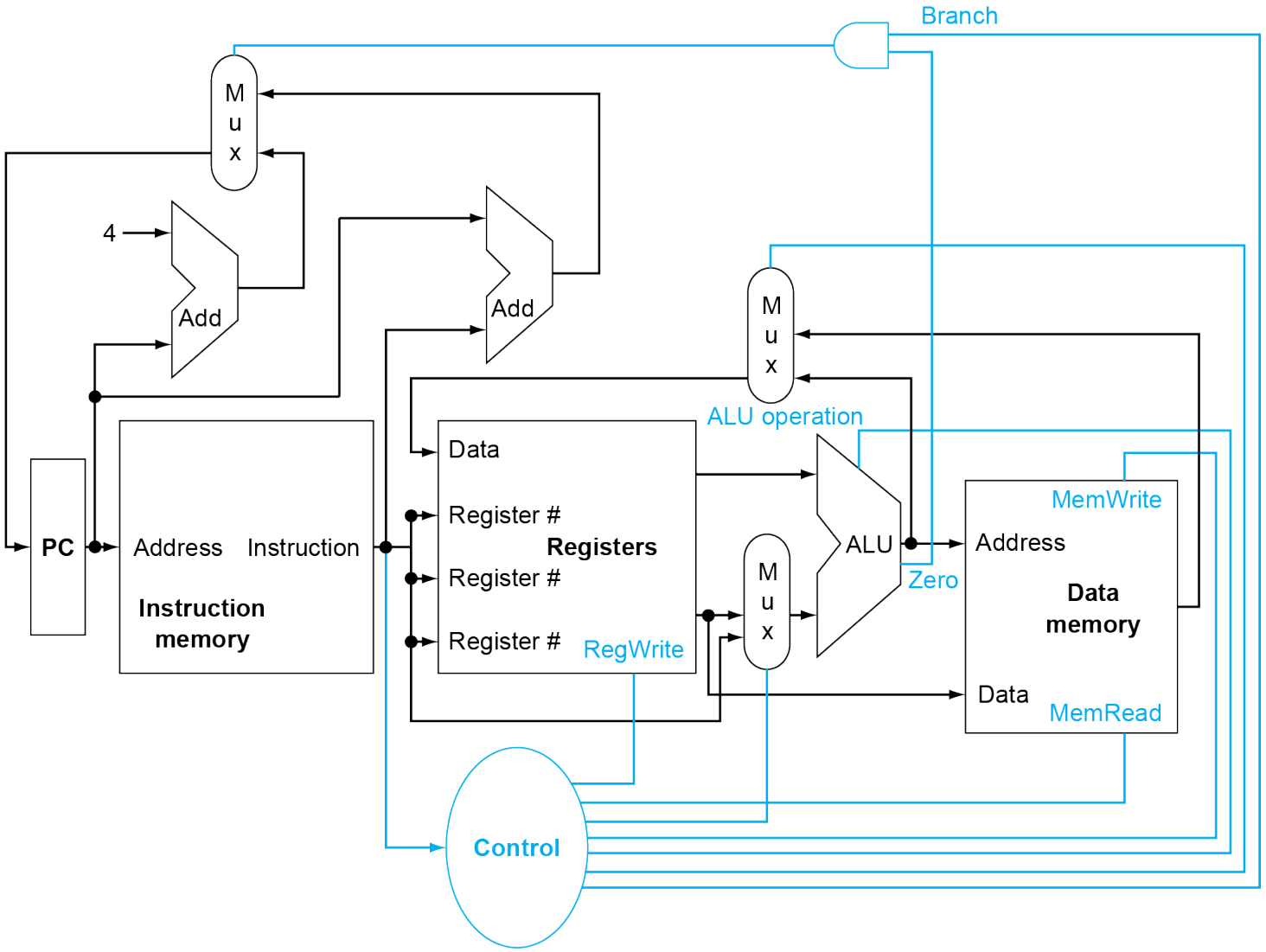
Optimització de processadors (1):

RISC-V single cycle

Els processadors (que denominarem CPU) estudiats en cursos anteriors estan dissenyats amb arquitectura Von Neumann. El funcionament dels quals es basen als punts:

- Els programes (quan s'executen) i les dades resten a una memòria comuna, que denominem memòria principal.
- Per tal d'executar els programes sempre es fa la mateixa seqüència :
  1. La CPU llegeix una instrucció de memòria
  2. Decodifica la instrucció (amb la Unitat de Control: UC)
  3. La UC genera les ordres per que la Unitat d'Execució (UE) i la resta del sistema duguin a terme totes les operacions d'aquesta instrucció.
  4. S'accedeix a memòria, si es necessita
  5. Un cop finalitzada l'execució de la instrucció es comença la mateixa seqüència per la següent instrucció.
- Això es repeteix fins acabar l'execució del programa.

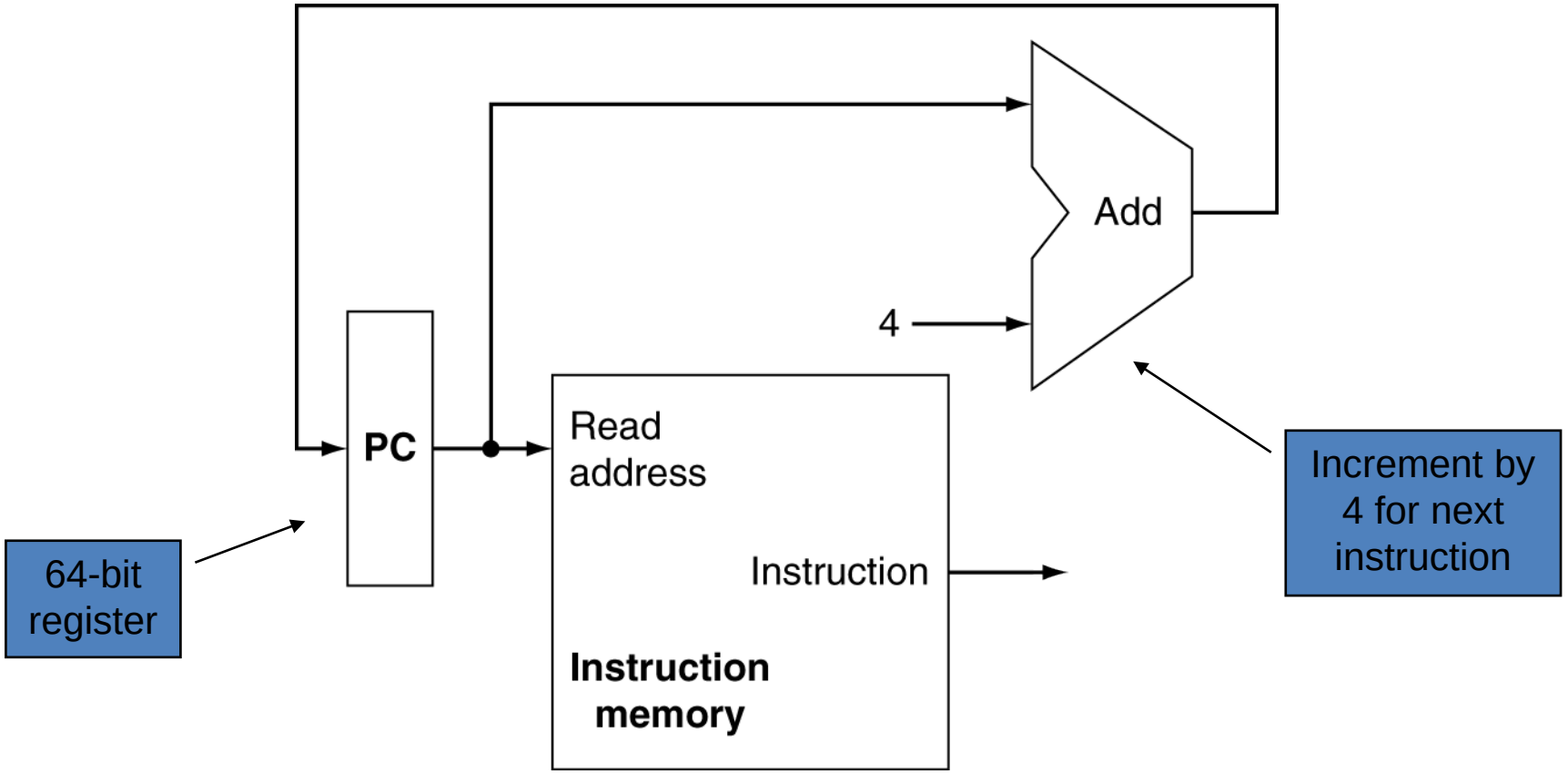


Exemple de CPU (arquitectura **Harvard**)

Construcció del “datapath” d’un processador RISC-V

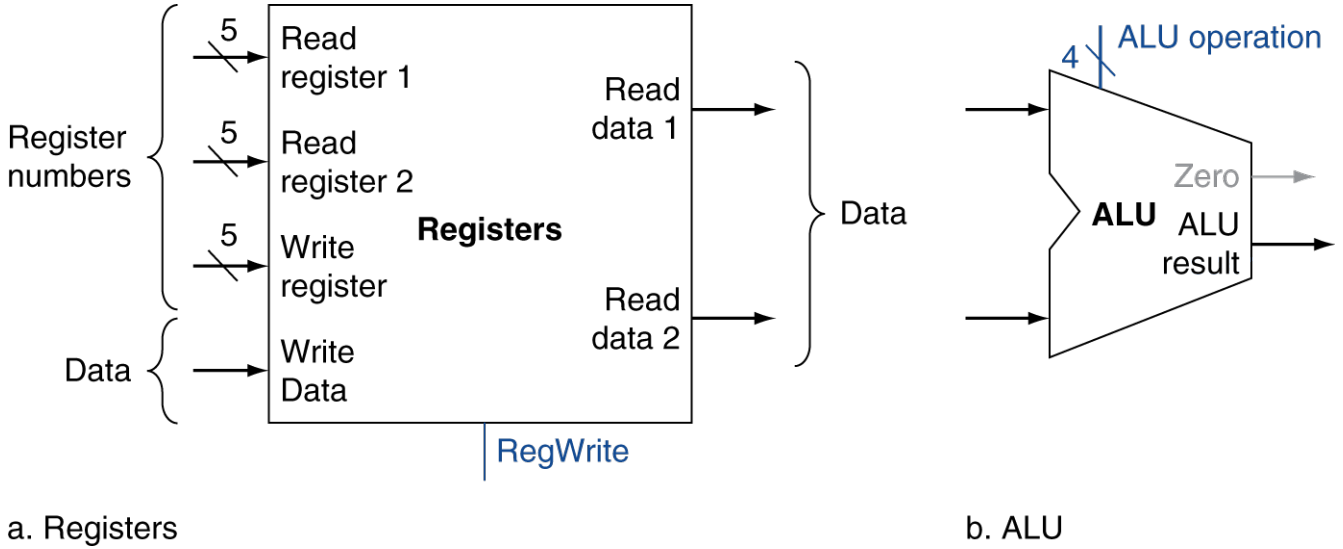
- Partirem del disseny presentat
- Poques instruccions i molt simples
  - Memory reference: ld, sd
  - Arithmetic/logical (R-type): add, sub, and, or
  - Control transfer: beq
- Com funciona cada bloc del processador exemple ?

“Instruction Fetch” o com introduïm les noves instruccions



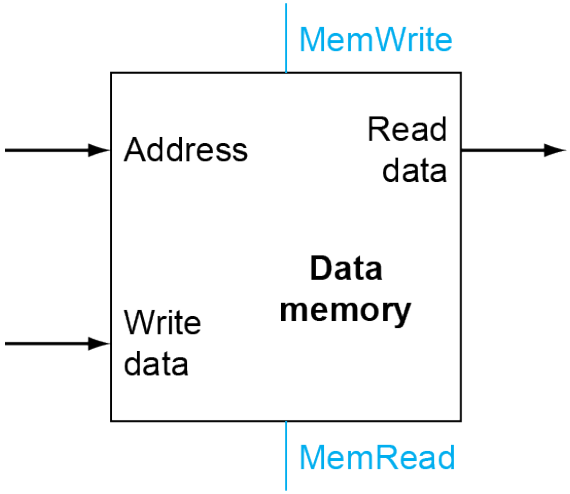
Instrucció tipus “R-Format”

- Llegeix operands del conjunt de registres, normalment dos registres
- Realitza operacions aritmètiques i lògiques
- Escriu el resultat en un registre

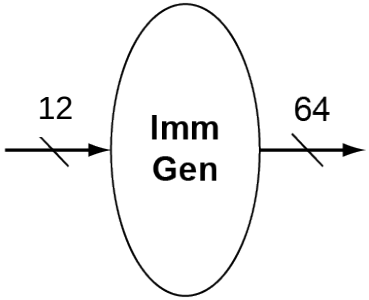


Instrucció tipus Load/Store

- Llegeix operands del conjunt de registres
- Calcula l'adreça emprant els 12 bits d'offset
  - Utilitza la ALU, en particular l'extensió de signe
- Load: llegeix la memòria i actualitza el conjunt de registres
- Store: Escriu el valor d'un registre a la memòria



a. Data memory unit

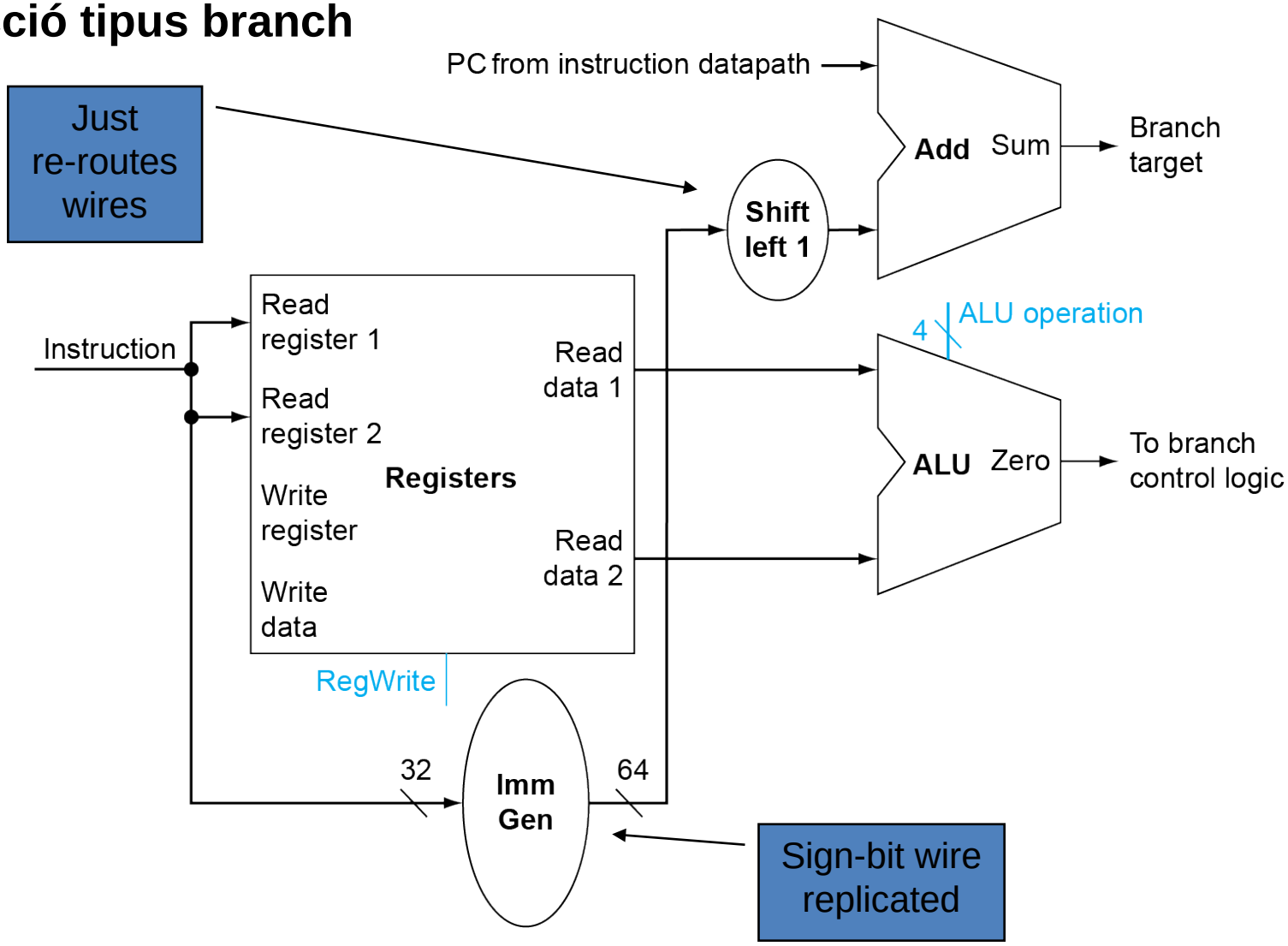


b. Immediate generation unit

Instrucció tipus branch

- Llegeix operands del conjunt de registres
- Compara els operands:
  - Utilitza la ALU, resta i comprova si el resultat és 0
- Calcula adreça destí
  - Desplaçament “sign-extended”
  - Desplaçament a l’esquerra d’1 bit (shift left)
  - Suma el valor final al PC

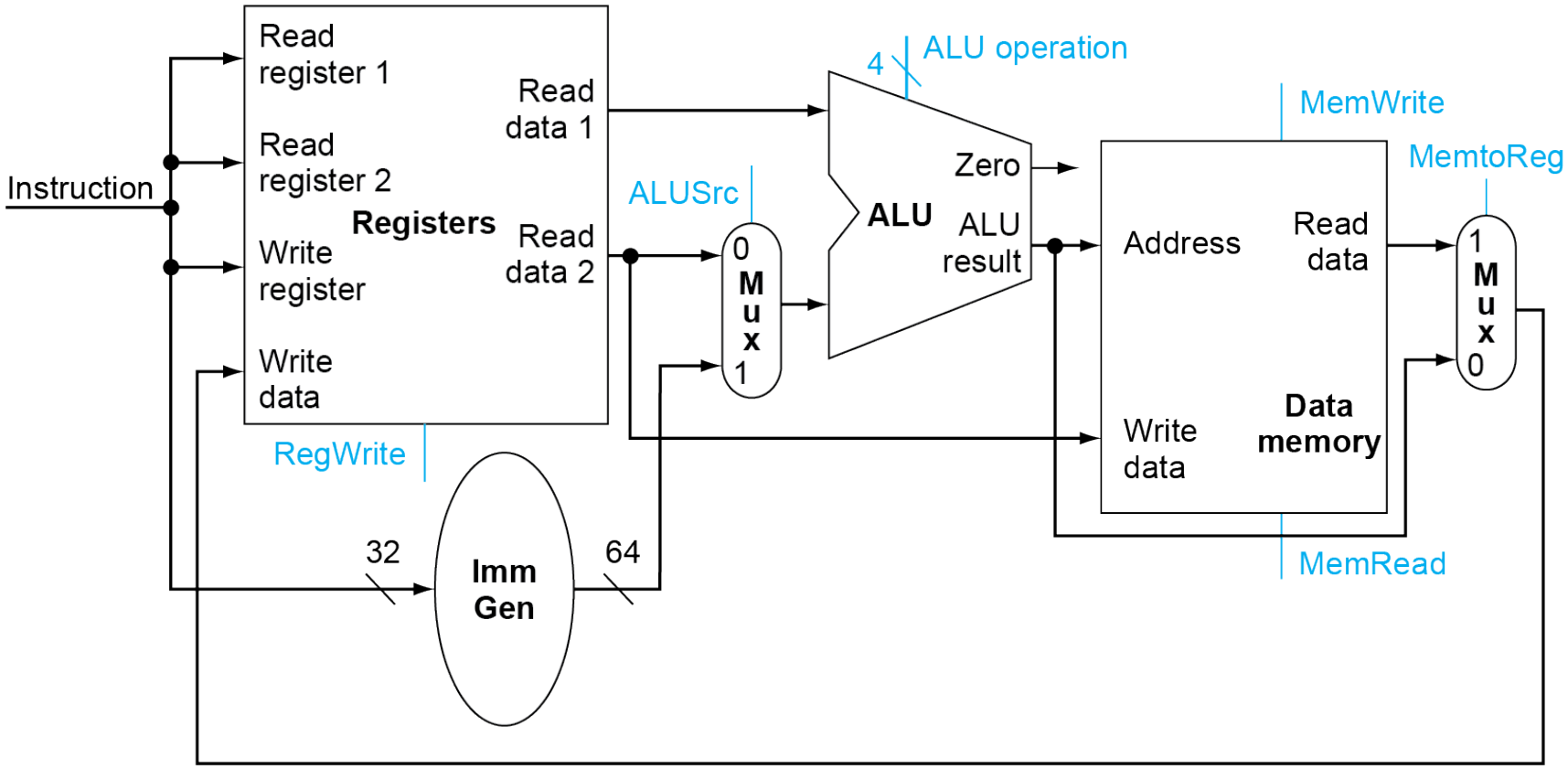
Instrucció tipus branch



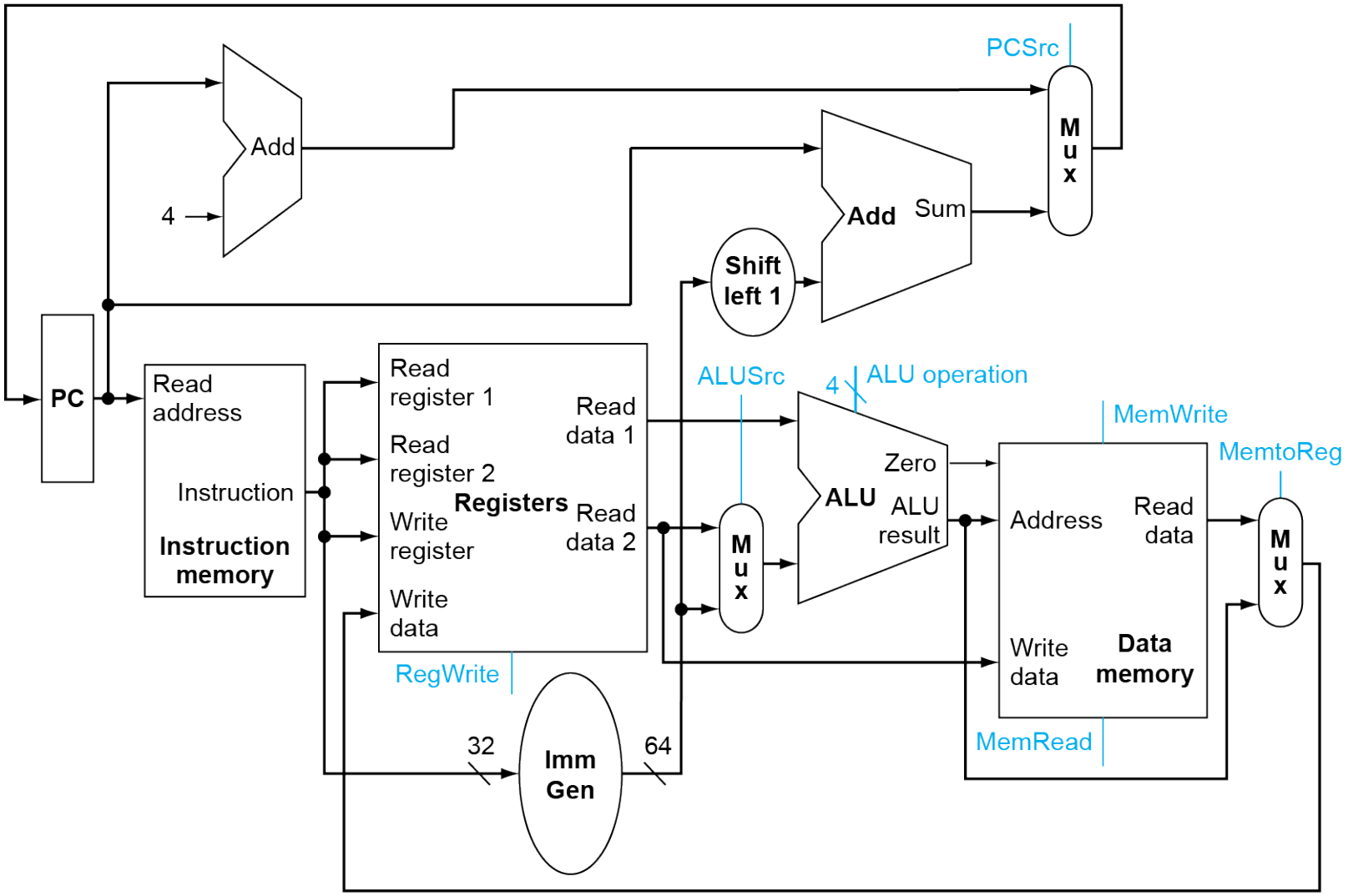
Resolem el “trencaclosques”

- Executem una instrucció en un cicle de rellotge:
  - Cada element del “datapath” només pot executar una funció en cada moment
  - Necessitem duplicar la memòria: una memòria d’instruccions i una memòria per a dades
- Utilitzarem multiplexors quan volguem alternar diferents “fonts” per a un mateix bloc

R-type/Load/Store Datapath



Full Datapath

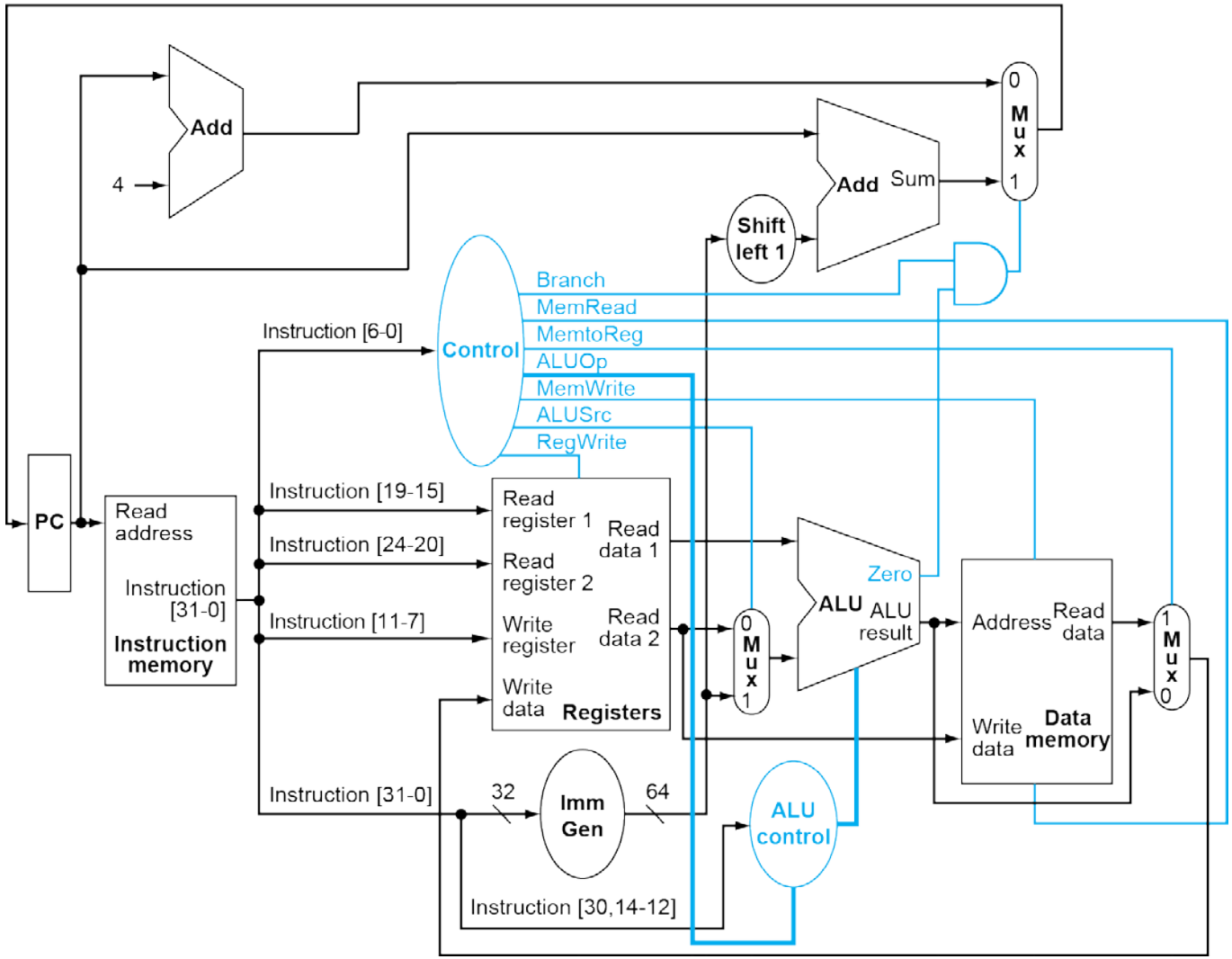


Unitat de control: Senyals de control es deriven de les instruccions

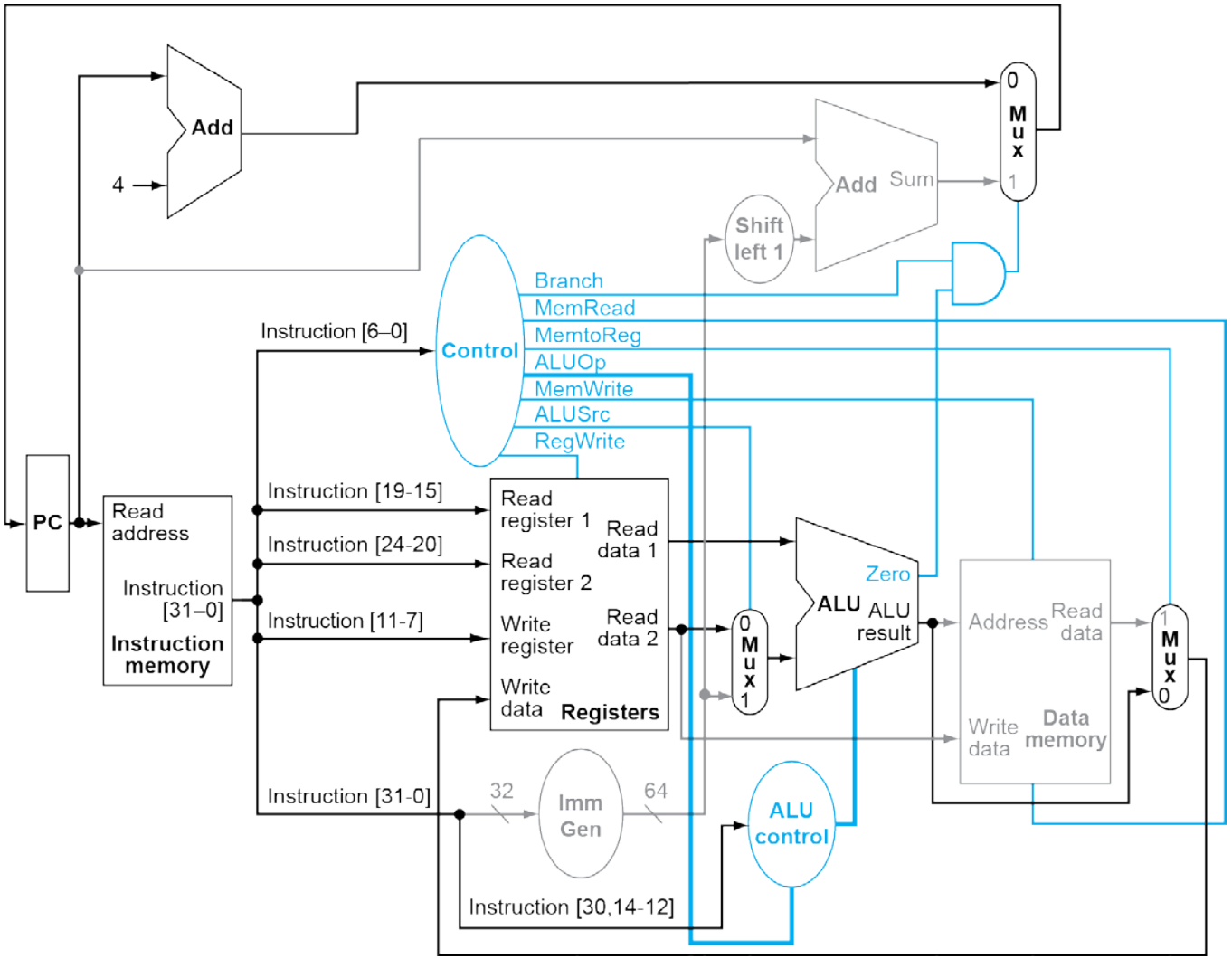
Name (Bit position)	Fields 31:25	24:20	19:15	14:12	11:7	6:0
(a) R-type	funct7	rs2	rs1	funct3	rd	opcode
(b) I-type	immediate[11:0]		rs1	funct3	rd	opcode
(c) S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode
(d) SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode

ALUOp		Funct7 field							Funct3 field			
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	Operation
0	0	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	0110
1	X	0	0	0	0	0	0	0	0	0	0	0010
1	X	0	1	0	0	0	0	0	0	0	0	0110
1	X	0	0	0	0	0	0	0	1	1	1	0000
1	X	0	0	0	0	0	0	0	1	1	0	0001

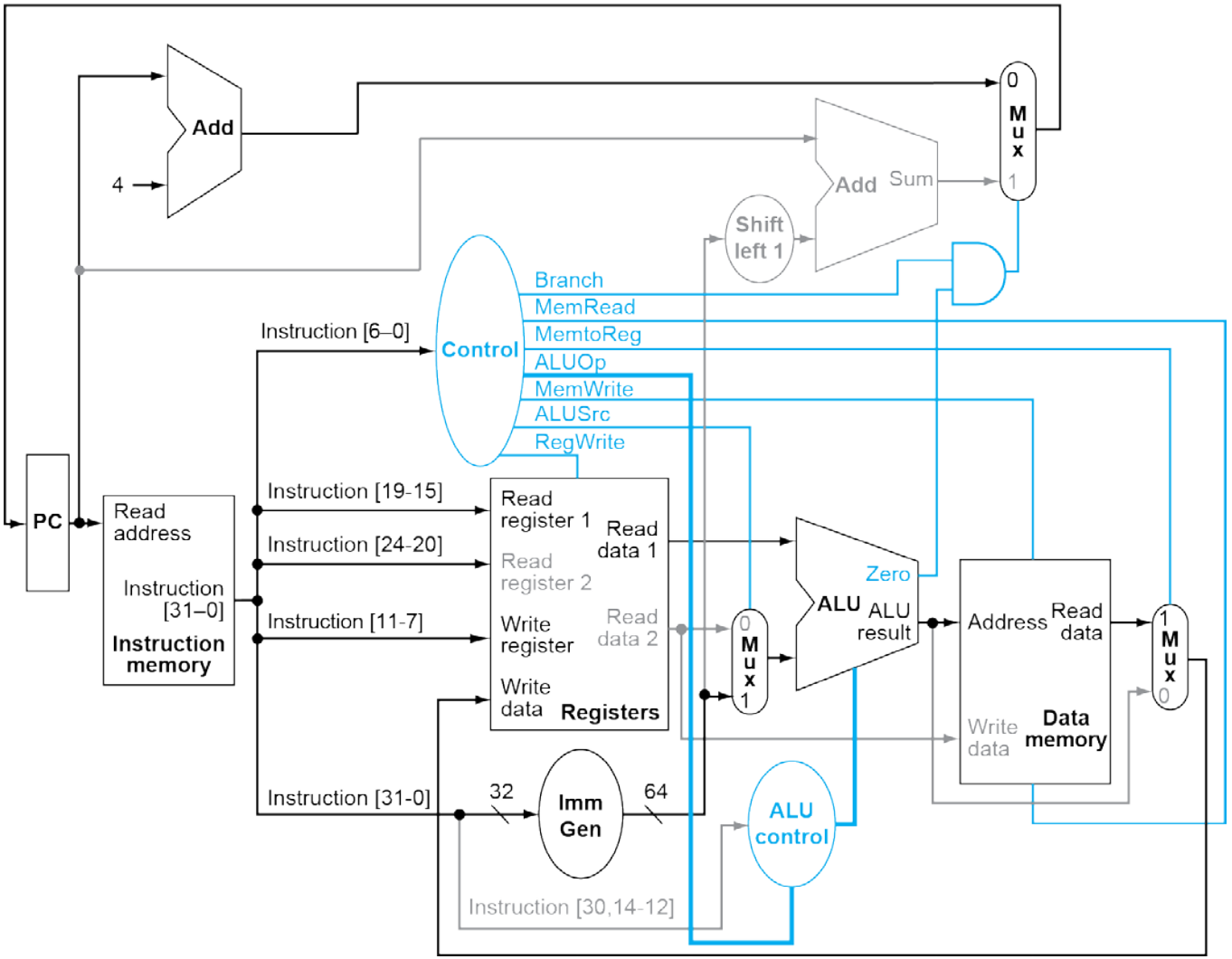
Full Datapath with control



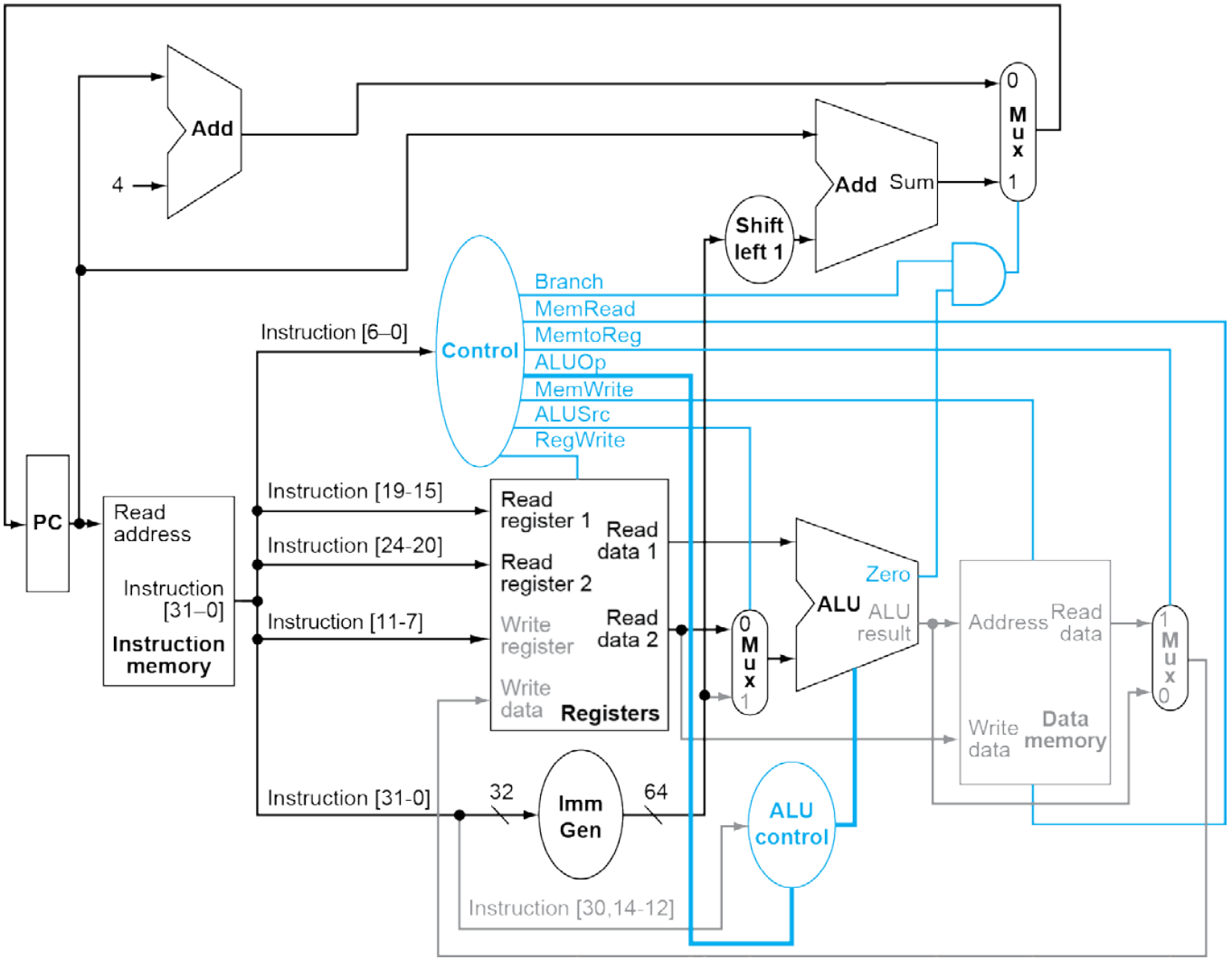
Exemple instrucció R-type



Exemple instrucció Load



Exemple instrucció beq (branch)



Problemes de rendiment en el processador RISC-V “single-cycle”

- El retard més gran determina el període del rellotge
  - Retard: Temps que triga cada bloc en executar la seva funció
  - Camí crític: Instrucció Load (Instruction memory → register file → ALU → data memory → register file)
- No és possible variar la freqüència del rellotge per a diferents instruccions
- El cas comú no és el més ràpid

Exemple:

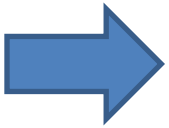
En el processador RISC-V "single-cycle" mostrat fins ara, les instruccions triguen en executar-se, segons els blocs que utilitzen, el següent:

Tipus d'Instrucció	Temps en executar-se
Load	800 ps
Store	700 ps
R-type (aritmètiques/lògiques)	600 ps
Branch (control)	500 ps

Quin és el període mínim de rellotge al que podrem fer funcionar aquest processador ?

Problemes de rendiment en el processador RISC-V “single-cycle”

- El retard més gran determina el període del rellotge
  - Retard: Temps que triga cada bloc en executar la seva funció
  - Camí crític: Instrucció Load (Instruction memory → register file → ALU → data memory → register file)
- No és possible variar la freqüència del rellotge per a diferents instruccions
- El cas comú no és el més ràpid



Necessitat d’augmentar prestacions/rendiment

Si volem augmentar les prestacions del processador (això significa trigar menys temps en fer el mateix) tenim 2 opcions:

1. Millorar la tecnologia fent dispositius més ràpids, reduint el cicle de rellotge (T_C) (o el que és equivalent augmentant f_C)
   - No precisa de canvis conceptuals, només tecnològics.

2. Redissenyar el processador per que cada instrucció trigui menys cicles de rellotge en executar-se.
   - Això significa dissenyar noves arquitectures que permetin disminuir els CPI del processador.

Estudiarem les tècniques basades al punt 2, tot i precisant que en realitat cada instrucció no trigarà forçosament menys en executar-se, sinó que la execució del conjunt d'instruccions que forma un programa doni un temps promig per instrucció inferior.

Els canvis arquitecturals per aconseguir el nostre objectiu es poden fer de dues maneres:

1. Implementant tècniques de segmentació d'instruccions. Això ho denominarem "**Paral·lelisme Temporal**", o processadors amb "***Pipeline***".

2. Amb arquitectures "**Escalars**" o "**Superescalars**" en les quals es poden executar vàries instruccions simultàniament a unitats d'execució diferents del mateix processador.

Ens ocuparem en primer lloc de les arquitectures de *pipelining,* que al cas ideal ens permetrien obtenir un CPI de 1. A la pràctica veurem que això és impossible però si que aconseguirem que CPI→1.

Amb arquitectures escalars ja veurem que es pot aconseguir CPI<1.

El que és més important és que veurem que les dues tècniques són compatibles.