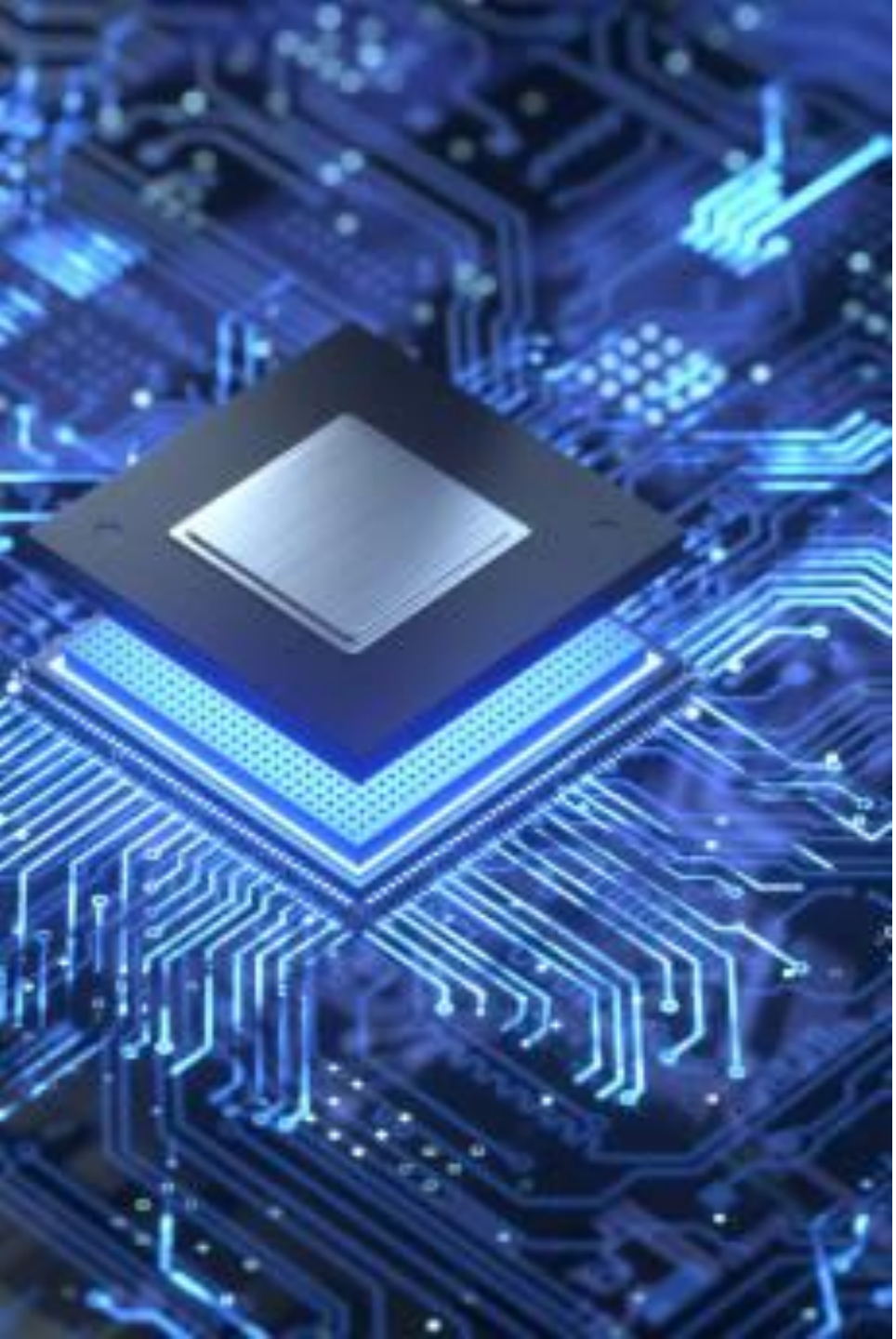


Tema 3

EL PROCESADOR

Índice

1. Introducción

- Recordar: rendimiento de un computador determinado por:
  - Recuento de instrucciones (RI)
    - Determinado por ISA y compilador
  - CPI y tiempo ciclo
    - Determinado por el hardware de la CPU
- Estudiaremos:
  - Versión simplificada monociclo
  - Versión segmentada más realista

1. Introducción

- Estudio simplificado: subconjunto de instrucciones del MIPS
  - Instrucciones de referencia a memoria (TIPO – I): lw, sw

```
lw $t3, 4($t1);   $t3 ← M[$t1+4])
sw $t0, 48($s3);  M[48+$s3] ← $t0
```

  - Instrucciones aritmético-lógicas (TIPO – R): add, sub, and, or, slt

```
add $t0, $s2, $t0 ; $t0 ← $s2 + $t0
slt $t0, $s3, Ss4 ; Si ($s3<$s4) entonces $t0=1 sino $t0=0
```

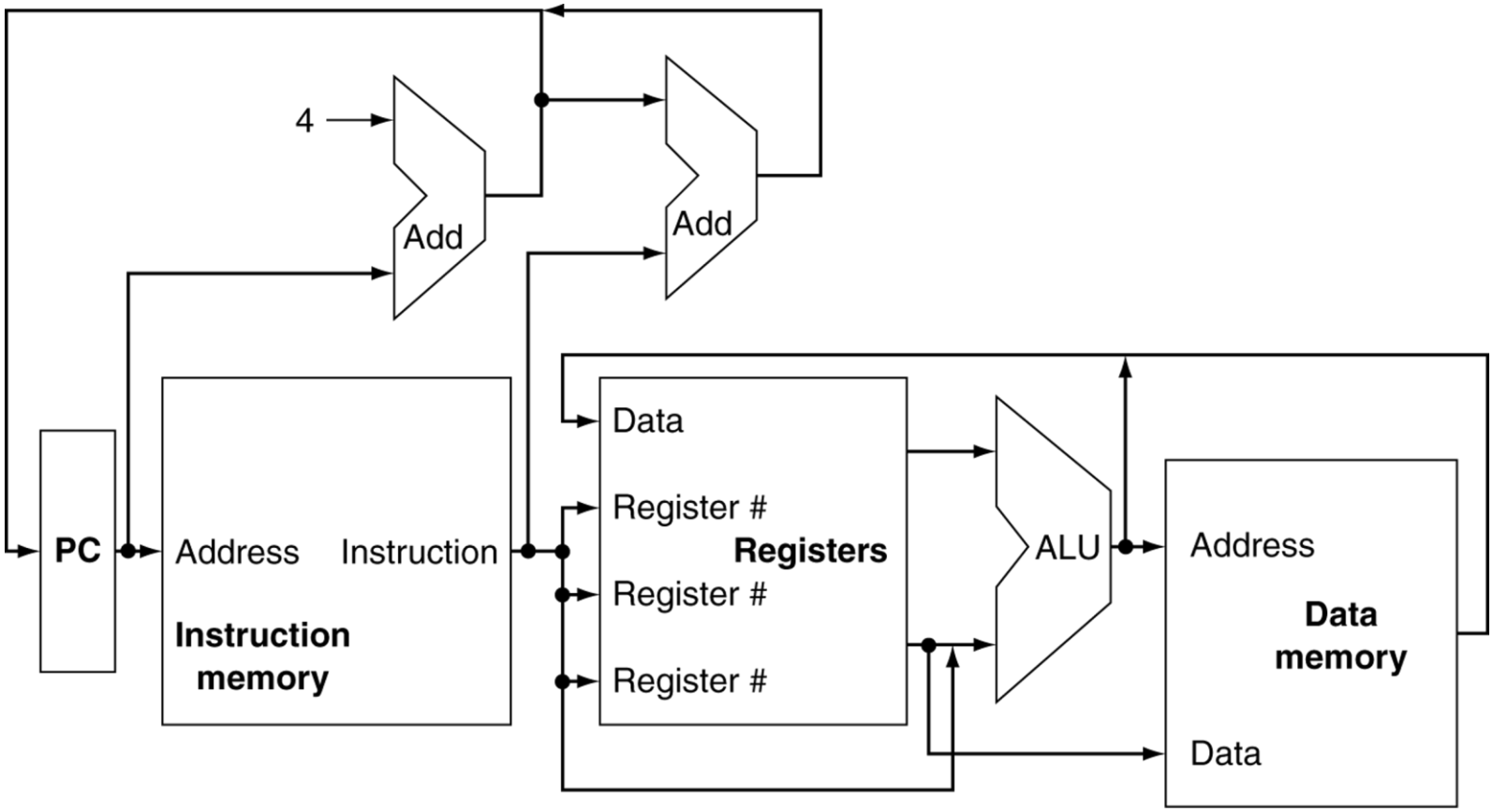
  - Instrucciones de control (TIPO – I o TIPO – J): beq, j

```
beq $t0, $zero, Salto ; Si $t0=$zero entonces ir a Salto
j Bucle ; Ir a Bucle
```

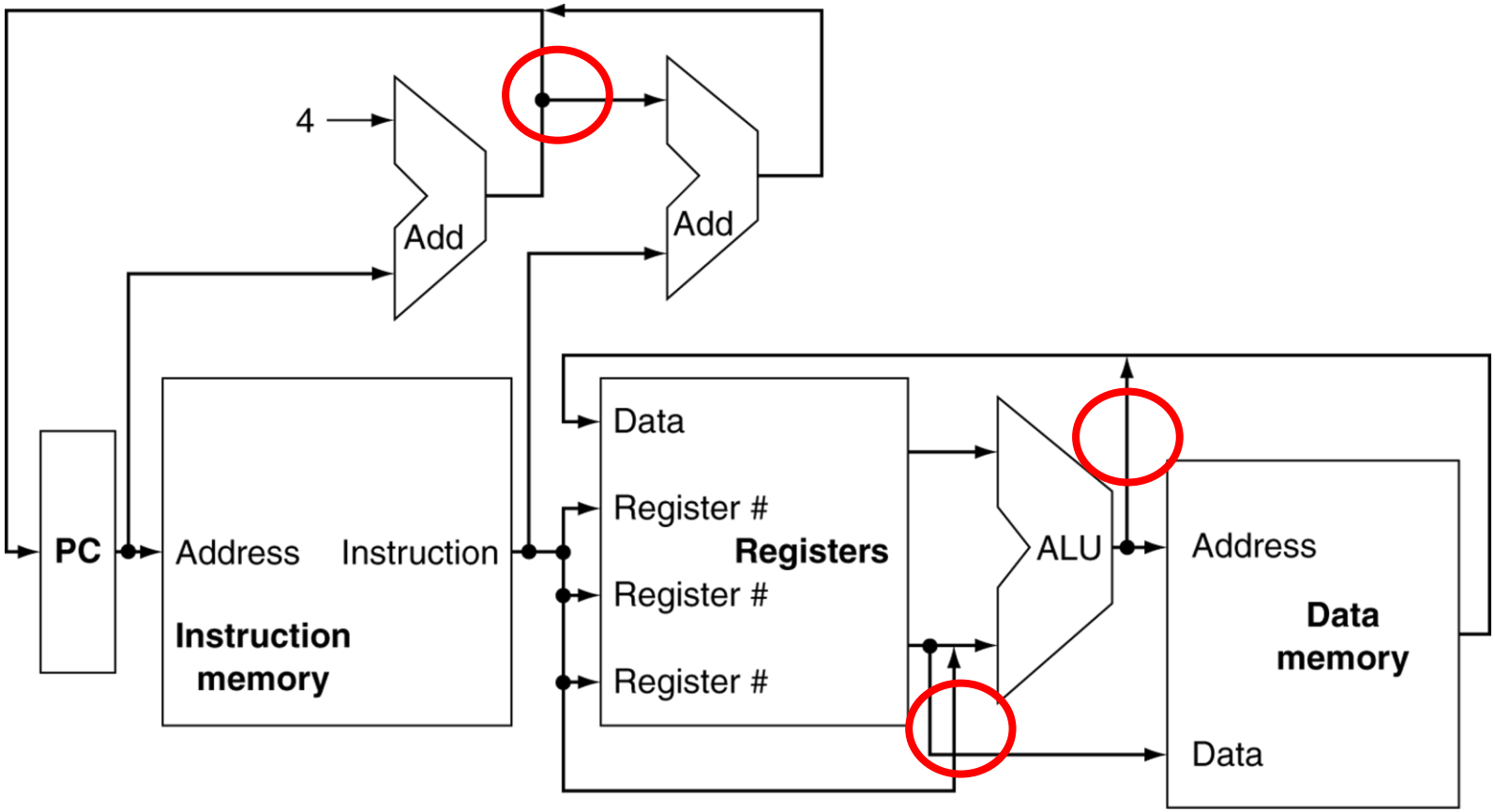
Ejecución de una instrucción

- El contador de programa (PC) proporciona la dirección de memoria donde se encuentra la instrucción. Se obtiene la instrucción.
- Se leen los registros.
- Dependiendo del tipo de instrucción
  - Se usa la ALU para calcular
    - Resultado aritmético
    - Dirección de memoria para load/store
    - Se calcula la dirección de salto
  - Se accede a memoria de datos para load/store
- Actualizar PC con la dirección de la siguiente instrucción (dirección de salto o PC+4)

Descripción general monociclo



Descripción general monociclo

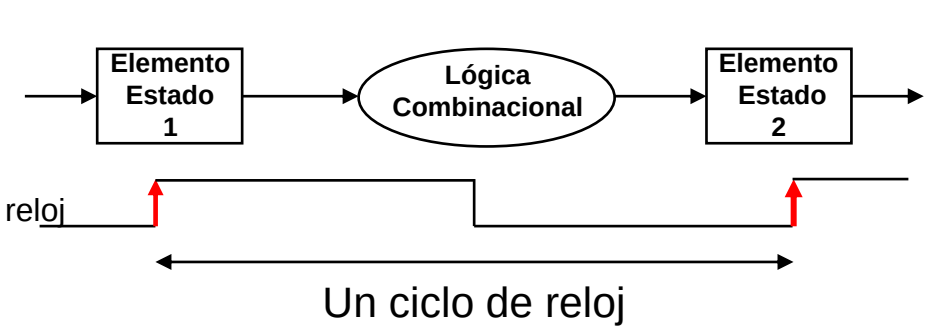


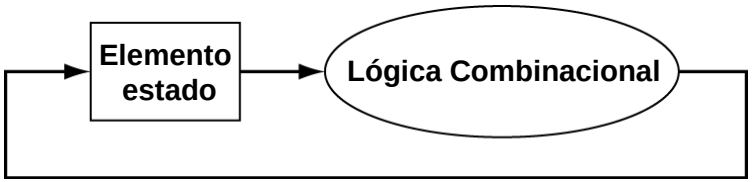
2. Convecciones de diseño lógico

- Dos tipos de unidades funcionales:
  - Elementos combinacionales
    - Operan con datos (ALU, multiplexores…)
  - Elementos secuenciales
    - Contiene estado (Registros)
    - Utilizan señal de reloj para determinar cuándo actualizar dato almacenado
    - Disparo por flanco de subida

Metodología de sincronización

- Ejecución típica:
  - Leer contenido de algún elemento de estado
  - Enviar valores a través de la lógica combinacional
  - Escribir resultado en un elemento de estado
- El retardo más largo determina el periodo de reloj





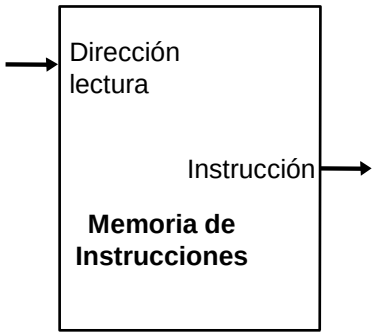
3. Construcción de la ruta de datos monociclo

- Ruta de datos:
  - Elementos que procesan datos y direcciones en la CPU
    - Registros, ALUs, multiplexores, memorias,…
- Construiremos la ruta de datos de forma incremental

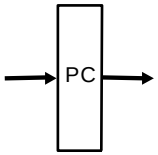
Búsqueda de la instrucción

- Acciones involucradas:
  - Obtener la instrucción de la memoria de instrucciones
  - Actualizar el valor del PC para que apunte a siguiente instrucción en secuencia

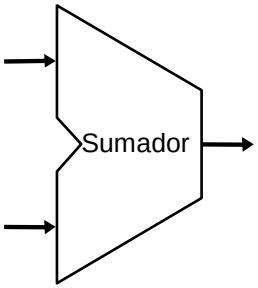
Elementos de la ruta de datos:



Memoria de Instrucciones

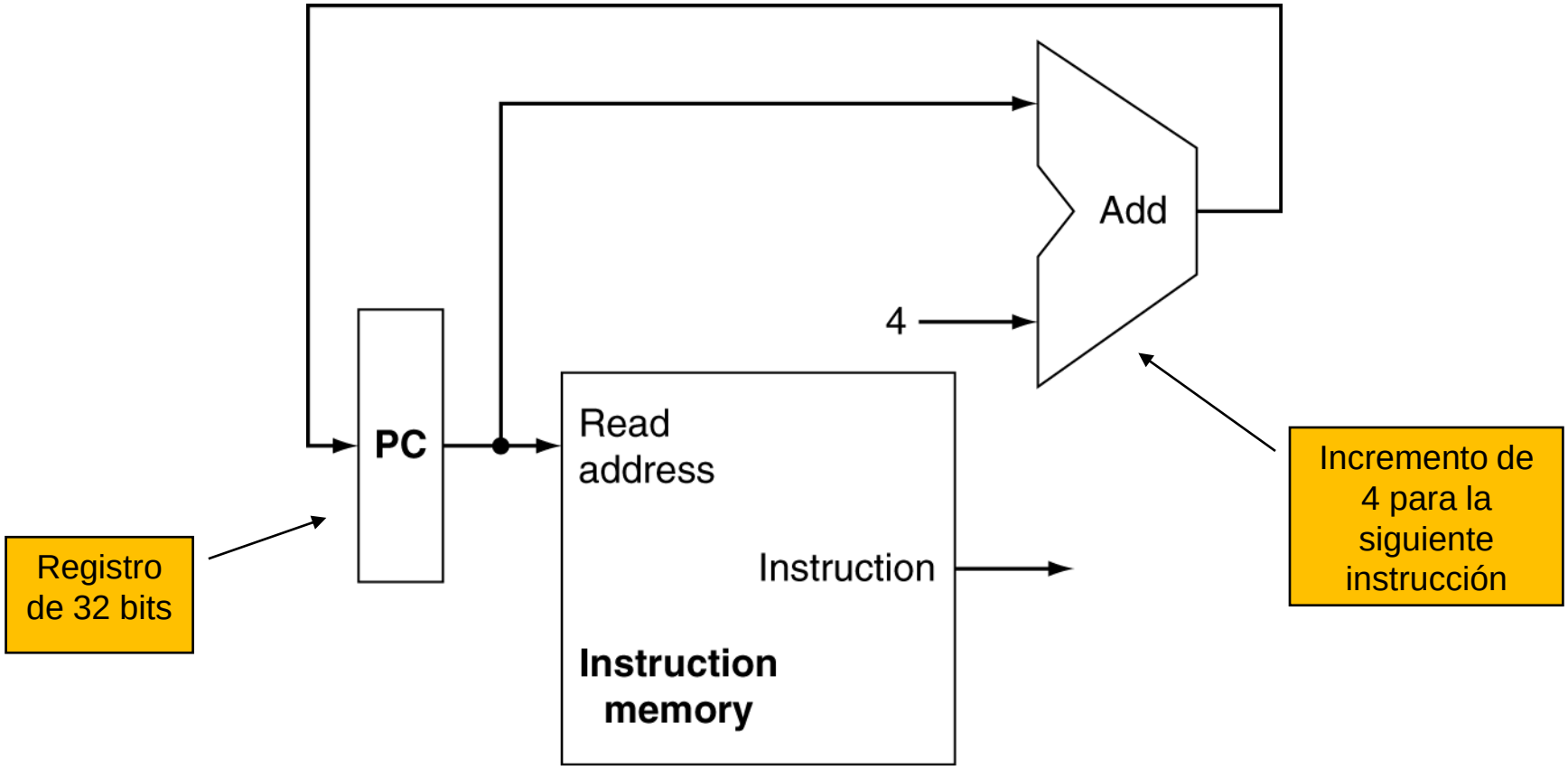


Contador del programa



Sumador

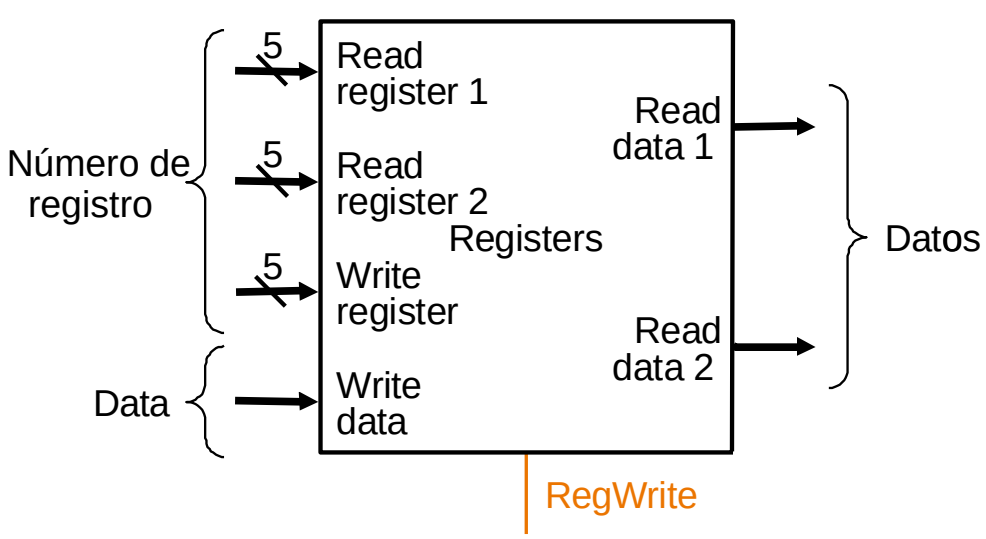
Búsqueda de la instrucción



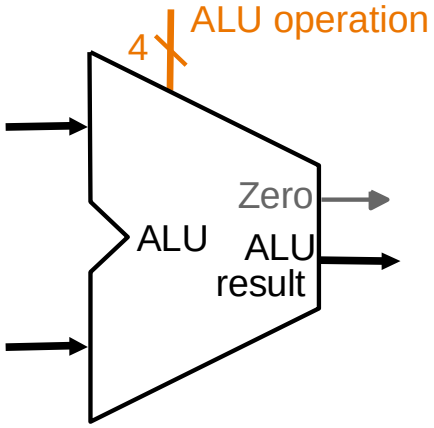
Instrucciones tipo R

- Acciones involucradas:
  - Leer dos registros operandos
  - Realizar la operación aritmética o lógica
  - Escribir en el registro resultado

add $t0, $s1, $s2
($8← $17+$18)



Registros

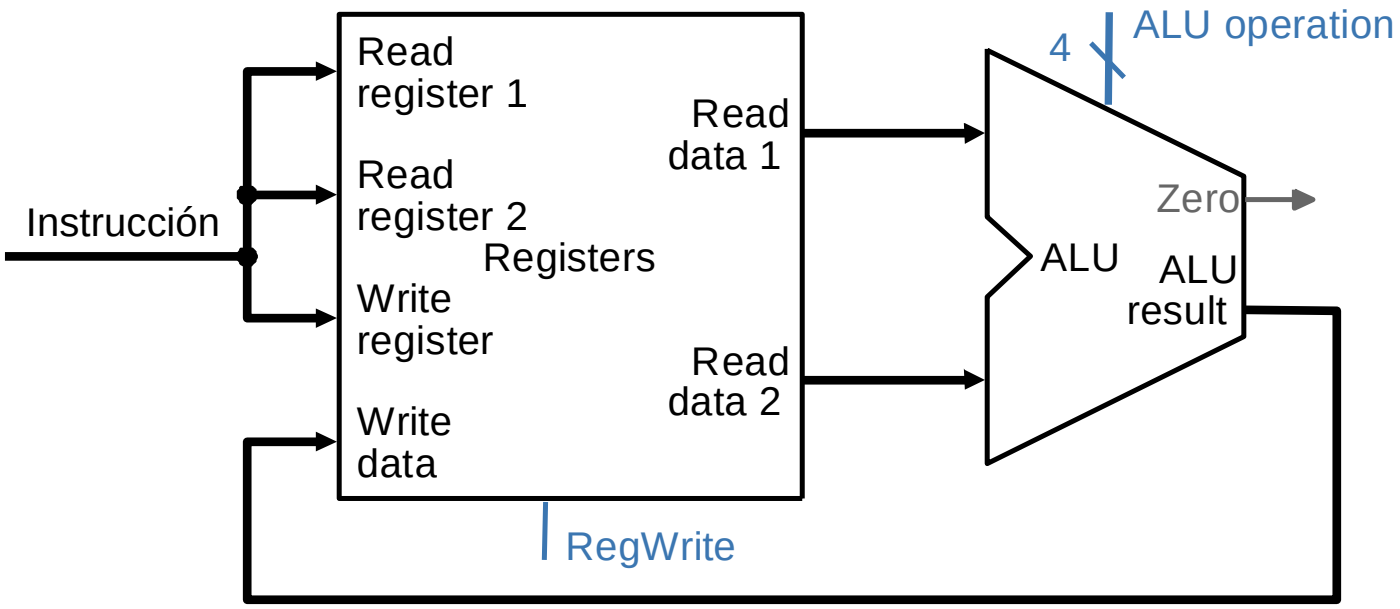


ALU

Instrucciones tipo R

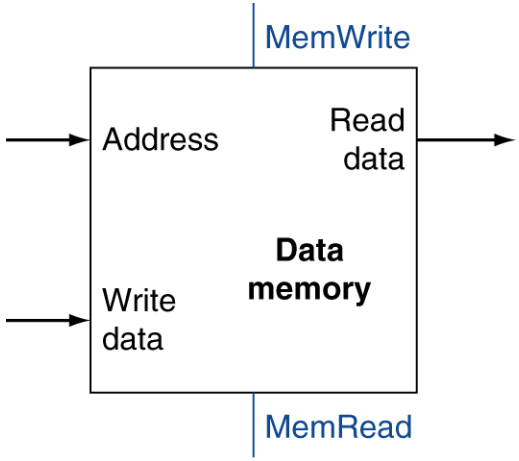
- Acciones involucradas:
  - Leer dos registros operandos
  - Realizar la operación aritmética o lógica
  - Escribir en el registro resultado

add $t0, $s1, $s2
($8← $17+$18)

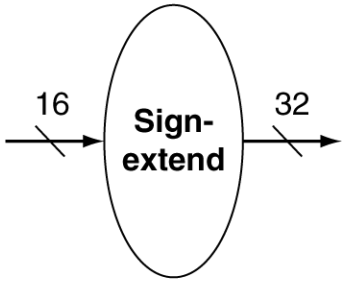


Instrucciones tipo I - lw y sw

- Acciones involucradas:
  - Leer registros operandos
  - Calcular dirección de memoria utilizando el desplazamiento de 16 bits (ALU y unidad de extensión de signo)
  - Carga (lw): escribir en registro el dato leído en memoria
  - Almacenamiento (sw): escribir en la memoria el registro leído



Memoria de datos



Unidad de extensión de signo

Carga de memoria:

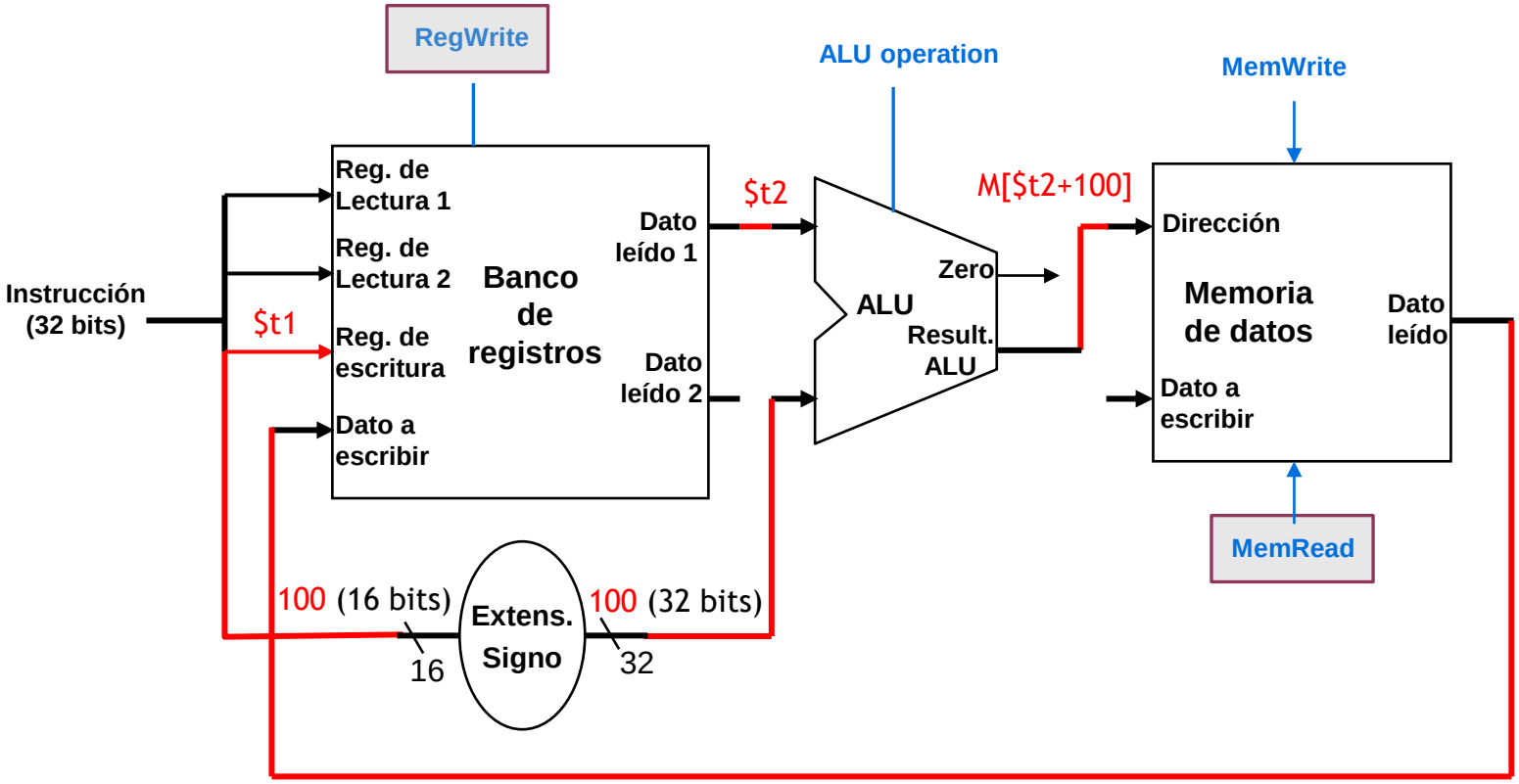
lw $t1, 100($t2) ($t1← M[$t2+100

Almacenamiento en memoria:

sw $t0, 48($s3); M[48+$s3] ← $t0

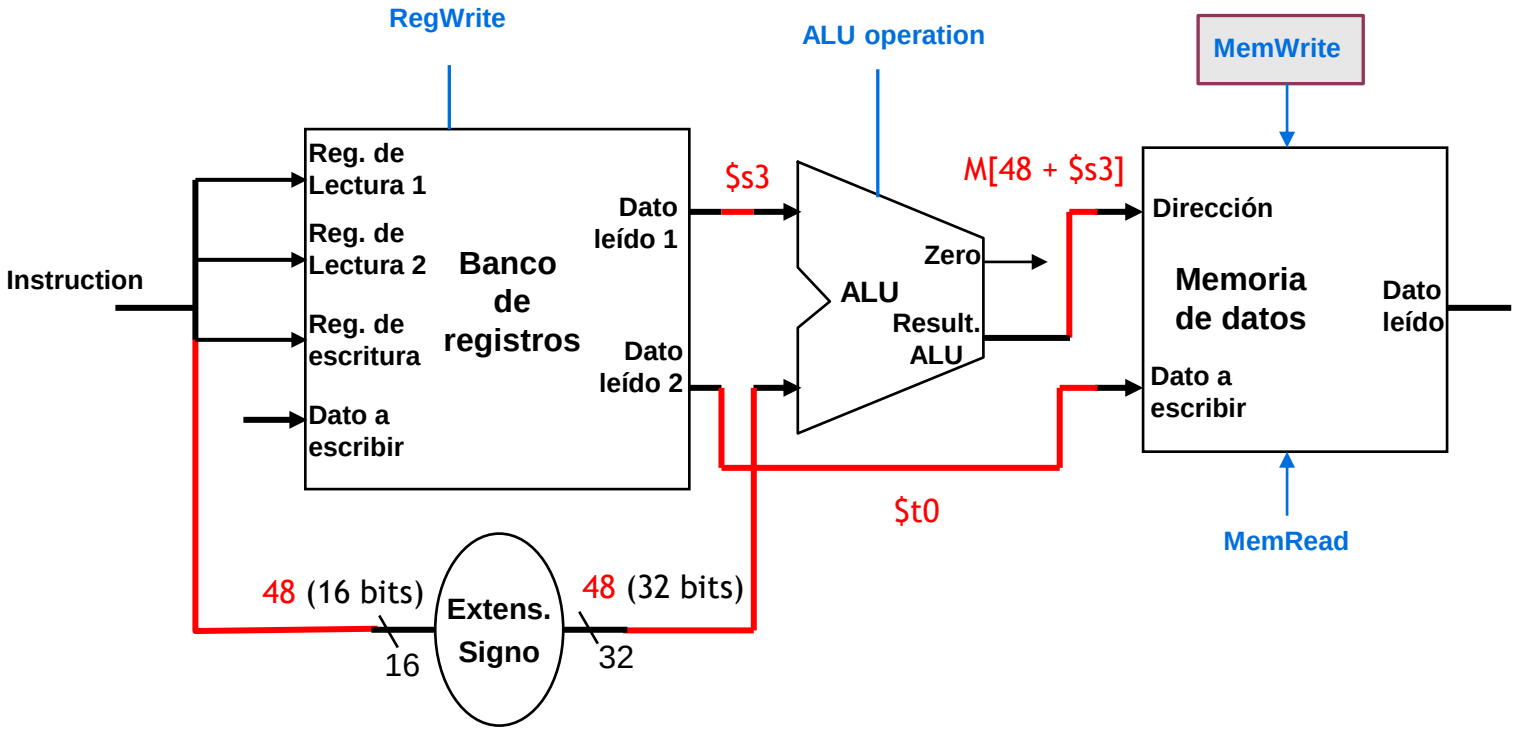
Carga desde memoria- lw

lw $t1, 100($t2); ($t1← M[$t2+100])



Almacenamiento en memoria - sw

sw $t0, 48($s3); M[48+$s3] ← $t0

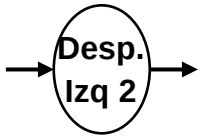


Instrucciones tipo I – beq

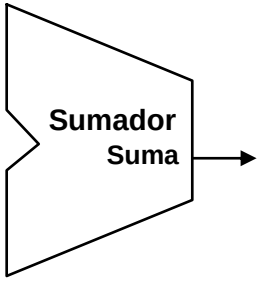
- Acciones involucradas:
  - Leer registros operandos
  - Comparar operandos (usar AU para restar y chequear el indicador Cero)
  - Calcular la dirección de salto:
    - Extensión de signo del campo desplazamiento
    - Desplazar dos bits a la izquierda
    - Sumar la dirección a PC+4 calculado en la búsqueda de la instrucción

Salto condicional:

beq $t1, $t2, 100 (si $t1=$t2 entonces PC ← PC+4+100)



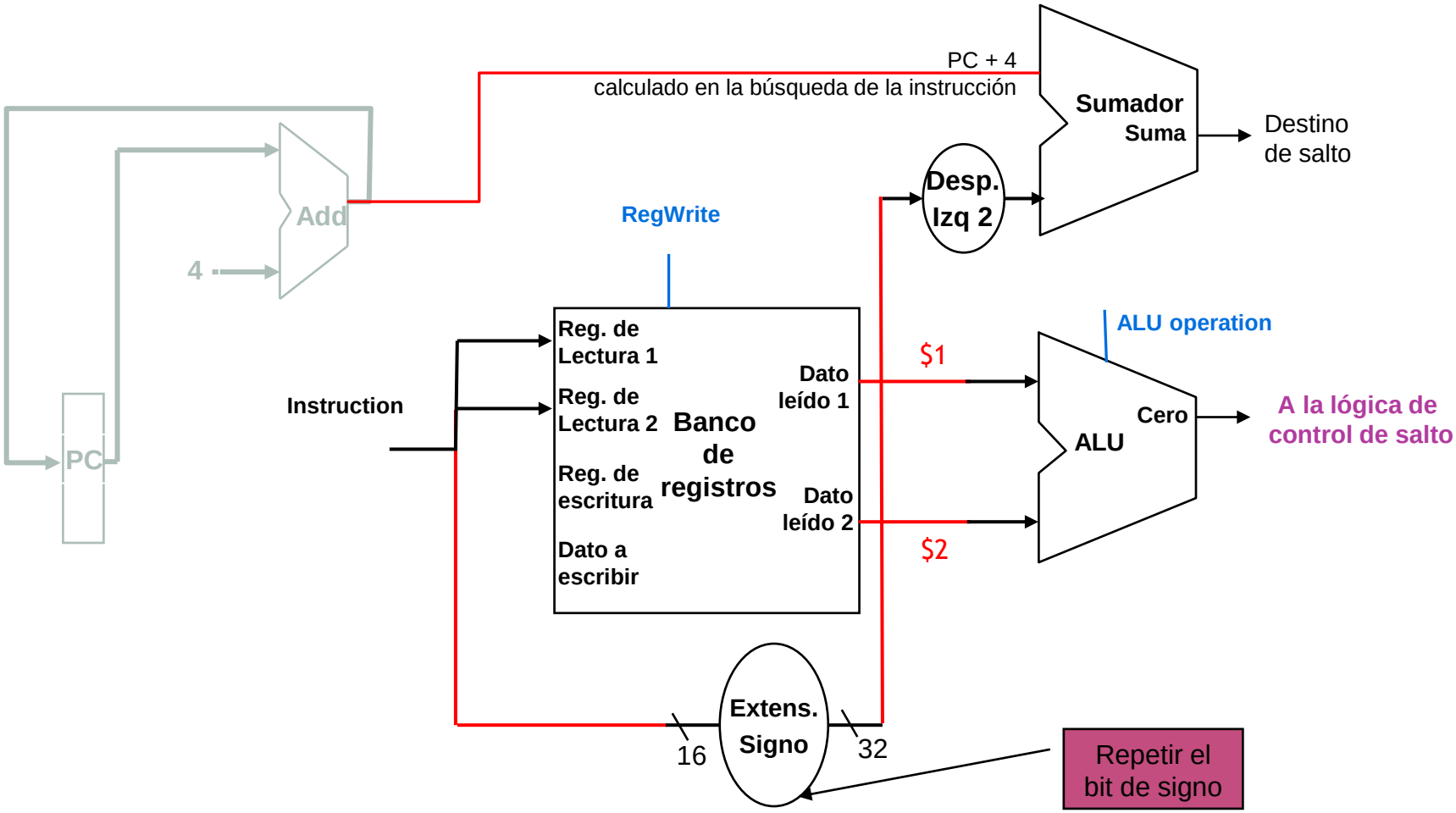
Desplazamiento a la izquierda



Sumador

Instrucciones tipo I – beq

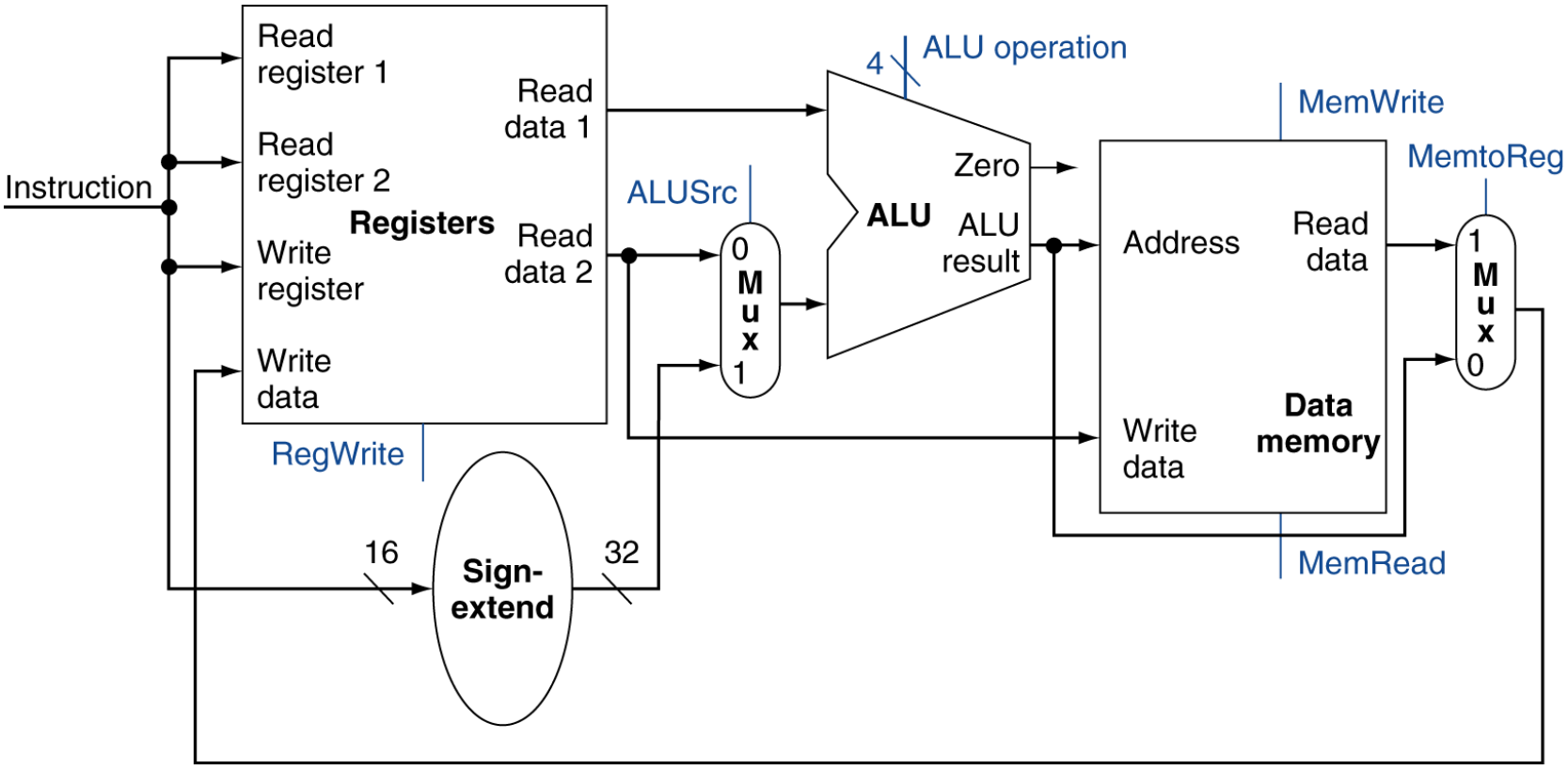
beq $t1, $t2, 100 (si $t1=$t2 entonces PC ← PC+4+100)



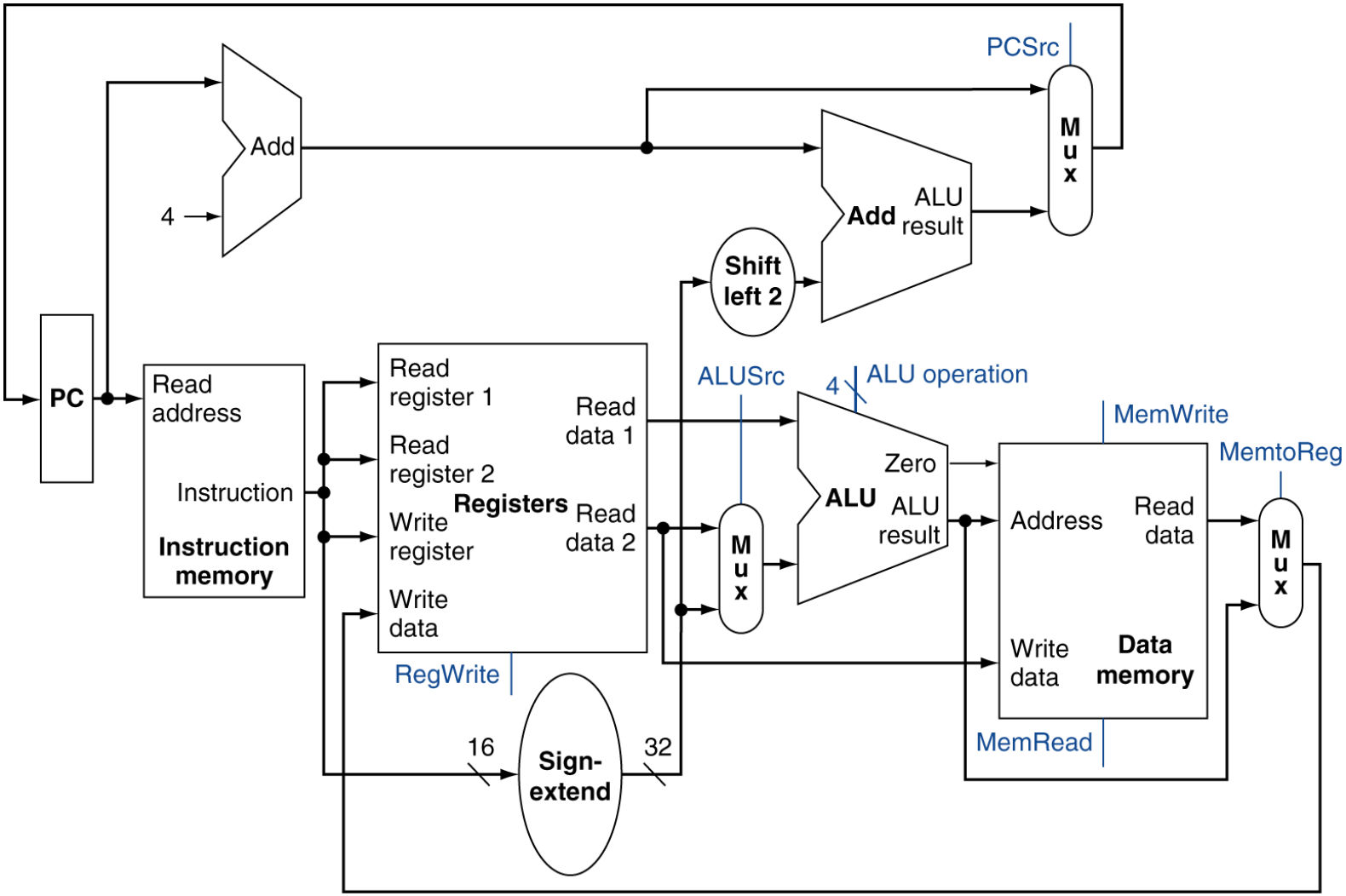
Componer la ruta de datos

- Combinar segmentos de la ruta de datos y añadir las líneas de control y los multiplexores necesarios
- Diseño monociclo: las instrucciones se ejecutan en un ciclo de reloj
  - Ningún elemento de la ruta de datos puede utilizarse más de una vez por instrucción (duplicidad de elementos (memorias separadas para instrucciones y datos, varios sumadores…)
  - Utilizar multiplexores para compartir elementos con distintas entradas para instrucciones diferentes
  - La duración del ciclo de reloj estará determinada por la instrucción más lenta.

Para instrucciones tipo R/Load/Store



Ruta de datos completa



El control de la ALU

- Señales de control a la ALU:

ALU operation (4 bits)	Operación
0000	AND
0001	OR
0010	suma
0110	resta
0111	Activar si menor que
1100	NOR

- La ALU se utiliza para:
  - Instrucciones lw/sw: **SUMA** → **ALU operation =0010**
  - Instrucciones de salto: **RESTA** → **ALU operation =0110**
  - Instrucciones Tipo R: **DEPENDE DEL CAMPO FUNCT**
    - **Ejemplo: add $8, $17, $18**

op	rs	rt	rd	shamt	funct
000000	10001	10010	01000	00000	100000

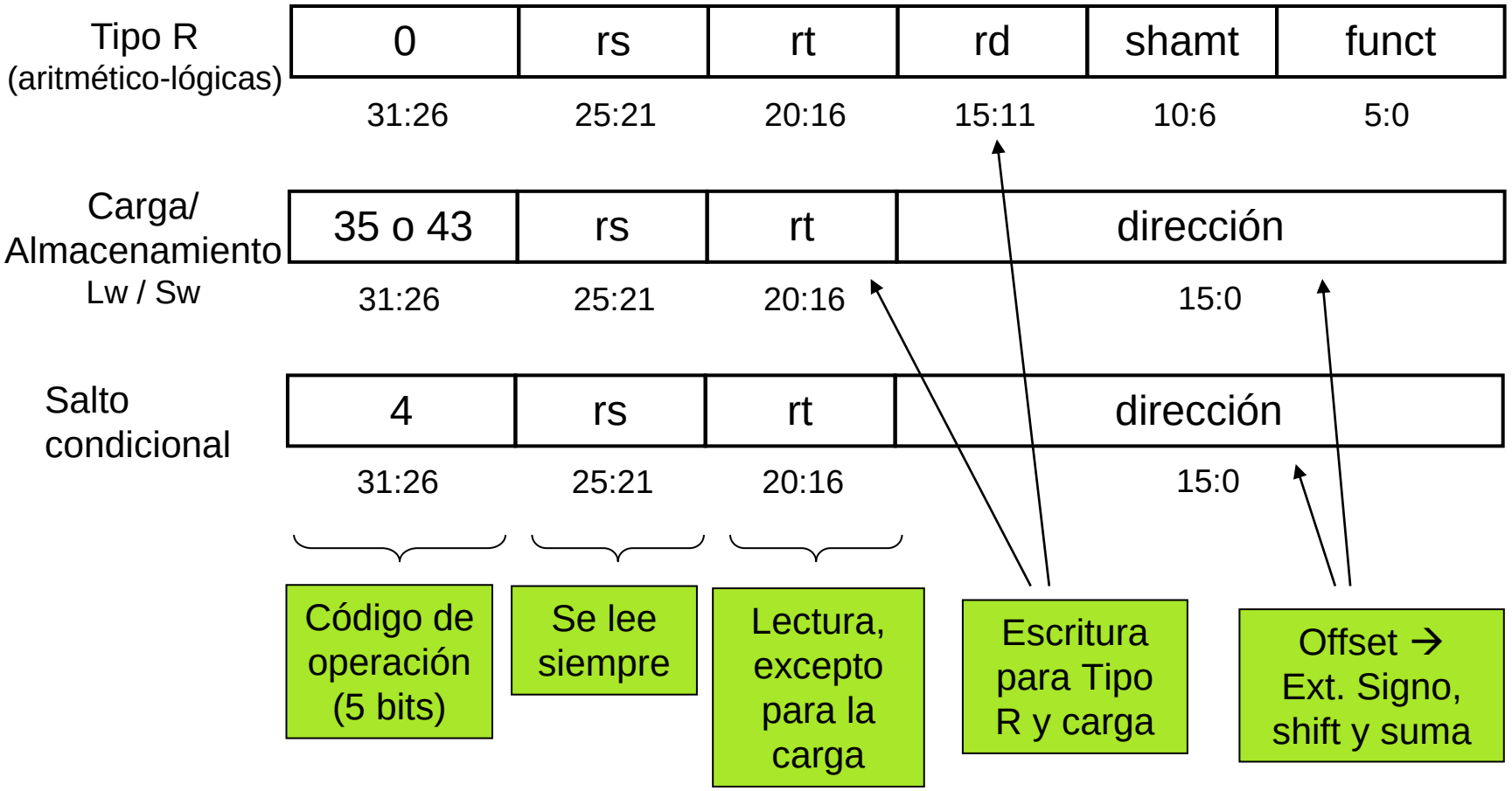
El control de la ALU

- Suponer la señal **ALUOp** de dos bits derivada del código de operación que identifique el tipo de instrucción
- Dos niveles de control: Control principal y Control ALU (reduce la complejidad del controlador principal y se incrementa la velocidad
- El control principal genera la señal ALUOp a partir del código de operación
- Un circuito combinacional obtendrá la entrada de control a la ALU

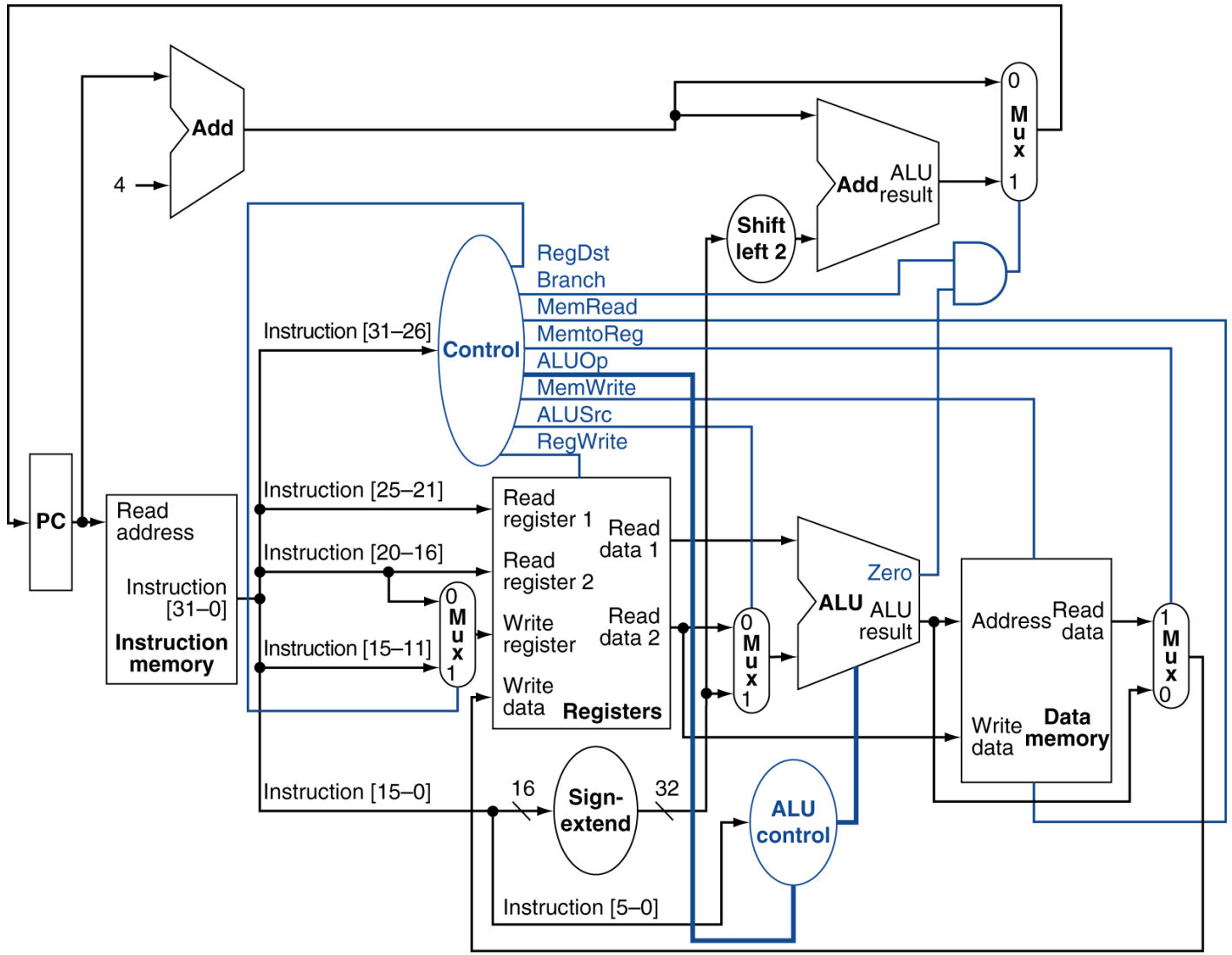
Instrucción	ALUOp	Operación	Campo funct	Acción de la ALU	Entrada a la ALU
lw	00	Cargar palabra	XXXXXX	Suma	0010
sw	00	Almacenar palabra	XXXXXX	Suma	0010
beq	01	Saltar si igual	XXXXXX	Resta	0110
R-type	10	Suma	100000	Suma	0010
		Resta	100010	Resta	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		Activar si menor que	101010	Activar si menor que	0111

La unidad de control principal

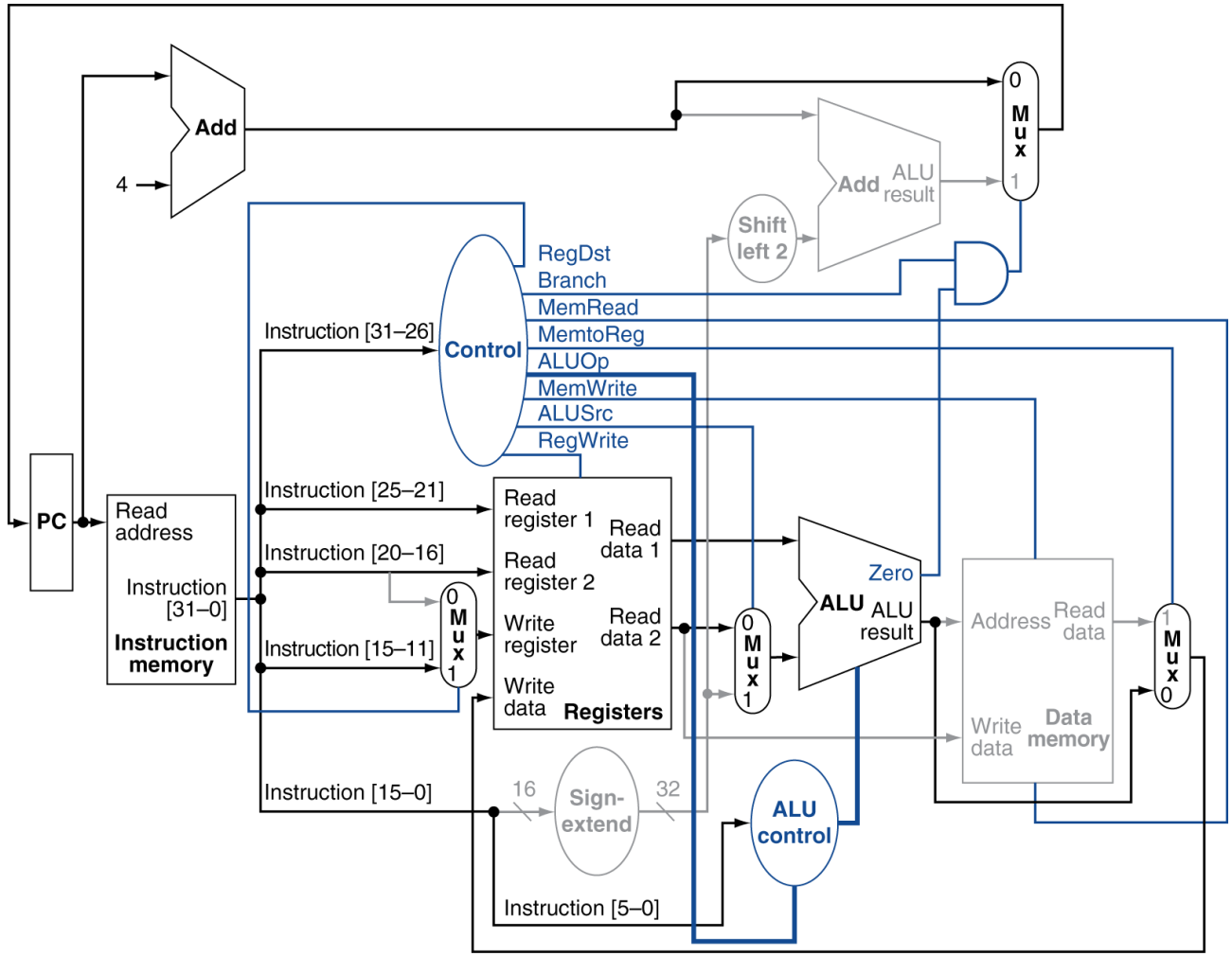
- Las señales de control se obtienen de las instrucciones:



Ruta de datos y control

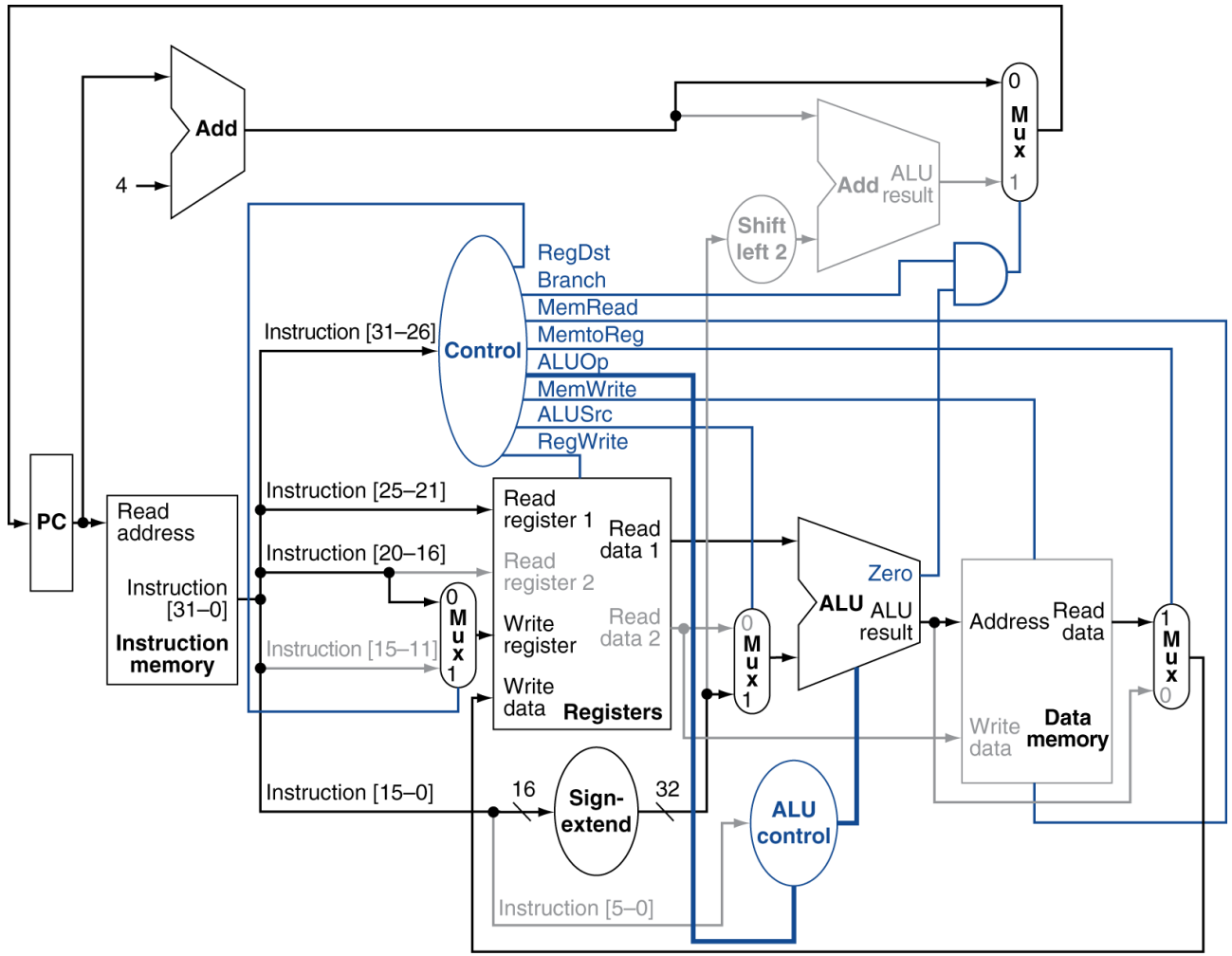


Instrucciones formato tipo R



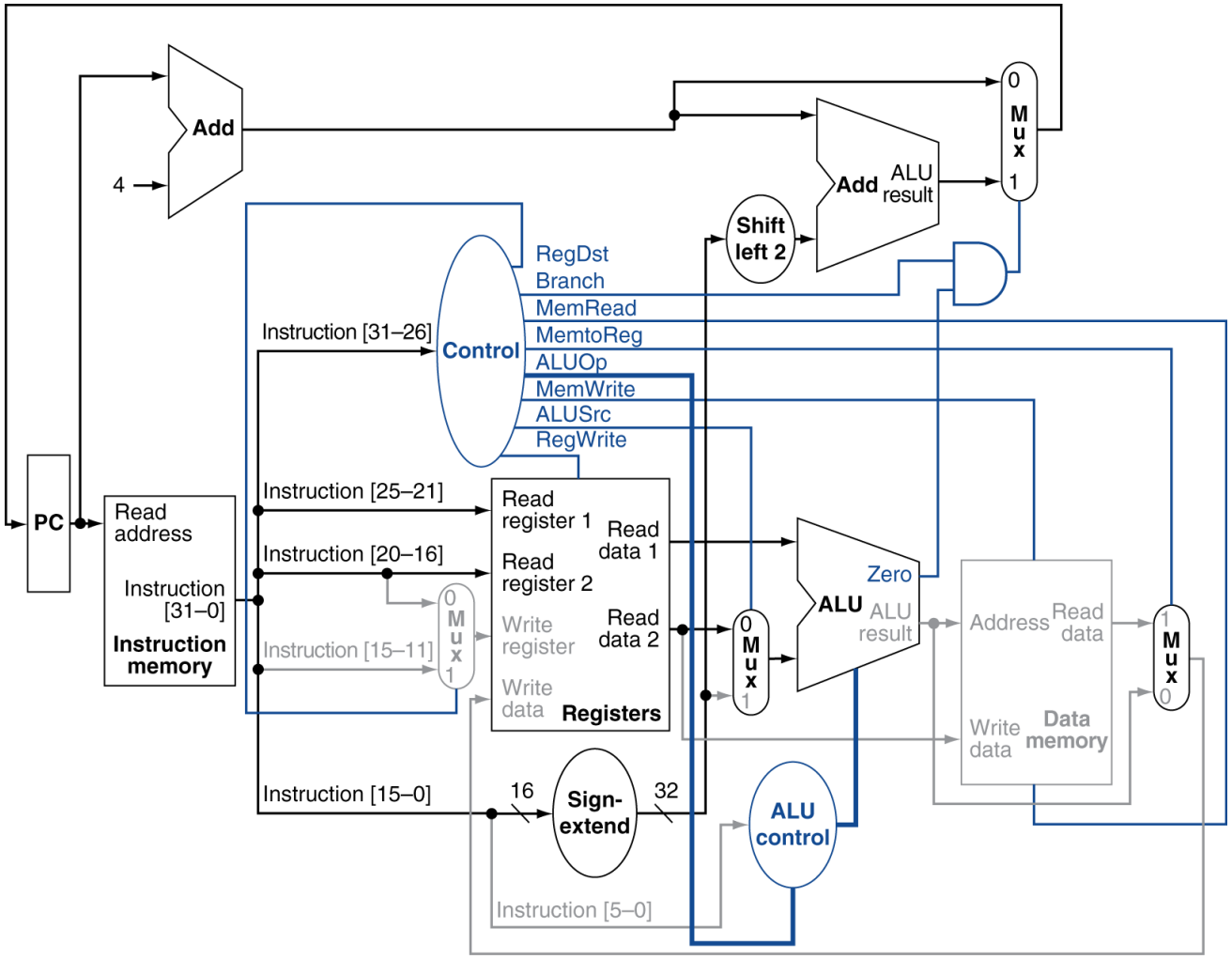
Instrucción	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0
Formato R	1	0	0	1	0	0	0	1	0

Instrucción de carga- lw



Instrucción	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0
Lw	0	1	1	1	1	0	0	0	0

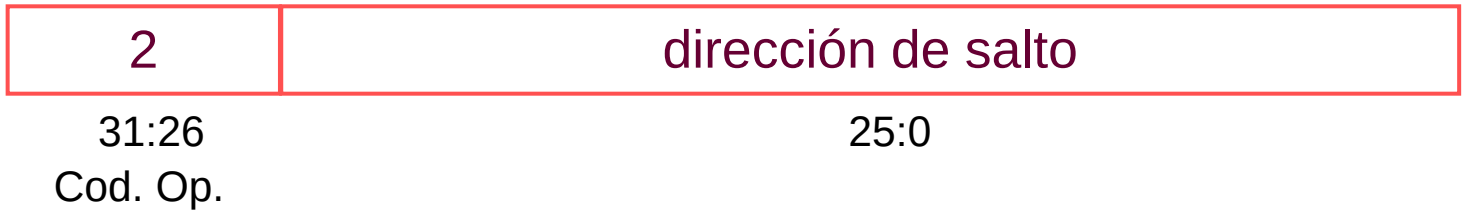
Instrucción de saltar si igual - beq



Instrucción	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUp0
Beq	X	0	X	0	0	0	1	0	1

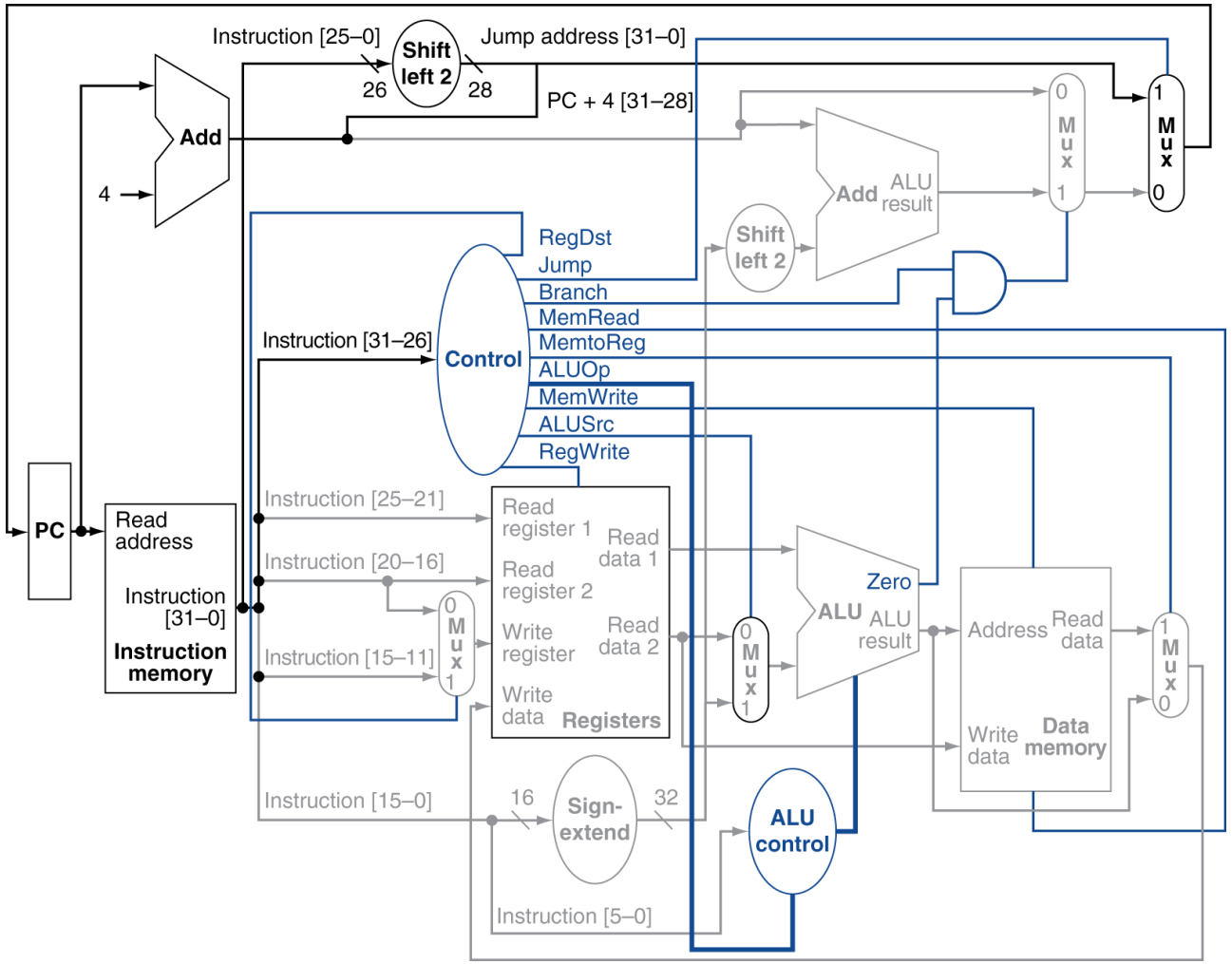
Añadir instrucción de salto incondicional - j

- Formato de la instrucción:



- La dirección se forma con la concatenación de:
  - 4 bits superiores del PC actualizado (PC+4 calculado en la búsqueda de la instrucción)
  - Los 26 bits de menor peso de la instrucción
  - 00
- Habrá que añadir una nueva señal de control que se active con la instrucción: ***jump***

Instrucción de salto incondicional - j

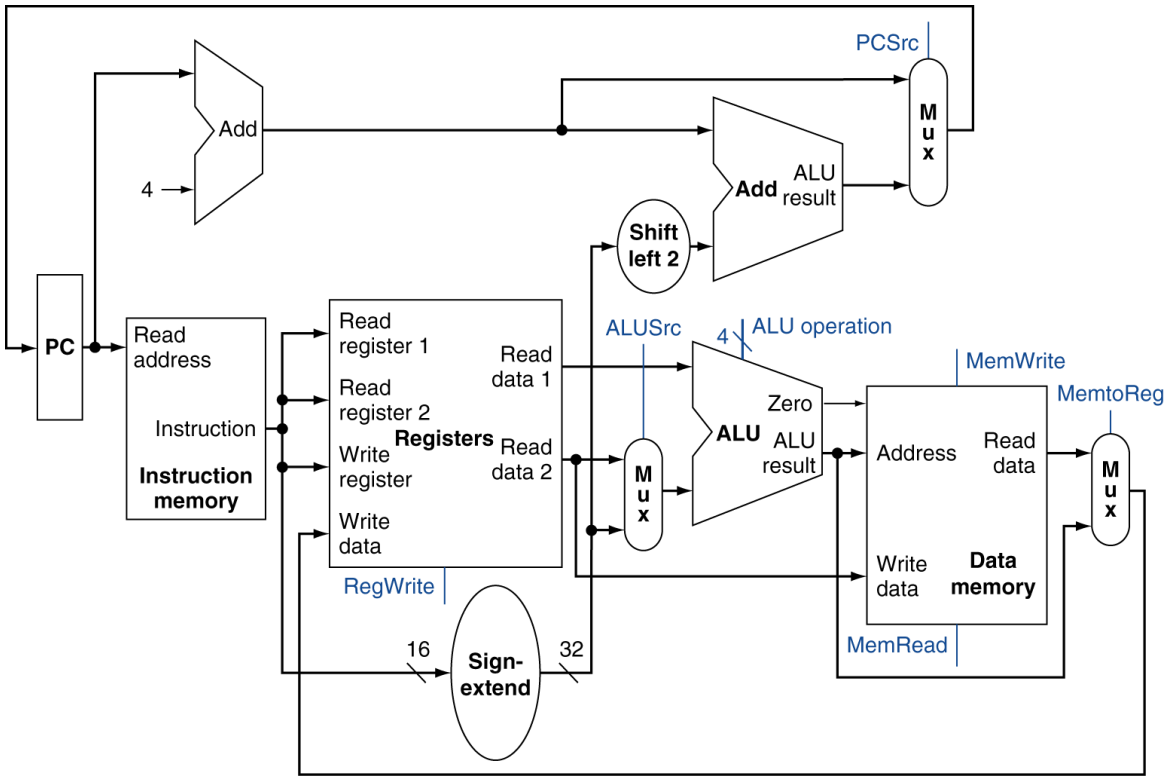


Instrucción	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Jump	Branch	ALUOp1	ALUp0
J	X	X	X	0	0	0	1	0	X	X

Ejercicio

- Calcular el tiempo de ciclo suponiendo retardos despreciables para todos los elementos de la ruta de datos monociclo excepto para:

Acceso a memoria (2 ns), ALU y sumadores (1 ns), acceso al banco de registro (0.5 ns)



Ejercicio (solución)

- Se toman como datos: Acceso a memoria (2 ns), ALU y sumadores (1 ns), acceso al banco de registro (0.5 ns), el resto de los elementos se supone que no cuentan.

Tipo Instr.	Unidades funcionales utilizadas					Total
Tipo - R						
Lw						
Sw						
Salto Cond.						
Salto Incond.						

Ejercicio (solución)

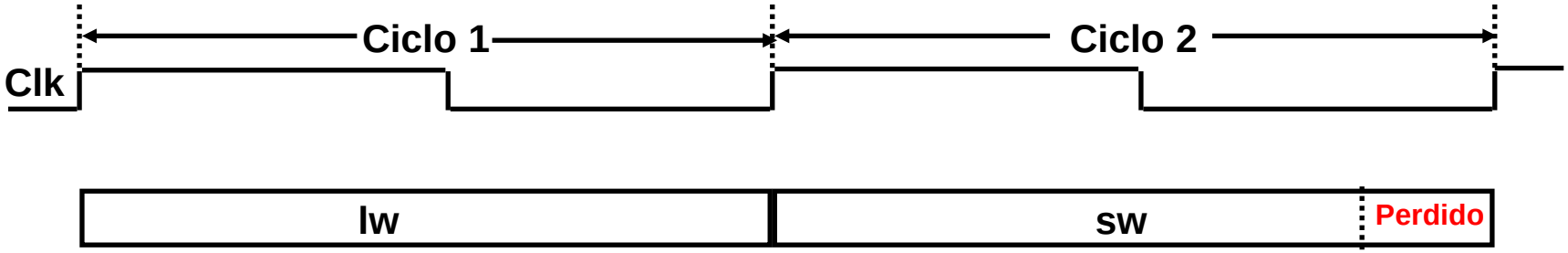
- Se toman como datos: Acceso a memoria (2 ns), ALU y sumadores (1 ns), acceso al banco de registro (0.5 ns), el resto de los elementos se supone que no cuentan.

- **El ciclo de reloj deberá ser de 6ns**

Tipo Instr.	Unidades funcionales utilizadas					Total
Tipo - R	Buscar Instr. (2 ns)	Acceso Reg. (0.5 ns)	ALU (1 ns)	Acceso Reg. (0.5 ns)		4 ns
Lw	Buscar Instr. (2 ns)	Acceso Reg. (0.5 ns)	ALU (1 ns)	Acceso memoria (2 ns)	Acceso Reg. (0.5 ns)	6 ns
Sw	Buscar Instr. (2 ns)	Acceso Reg. (0.5 ns)	ALU (1 ns)	Acceso memoria (2 ns)		5.5 ns
Salto Cond.	Buscar Instr. (2 ns)	Acceso Reg. (0.5 ns)	ALU (1 ns)			3.5 ns
Salto Incond.	Buscar Instr. (2 ns)					2 ns

Problemas de rendimiento

- La duración del ciclo de reloj determinado por la instrucción más lenta (lw):



  - ¿Qué ocurriría si tuviéramos una instrucción muy complicada, por ejemplo, una operación en coma flotante?

- No es viable duraciones del ciclo de reloj distintas para distintas instrucciones
- Mejoraremos el rendimiento mediante segmentación