Codificación de instrucciones del MIPS (I)

28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	bltz/gez	jump	jump & link	branch eq	branch ne	blez	bgtz
1(001)	add immediate	addiu	set less than imm.	set less than imm. unsigned	andi	ori	xori	load upper immediate
2(010)	TLB	FIPt						
3(011)								
4(100)	load byte	load half	lwl	load word	load byte unsigned	load half unsigned	lwr	
5(101)	store byte	store half	swl	store word			swr	
6(110)	load linked word	lwc1						
7(111)	store cond. word	swc1						

Codificación de instrucciones del MIPS(II)

2-0 5-3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	Shift left logical		shift right logical	sra	sllv		srlv	srav
1(001)	Jump Register	jalr			syscall	break		
2(010)	mfhi	mthi	mflo	mtlo				
3(011)	mult	multu	div	divu				
4(100)	add	addu	subtract	subu	and	or	xor	nor
5(101)			set l.t.	set l.t. unsigned				
6(110)								
7(111)								

Tradución a/del lenguaje de máquina

Instruction	Format	op	rs	rt	rd	shamt	funct	address
add	R	0	reg	reg	reg	0	32_{ten}	n.a.
sub (subtract)	R	0	reg	reg	reg	0	34_{ten}	n.a.
add immediate	I	8_{ten}	reg	reg	n.a.	n.a.	n.a.	constant
lw (load word)	I	35_{ten}	reg	reg	n.a.	n.a.	n.a.	address
sw (store word)	I	43_{ten}	reg	reg	n.a.	n.a.	n.a.	address

Cómo se traducen las siguientes instrucciones?

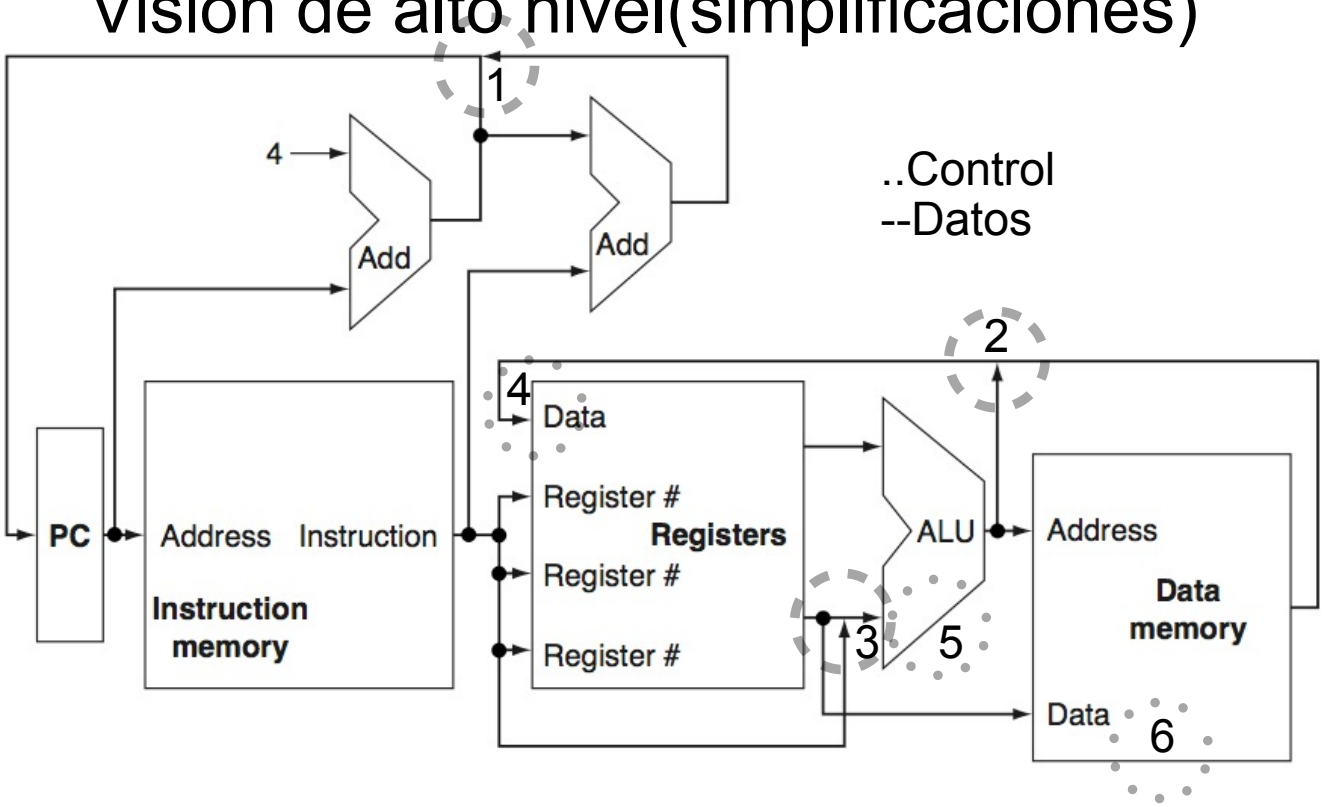
addiu $t0,$zero,0xf

lw $t1,-16($fp)

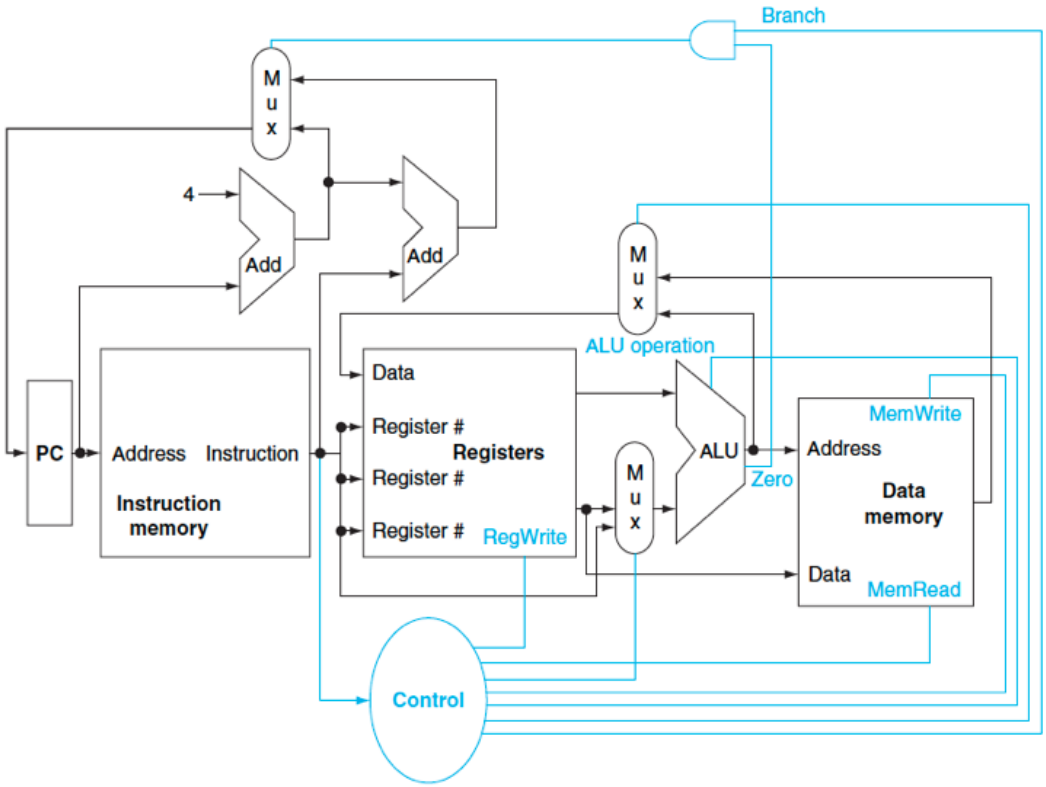
A que sentencia assembly corresponde la instrucción

00af8020 hex

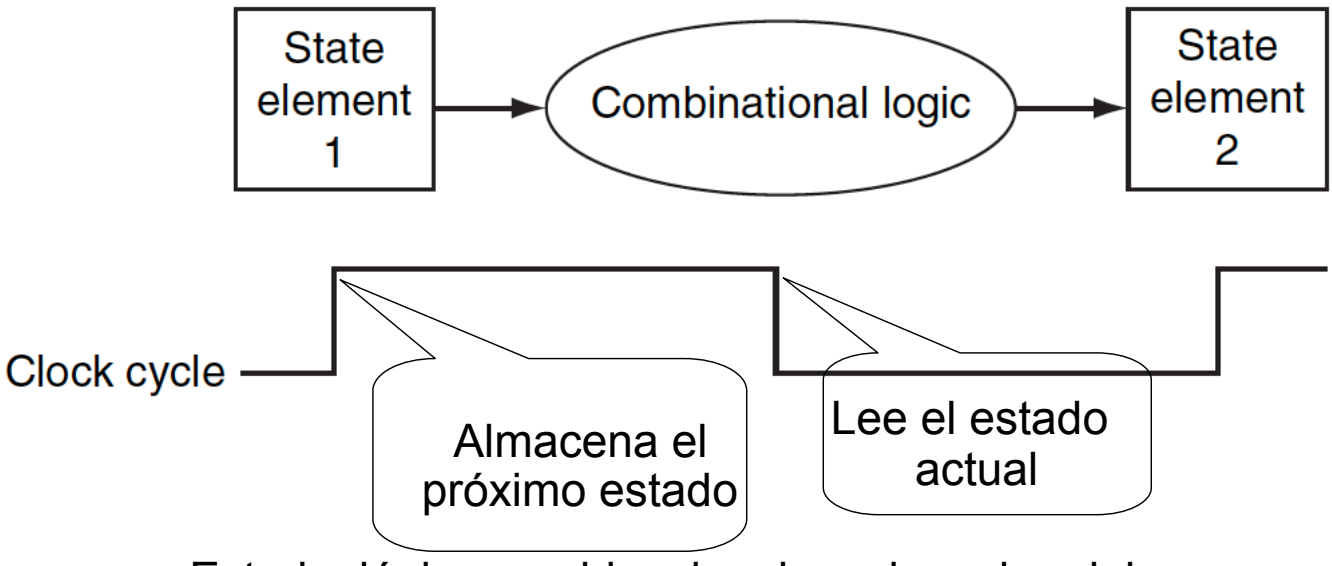
Visión de alto nivel(simplificaciones)



Señales de control

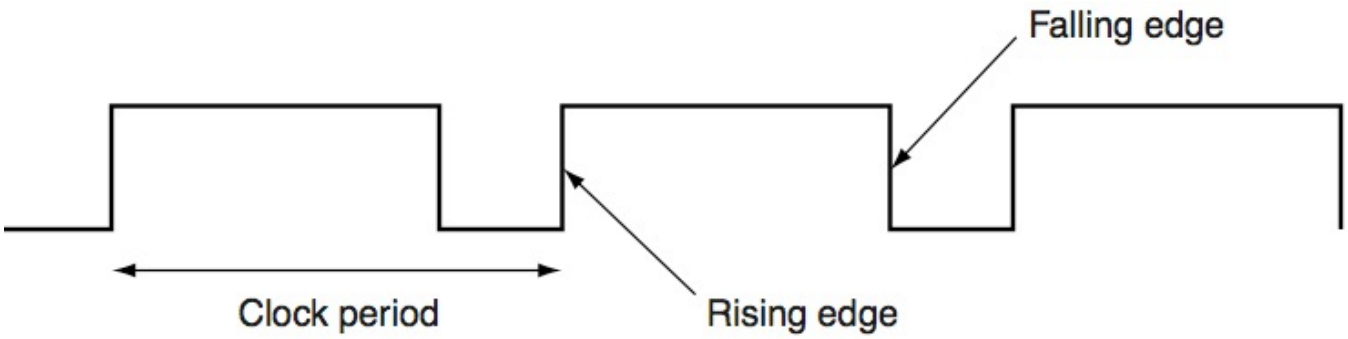


Sincronización por reloj



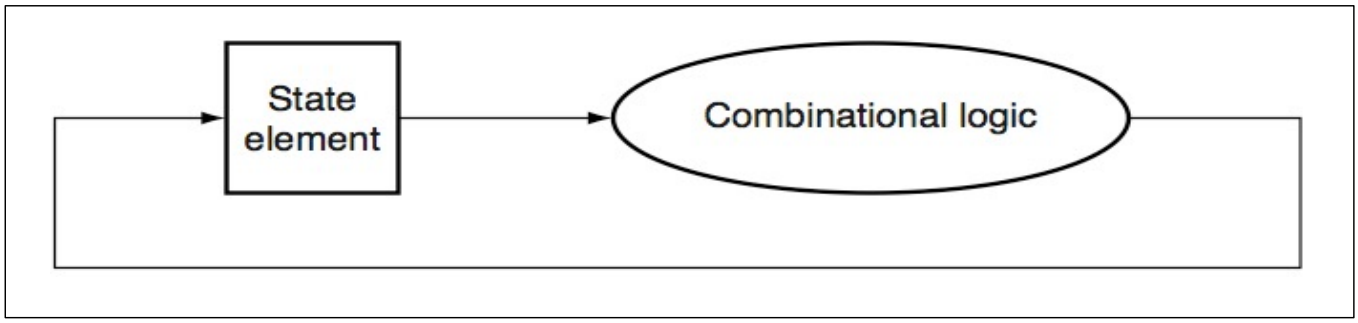
Estado, lógica combinacional y pulsos de reloj.

Señal del reloj



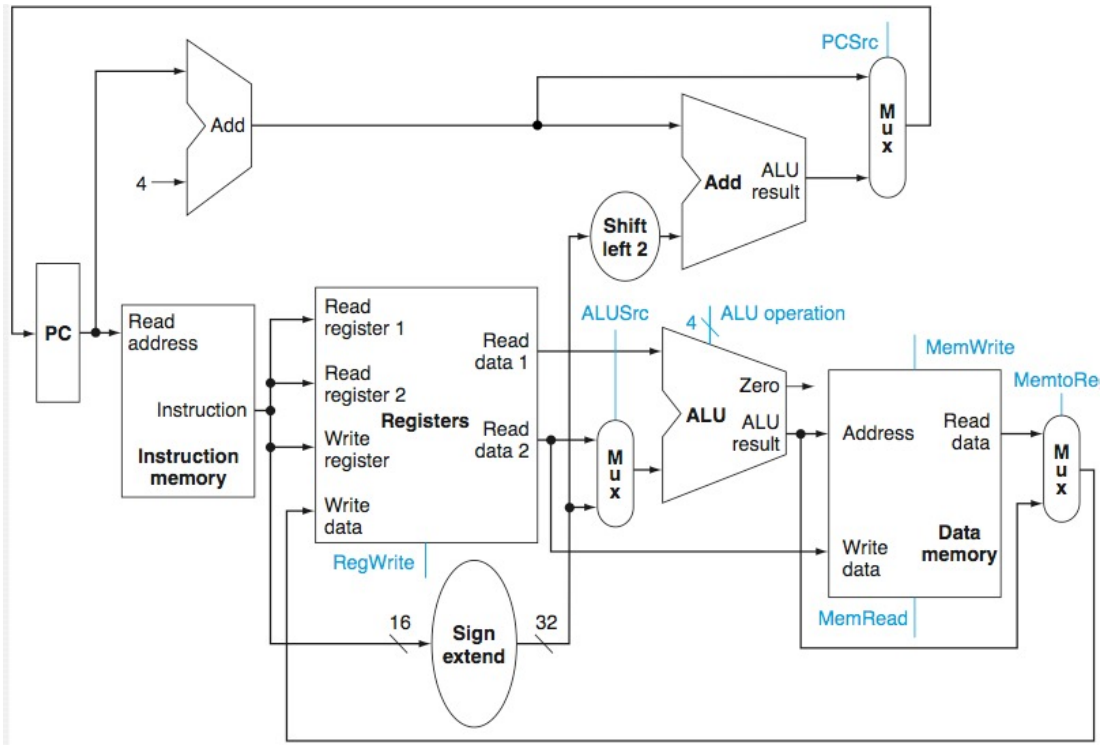
Disparo por flancos – (edge-triggered)

Cambio de estado

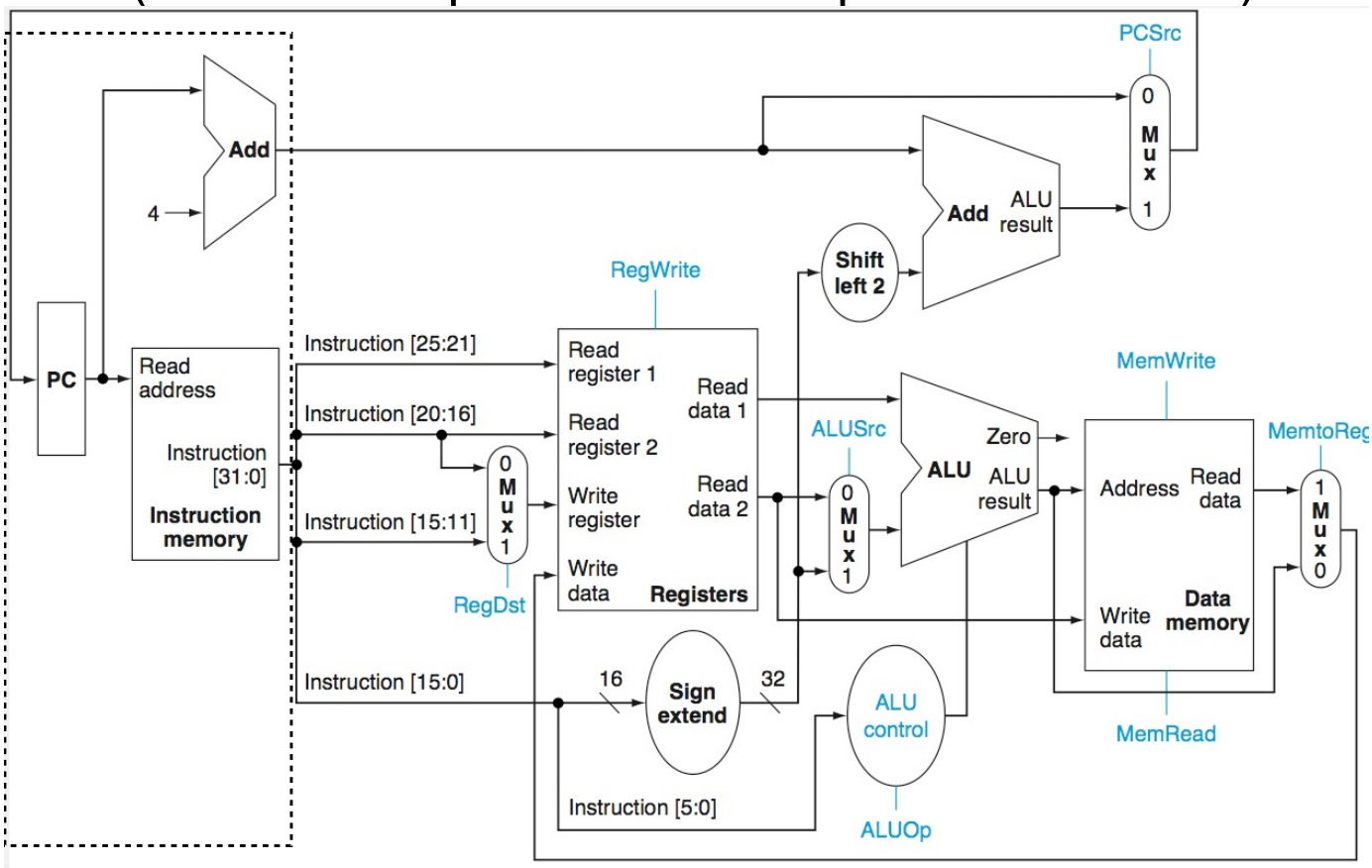


Elemento de estado como entrada y salida de lógica combinacional

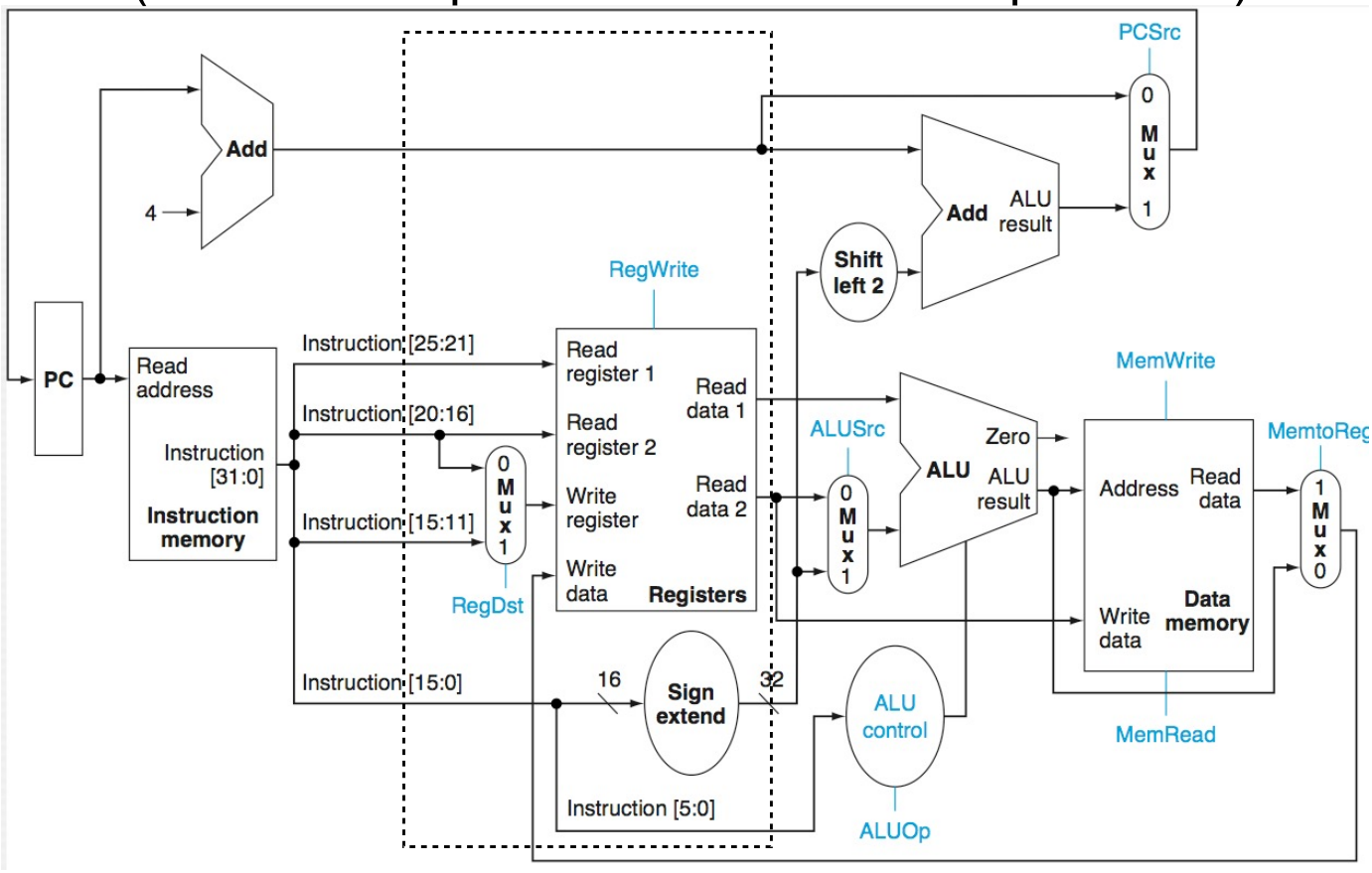
Camino de datos para distintos tipos de instrucciones



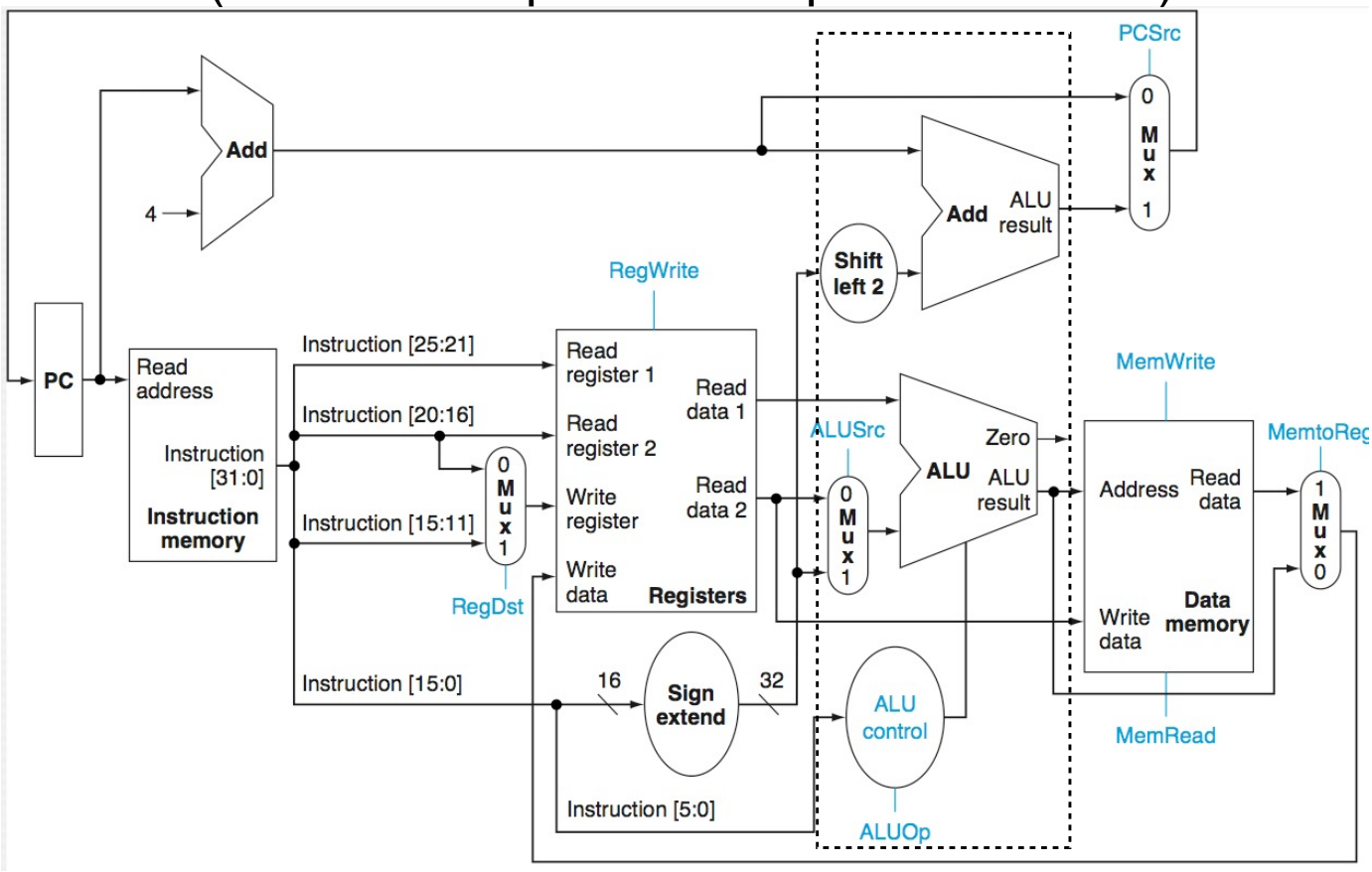
Datos, multiplexores y líneas de control (Instrucción tipo R- Fase Búsqueda instrucción)



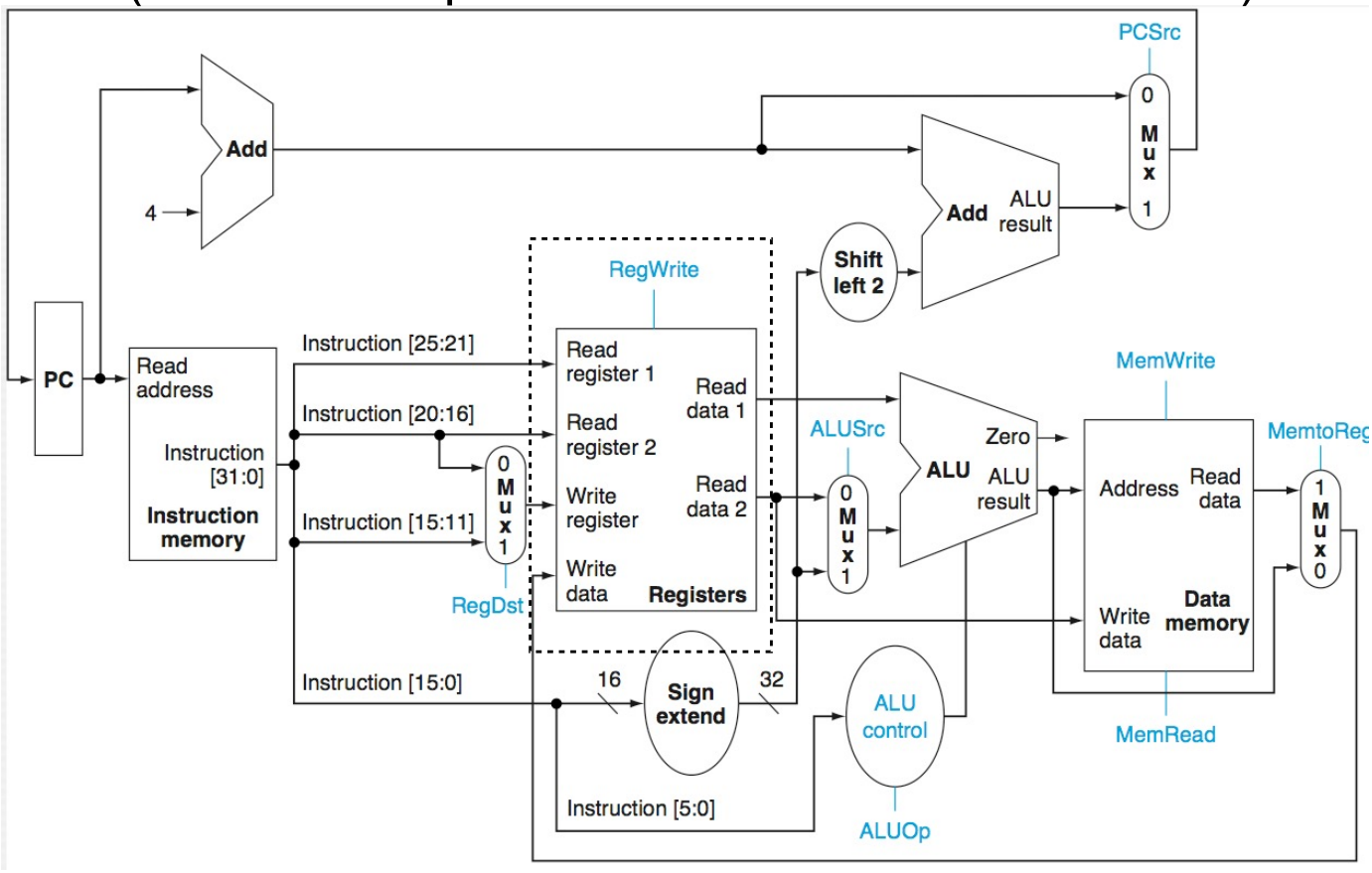
Datos, multiplexores y líneas de control (Instrucción tipo R- Fase lectura de operandos)



Datos, multiplexores y líneas de control (Instrucción tipo R- Fase procesamiento)



Datos, multiplexores y líneas de control (Instrucción tipo R- Fase escritura de resultado)



Señales de la unidad de control

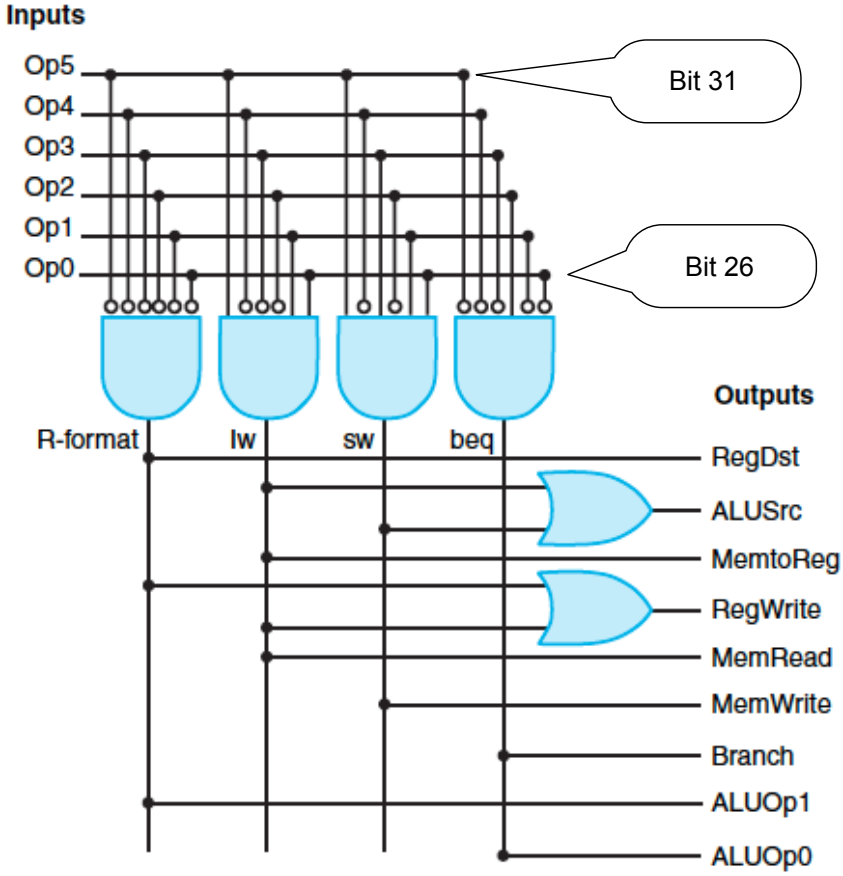
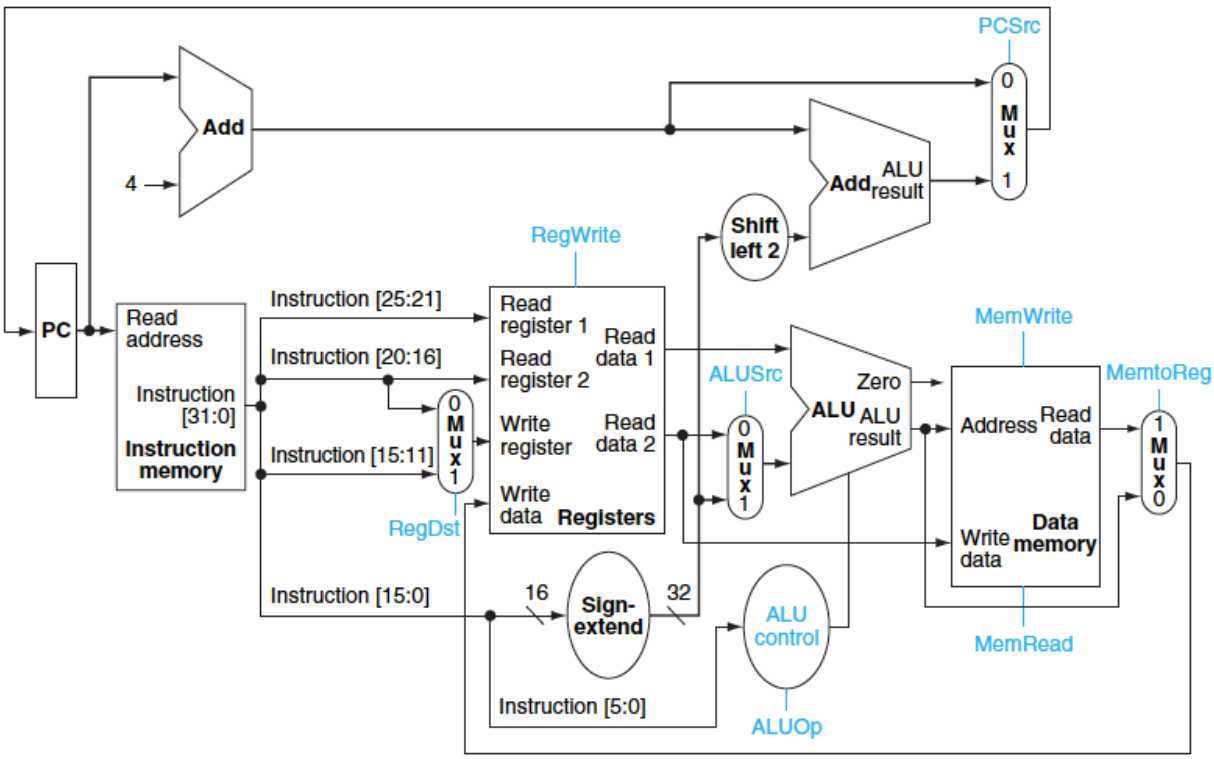


Tabla de verdad de las señales de control

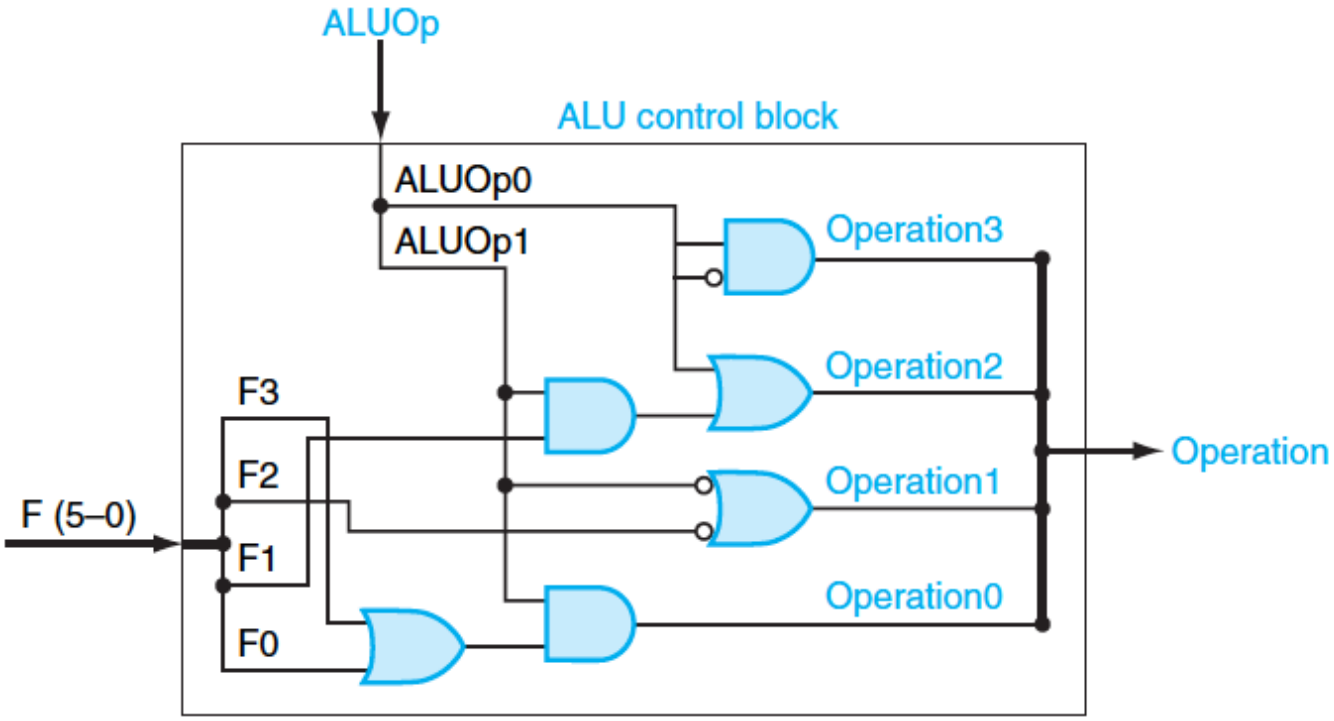
Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

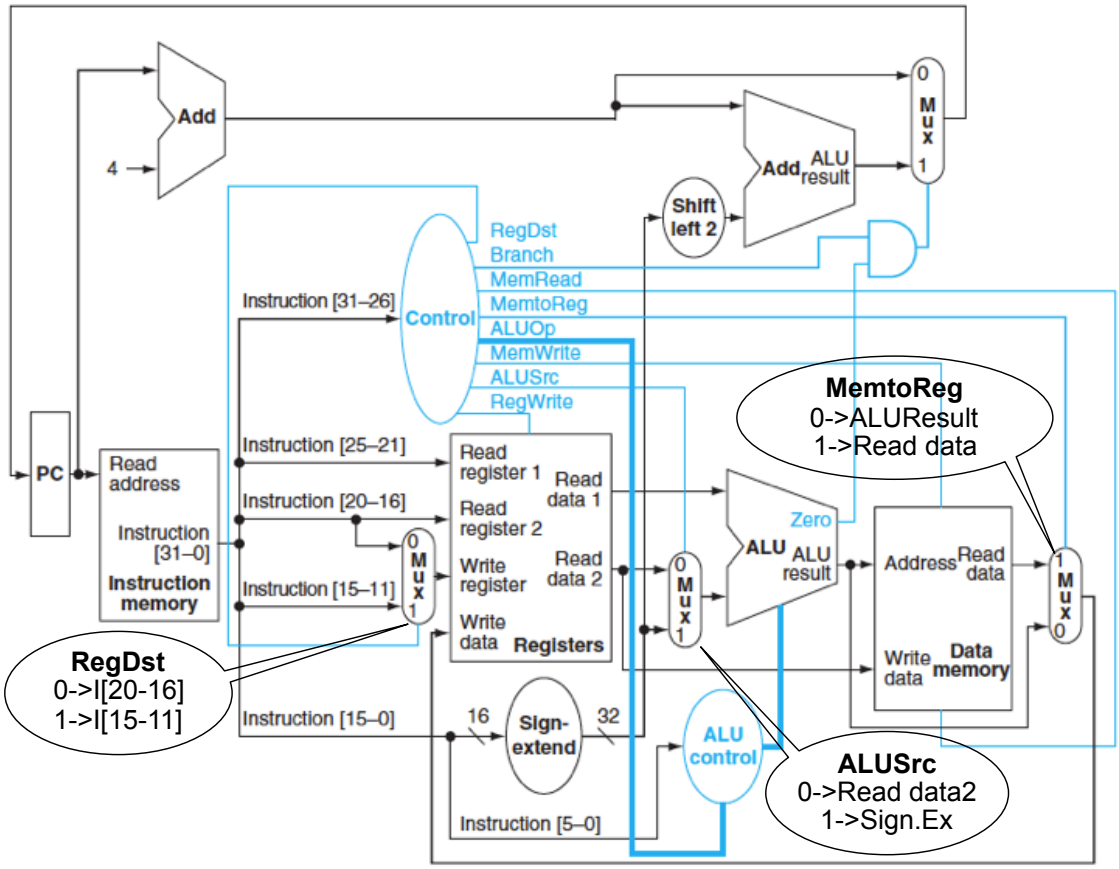


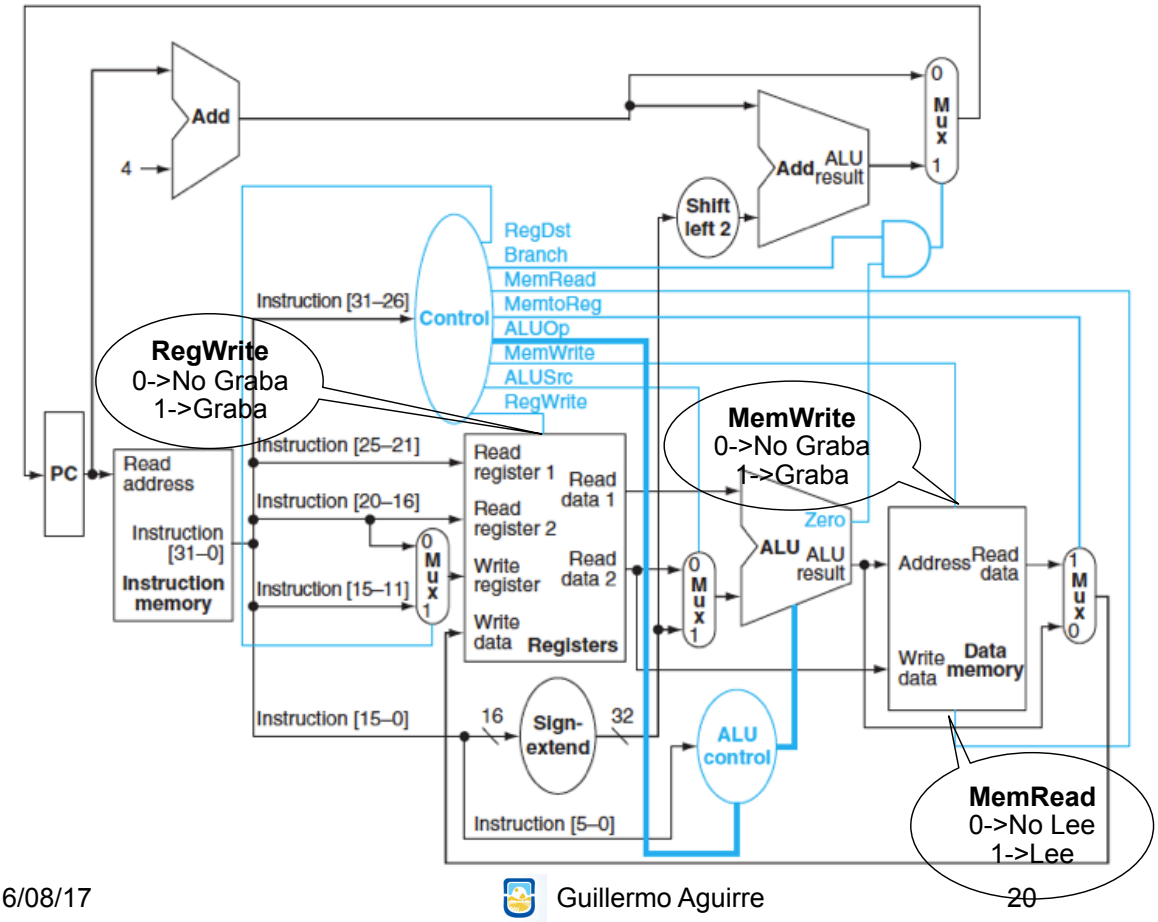
Control de la ALU

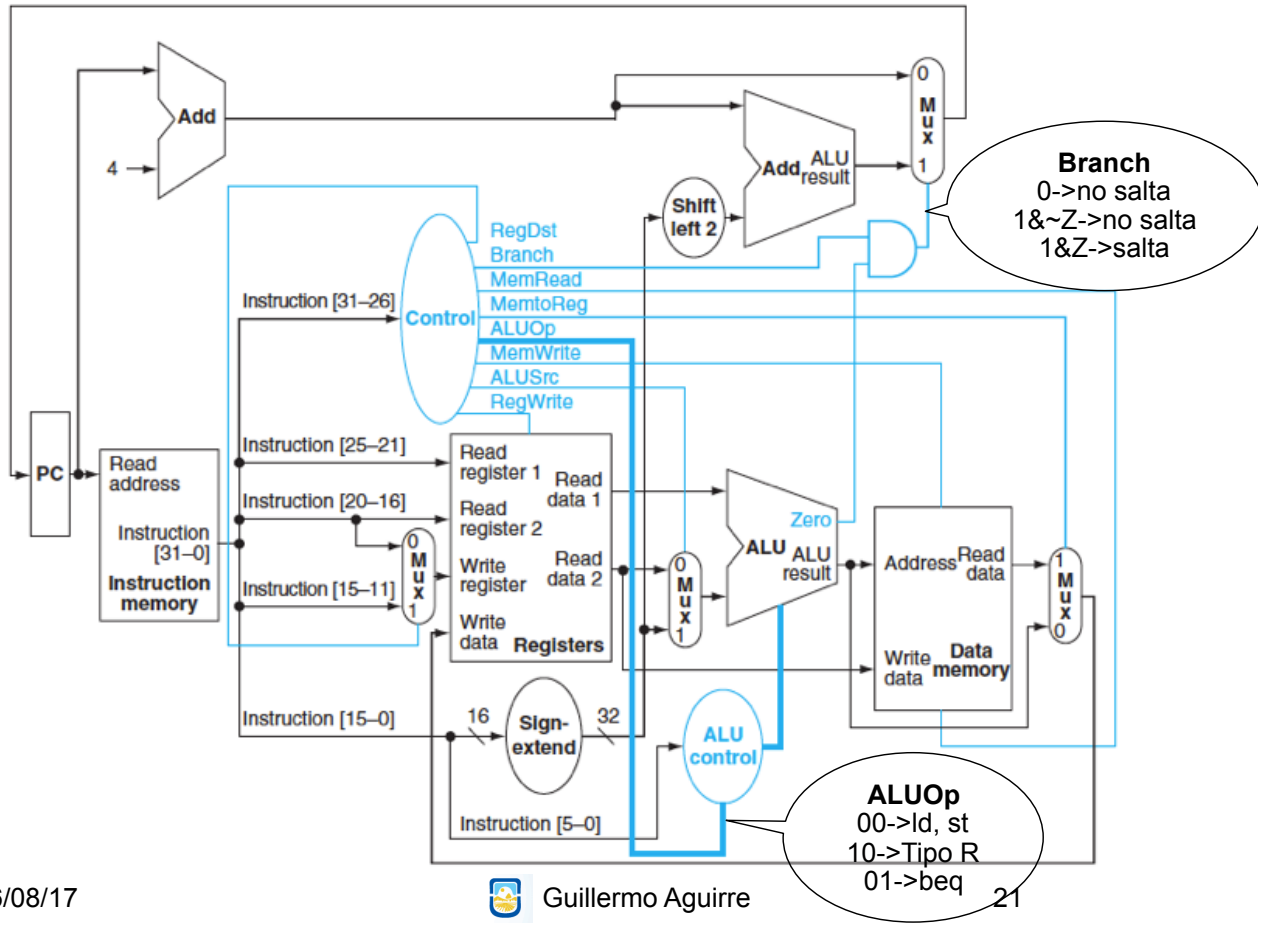
Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100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

Control de la ALU




Add
4
Add
ALU result
Shift left 2
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU control
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
MemtoReg
0->ALUResult
1->Read data
RegDst
0->I[20-16]
1->I[15-11]
ALUSrc
0->Read data2
1->Sign.Ex


Add
4
Add
ALU result
Shift left 2
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU control
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
RegWrite
0->No Graba
1->Graba
MemWrite
0->No Graba
1->Graba
MemRead
0->No Lee
1->Lee



Problemas de la implementación en un solo ciclo

Existe un único ciclo para todas las clases de instrucciones.

La duración del ciclo es igual a la demora del camino más largo.

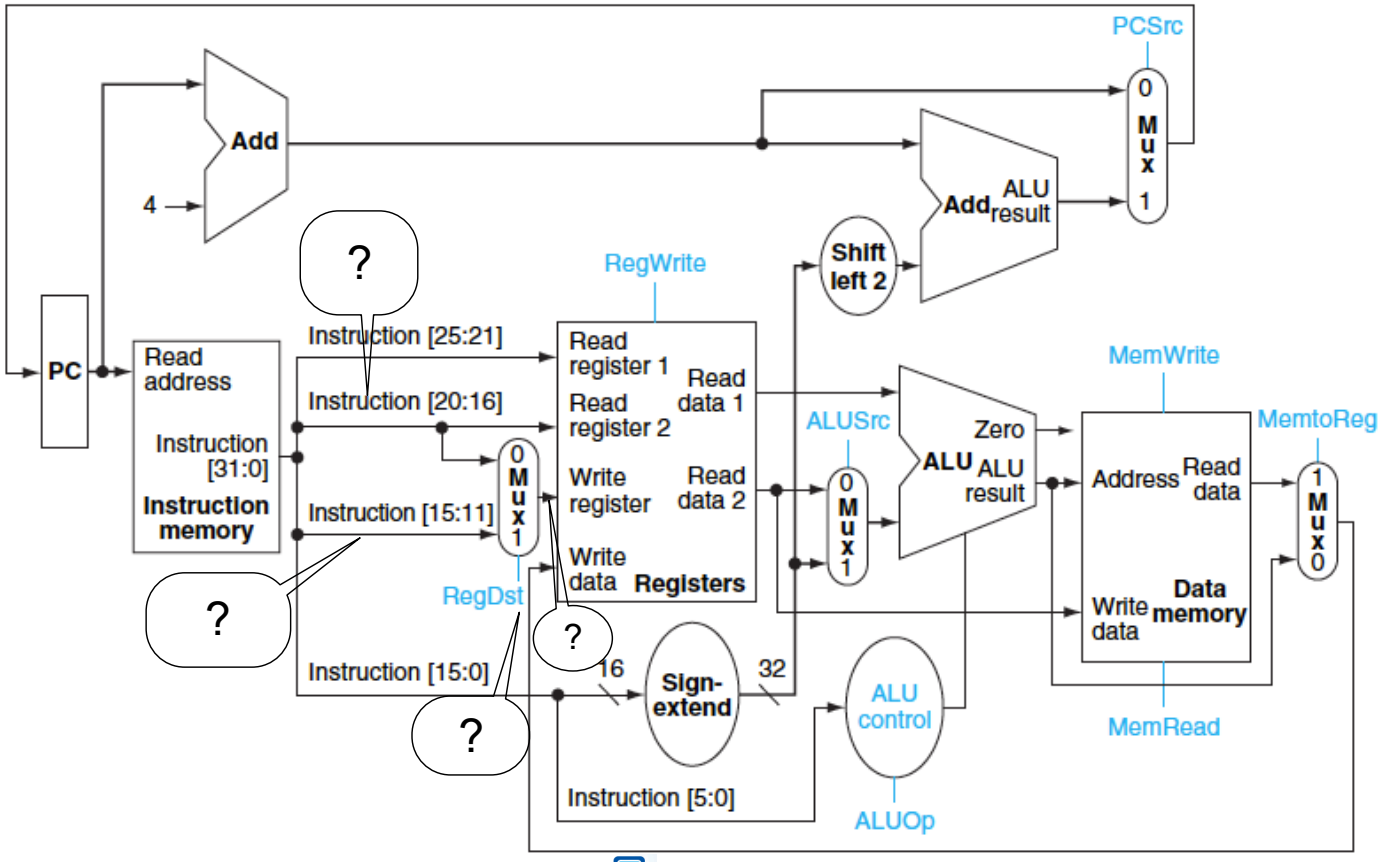
Un load usa cinco unidades.

Las instrucciones más cortas no se benefician.

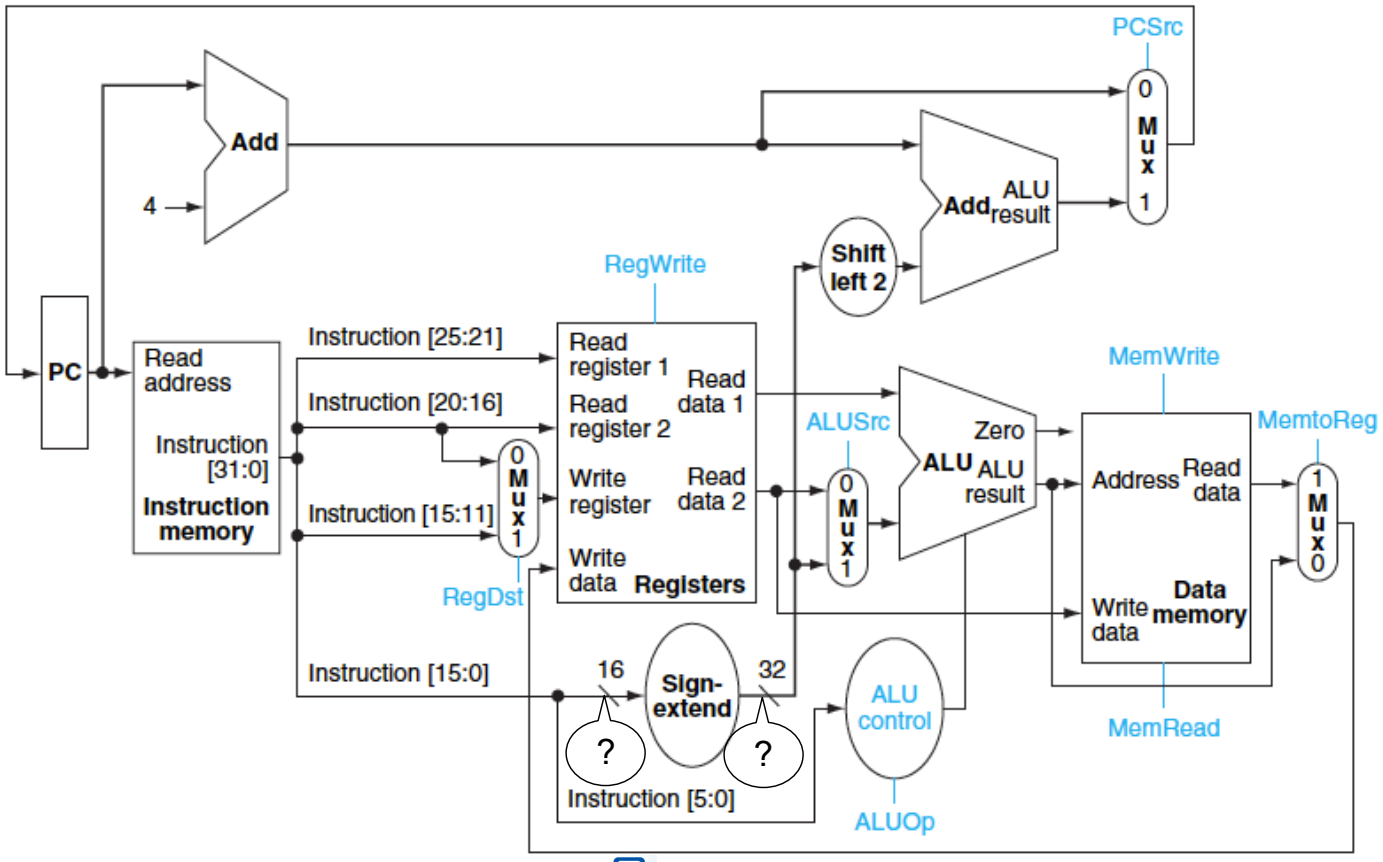
El precio es mayor con instrucciones de punto flotante.

Cada unidad puede ser usada una sola vez.

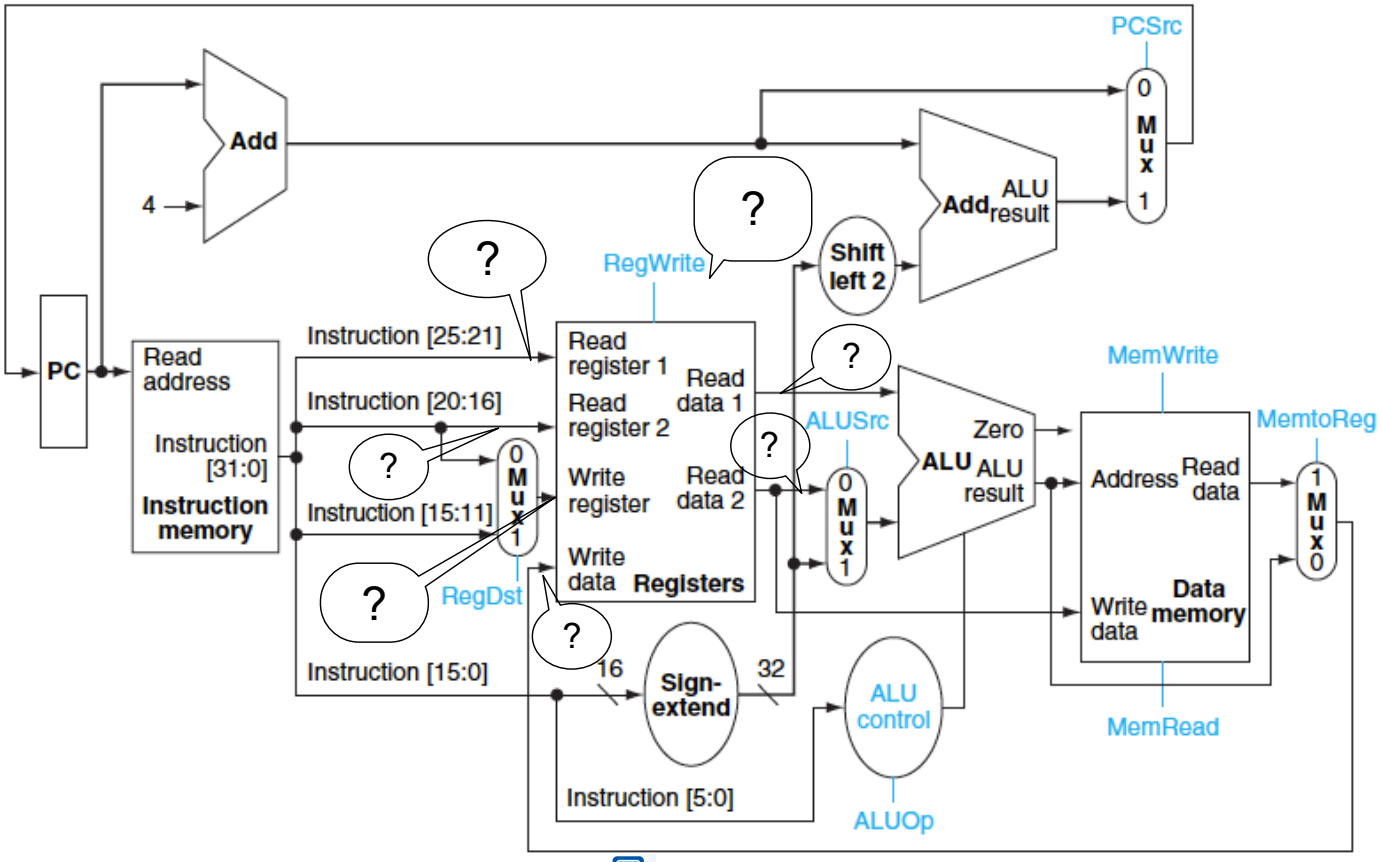
Complete para: lw $a2, 8($t1)



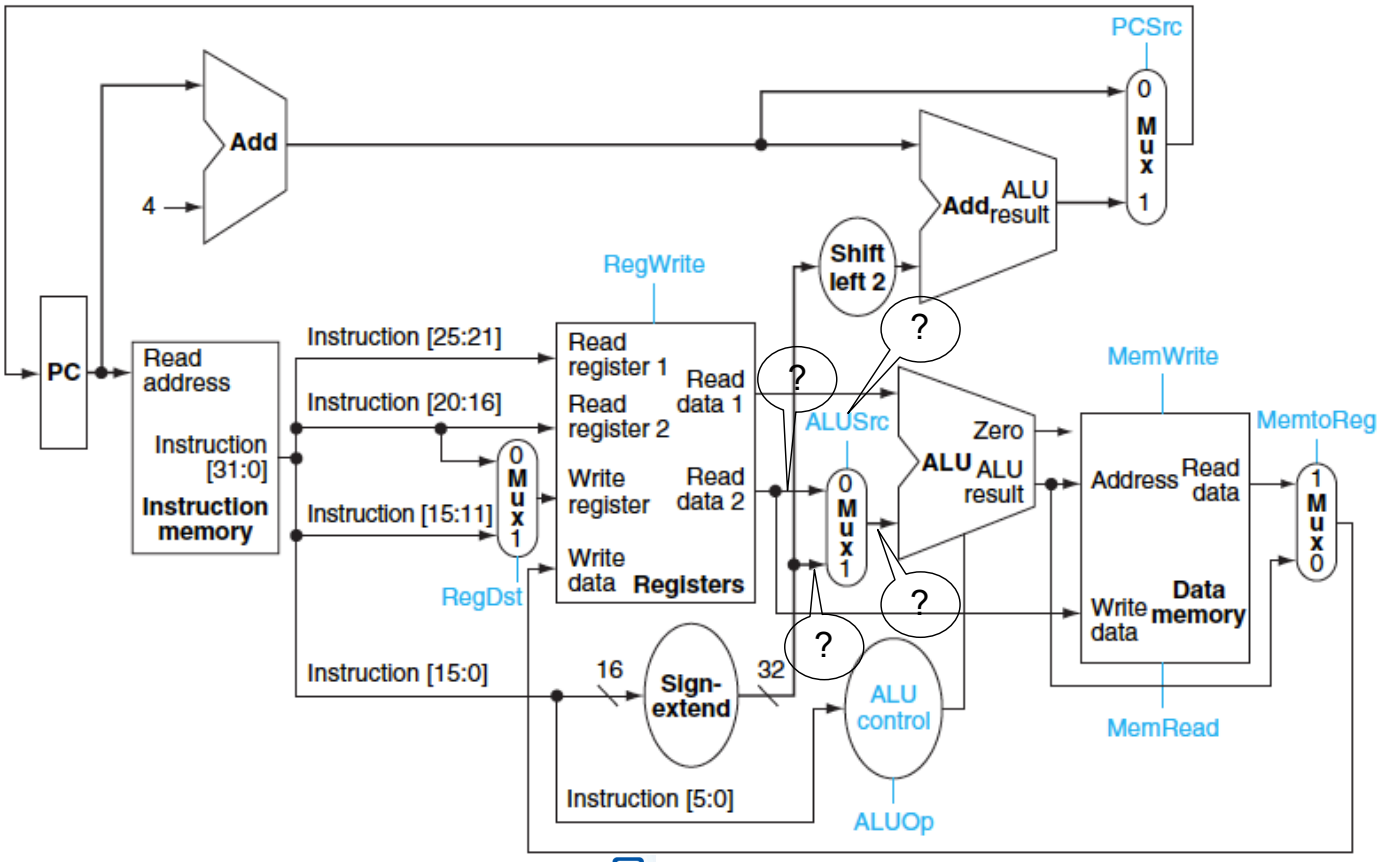
Complete para: lw $a2, 8($t1)



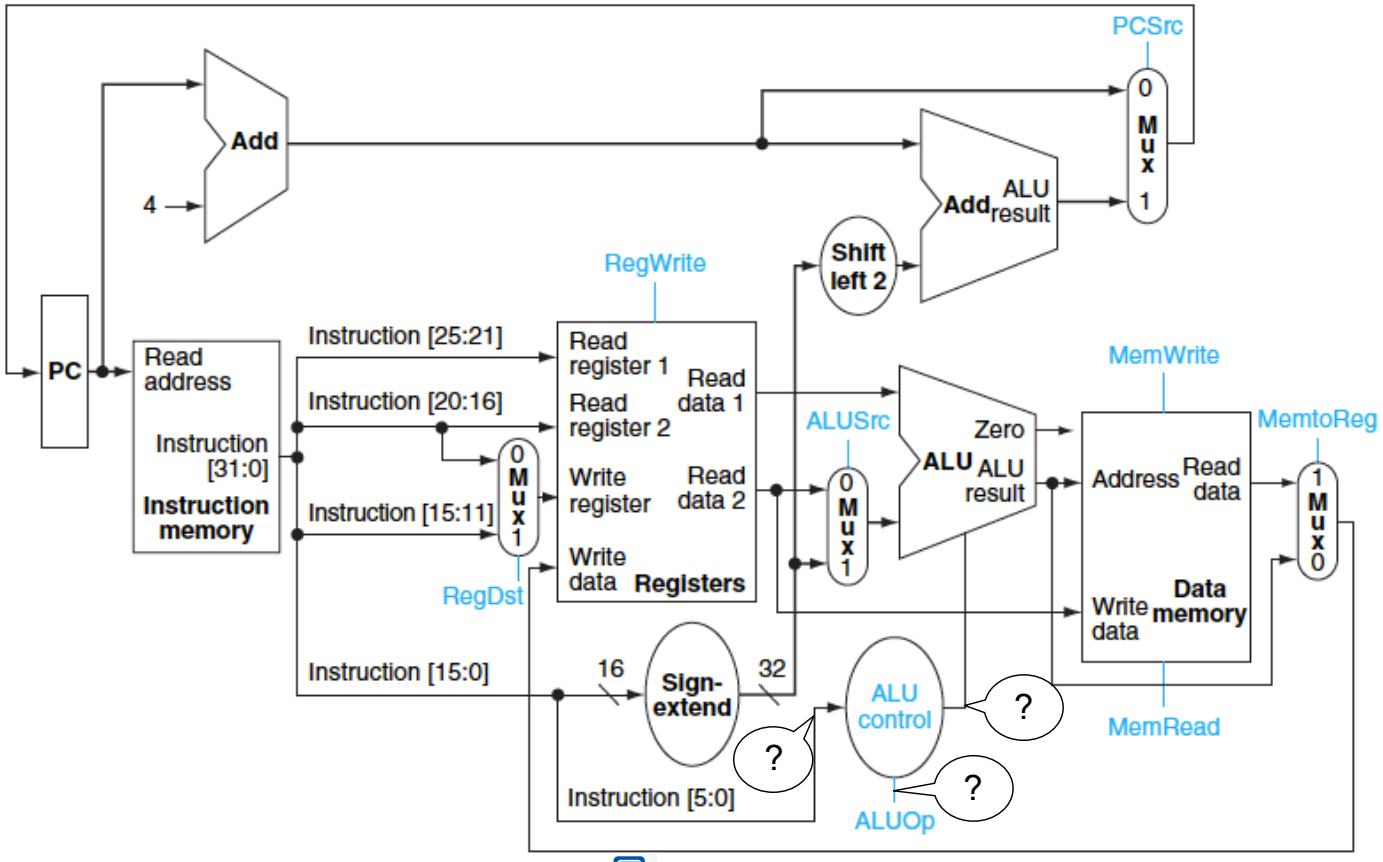
Complete para: lw $a2, 8($t1)



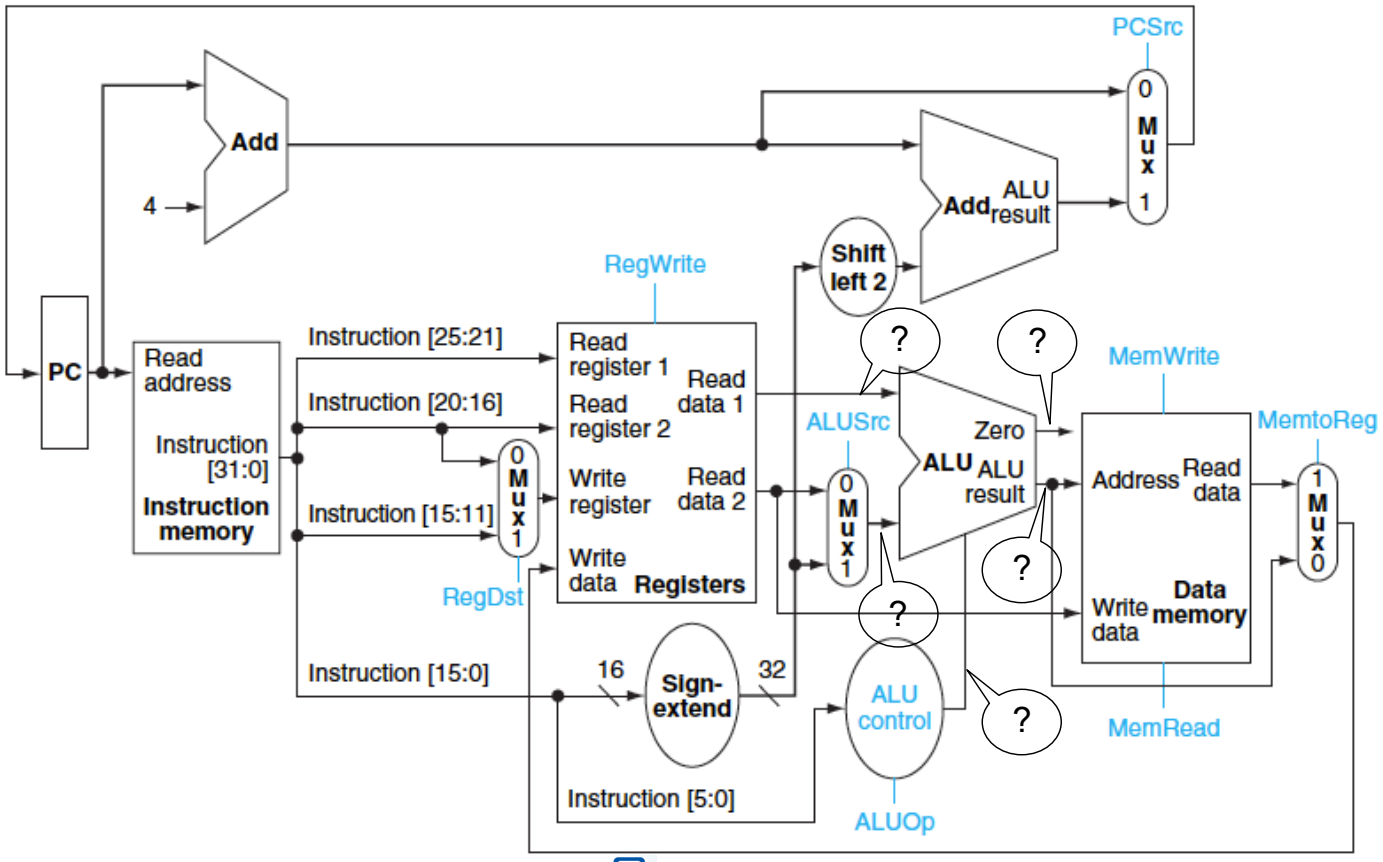
Complete para: lw $a2, 8($t1)



Complete para: lw $a2, 8($t1)



Complete para: lw $a2, 8($t1)



¿Qué vimos?

Principales Unidades Funcionales.

Camino de datos de los distintos tipos de instrucciones.

Estado y cambio de estado.

Sincronización mediante reloj.

Camino de datos y señales de control.

Ineficiencia de la implementación uniciclo