El procesador

Ruta de datos y de control

Katia Leal Algara, URJC
Juan González Gómez, URJC

Objetivos

- Principios y técnicas utilizadas en la implementación de un procesador
  - Ruta de datos
  - Ruta de control
- Examinaremos dos implementaciones RISC-V
  - Procesador monociclo
  - Procesador multiciclo
- Nos centraremos en el estudio de una arquitectura RISC-V simplificada, el ***nanoRISC-V***
  - Memory reference: **`ld (I)`**, **`sd (I)`**
  - Arithmetic/logical: **`add`**, **`sub`**, **`and`**, **`or (R)`**
  - Control transfer: **`beq (SB)`**

Ejecución de una instrucción

- PC → memoria de instrucciones (MI), buscar siguiente instrucción
- Número de registro → banco de registro (BR), lectura de operandos
- Dependiendo de la clase de instrucción
  - Uso de la ALU para calcular
    - Resultado aritmétido
    - Dirección de memoria para load/store
    - Comparación de registros
  - Acceso a la memoria de datos (MD) paara load/store
  - PC ← dirección destino de salto o PC + 4

Ejecución de una instrucción

- **Fetch (IF)**: buscar en memoria la instrucción apuntada por el PC. Actualización del PC
- **Decode (ID)**: decodificación de la instrucción, separar los diferentes campos. Si es necesario, se leen 1 o 2 operandos de los registros
- **Execution (EX)**: ejecución de la operación indicada en el opcode
- **Memory Access (MEM)**: si es necesario, acceder a memoria para leer o escribir
- **Writeback (WB)**: si es necesario, se vuelca un resultado a un registro

Ejecución de una instrucción: ld/sd-I

IF	- Búsqueda en la MI de la instrucción apuntada por PC. - Actualización del PC	MI[PC] PC ← PC+4
ID	- Decodificación y lectura de registros - Extensión de signo para inmediato	[RS1] ext(Inmediato) Si es **Store**: [RD]
EX	- Suma en la ALU del registro base + el desplazamiento	[RS1]+ext(Inmediato)
MEM	- Acceso a la dirección de memoria calculada en la etapa anterior	**Load**: MD[[RS1]+ext(Inmediato)] **Store**: MD[[RS1]+ext(Inmediato)] ←[RD]
WB	- En caso se Load, se escribe el contenido de memoria en el registro RD	[RD] ← MD[[RS1]+ext(Inmediato)]

Ejecución de una instrucción: ALU-I

IF	- Búsqueda en la MI de la instrucción apuntada por PC. - Actualización del PC	MI[PC] PC ← PC+4
ID	- Decodificación y lectura del registro RS - Extensión de signo para inmediato	[RS1] ext(Inmediato)
EX	- La ALU realiza la operación que indique el opcode	[RS1] OP ext(Inmediato)
MEM		
WB	- Se escribe el resultado de la operación en el registro RD	[RD] ← [RS1] OP ext(Inmediato)

Ejecución de una instrucción: ALU-R

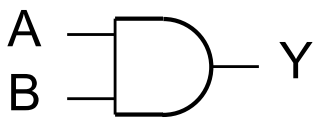
IF	- Búsqueda en la MI de la instrucción apuntada por PC. - Actualización del PC	MI[PC] PC ← PC+4
ID	- Decodificación y lectura del registro RS - Extensión de signo para inmediato	[RS1] [RS2]
EX	- La ALU realiza la operación que indique el opcode	[RS1] OP [RS2]
MEM		
WB	- Se escribe el resultado de la operación en el registro RD	[RD] ← [RS1] OP [RS2]

Ejecución de una instrucción: Branch-SB

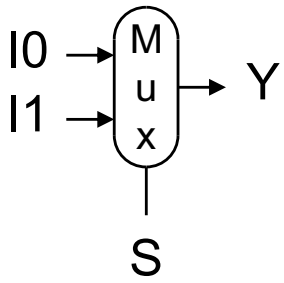
IF	- Búsqueda en la MI de la instrucción apuntada por PC - Actualización del PC	MI[PC] PC ← PC+4
ID	- Decodificación de la instrucción - Lectura de los registros RS1 y RS2 - Extensión de signo para inmediato	[RS1] [RS2] ext(Inmediato)
EX	- Cálculo de la dirección de salto: PC + Inmediatox2 - Evaluación condición de salto en RS1 y RS2 - Si evaluación positiva, se carga el PC con la dirección de salto	[PC] + ext(Inmediato)x2 cond ([RS1],[RS2]) Si con = TRUE PC ← [PC] + ext(Inmediato)x2
MEM		
WB		

Elementos combinacionales

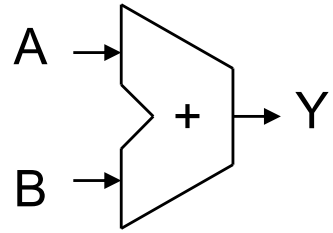
- Puerta AND
  - Y = A & B



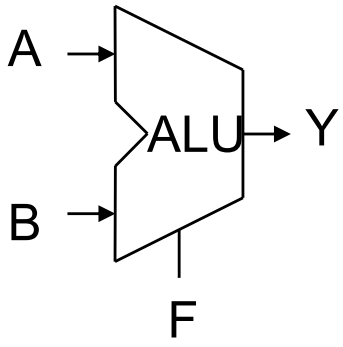
- Multiplexor
  - Y = S ? I1 : I0



- Sumador
  - Y = A + B



- ALU
  - Y = F(A, B)



Unidad Aritmético-Lógica: ALU

- Opera sobre los datos de una instrucción
- Tipo de operaciones:
  - desplazamientos
  - lógicas
  - aritméticas
- En la mayoría de los casos, es un **simple sumador-restador**
- Multiplicaciones y divisiones
  - Vía hardware: más rápido, pero ALU más cara
  - Vía software: ALU sumador-restador + algoritmo, más lento pero ALU más barata
- ALUs en **complemento a dos**
  - Circuito de la ALU más sencillo

Procesadores secuenciales

- Hasta que no termina de ejecutar una instrucción no comienza a ejecutarse la siguiente
- Según el método de temporización escogido tenemos:
  - **Procesador monociclo**
    - Cada instrucción se completa en un único ciclo de reloj
    - CPI = 1
    - La duración del ciclo de reloj viene fijada por la instrucción que más tarde en ejecutarse
  - **Procesador multiciclo**
    - Cada instrucción puede tardar más de un ciclo en ejecutarse
    - CPI > 1
    - La duración del ciclo de reloj es menor que para monociclo
    - Se establece que la duración de un ciclo = duración de la etapa más larga

Caso de estudio: nanoRISC-V

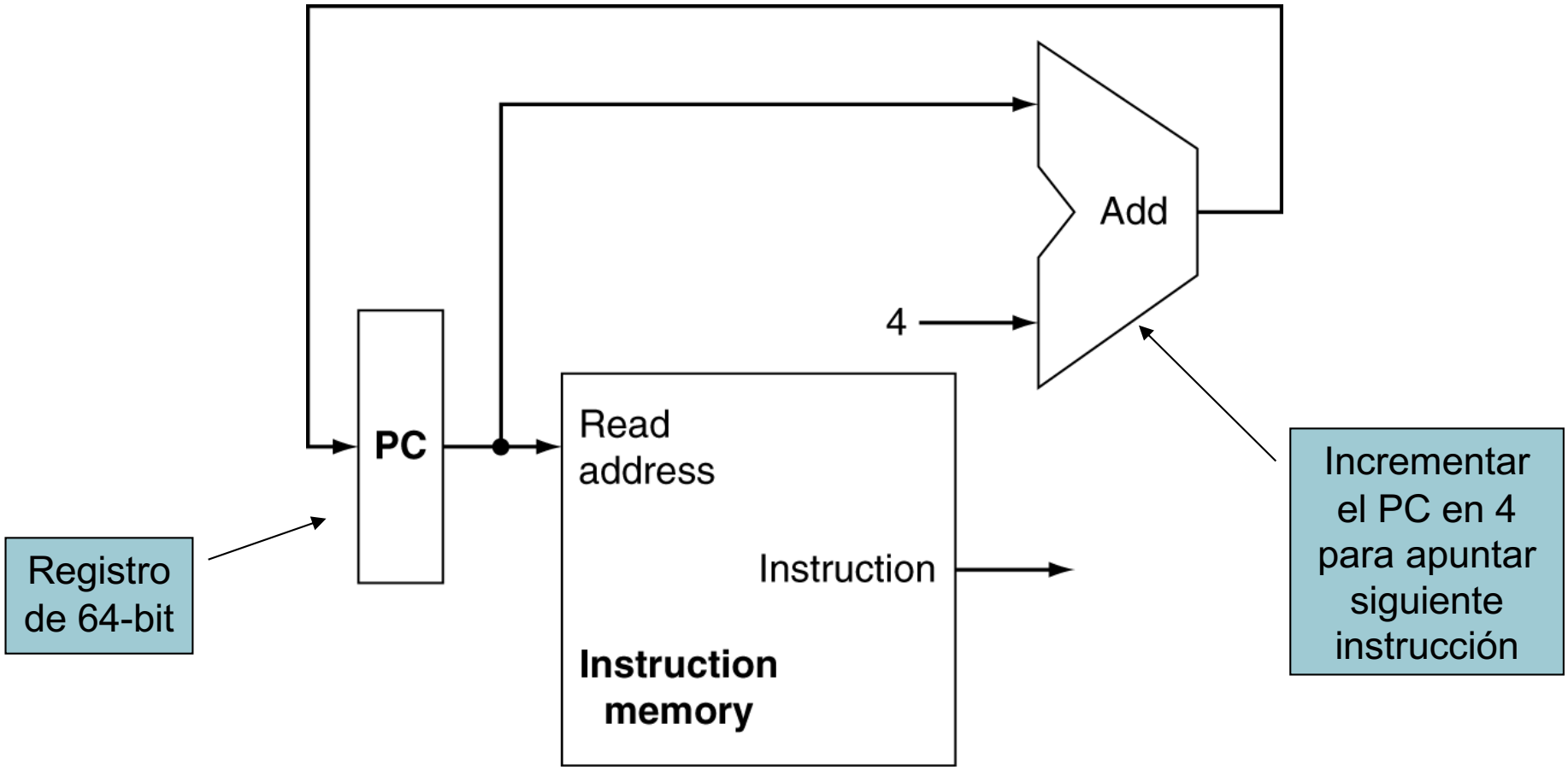
- Instrucciones soportadas por el nanoRISC-V
  - Acceso a memoria: `ld (I)`, `sd (I)`
  - Aritmético-lógicas: **`add`**, **`sub`**, **`and`**, **`or (R)`**
  - Control de flujo: **`beq (SB)`**

Ins.	Pseudocódigo	Descripción	Opcode	Funct3	Funct7
LD	LD RD,Imm (RS1)	R[RD]=M[R[RS1]+Imm](63:00)	0000011	011	-
SD	SW RD,Imm(RS1)	M[R[RS1]+Imm](63:00)=R[RD]	0100011	011	-
ADD	ADD RD,RS1,RS2	R[RD]=R[RS1]+R[RS2]	0110011	000	0000000
SUB	SUB RD,RS1,RS2	R[RD]=R[RS1]-R[RS2]	0110011	000	0100000
AND	AND RD,RS1,RS2	R[RD]=R[RS1]&R[RS2]	0110011	111	0000000
OR	OR RD,RS1,RS2	R[RD]=R[RS1]|R[RS2]	0110011	110	0000000
BEQ	BEQ RS1,RS2,destino	if(R[RS1]==R[RS2])PC=PC+{Imm}	1100011	000	-

Diseño ruta de datos y de control

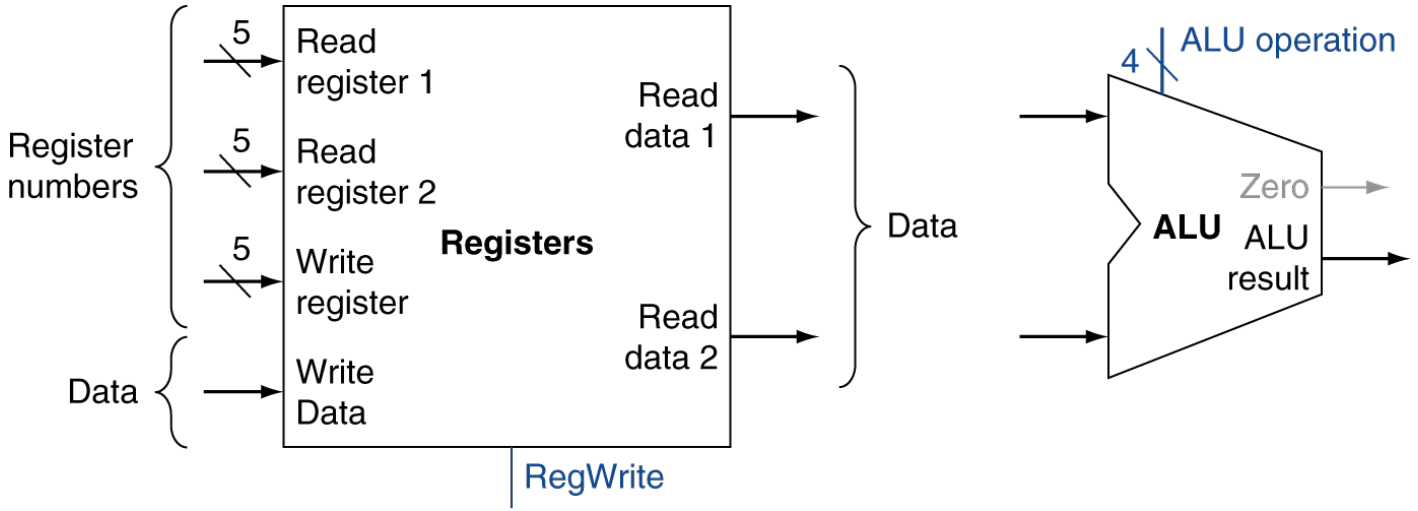
1. Análisis del repertorio de instrucciones a ejecutar
2. Establecer la **metodología de temporización**
3. Seleccionar los **módulos** necesarios para operar y almacenar datos y que formarán la ruta de datos teniendo en cuenta el repertorio de instrucciones y la metodología de temporización
4. **Ensamblar la ruta de datos** conectando los módulos escogidos e identificando los puntos de control
5. Determinar los valores de los **puntos de control** para cada instrucción del repertorio
6. Diseñar la **unidad de control**
7. Optimizar el diseño obtenido: **segmentación**

Etapa IF-Instruction Fetch



Instrucciones de tipo R

- Leer dos operandos/registros fuente
- Realizar una operación aritmética/lógica
- Almacenar el resultado en el registro destino

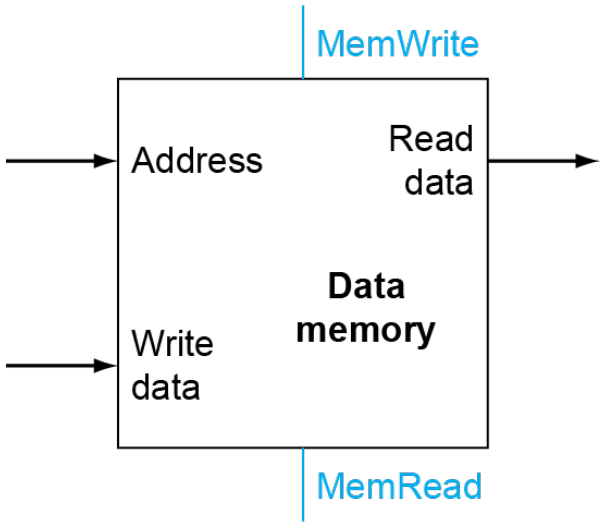


a. Registers

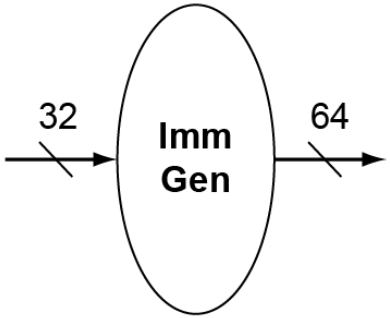
b. ALU

Instrucciones Load/Store

- Leer operandos/registros fuente
- Calcular dirección de memoria usando el offset de 12-bit
  - Usar ALU, pero extender de signo el offset
- Load: Leer memoria de datos y actualizer registro
- Store: Escribir en memoria de datos el valor del resgistro



a. Data memory unit

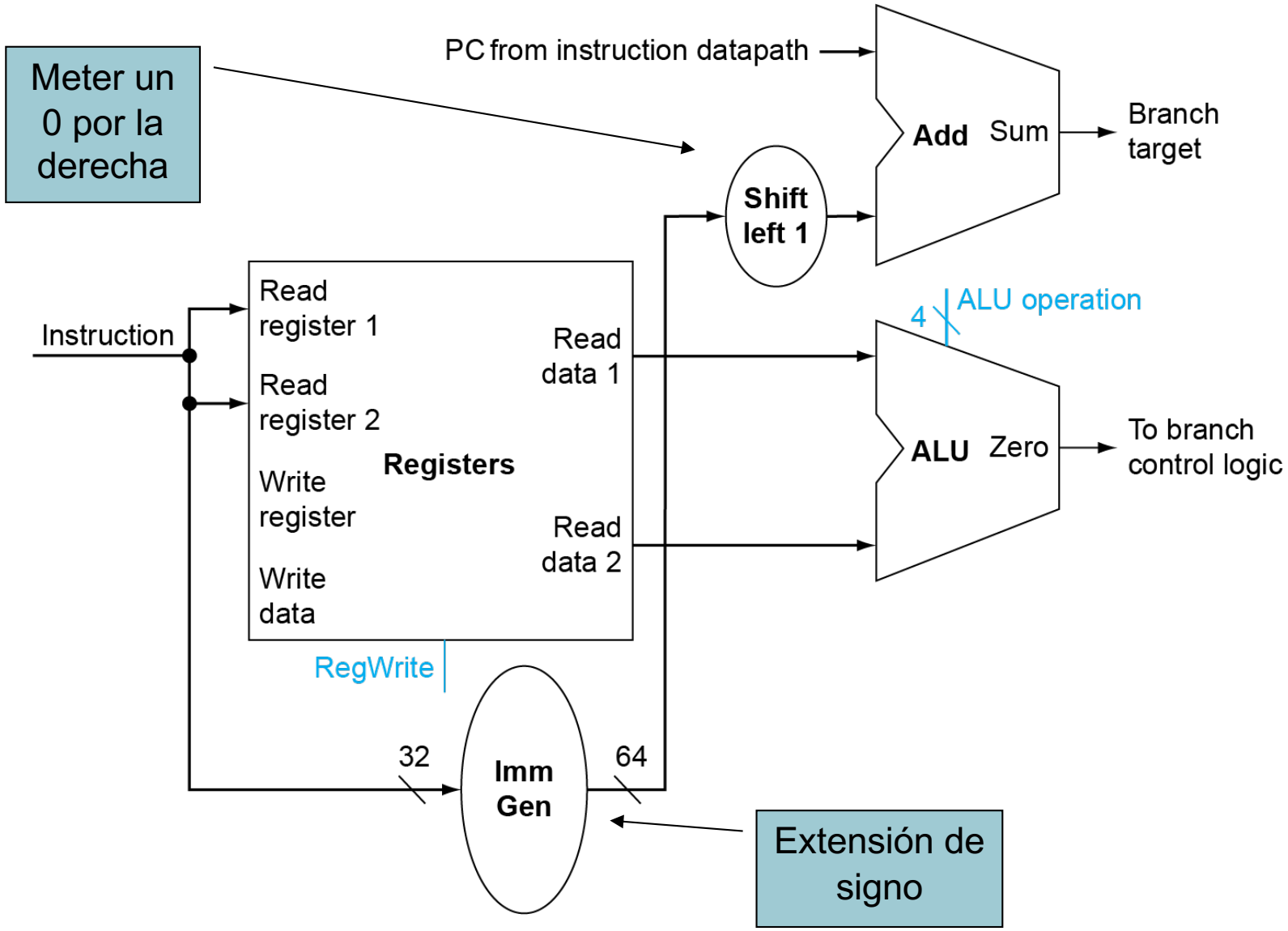


b. Immediate generation unit

Instrucciones de salto condicional

- Leer los operandos/registros fuente
- Comparar los operandos
  - Usar ALU, restrar y comprobar salida Zero
- Calcular dirección destino de salto
  - Extensión de signo del desplazamiento
  - Shift left 1, meter un 0 por la derecha
  - Sumar el resultado al PC

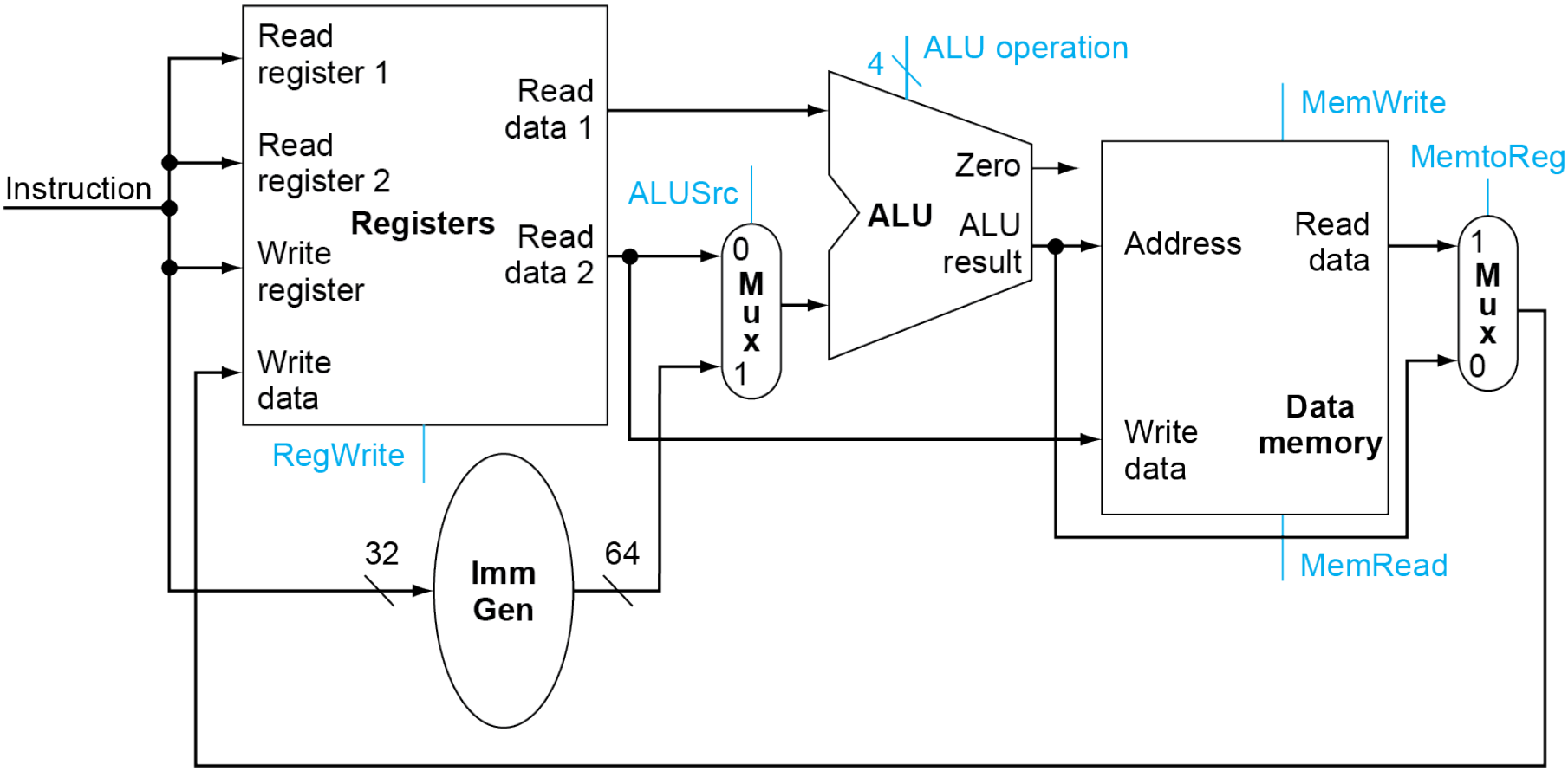
Instrucciones de salto condicional



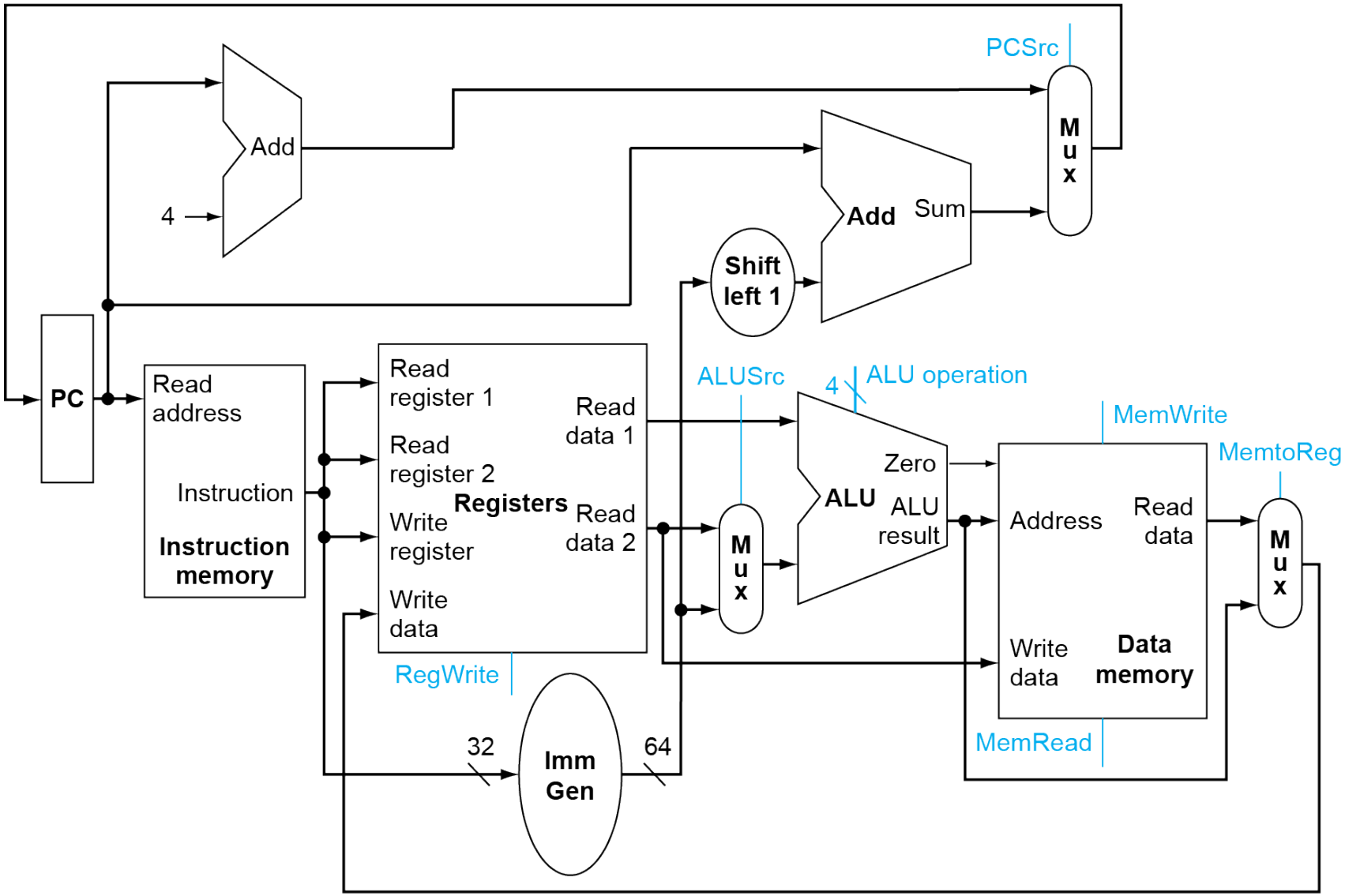
Disposición de los elementos

- El procesador monociclo ejecuta una instrucción en un ciclo de reloj
  - Cada elemento de la ruta de datos solo puede ejecutar una tarea a la vez
  - Por lo tanto, la memoria de datos y de instrucciones deben estar separadas
- Usar multiplexores donde se utilizan fuentes de datos alternativas para diferentes instrucciones

Ruta de datos de las instrucciones tipo R/Load/Store



Ruta de datos del nanoRISC-V monociclo



Ejercicio 1

- Cálculo de la frecuencia máxima de funcionamiento de un procesador nanoRISC-V monociclo
- Latencias componentes ruta de datos:
  - Lectura de la memoria de instrucciones: 0,3 ns
  - Lectura de la memoria de datos: 0,35 ns
  - Escritura en la memoria de datos: 0,5 ns
  - Lectura y escritura en el banco de registros: 0,05 ns
  - Operación aritmético-lógica en la ALU: 0,25 ns
  - Suma para preparar el siguiente PC: 0,1 ns
  - Suma del PC y el desplazamiento del salto: 0,1 ns
- ¿Cuál será el periodo de reloj del procesador? ¿Cuál será su frecuencia máxima?

Unidades de control del nanoRISC-V monociclo

- Unidad de control **global**
  - Decodifica el campo Opcode
  - Configuración global de la ruta de datos
- Unidad de control **local** a la ALU
  - Decodifica los campos Funct
  - Genera la señal ALU Control dependiendo de la operación concreta a realizar
- **Decodificación multinivel**
  1. La U.C global decodifica la instrucción leyendo su *opcode*
  2. El control local de la ALU realiza una segunda decodificación leyendo el/los campos *Funct*

Unidad de Control Global

- Las señales de control derivan de la instrucción

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

Unidad de Control Global

- **Entrada** ← *Opcode*
- **Salidas** → Genera los valores adecuados para las diferentes señales de control por medio de una ***tabla de verdad***: puntos de control en la ruta de datos
- PCSrc no se genera directamente
  - Cuando la instrucción es un salto, Branch = 1
  - Branch AND resultado evaluación condición
- *Las señales de control permanecen activas hasta que finaliza el ciclo. Cuando llega una nueva subida del flanco de reloj, se vuelve a comenzar el proceso*

Unidad de Control Local

- ALU Control
- Uso de la ALU
  - Load/Store: F = suma
  - Branch: F = resta
  - Tipo R: F depende del código de operación

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract

Unidad de Control Local

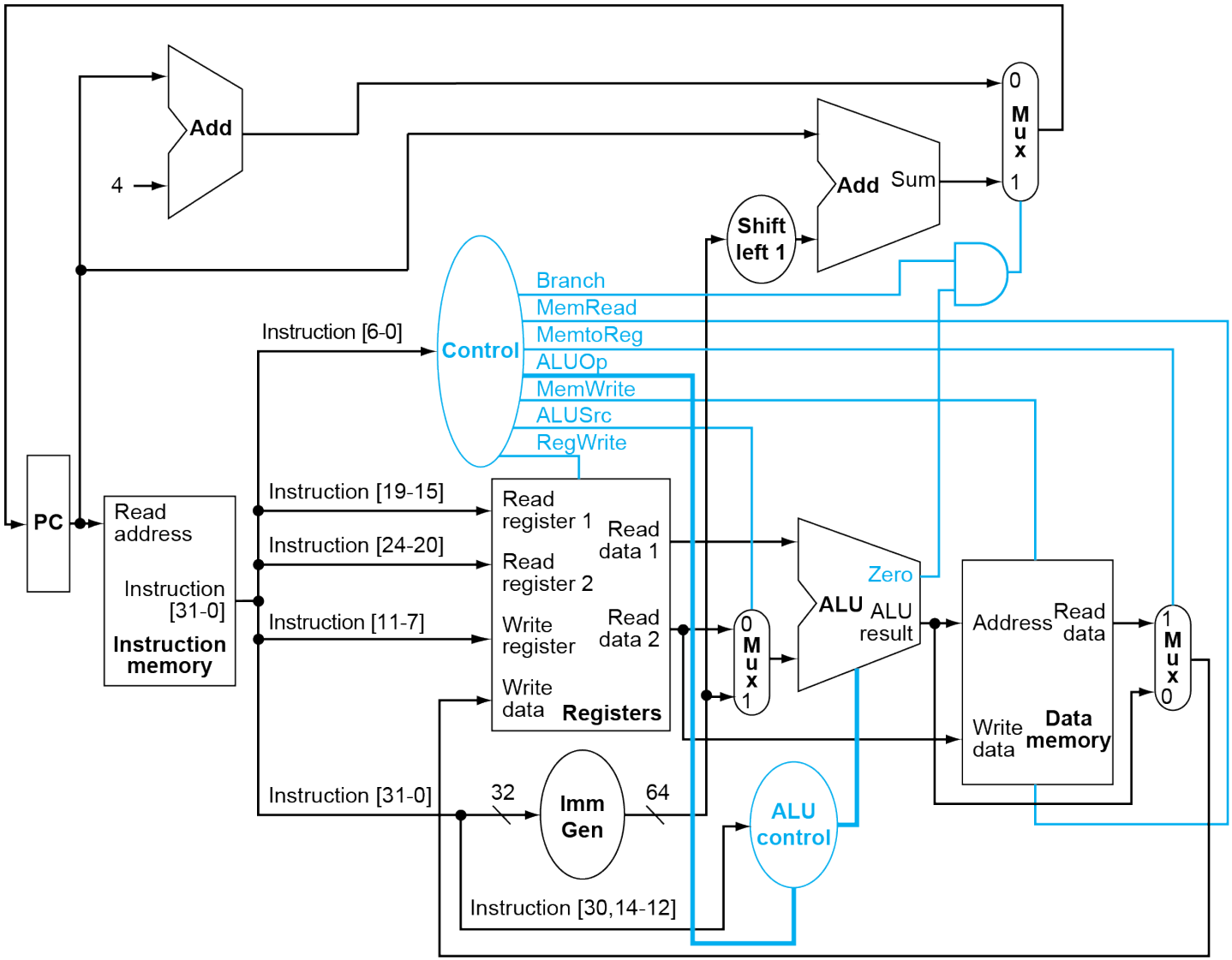
- ALU Control
- 2-bit ALUOp derivado del opcode

opcode	ALUOp	Operation	Opcode field	ALU function	ALU control
ld	00	load register	XXXXXXXXXXX	add	0010
sd	00	store register	XXXXXXXXXXX	add	0010
beq	01	branch on equal	XXXXXXXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001

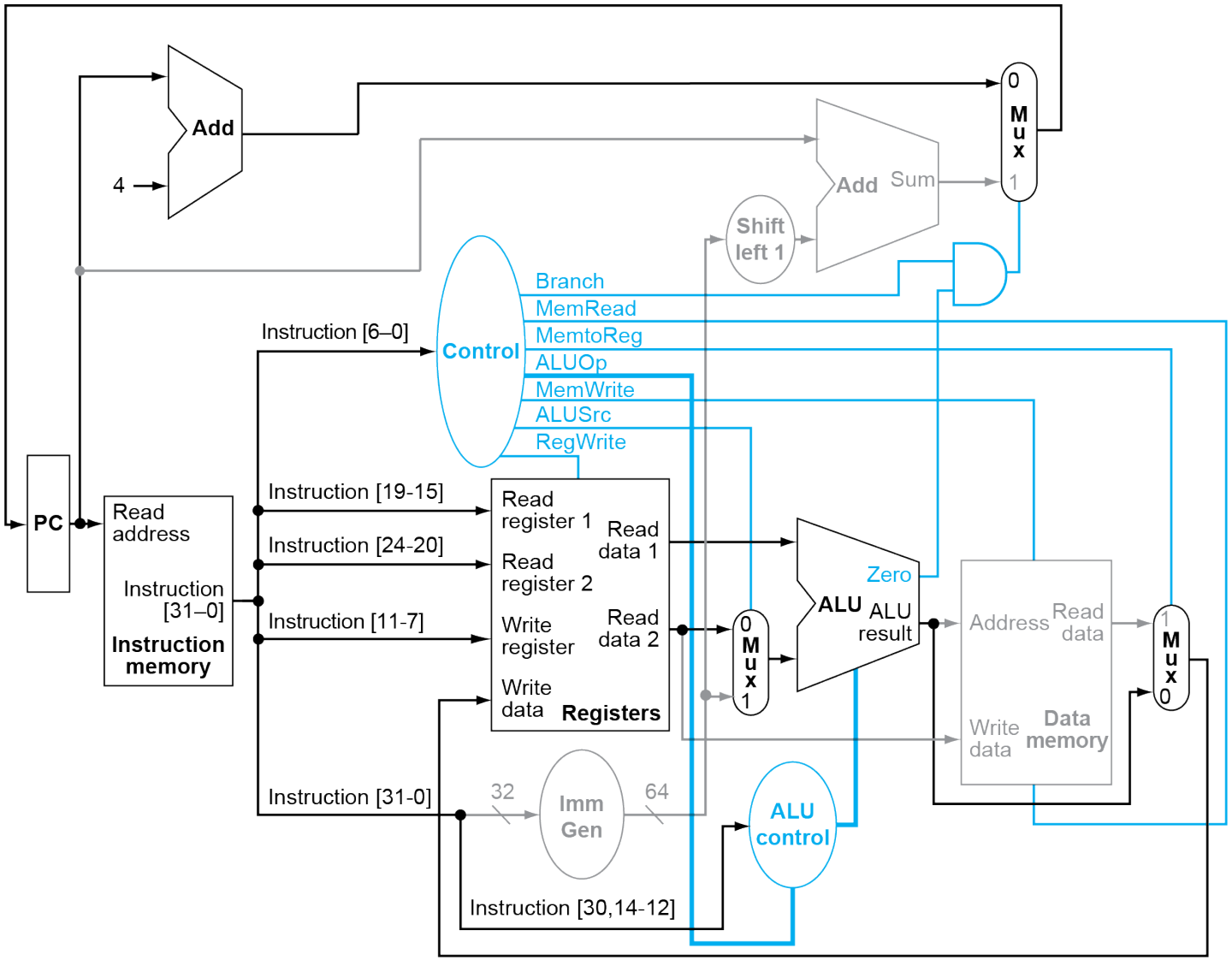
Unidad de Control Local

- ALU Control
- **Entrada** ← *ALUOp, Funct*
- **Salidas** → Genera la señal ALUControl por medio de una ***tabla de verdad***: puntos de control en la ruta de datos
- *Las señales de control permanecen activas hasta que finaliza el ciclo. Cuando llega una nueva subida del flanco de reloj, se vuelve a comenzar el proceso*

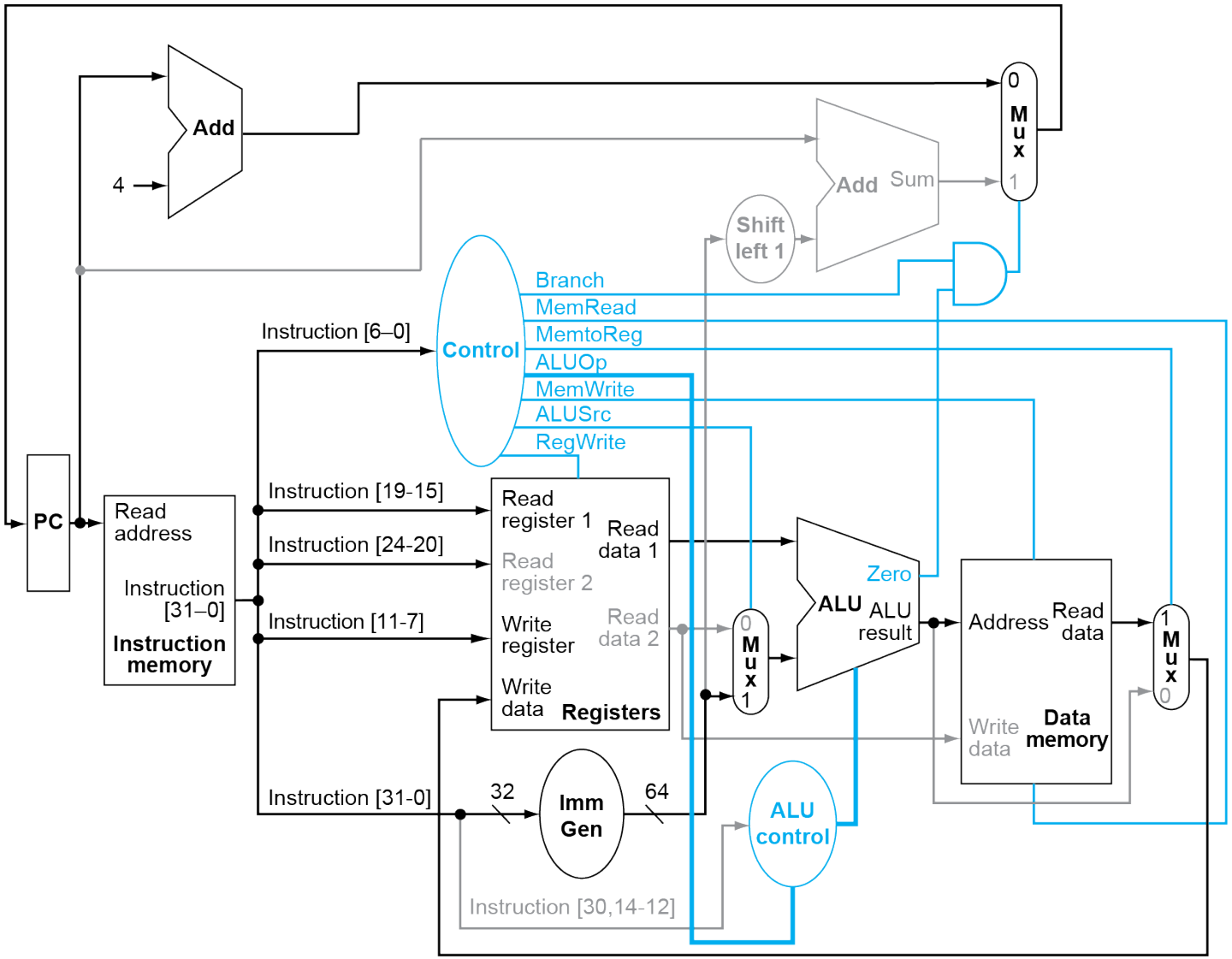
Ruta de Datos y Unidad de Control del nanoRISC-V monociclo



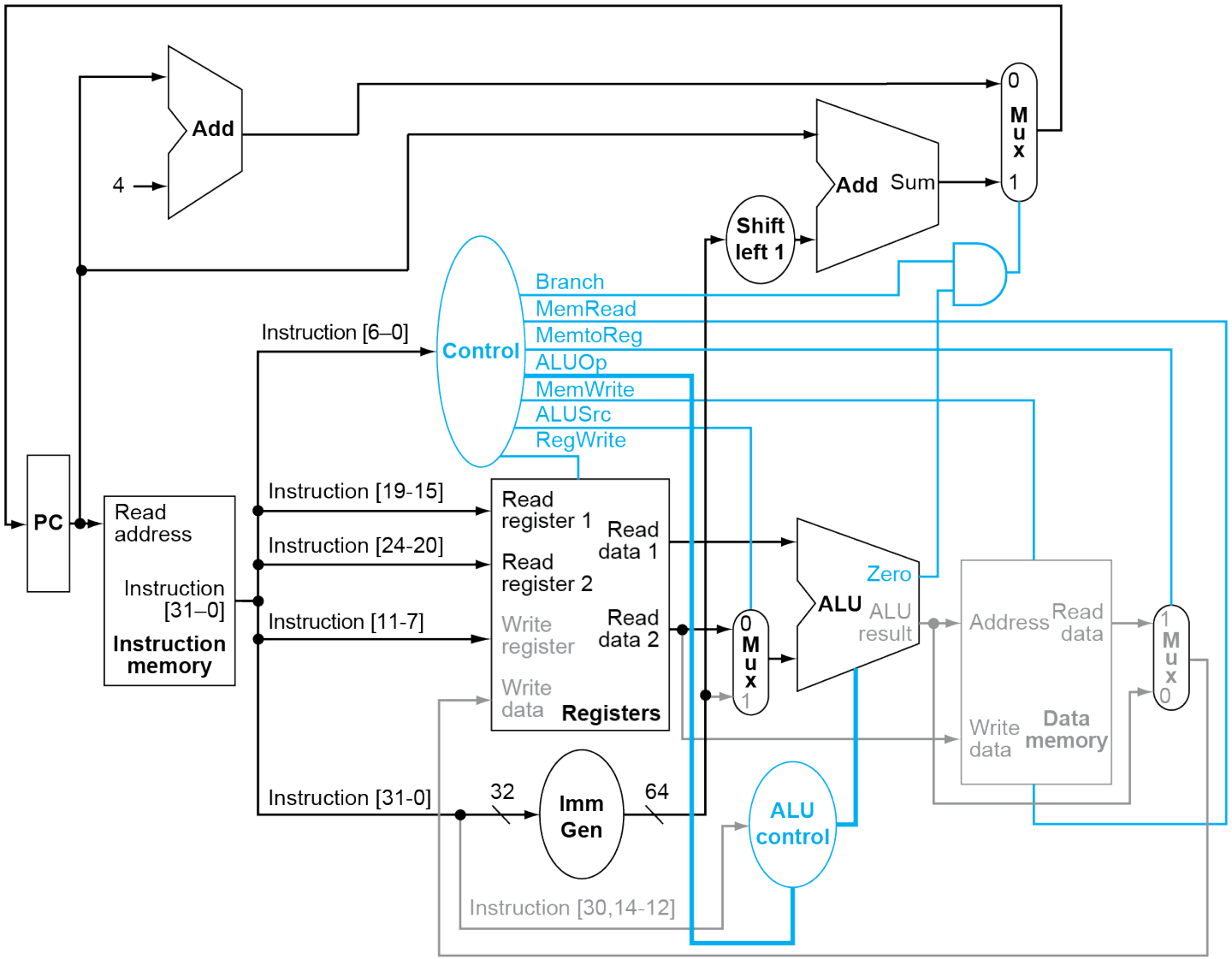
Instrucción tipo R



Instrucción Load



Instrucción BEQ



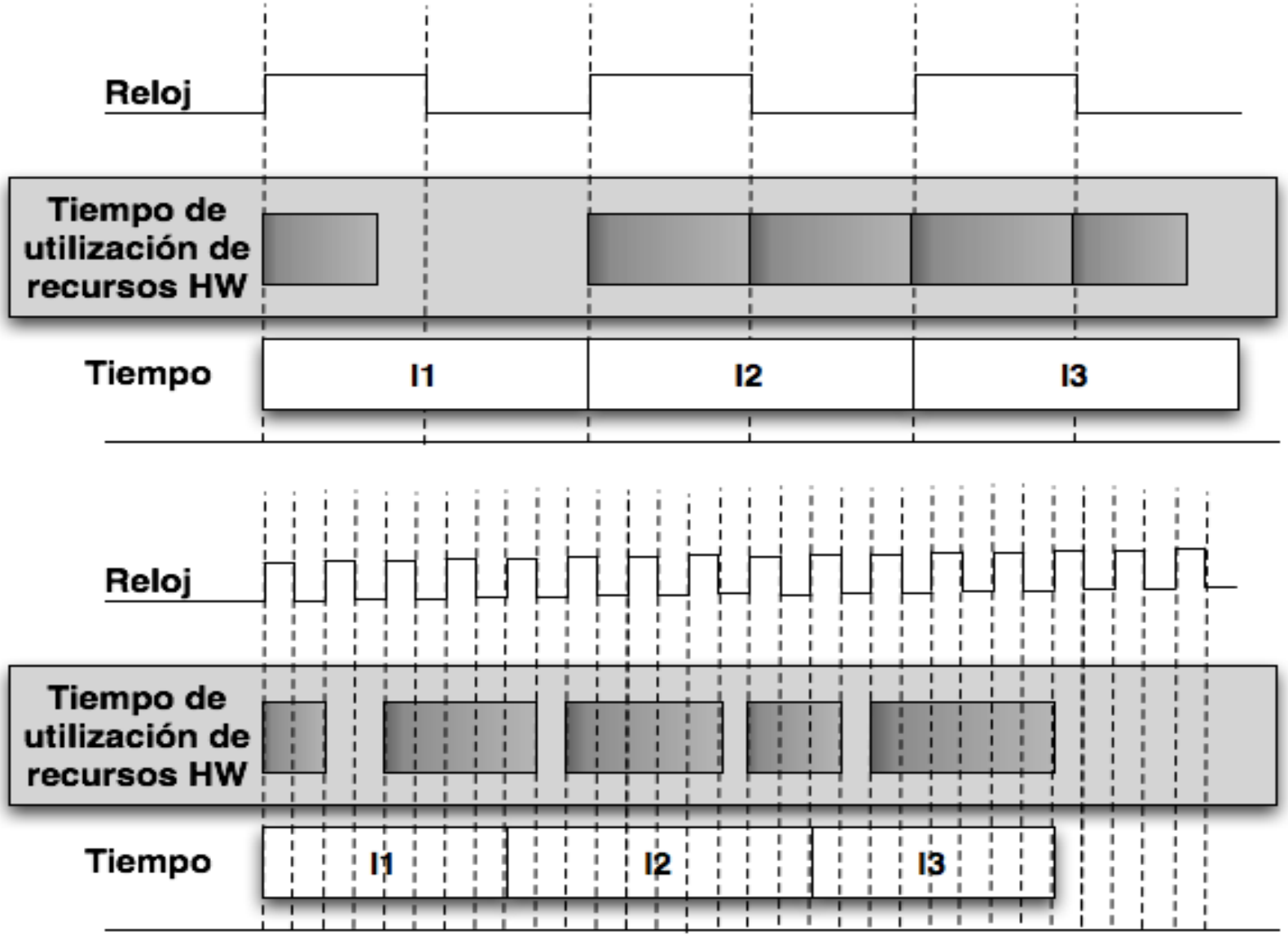
Rendimiento

- La instrucción más lenta determina la duración del ciclo de reloj
- No es posible variar el período para diferentes instrucciones
- Viola el principio de diseño
  - Hacer el caso común rápido
  - Mejora del rendimiento por medio del procesador multiciclo

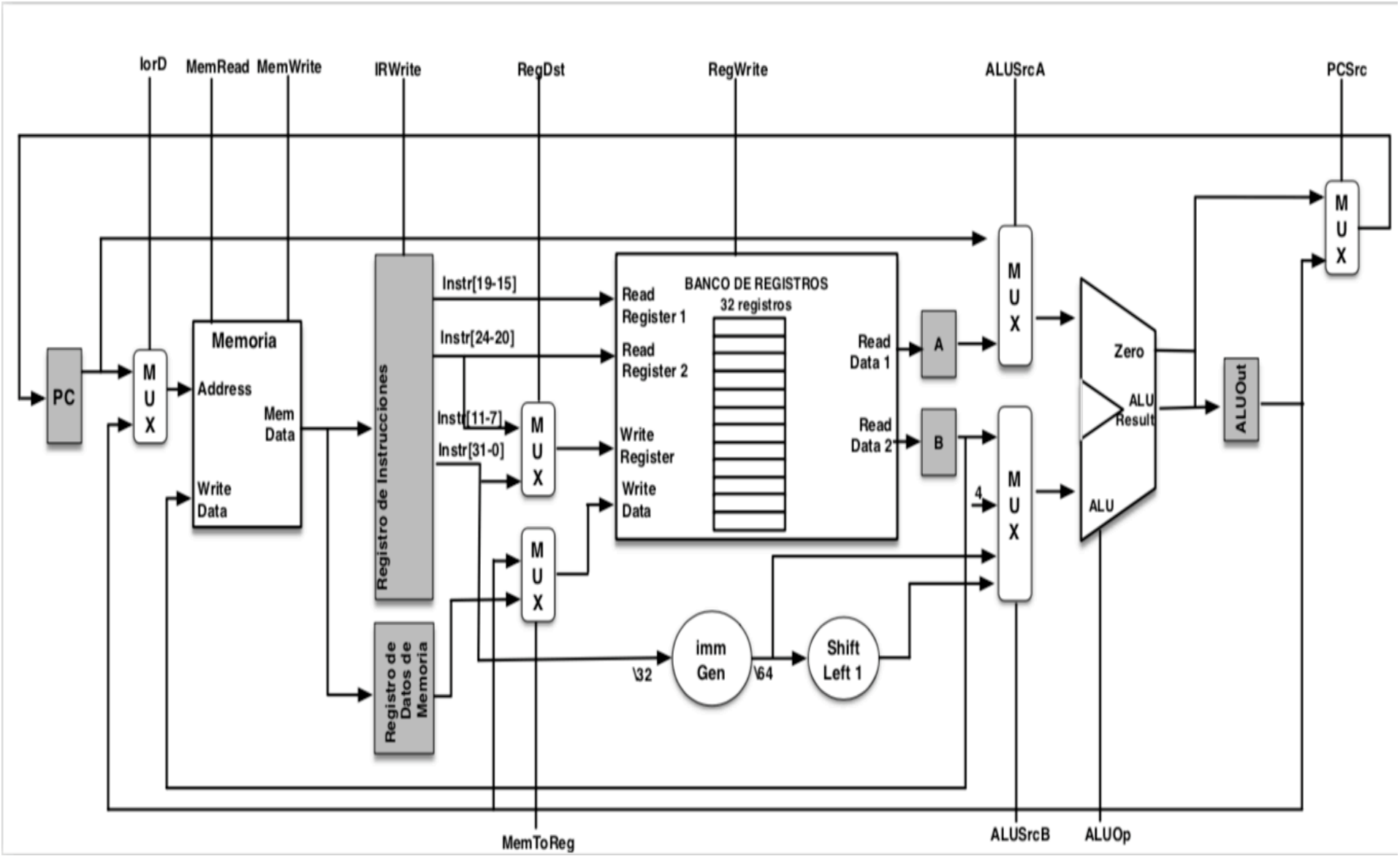
Diseño de la Ruta de Datos del nanoRISC-V multiciclo

- Se utiliza una **división del trabajo en etapas** típica de los procesadores con repertorio RISC
- Cada etapa está relacionada con el hardware de la ruta de datos que se utiliza
- Cada ***etapa*** debe completarse en 1 ciclo de reloj
- Se reduce el periodo del procesador
- El CPI medio será mayor que 1: nos alejamos del ideal
- Cada instrucción tarda en ejecutarse tantos ciclos como sea necesario

Utilización de recursos



Ruta de Datos del nanoRISC-V multiciclo



Ruta de datos nanoRISC-V multiciclo

- Etapa **IF** → Registro de instrucción IR
- Etapa **ID** → Lectura de operandos, A y B
- Etapa **EX** → Operandos fuente ALU en A y B
- Etapa **MEM** → De memoria a MDR
- Etapa **WB** → De ALUOut a registro
- No son necesarios sumadores extra
- No son necesarias dos memorias separadas
- Un mismo recurso puede usarse en diferentes etapas de la ejecución de una instrucción
- ¿Cuántas veces se utiliza la ALU en la ejecución de la instrucción BEQ?

Ejercicio 2

- Comparación versión monociclo nanoRISC-V con versión multiciclo
- Latencias de los elementos de la ruta de datos:
  - Lectura de la memoria: 0,3 ns
  - Escritura en la memoria: 0,45 ns
  - Lectura y escritura en el banco de registros: 0,05 ns
  - Operación aritmético-lógica en la ALU: 0,25 ns
  - Suma para preparar el siguiente PC: 0,1 ns
  - Suma del PC y el desplazamiento del salto: 0,1 ns
- Determinar si hay ganancia por convertir el procesador monociclo en multiciclo. El porcentaje de cada tipo de instrucción es:
  - Load: 20%
  - Store: 10%
  - Aritmético-lógicas: 35%
  - BEQ: 35%

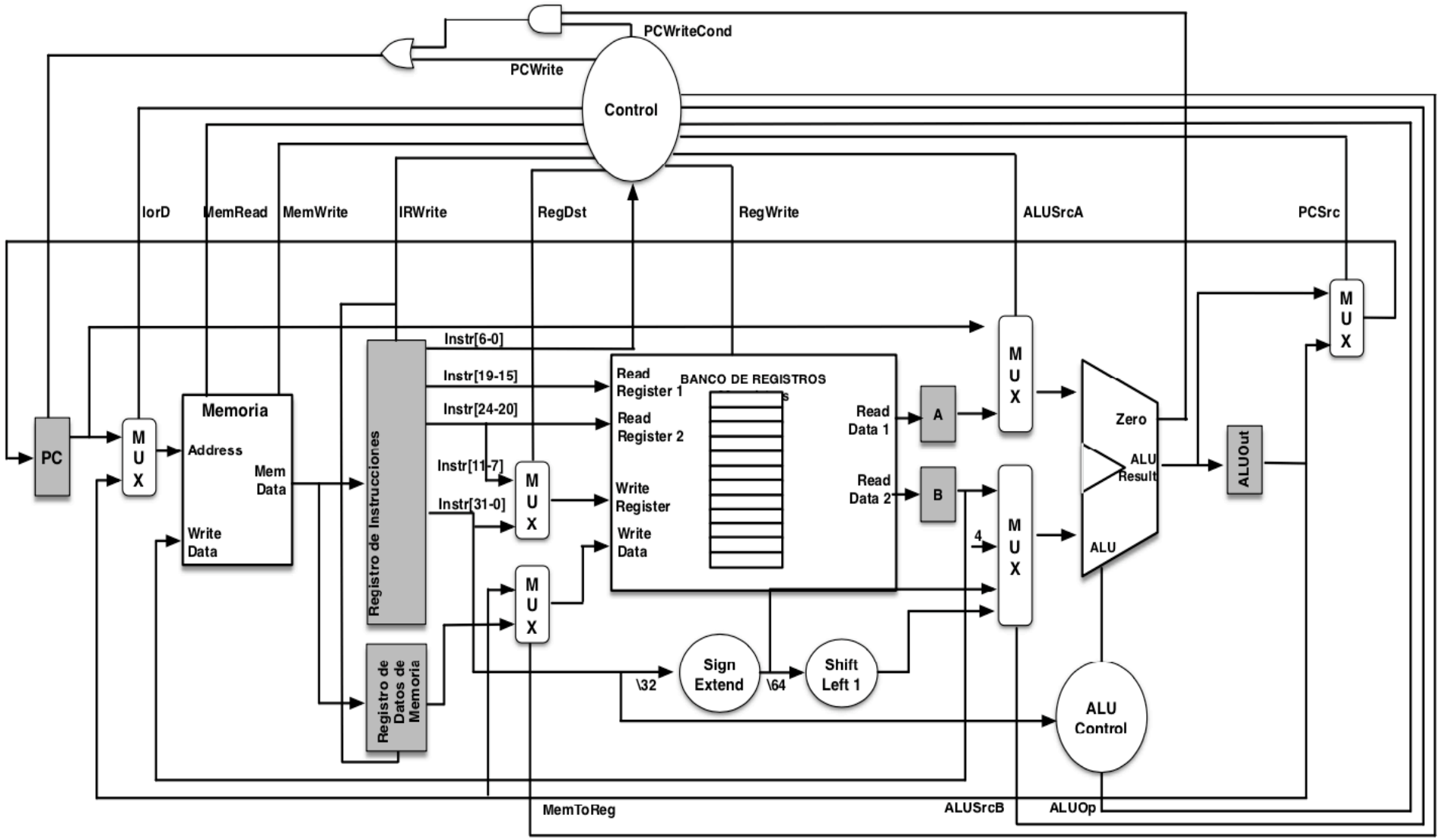
Ejercicio 3

- Periodo de reloj del nanoRISC-V multiciclo: registros intermedios
- Latencias de los elementos de la ruta de datos:
  - Lectura de la memoria: 0,3 ns
  - Escritura en la memoria: 0,45 ns
  - Lectura y escritura en el banco de registros: 0,05 ns
  - Operación aritmético-lógica en la ALU: 0,25 ns
  - Suma para preparar el siguiente PC: 0,1 ns
  - Suma del PC y el desplazamiento del salto: 0,1 ns
- ¿Cuál será el periodo de reloj del procesador multiciclo si los registros intermedios tienen una latencia de 0,1 ns?

Puntos de control de procesador multiciclo

- Los puntos de control **no se pueden dar como una tabla de verdad**, ya que no se mantienen constantes a lo largo de toda la ejecución
- Los valores de las señales se van modificando en los diferentes ciclos de reloj y dependen de la etapa en la que se encuentra la instrucción
- Señales a generar: *IorD, MemRead, MemWrite, MemToReg, RegDst, RegWrite, ALUSrcA, ALUSrcB, ALUOp, PCSrc, PCWrite, PCWriteCond e IRWrite*

Ruta de Datos y Unidad de Control del nanoRISC-V multiciclo



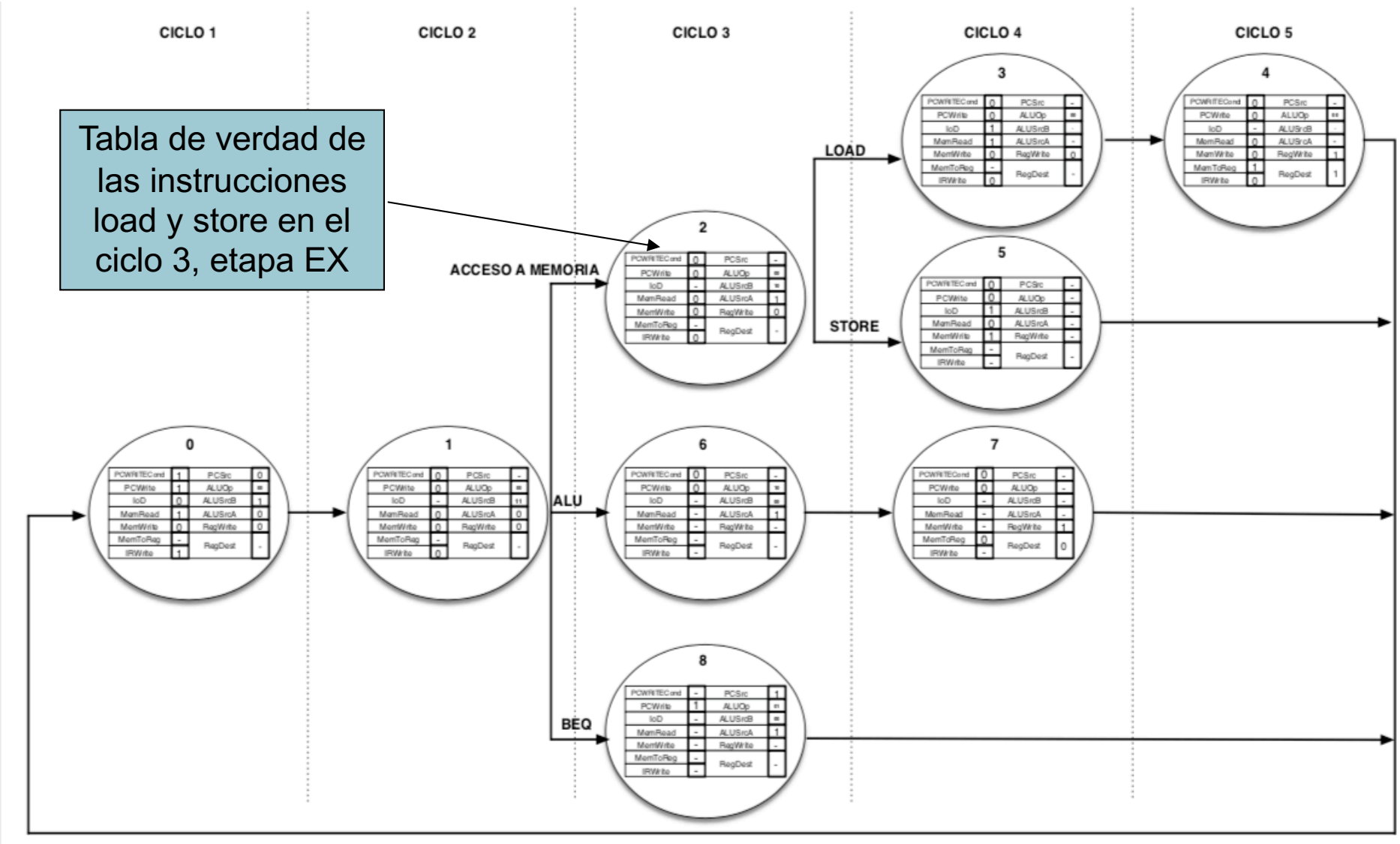
Unidades de Control del nanoRISC-V multiciclo

- La decodificación es multinivel, pero cambia la forma de generar la señales de control con respecto al nanoRISC-V monociclo
- Unidad de control **global**
  - Se puede diseñar como **máquina de estados** (***cableada, hardware***) o mediante **microprograma (*software*)**

Unidad de control *cableada*

- 9 estados que se ejecutan en un máximo de 5 ciclos
- Los ciclos IF e ID se ejecutan para todas las instrucciones por igual
- Cada tipo de instrucción evoluciona por unos estados diferentes
- **Problemas** cuando se modifica una única instrucción, **hay que rediseñar toda la unidad de control**
- **Ventaja**: rapidez

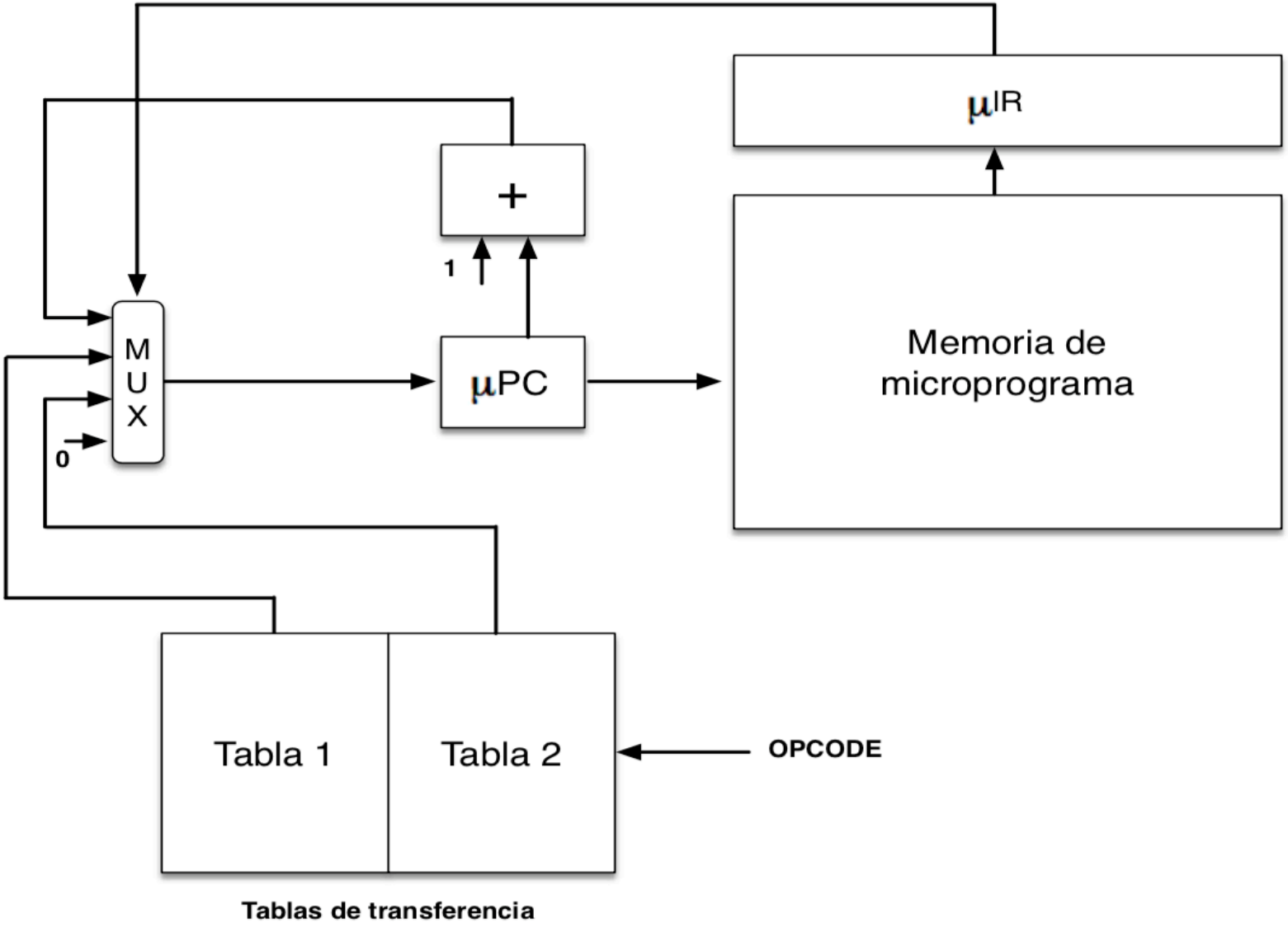
Unidad de control *cableada*



Unidad de control *microprogramada*

- **Memoria ROM** que almacena vectores con el valor de las señales de control que se deben generar en cada momento
- Ejecutar una instrucción es equivalente a leer palabras de esta memoria en un orden determinado
- Cada palabra es una **microinstrucción**
- El conjunto de micorinstrucciones que permiten ejecutar una instrucción es un **microprograma**
- **Ventajas**
  - Más flexible
  - Ocupa menos área
- **Desventaja**
  - Más lenta que la cableada
- Recuperar 3 o 4 microinstrucciones de la memoria, llevarlas al registro de microinstrucción (μIR), dejar tiempo para que se estabilicen las señales de control generadas, etc

Unidad de control *microprogramada*



Unidad de control *microprogramada*

- Hoy día la microprogramación ha desaparecido casi por completo
- Existen herramientas avanzadas para diseñar complejas unidades de control con millones de transistores. Estas herramientas garantizan la ausencia de errores de diseño
- Las unidades de control cableadas tienen un rendimiento significativamente mayor que cualquier unidad microprogramada, resultando más competitivas

Excepciones e Interrupciones

- Eventos inesperados que requieren de un cambio en el flujo de control
- Excepción
  - Se produce dentro de la CPU
    - Código de operación indefinido, syscall, …
- Interrupción
  - La genera el controlador de un dispositivo de E / S
- Atenderlas sin sacrificar el rendimiento es difícil

Tratamiento de excepciones

- Hasta ahora siempre hemos tenido un funcionamiento correcto del procesador
- El tratamiento de excepciones consiste en transferir el control a otro programa qué:
  - Salve el estado del procesador cuando se produzca la excepción
  - Corrija la causa de la excepción
  - Restaure el estado del procesador
  - Repita la ejecución de la instrucción causante de la excepción para poder continuar con la ejecución por el punto en el que se encontraba
- A este programa se le denomina *Rutina de Tratamiento de Excepción* o **RTE**

Excepción Vs Interrupción

- **Excepción**: evento no planificado que interrumpe la ejecución de un programa
- **Interrupción**: una excepción que proviene de fuera del microprocesador

Tipo de evento	¿Origen?	Terminología RISC-V
Petición de E/S	Externo	Interrupción
Llamada al SSOO desde un programa de usuario, *syscall*	Interno	Excepción
Desbordamiento aritmético	Interno	Excepción
Instrucción no definida	Interno	Excepción
Mal funcionamiento hardware	Ambos	Excepción o interrupción

Tipos de excepciones

- Interrupciones de E/S
- Llamadas al Sistema Operativo, *syscall*
- Puntos de ruptura
- **Códigos de operación inválidos**
- **Overflow o desbordamiento en la ALU**
- Fallos de página
- Accesos a memoria no alineados
- Violación de zonas protegidas de memoria
- Fallos de hardware
- Fallos de alimentación

Hardware necesario para soportar excepciones

- Salvar el PC de la instrucción causante de la excepción:
  - En el *Supervisor Exception Program Counter* (**SEPC**)
  - SEPC almacena PC - 4
  - Necesitamos un restador para realizar PC - 4
- Salvar el motivo de la excepción:
  - En el *Registro Supervisor Exception Cause Register* (**SCAUSE**)
  - 64 bits, pero la mayoría vacíos. Campo código excepción: 2 para código de operación indefinido, 12 para mal funcionamiento del hardware, …
- Saltar al manejador: se debe cargar en el PC la dirección de memoria de la RTE
  - $0000\ 0000\ 1C09\ 0000_{hex}$

Señales de control

- **ALU_overflow** e **Ilegal_opcode**
- **Exception**: para escribir el código de las excepciones en el registro **SCAUSE**
- **ExceptionWrite** y **SEPCWrite**: para controlar la escritura en los dos nuevos registros
- **PCWrite**: para controlar la carga del PC

Acciones de la RTE

- Leer la causa y transferir el control al manejador correspondiente
- Determinar la acción necesaria
- Si la excepción es recuperable
  - Se emprende la acción correctiva
  - Se usa SEPC para retornar el programa
- Si no es recuperable
  - Termina el programa
  - Se informa del error usando SEPC, SCAUSE, …

Unidades de Control y excepciones

- **Monociclo**: sólo hay que añadir las nuevas señales a la tabla de verdad
- **Multiciclo**:
  - **Modificar la máquina de estados** con la que se diseña la unidad de control
  - Añadir **nuevos microprogramas** a la unidad de control microprogramada

Nuevos estados nanoRISC-V multiciclo