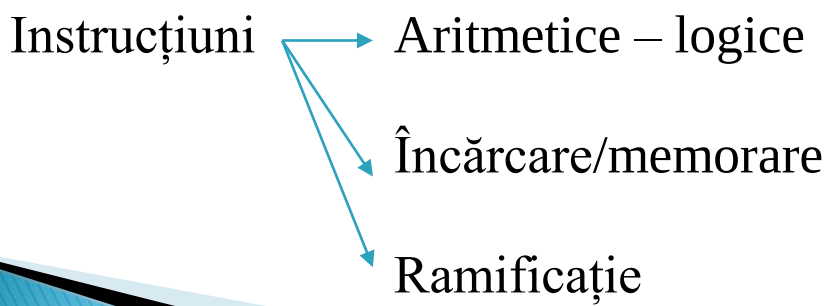
Pipeline-ul poate fi întrerupt de anumite evenimente interne sau de evenimente externe – excepții / întreruperi.

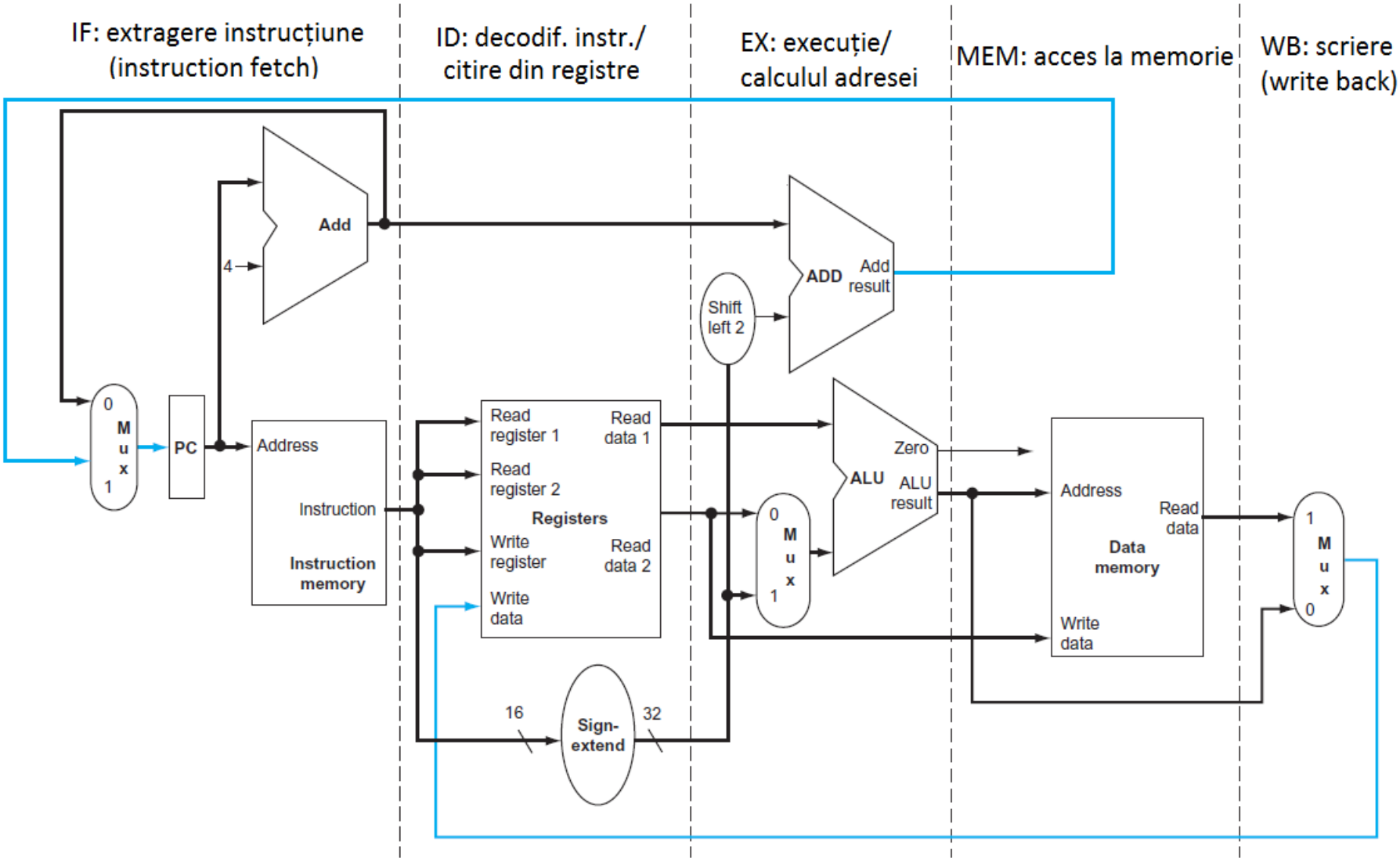
Pipeline-ul trebuie golit și starea procesului salvată pentru a putea fi reluată.

MIPS posedă: o colecție de registre generale, PC, o memorie cache de instrucțiuni și una de date

Fiecare registru general dar și PC-ul are lungimea de 32 de biți – 4 bytes – 1 cuvânt

Procesul MIPS prezintă o arhitectură de tip load-store, toate instrucțiunile au registre sau operanzi imediați.




IF: extragere instrucțiune (instruction fetch)
ID: decodif. instr./ citire din registre
EX: execuție/ calculul adresei
MEM: acces la memorie
WB: scriere (write back)
Add
4
0
M
u
x
1
PC
Address
Instruction
Instruction memory
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
ADD
Add result
0
M
u
x
1
ALU
Zero
ALU result
Address
Data memory
Read data
Write data
1
M
u
x
0
16
Sign-extend
32

- IF – instrucțiunea este citită din memorie de la adresa indicată de PC. Se presupune că instrucțiunea nu este o instrucțiune de tip branch și de aceea se incrementează PC către următoarea instrucțiune.

- ID – Instrucțiunea este decodificată și tipul ei este identificat. Alte task-uri precum extensia la 32 de biți sunt executate.

- EX – în cazul unei instrucțiuni aritmetico-logice, ALU realizează operația aritmetică sau logică. În caz de LOAD sau STORE adresa addr = Rs + depl se determină (depl este mărit la 32 de biți în stagiul ID) – în caz de branch, PC-ul va indica valoarea corectă pentru următoarea instrucțiune

- MEM – În cazul LOAD, conținutul ***Mem[addr]*** este citită (din memoria cache). Dacă instrucțiunea este STORE, conținutul acelei locații este modificat. Dacă nu avem nici una dintre aceste două instrucțiuni, atunci în acest stagiu nu se întâmplă nimic.

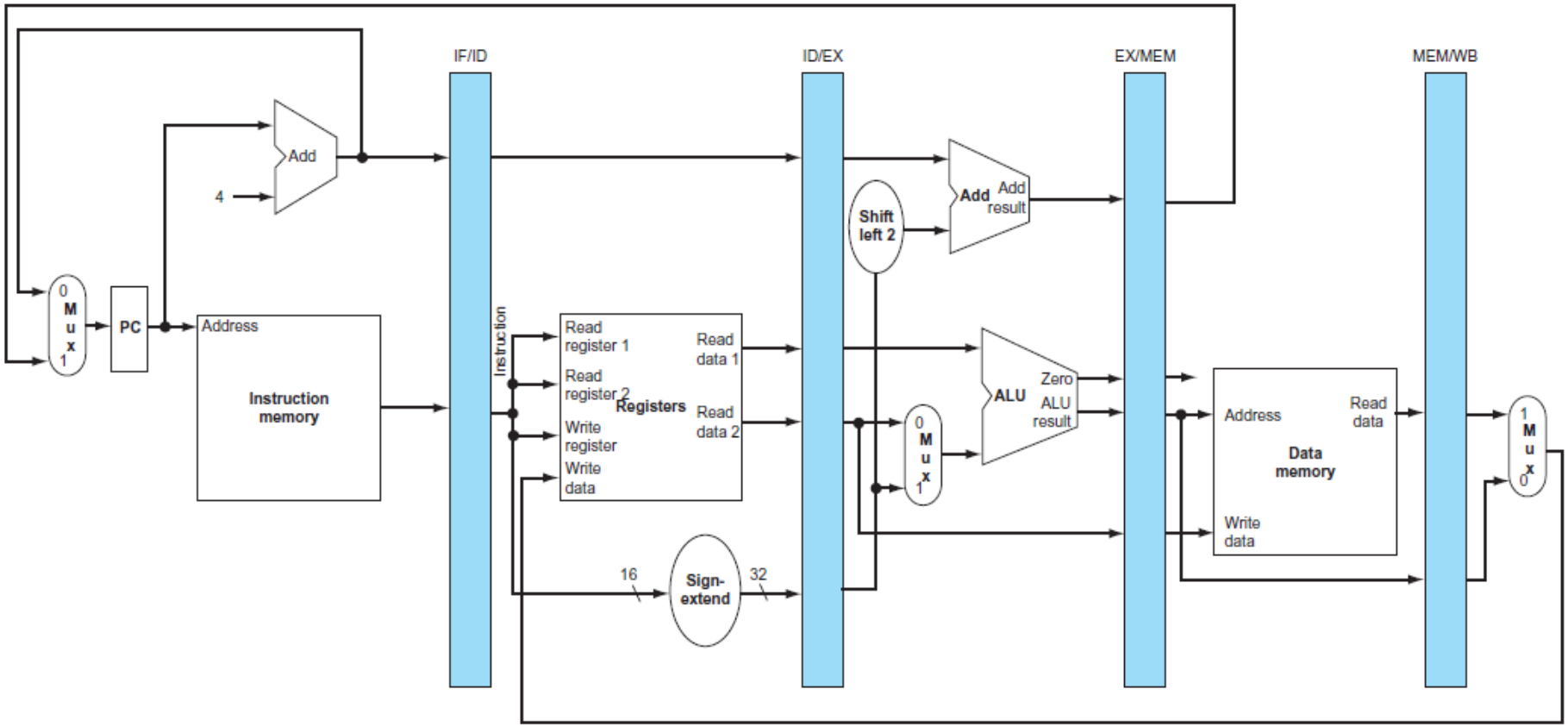
- WB – Dacă instrucțiunea este LOAD sau aritmetică (nu STORE) – rezultatul este memorat în registrul rezultat sau destinație.

Excepții

- ✓ Stagiul write-back – plasează rezultatele înapoi în fișierele de registre la mijlocul căii de date.

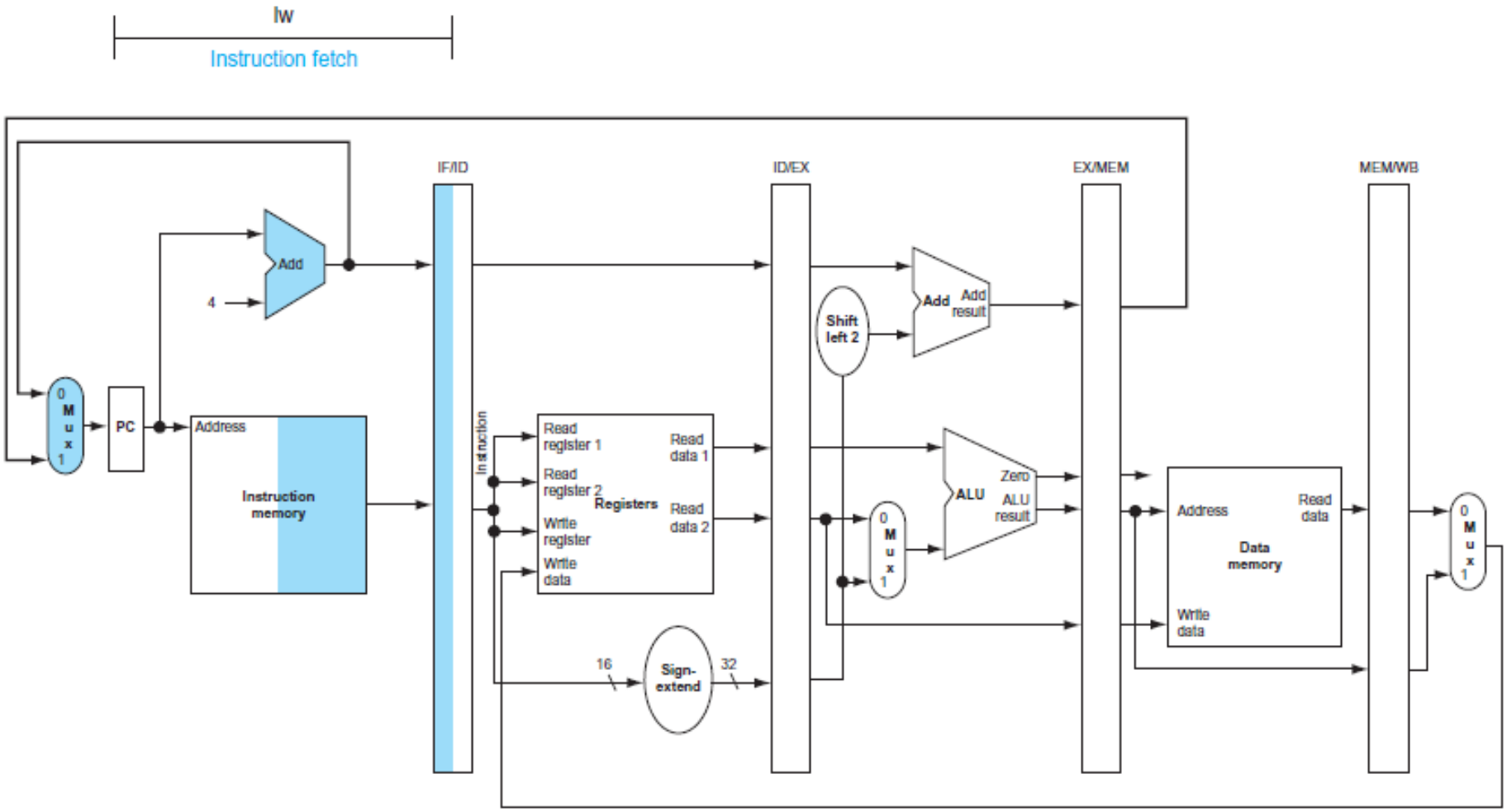
- ✓ Selecția următoarei valori din PC, care alege între incrementarea lui PC și adresa de ramificație din stagiul MEM.

Versiunea pipe a căii de date

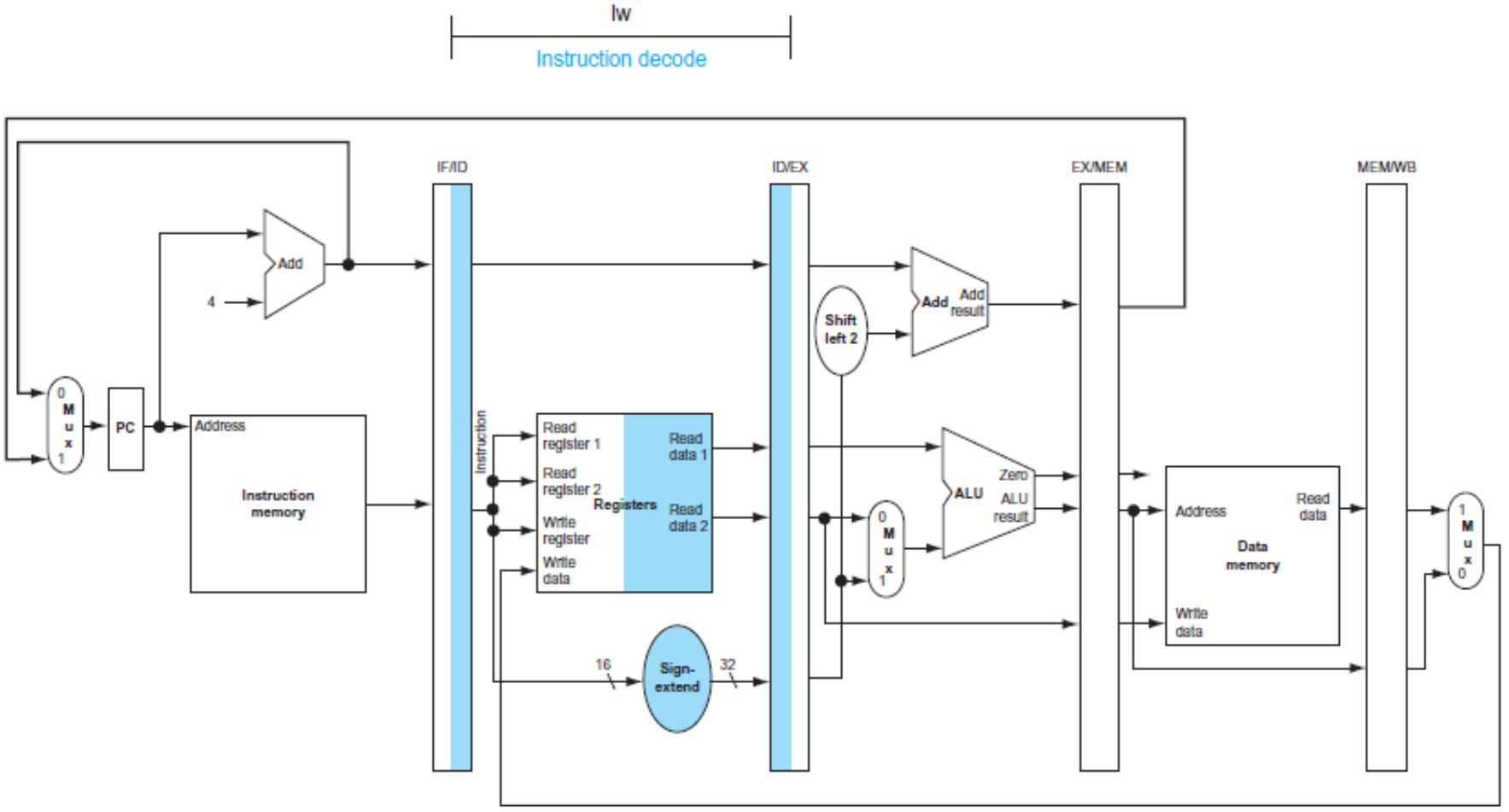


Registrele sunt suficient de mari pentru a permite stocarea datelor transferate de la un stagiu la celălalt: IF/ID = 64 biți, ID/EX = 128 (=32 *4), EX/MEM = 97 (= 32*3 +1), MEM/WB = 64 (=32*2) biți.

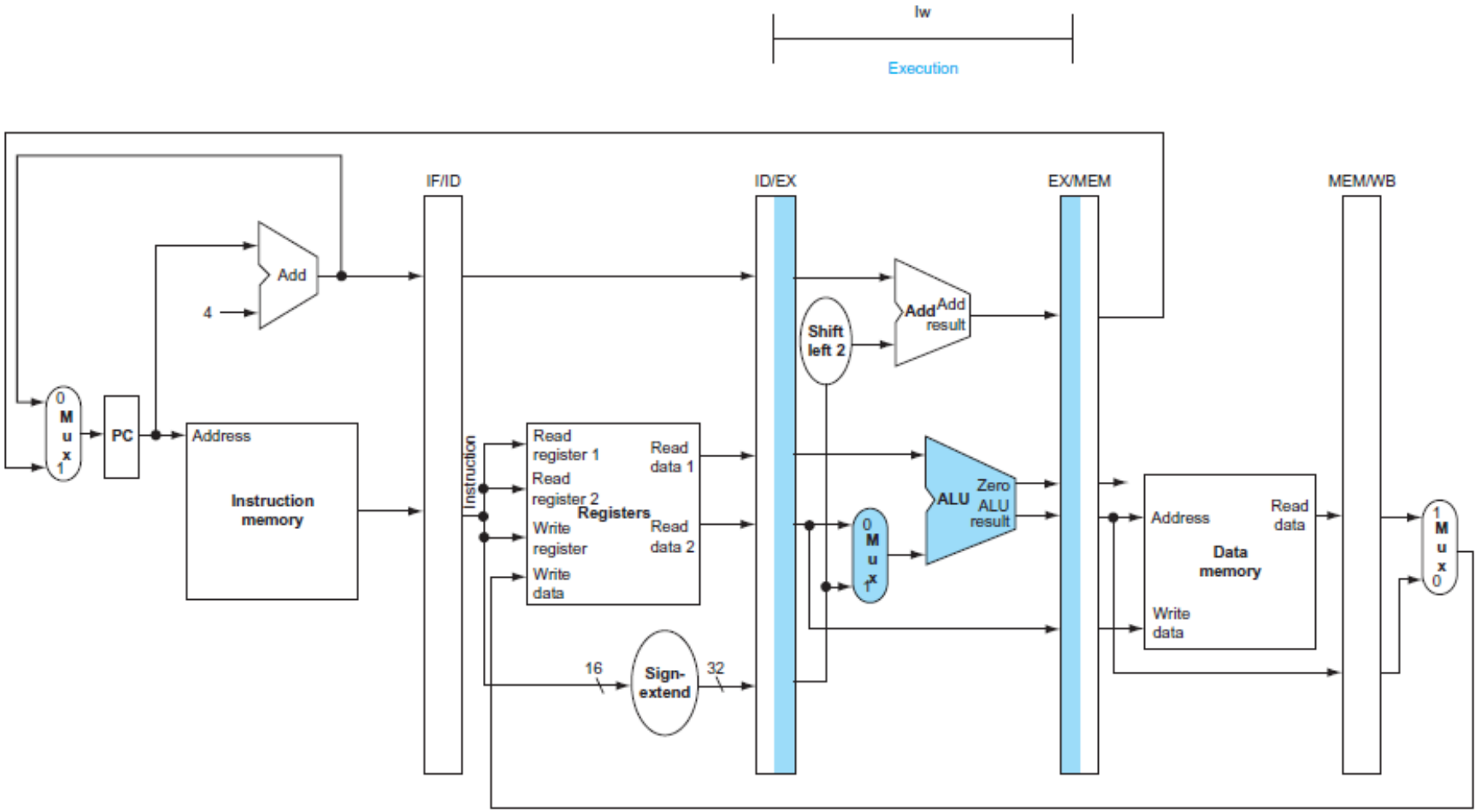
Execuția instrucțiunii lw



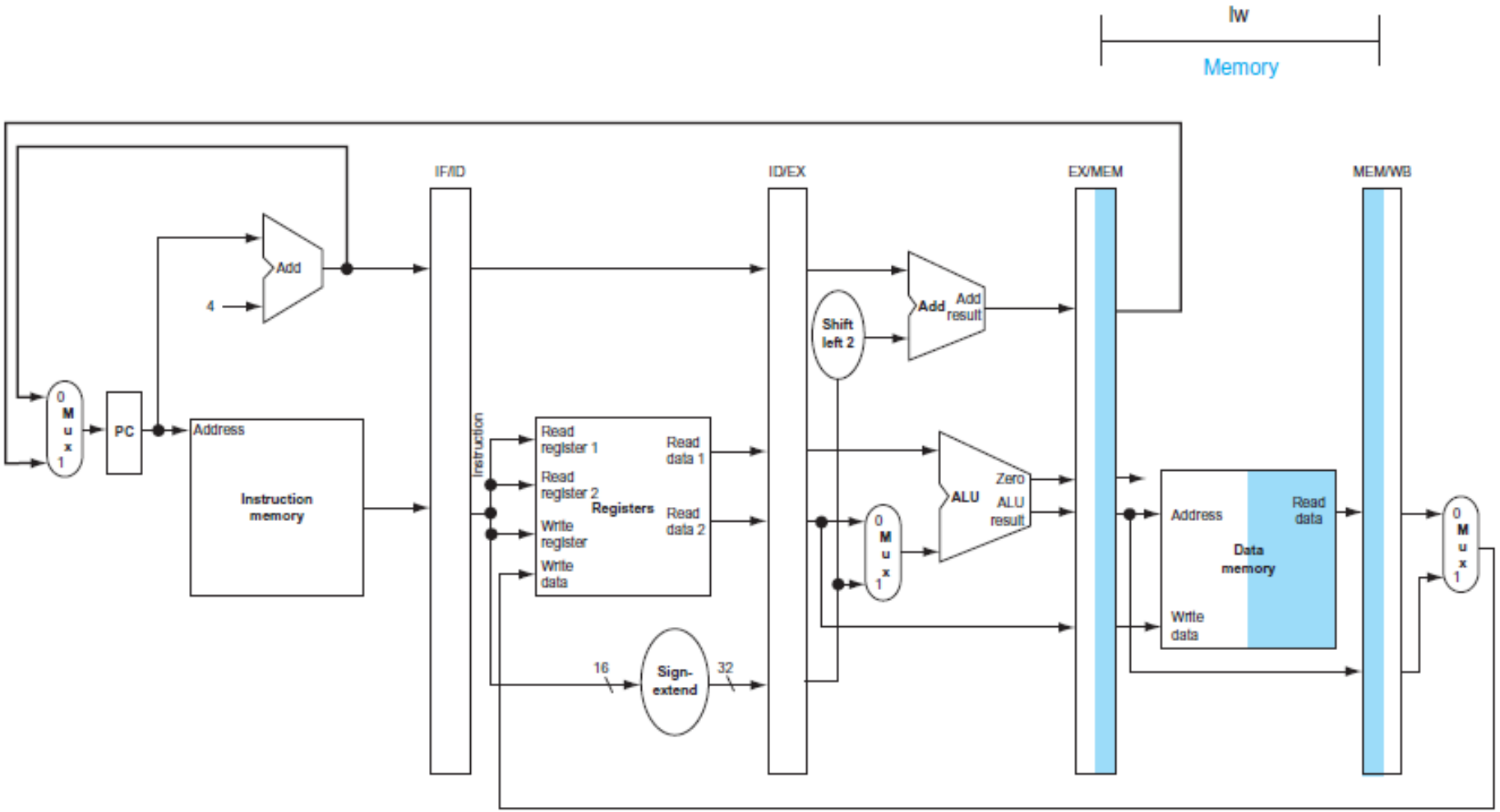
Execuția instrucțiunii lw



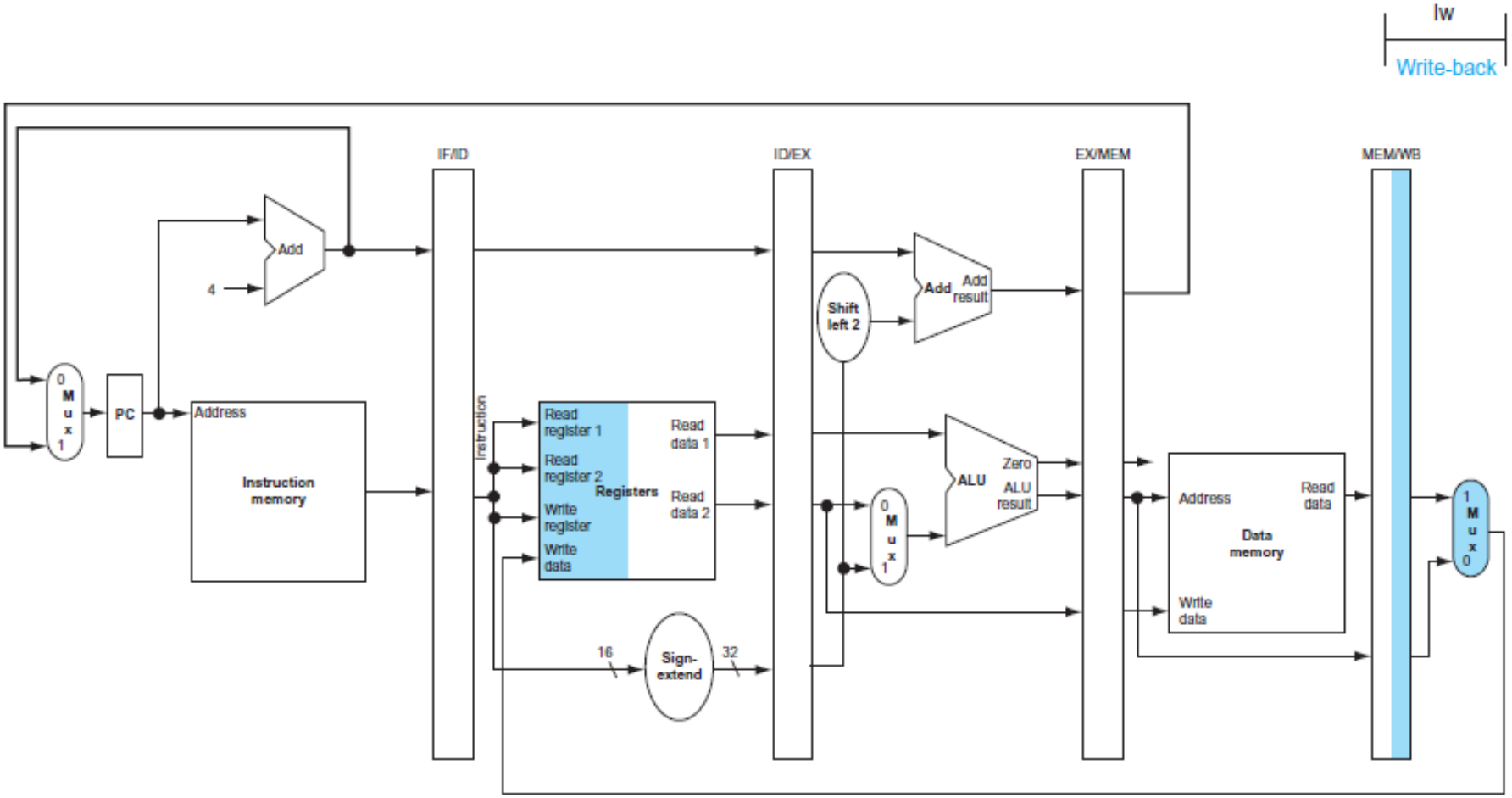
Execuția instrucțiunii lw



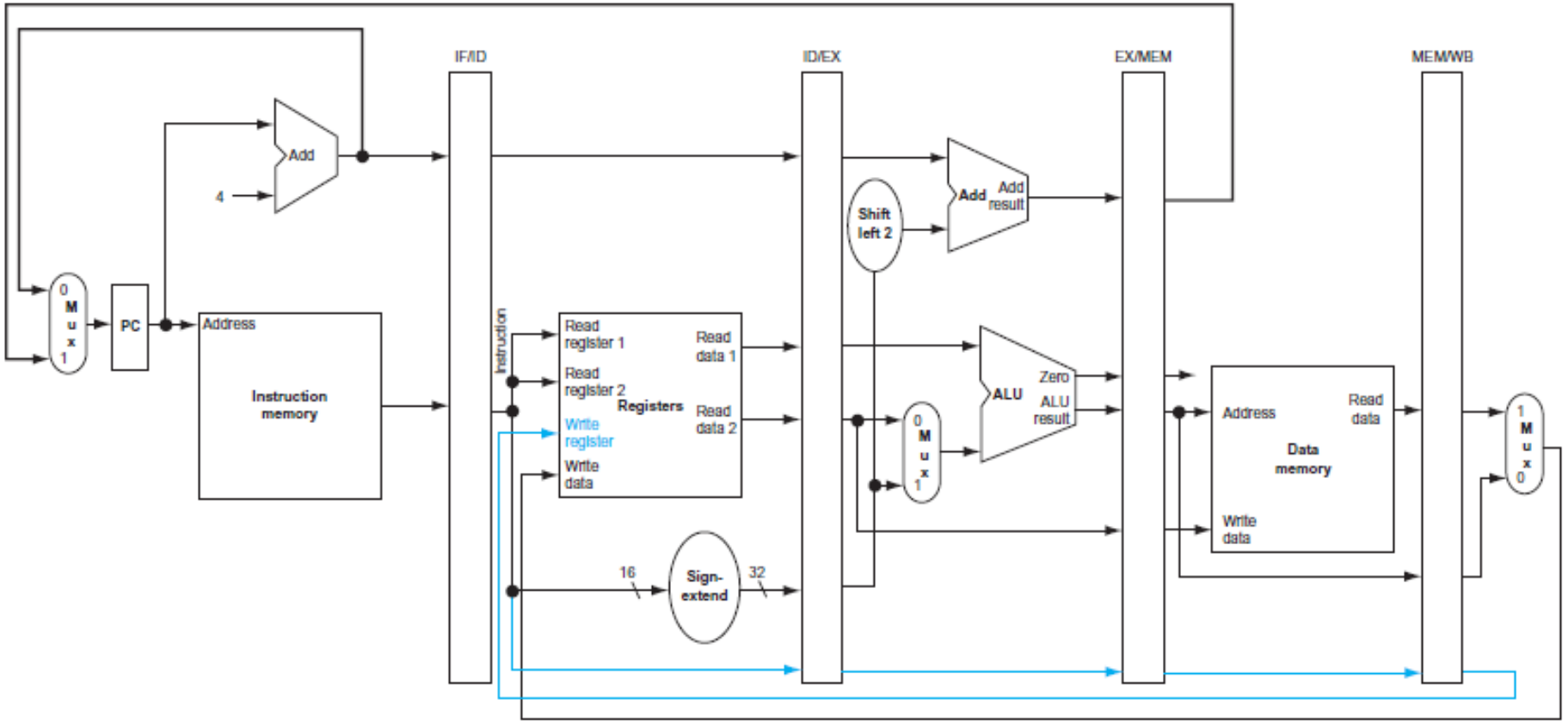
Execuția instrucțiunii lw



Execuția instrucțiunii lw

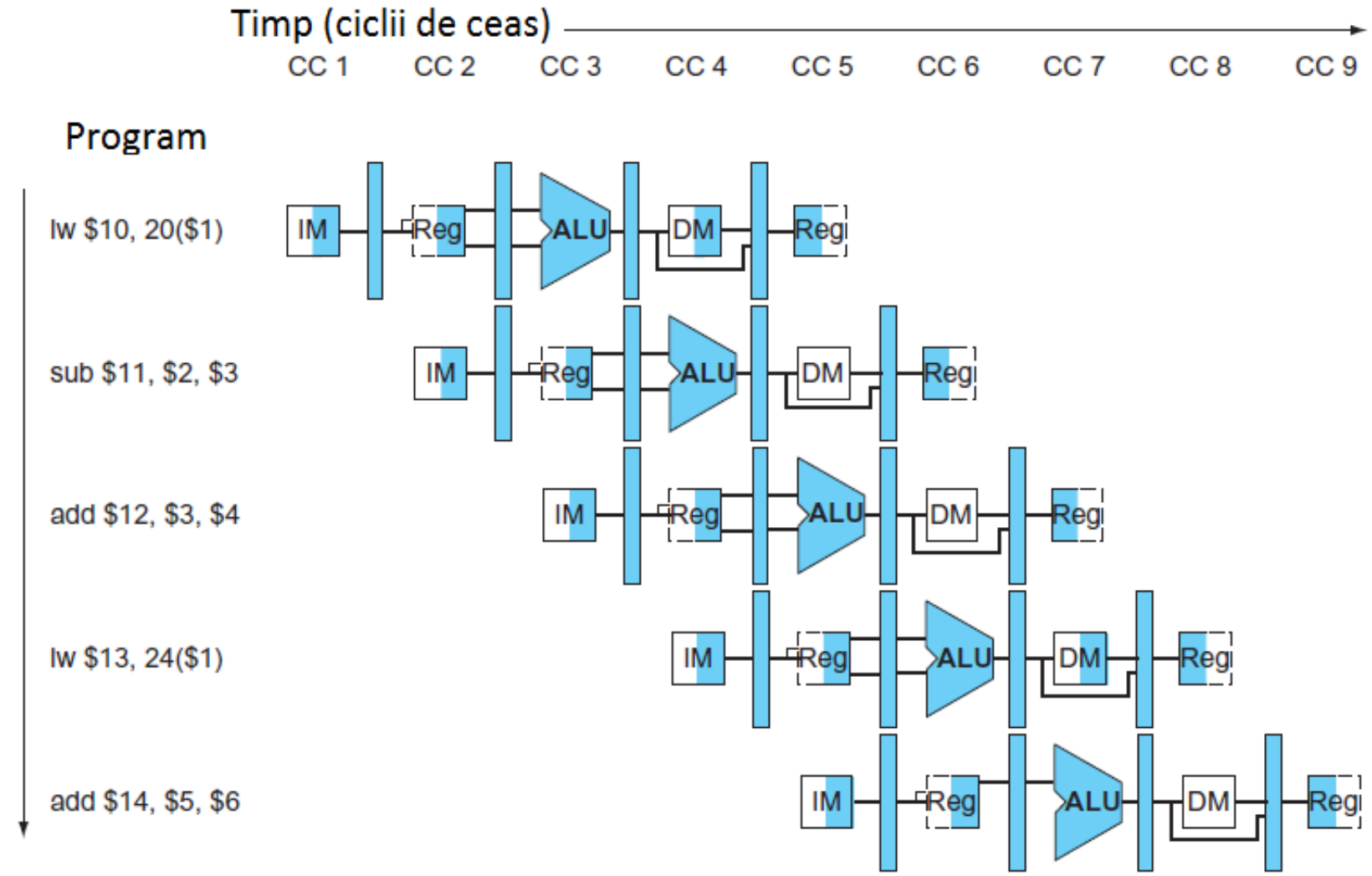


Execuția instrucțiunii lw – versiunea corectă, cu ultimul bug eliminat

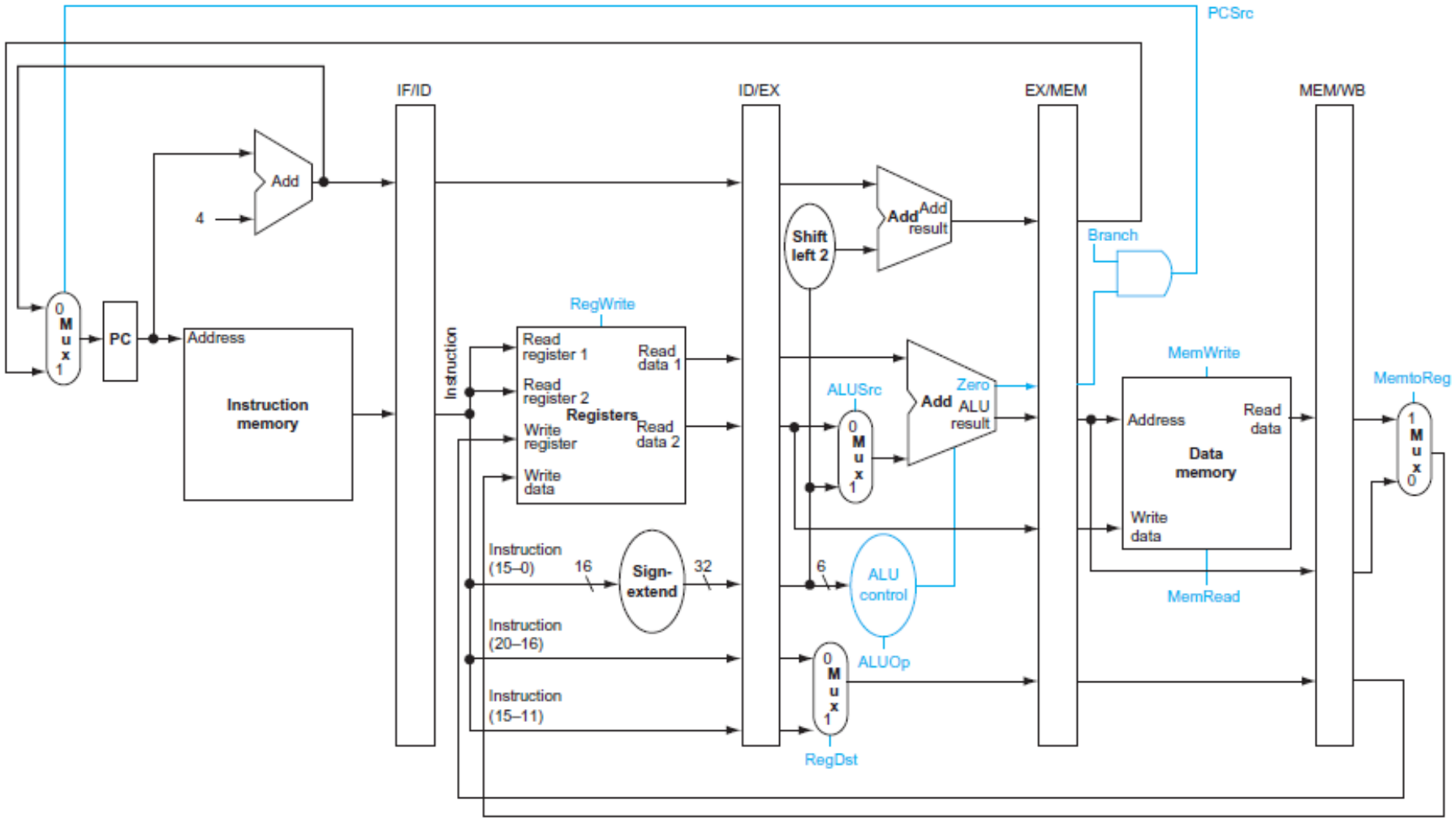


Excepția Write-Back a fost rezolvată. ***Numărul registrului provine de la MEM/WB prin intermediul căii de date. Ne trebuie încă 5 biți să adăugăm la registre.***

Exemplu de diagramă pipeline pentru 5 instrucțiuni:



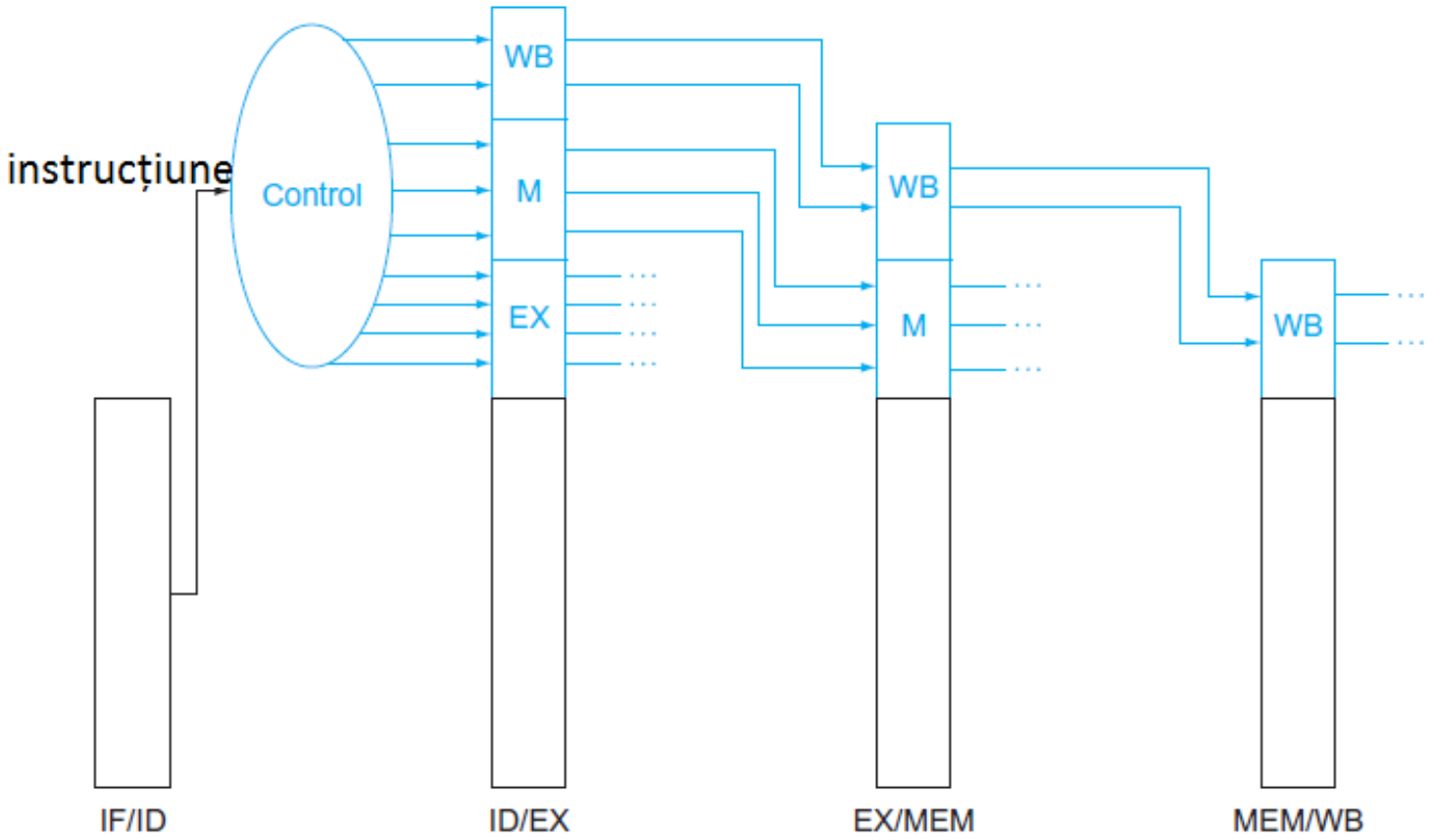
Controlul în pipeline



Instruction opcode	ALUOp	Instruction operation	Function code	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	RegDst	ALUOp1	ALUOp0	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Liniile de control pentru cele 3 stagii



Versiunea finală de control