



Tema 5:

Diseño monociclo del procesador

Fundamentos de computadores II



José Manuel Mendías Cuadros

*Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid*





Introducción

- El método más fiable para conocer el rendimiento de un computador es **midiendo el tiempo que tarda en ejecutar programas**.
 - El computador que los ejecute **más rápido** tendrá **mayor rendimiento**.
- Para una arquitectura dada, el tiempo de ejecución de un programa depende principalmente del:
 - **Número de instrucciones** que tenga el programa.  **depende del programador**
 - **Número de ciclos** que tarde cada instrucción.  **dependen del diseño HW**
 - **Tiempo de ciclo** (frecuencia de reloj).
- Comúnmente **número de ciclos y tiempo de ciclo tienen son inversos**:
 - **Procesador monociclo**: 1 ciclo por instrucción, tiempo de ciclo largo.
 - **Procesador multiciclo**: Varios ciclos por instrucción, tiempo de ciclo corto.

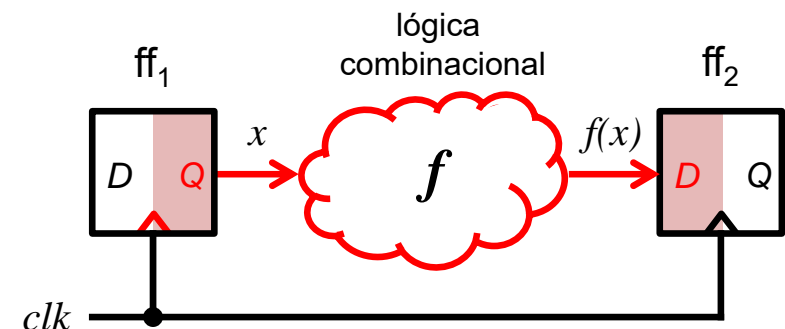
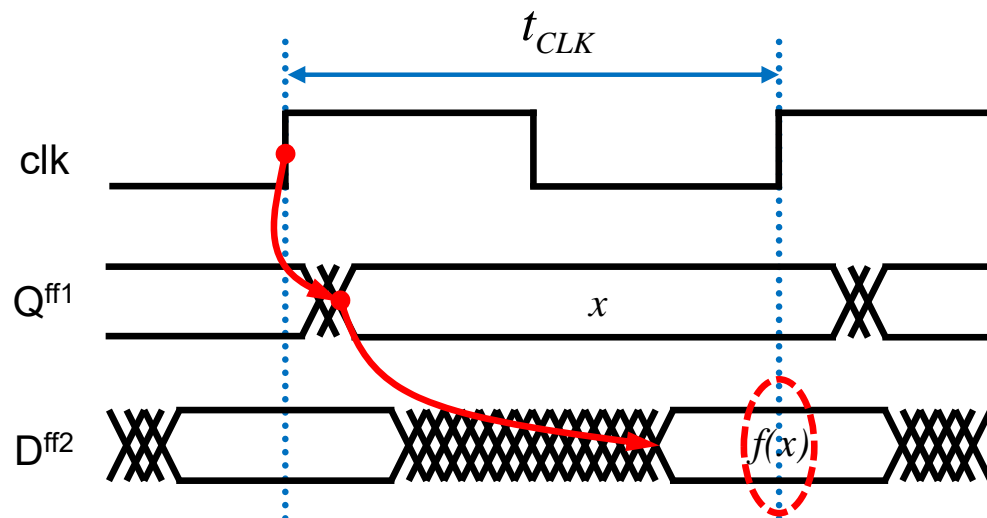


Introducción

- Para **diseñar un procesador** se usan **técnicas de diseño algorítmico** en donde la **especificación** del circuito es su **arquitectura**.
- El procesador estará formado por 2 elementos:
 - **Ruta de datos**: realiza operaciones y almacena resultados.
 - Al menos, deberá incluir tantos **elementos de almacenamiento** como estén **definidos en la arquitectura** (sean visibles por el programador).
 - Deberá incluir los **elementos funcionales** que sean necesarios para **realizar todas las operaciones** del repertorio de instrucciones.
 - **Controlador**: secuencia la realización de las transferencias entre registros definidas para cada instrucción del repertorio.
- Según la **estrategia de diseño** elegida tendremos:
 - **Procesadores monociclo**: **todas las transferencias** de entre registros implicadas en una instrucción se realizan **en un único ciclo de reloj**.
 - **Procesadores multiciclo**: **las transferencias** entre registros implicadas en una instrucción **se reparten entre varios ciclos de reloj** consecutivos.

Introducción

- Los procesadores se diseñarán según el **modelo de temporización síncrona por flanco** (de subida) de reloj global:
 - El reloj llega a todos los biestables del sistema y todos ellos **cambian de estado simultáneamente en el flanco (de subida) del reloj**.
 - Los nuevos **valores se propagan a través de las redes combinatoriales** hasta estabilizarse en las entradas de los biestables.
 - Se repite el proceso indefinidamente en cada ciclo de reloj.
 - El **tiempo de ciclo del reloj debe ser lo suficientemente largo** para que todos los sistemas combinatoriales alcancen su **régimen permanente**.





RISC-V de arquitectura reducida

Repertorio de instrucciones (i)

- Se diseñará una **microarquitectura** capaz de ejecutar un **subconjunto** del **repertorio de instrucciones del RISC-V32** que opere únicamente con datos de **32 bits**.
- **Instrucciones con acceso a memoria**

lw rd, imm _{12b} (rs1)	rd ← Mem[rs1 + sExt(imm)]	tipo-I
sw rs2, imm _{12b} (rs1)	Mem[rs1 + sExt(imm _{12b})] ← rs2	tipo-S

- **Aritmético-lógicas con ambos operandos en registros**

add rd, rs1, rs2	rd ← rs1 + rs2	tipo-R
sub rd, rs1, rs2	rd ← rs1 - rs2	tipo-R
and rd, rs1, rs2	rd ← rs1 & rs2	tipo-R
or rd, rs1, rs2	rd ← rs1 rs2	tipo-R
stl rd, rs1, rs2	rd ← if (rs1 < rs2) then (1) else (0)	tipo-R

RISC-V de arquitectura reducida

Repertorio de instrucciones (ii)



■ Aritmético-lógicas con un operando inmediato

<code>addi rd, rs1, imm_{12b}</code>	$rd \leftarrow rs1 + sExt(imm)$	tipo-I
<code>andi rd, rs1, imm_{12b}</code>	$rd \leftarrow rs1 \& sExt(imm)$	tipo-I
<code>ori rd, rs1, imm_{12b}</code>	$rd \leftarrow rs1 sExt(imm)$	tipo-I
<code>stli rd, rs1, imm_{12b}</code>	$rd \leftarrow if (rs1 < sExt(imm)) then (1) else (0)$	tipo-I

■ Instrucciones de salto condicional

<code>beq rs1, rs2, imm_{13b}</code>	$PC \leftarrow if (rs1 = rs2)$ $then (PC + sExt(imm_{12:1} \ll 0)) else (PC+4)$	tipo-B
--	---	--------

■ Instrucciones de salto a función

<code>jai rd, imm_{21b}</code>	$PC \leftarrow PC + sExt(imm_{20:1} \ll 0), rd \leftarrow PC+4$	tipo-J
--	---	--------

RISC-V de arquitectura reducida

Formato de instrucción



- Los formatos de instrucción y codificación de campos serán los **mismos** que en la arquitectura completa.

31	25 24	20 19	15 14	12 11	7 6	0	
funct7	rs2	rs1	funct3	rd	op		<i>tipo-R</i>
imm _{11:0}		rs1	funct3	rd	op		<i>tipo-I</i>
imm _{11:5}	rs2	rs1	funct3	imm _{4:0}	op		<i>tipo-S</i>
imm _{12,10:5}	rs2	rs1	funct3	imm _{4:1,11}	op		<i>tipo-B</i>
imm _{20,10:1,11,19:12}				rd	op		<i>tipo-J</i>



RISC-V de arquitectura reducida

Formato de instrucción: codificación de campos

Instrucción	Tipo	funct7 bits 31:25	funct3 bits 14:12	op bits 6:0
lw	I	—	010	0000011
sw	S	—	010	0100011
add	R	0000000	000	0110011
sub	R	0100000	000	
slt	R	0000000	010	
or	R	0000000	110	
and	R	0000000	111	
addi	I	—	000	0010011
slti	I	—	010	
ori	I	—	110	
andi	I	—	111	
beq	B	—	000	1100011
jal	J	—	—	1101111



Diseño de la ruta de datos

Elementos de almacenamiento (i)

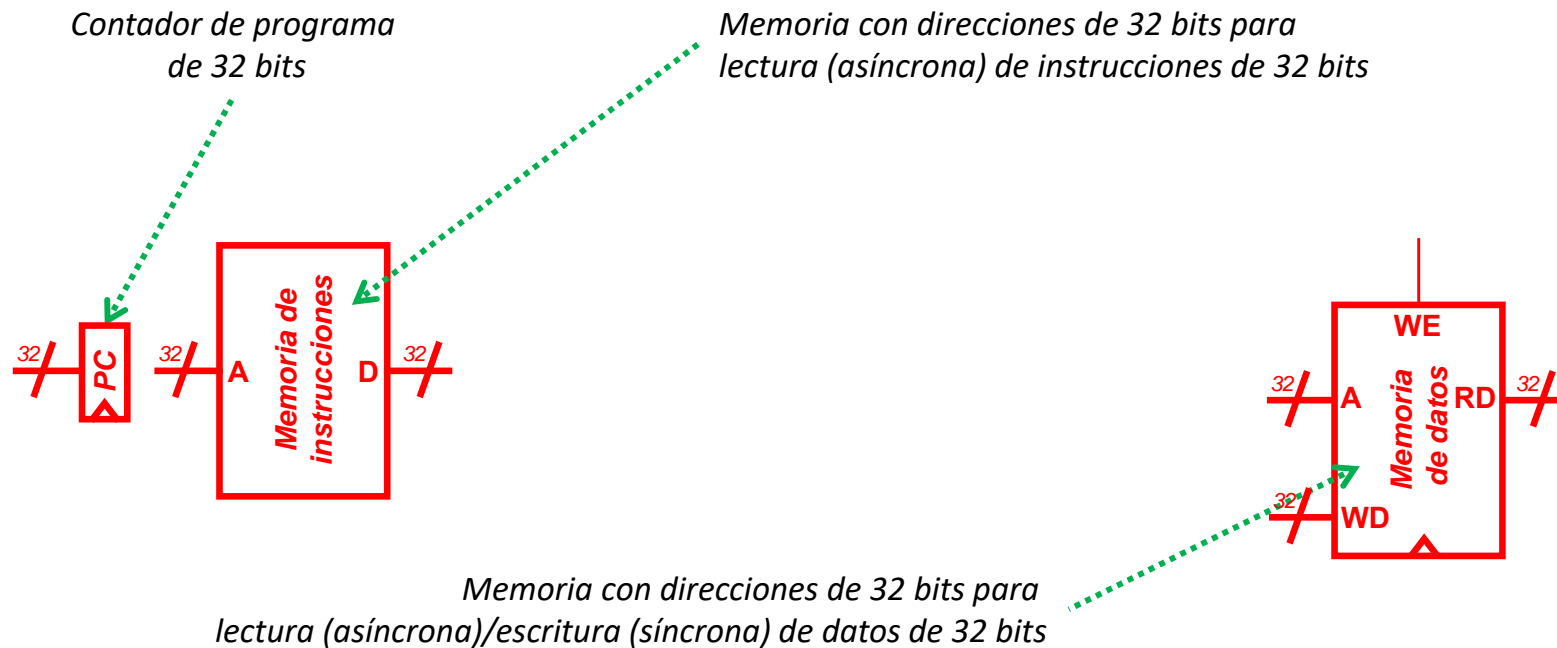
- Los **elementos de almacenamiento** serán los mismos que son visibles al programador: **Memoria**, **PC** y **32 registros** de propósito general.
- La **Memoria** tendrá un **comportamiento idealizado**:
 - Integrada en la CPU.
 - Direccionable por bytes, pero capaz de aceptar/ofrecer 4 bytes por acceso.
 - Tiempo de acceso inferior al tiempo de ciclo del procesador.
 - Dividida en dos porque las **instrucciones deben leerse** de memoria en el mismo **ciclo de reloj** en que se **leen/escriben los datos**.
 - **Memoria de instrucciones**: se comportará como una ROM combinacional.
 - **Memoria de datos**: se comportará como un gran Banco de Registros con doble puerto de datos para entrada y salida separada de los mismos.
- El **Contador de Programa** será un **array de biestables D**:
 - Al ejecutarse las instrucciones en un único ciclo, el PC **debe actualizarse en todos los ciclos** y no requiere tener una señal que controle su carga.



Diseño de la ruta de datos

Elementos de almacenamiento (ii)

- Los **elementos de almacenamiento** serán los mismos que son visibles al programador: **Memoria**, **PC** y **32 registros** de propósito general.

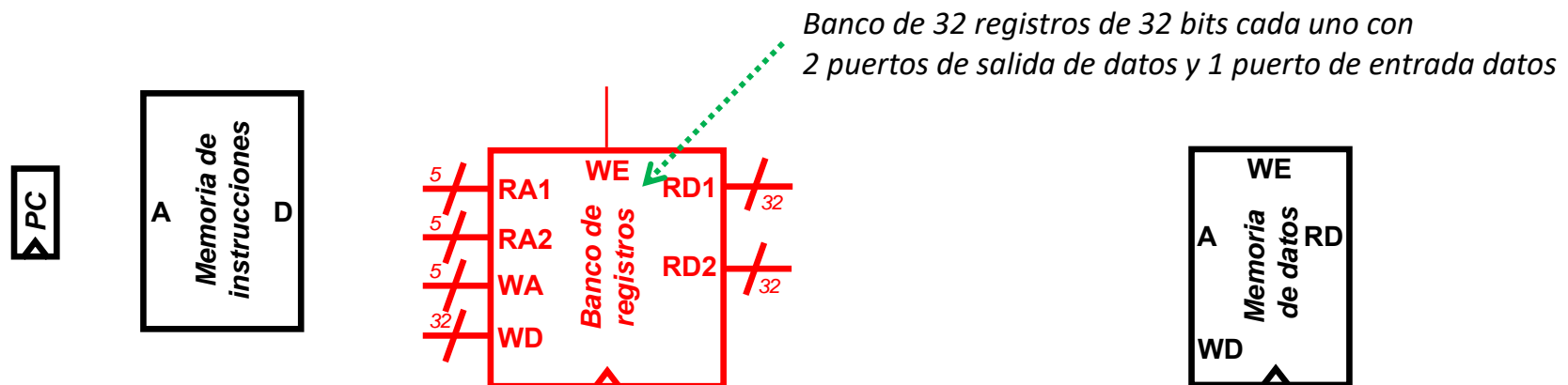




Diseño de la ruta de datos

Elementos de almacenamiento (iii)

- Los **elementos de almacenamiento** serán los mismos que son visibles al programador: **Memoria**, **PC** y **32 registros** de propósito general.



- Los 32 registros se organizan en un **Banco con 3 puertos**:
 - En un procesador monociclo, las instrucciones tipo-R en el **mismo ciclo de reloj** leen del Banco **2 registros** y escriben **1 registro**.

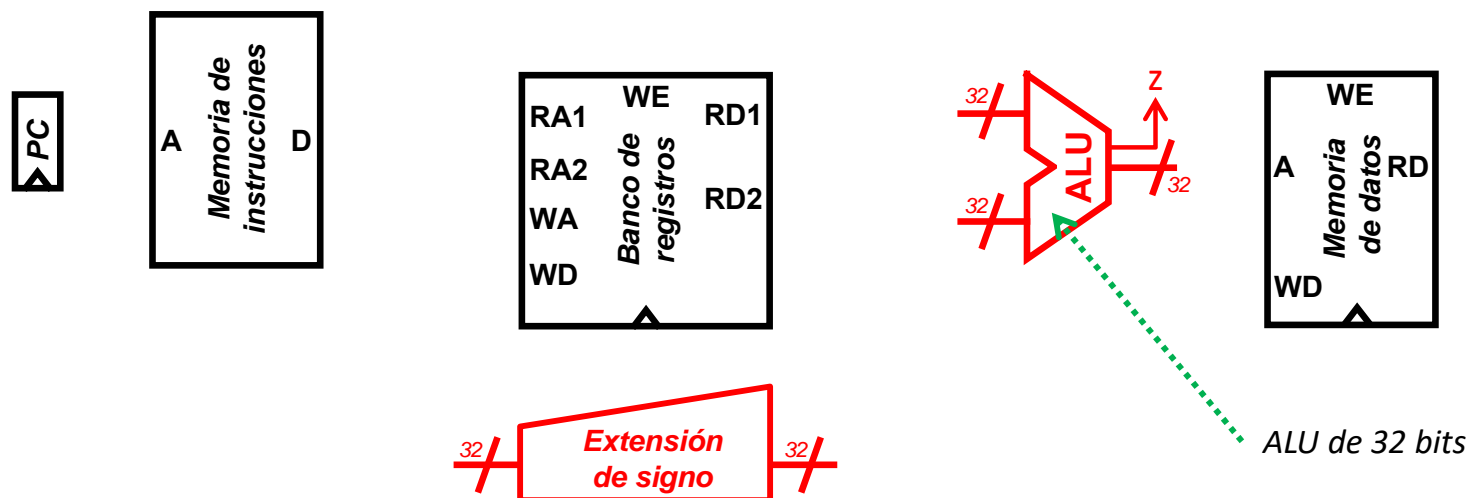
`add rd, rs1, rs2 $BR[rd] \leftarrow BR[rs1] + BR[rs2], PC \leftarrow PC+4$`



Diseño de la ruta de datos

Elementos funcionales (i)

- Dispondrá de una **ALU** y un **Extensor de Signo** combinacionales:
 - La **ALU** realizará **todas las operaciones aritmético-lógicas** del repertorio.
 - Con un **flag Z** para **realizar mediante resta la comparación** de igualdad en instrucciones **beq**
 - El **Extensor de Signo** construirá el **operando inmediato de 32 bits** a partir de los campos imm (de menor anchura) de las instrucciones.

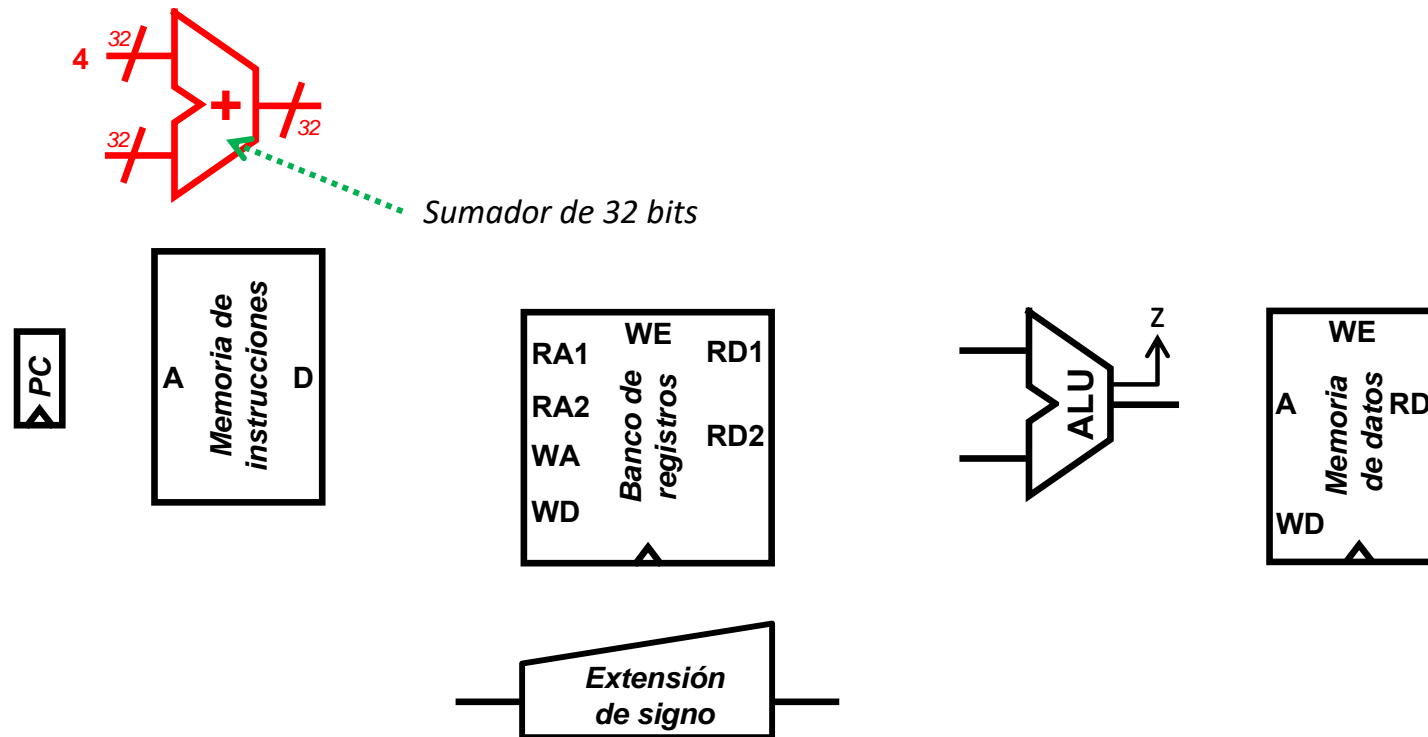




Diseño de la ruta de datos

Elementos funcionales (ii)

- Dispondrá de un **sumador adicional** para **incrementar el PC**
 - En un procesador monociclo, en el **mismo ciclo de reloj** en que **opera la ALU** se **actualiza el PC** con la dirección de la instrucción siguiente.



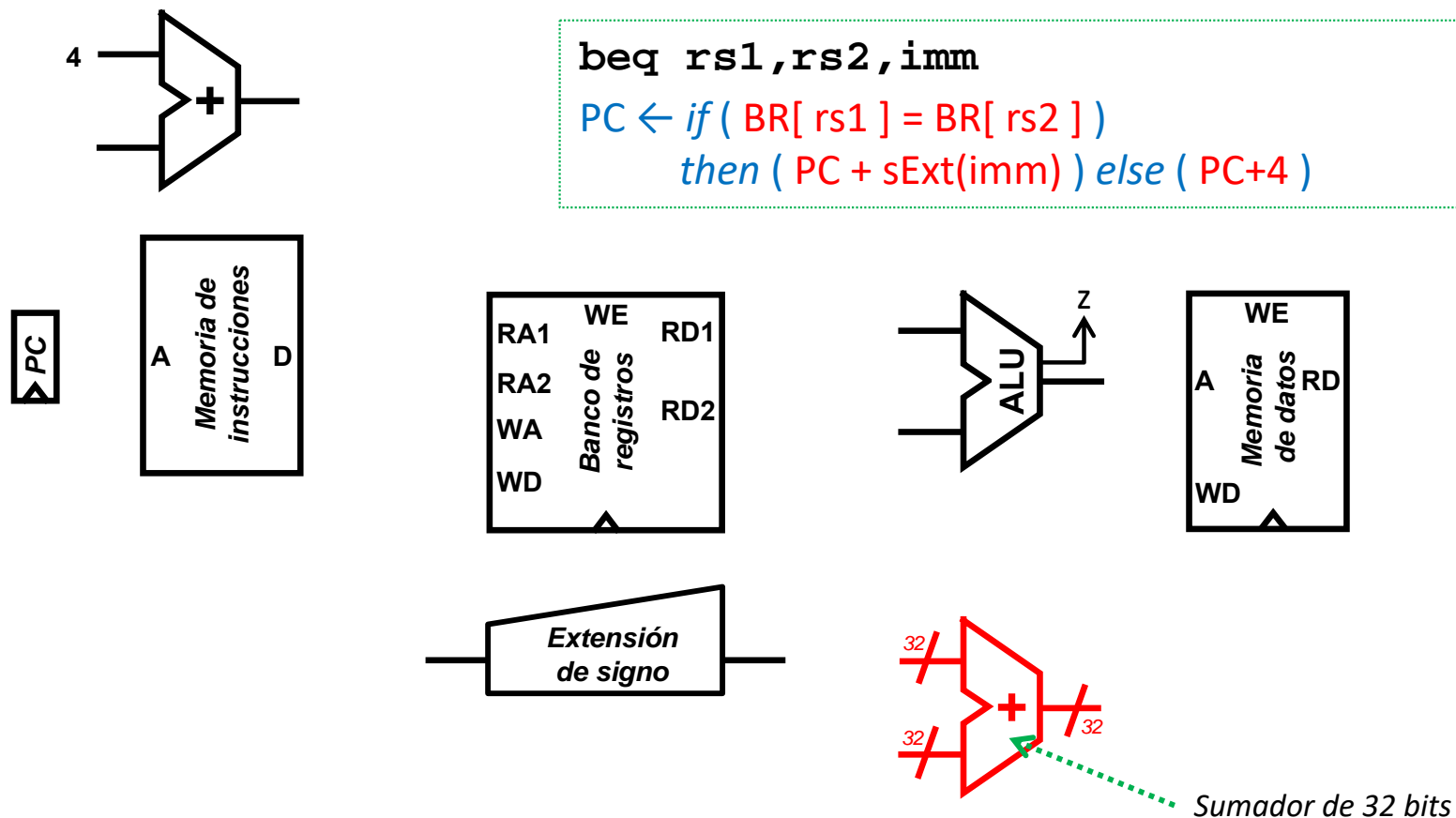
- Siempre **suma 4** porque la Memoria es direccionable por bytes y las **instrucciones son de 32 bits** (4 bytes).



Diseño de la ruta de datos

Elementos funcionales (iii)

- Dispondrá de otro **sumador adicional** para **calcular direcciones de salto**
 - En un procesador monociclo, las instrucciones tipo **beq** en **mismo ciclo de reloj** operan en la ALU, efectúan **PC+4** y **calculan la dirección de salto**.

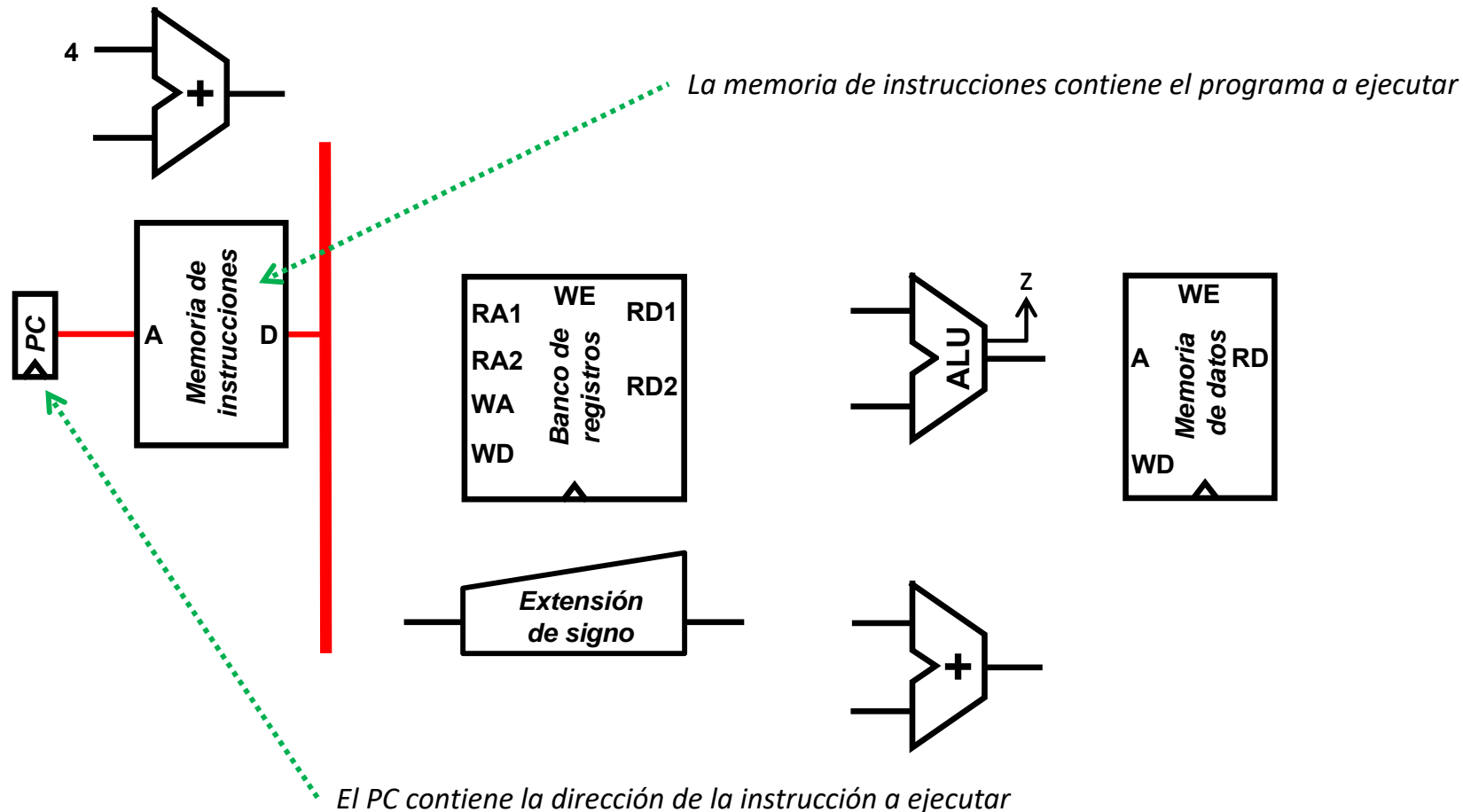




Diseño de la ruta de datos

Lectura de la instrucción

- La ejecución de toda instrucción comienza con su lectura de Memoria.
 - El PC y la Memoria de Instrucciones se conectan para leer la instrucción a ejecutar (aquella cuya dirección está almacenada en el PC).



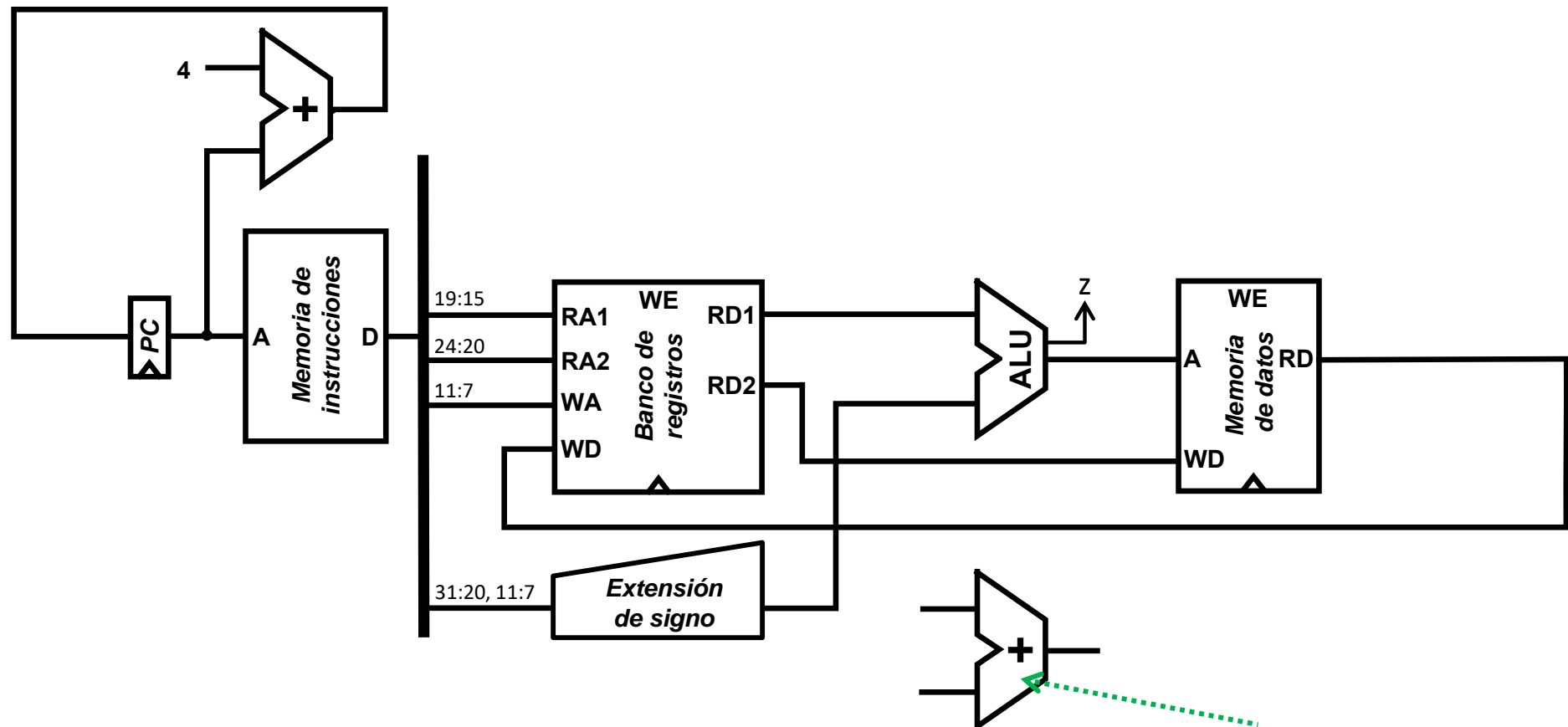
Este sumador solo se usa para calcular direcciones de salta en instrucciones jal/beq



Diseño de la ruta de datos

Ruta de datos para instrucciones **lw/sw**

- Esta ruta de datos puede ejecutar **cualquier secuencia** de instrucciones **lw** y/o **sw**, la seguiremos ampliando.



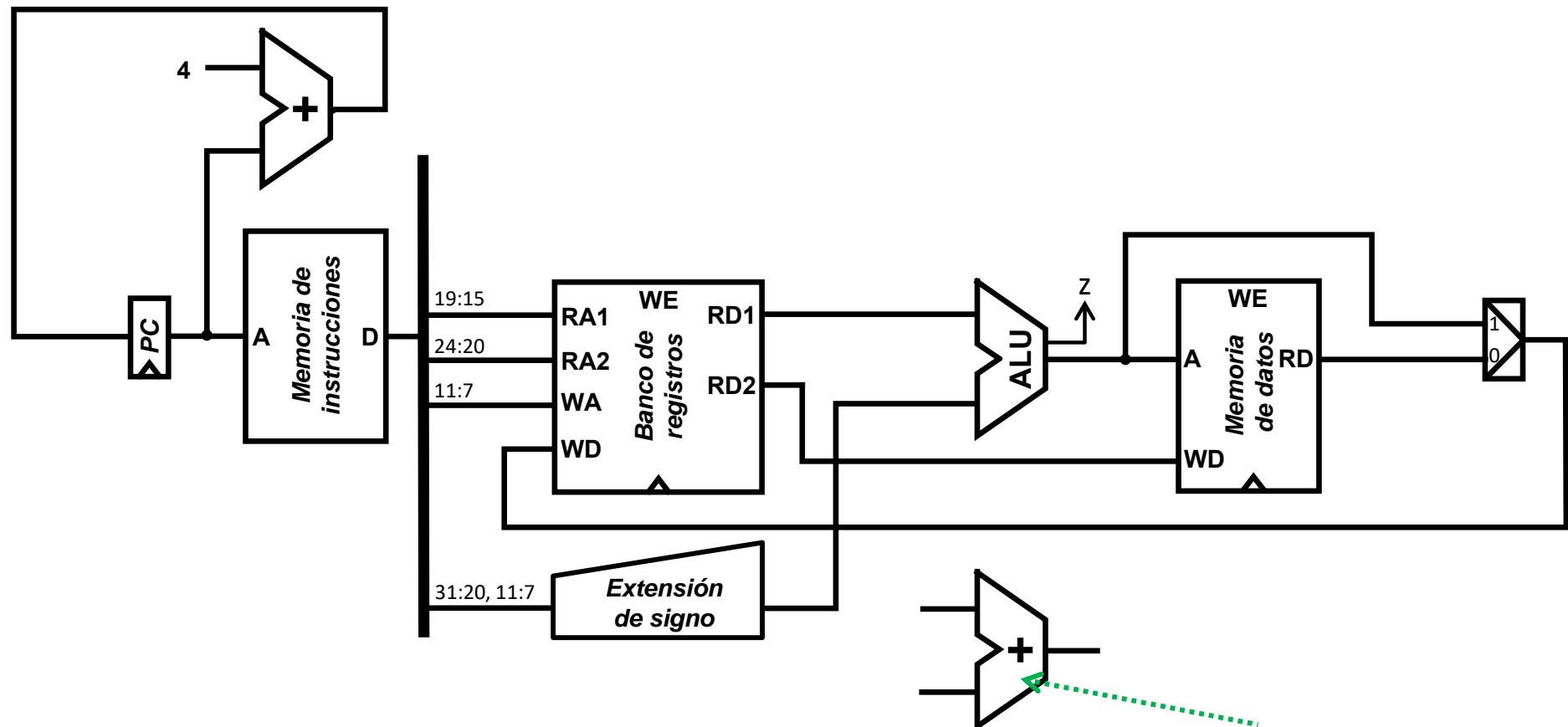
Este sumador solo se usa para calcular direcciones de salta en instrucciones **jal/beq**



Diseño de la ruta de datos

Ruta de datos para instrucciones **lw/sw/addi**

- Esta ruta de datos puede ejecutar **cualquier secuencia** de instrucciones **lw, sw**, y/o **aritmético-lógica con operando inmediato**.



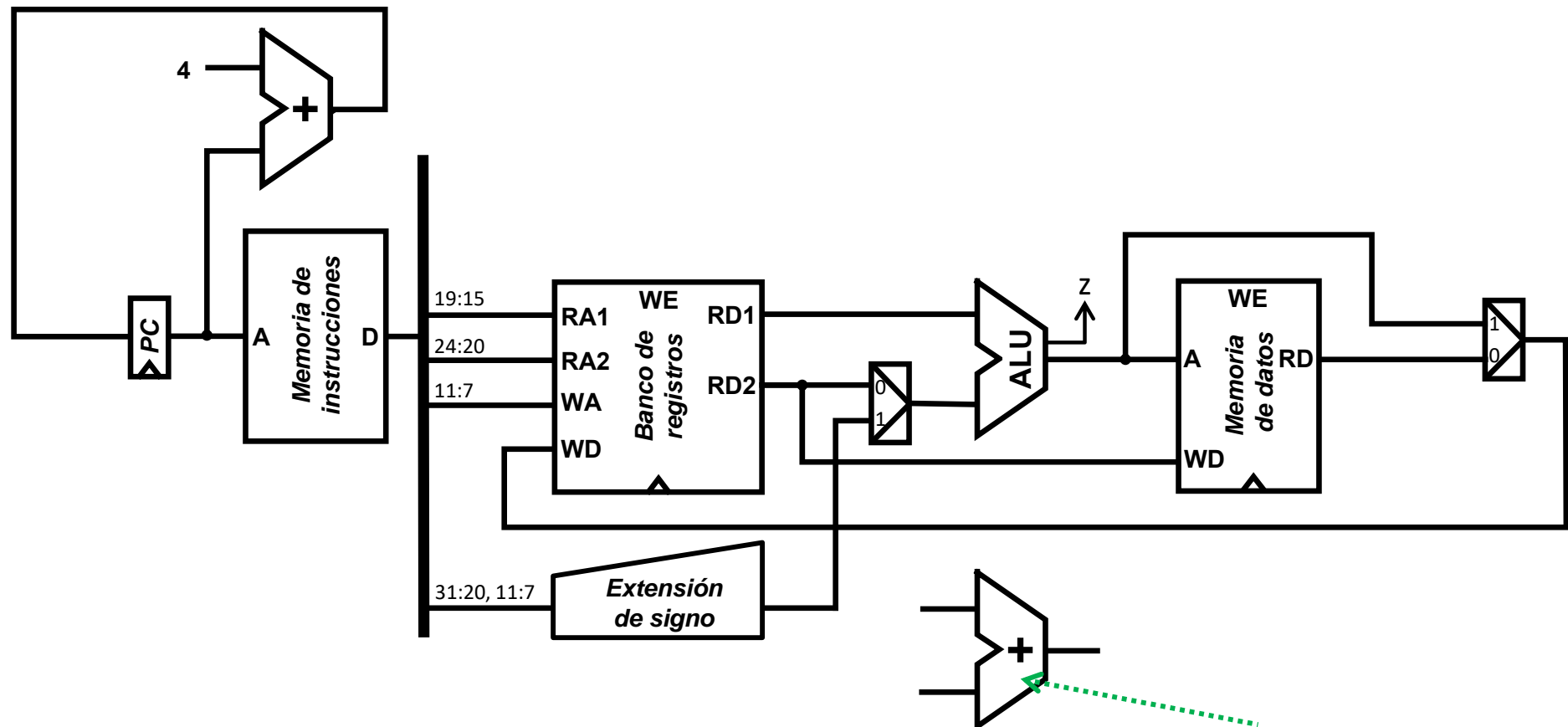
Este sumador solo se usa para calcular direcciones de salta en instrucciones **jal/beq**



Diseño de la ruta de datos

Ruta de datos para instrucciones **lw/sw/addi/add**

- Esta ruta de datos puede ejecutar **cualquier secuencia** de instrucciones **lw, sw, y/o aritmético-lógica**.



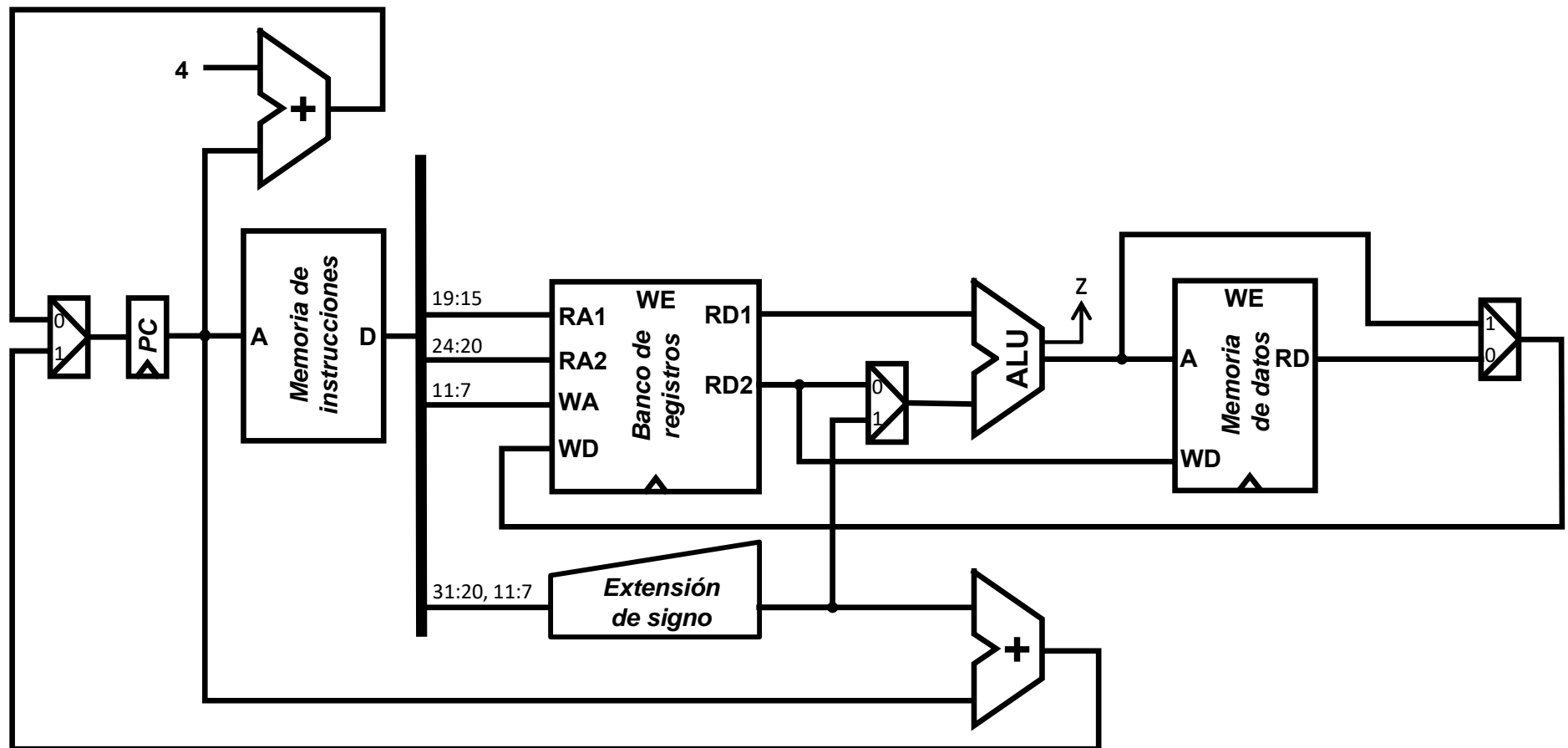
Este sumador solo se usa para calcular direcciones de salta en instrucciones **jal/beq**



Diseño de la ruta de datos

Ruta de datos para instrucciones **lw/sw/addi/add/beq**

- Esta ruta de datos puede ejecutar **cualquier secuencia** de instrucciones **lw**, **sw**, **aritmético-lógica**. y/o de salto condicional.

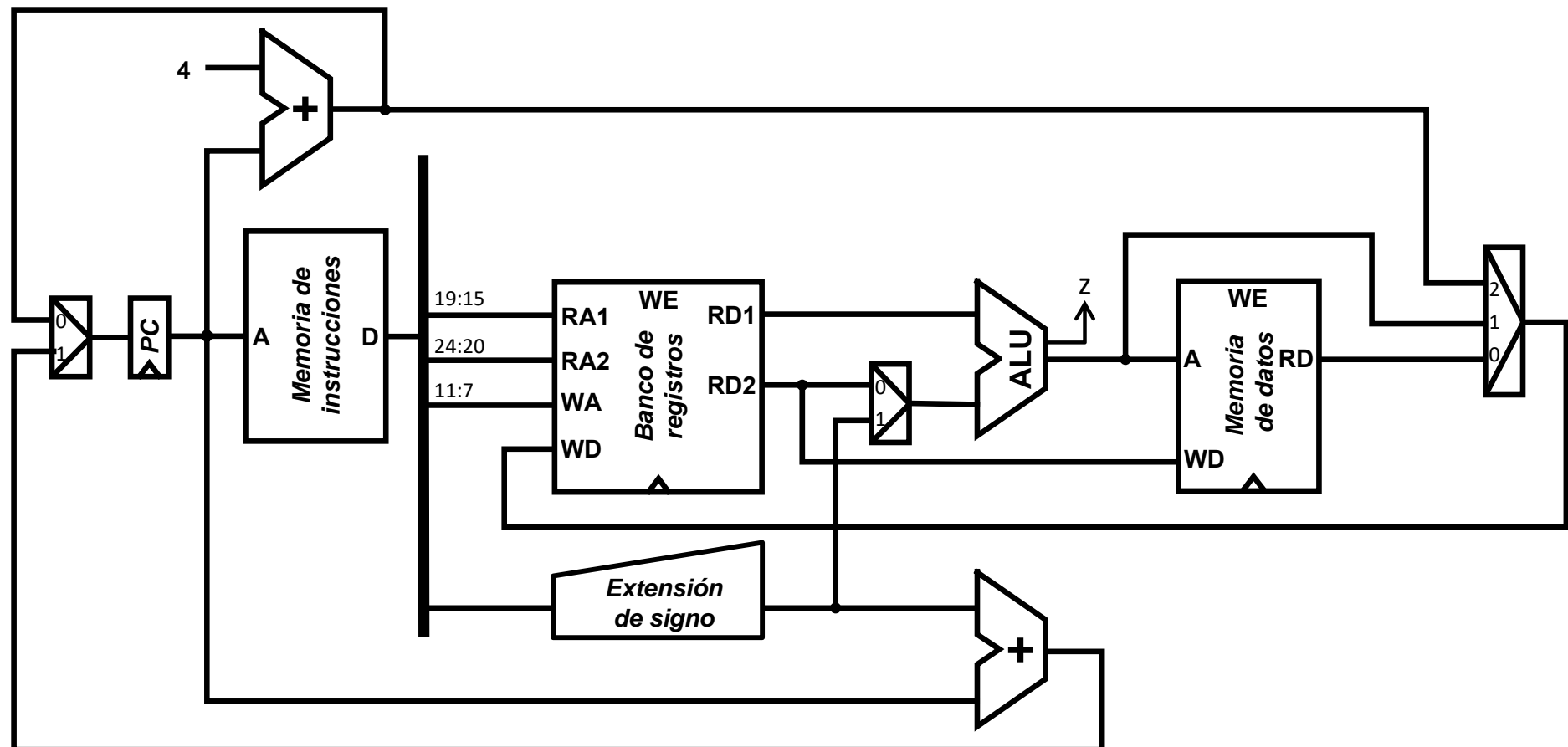




Diseño de la ruta de datos

Ruta de datos RISC-V reducido

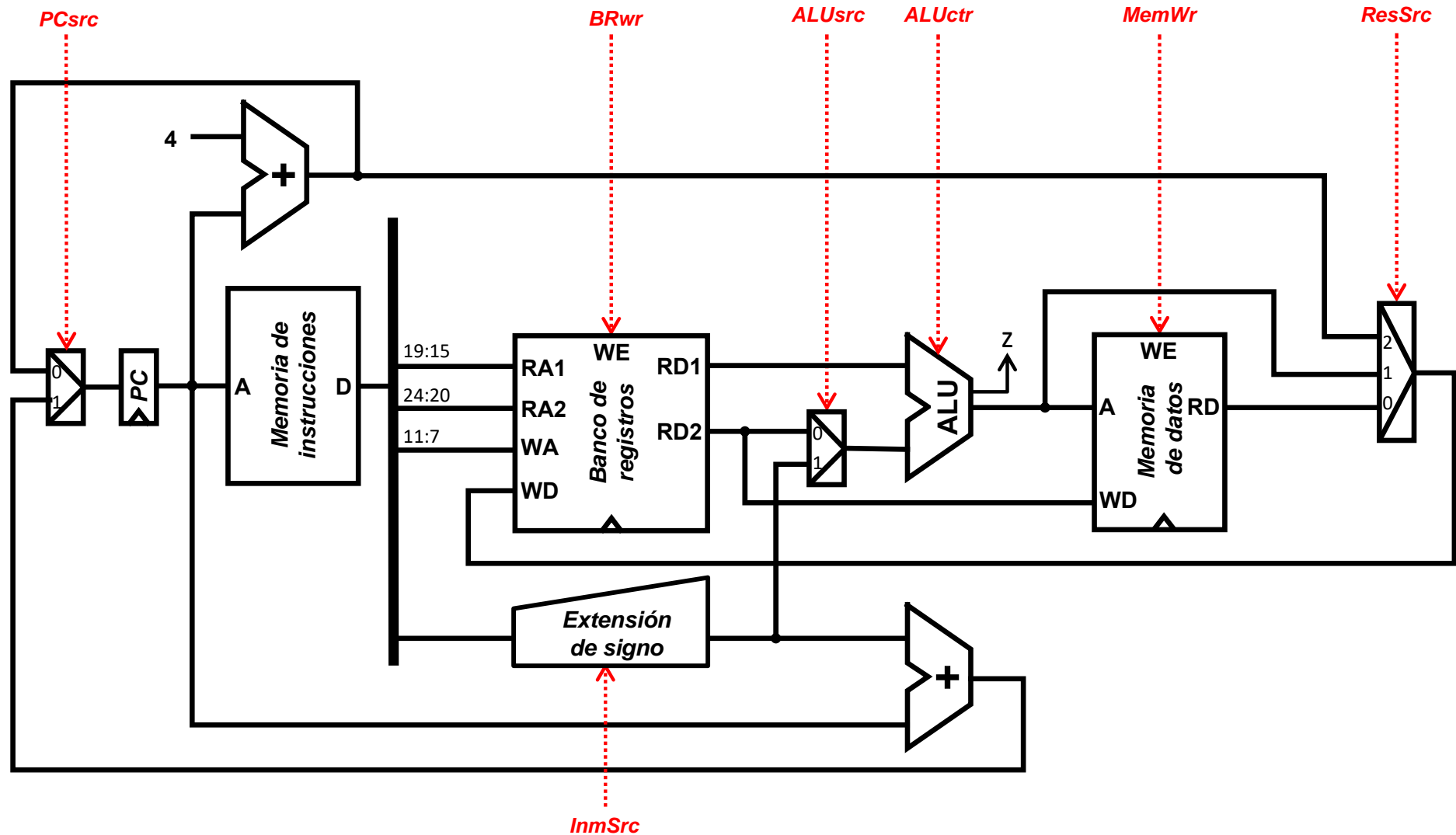
- Esta ruta de datos puede ejecutar **cualquier secuencia** de instrucciones del **repertorio del RISC-V reducido**.





Diseño de la ruta de datos

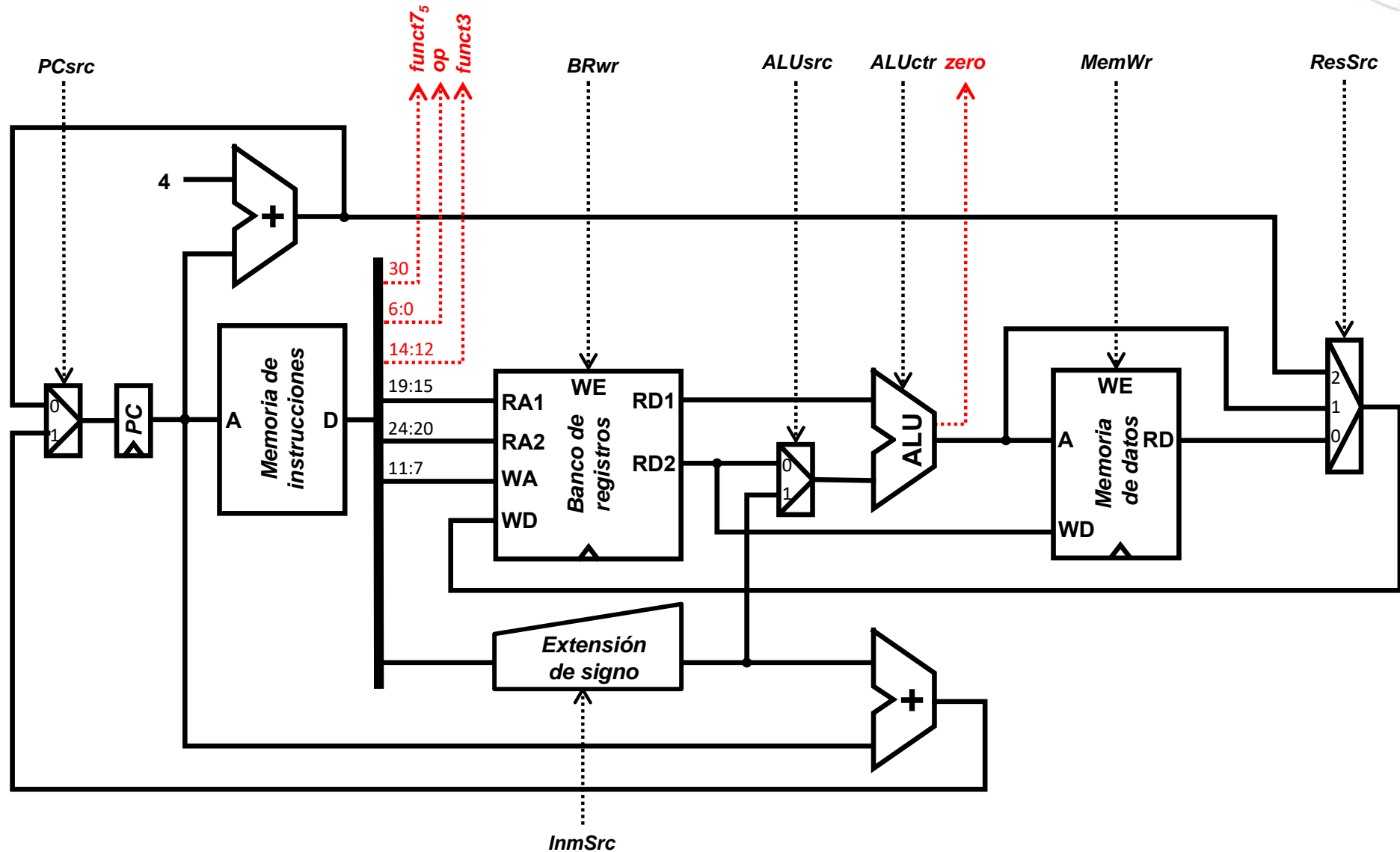
Señales de control





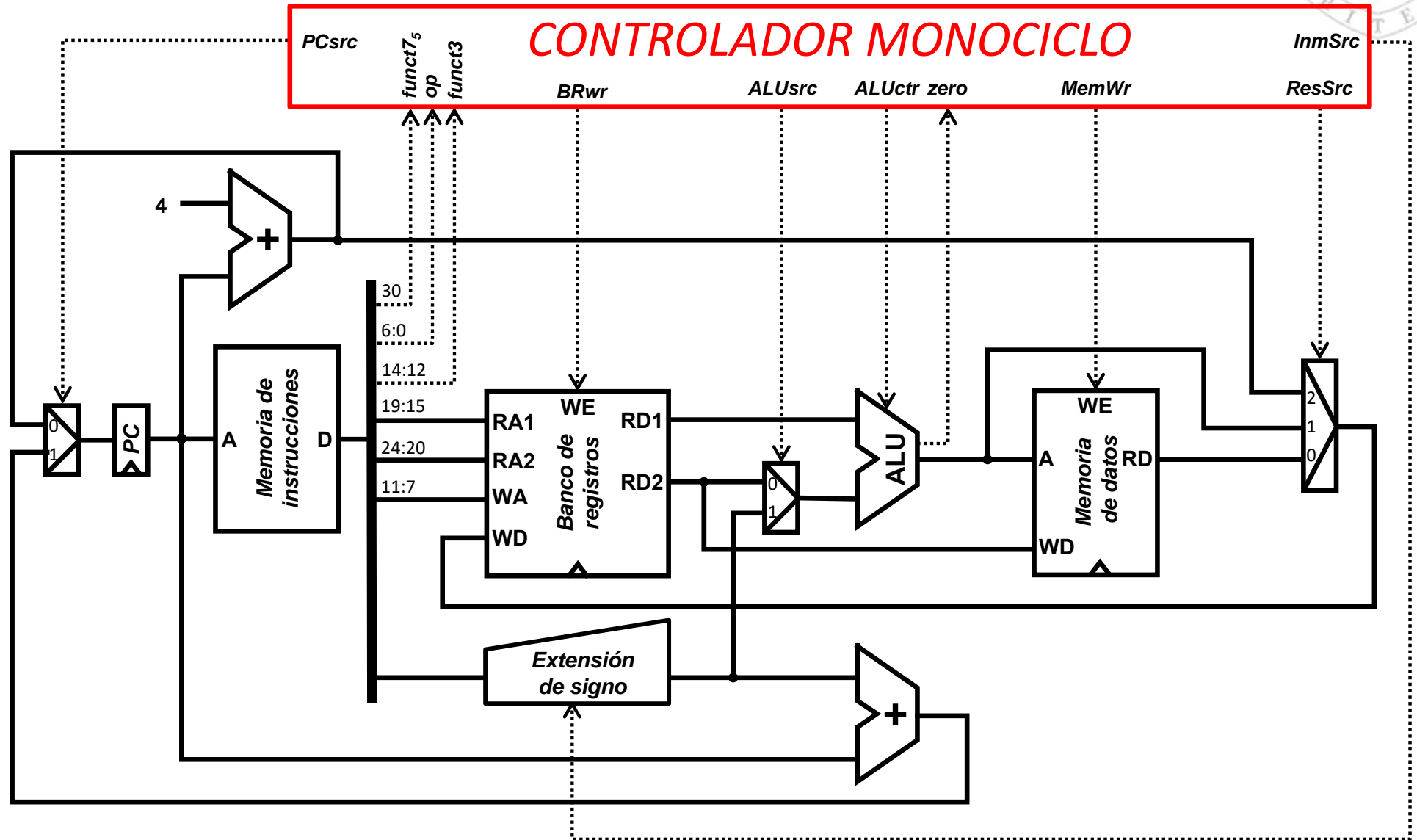
Diseño de la ruta de datos

Señales de estado



Diseño de la ruta de datos

Conexión con el controlador



Diseño de la ruta de datos

Conexión del reloj y reset

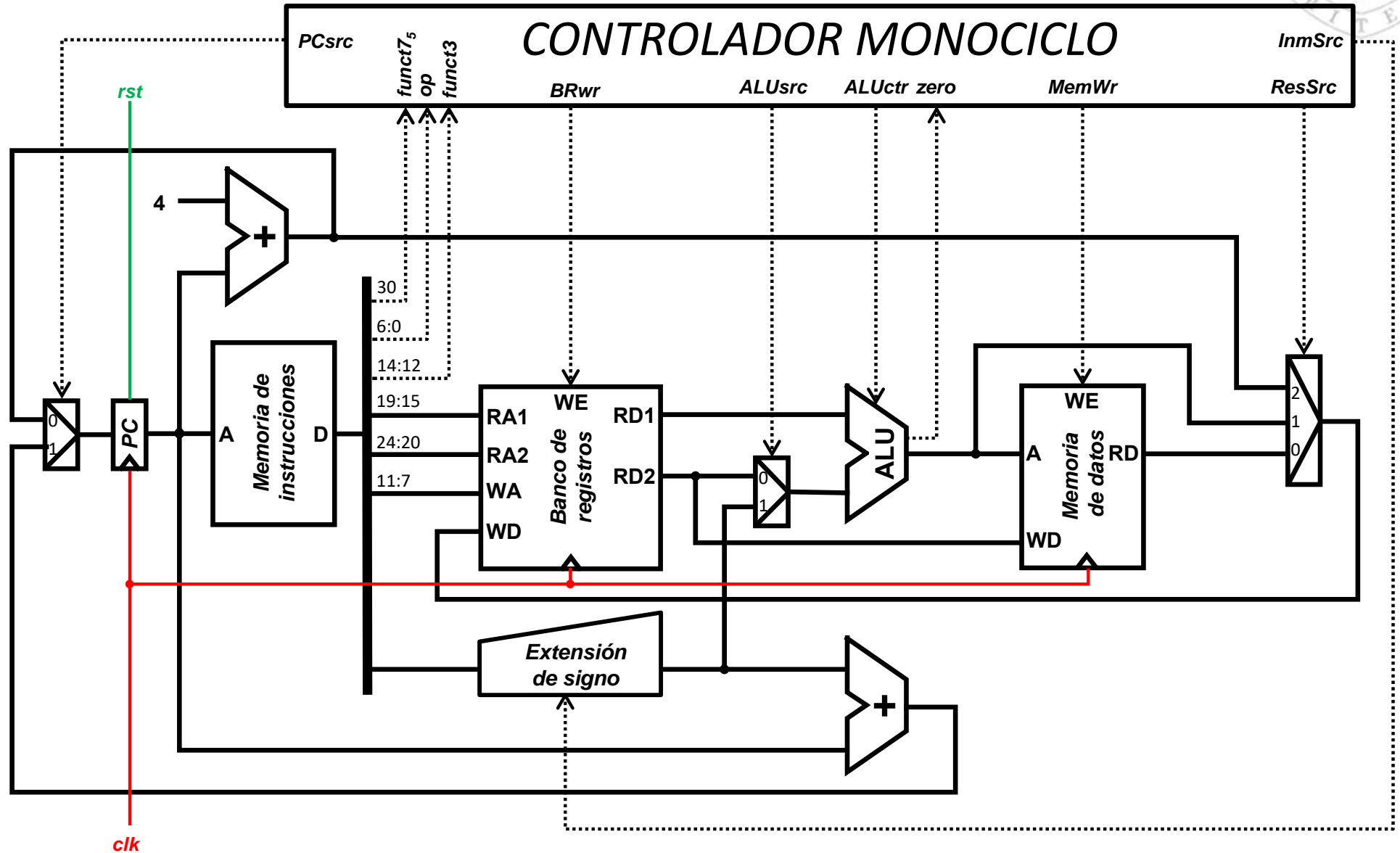


versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

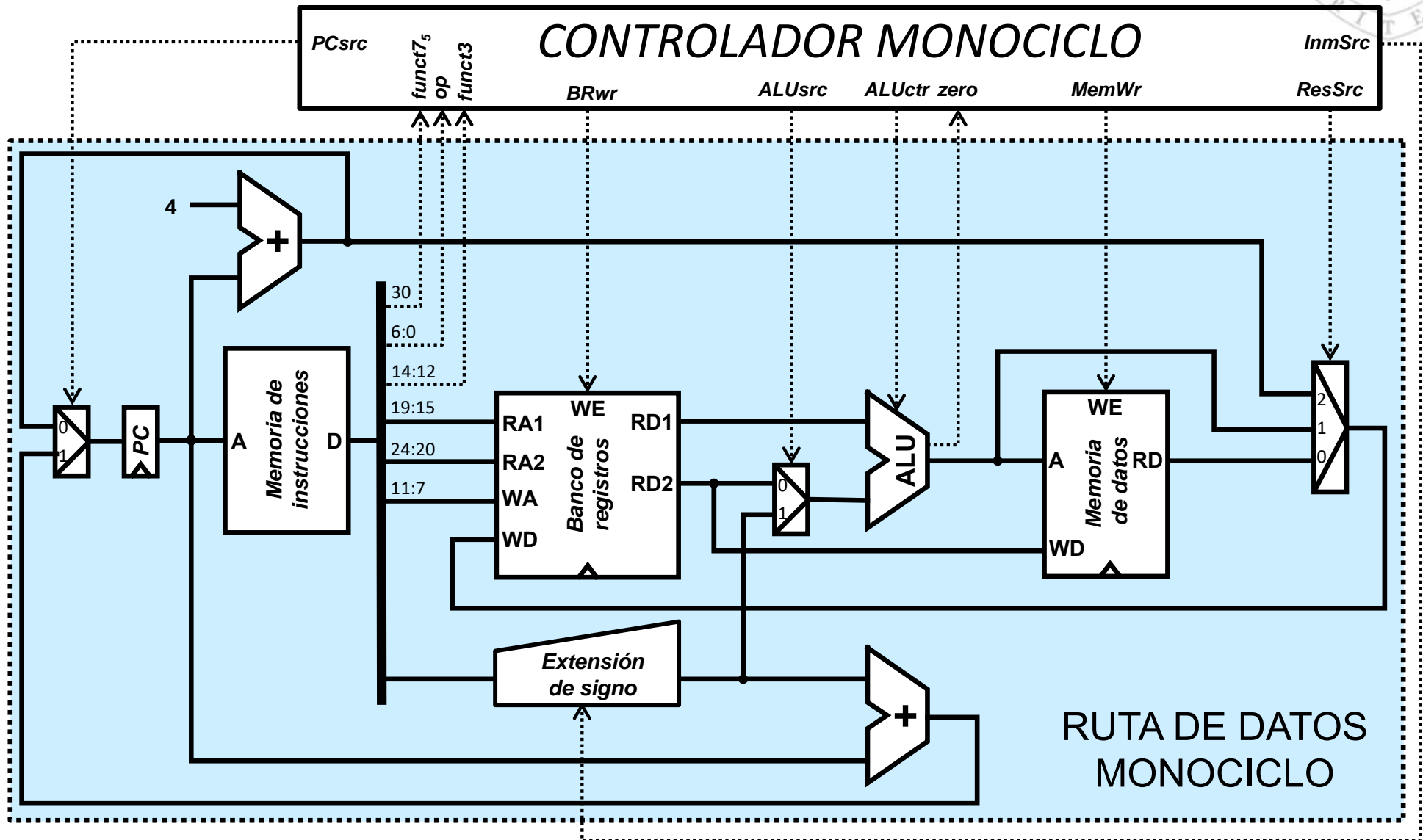
26





Diseño de la ruta de datos

Estructura del sistema completo

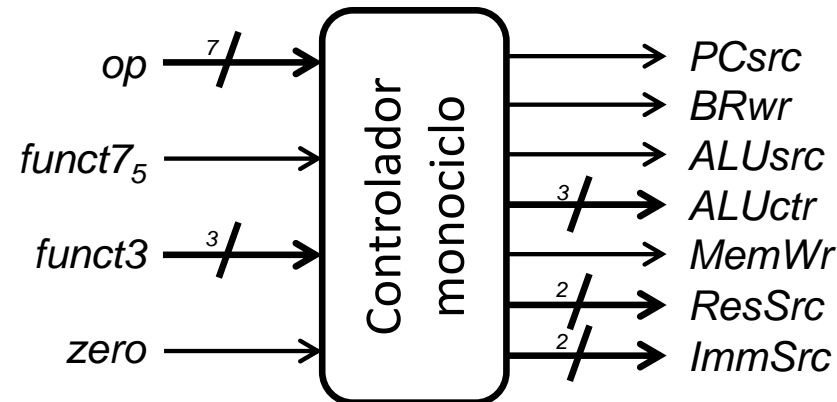




Diseño del controlador

Estructura del controlador (i)

- Al ser un procesador monociclo, el controlador será un decodificador puramente combinacional.

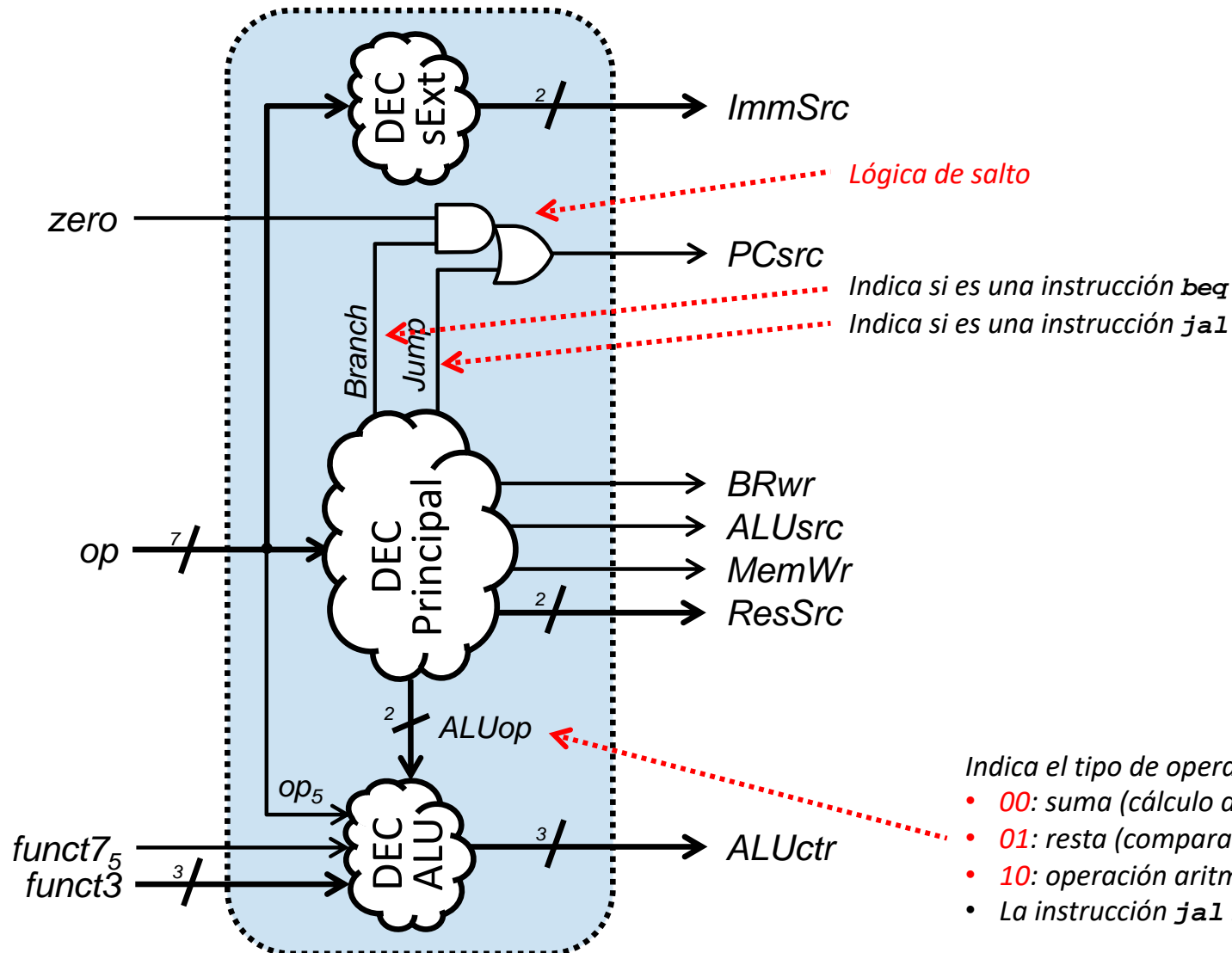


- Para simplificar su diseño estará formado por 4 subcircuitos:
 - **Decodificador principal**: decodifica el código de operación **op** para controlar la actividad global de la ruta de datos.
 - **Decodificador local al extensor de signo**: decodifica el código de operación **op** para controlar el tipo de extensión a realizar.
 - **Decodificador local a la ALU**: decodifica los códigos de operación aritmética (**funct3** y **funct7**) para controlar la operación que realiza la ALU.
 - **Lógica de salto**: determina si el procesador salta o no.



Diseño del controlador

Estructura del controlador (ii)

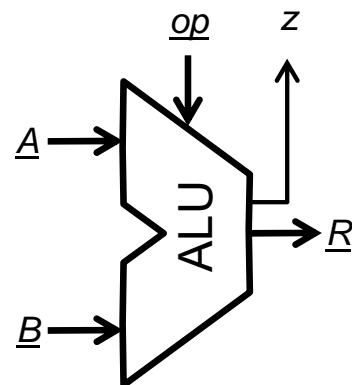
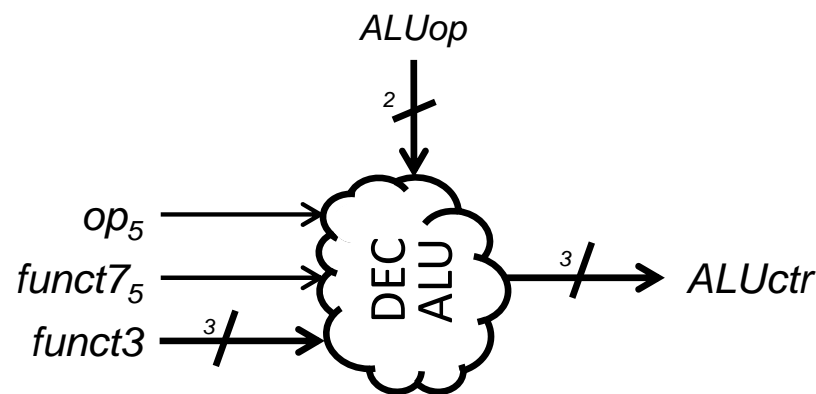




Diseño del controlador

Diseño del controlador: DEC local a la ALU

- Este subcircuito indica a la ALU el tipo de operación que deber realizar.
 - Adapta la codificación de la operación en la instrucción a la codificación de operaciones de la ALU.



op	R
000	$\underline{A} + \underline{B}$
001	$\underline{A} - \underline{B}$
010	$\underline{A} \& \underline{B}$
011	$\underline{A} \underline{B}$
101	if ($\underline{A} < \underline{B}$) then 1 else 0
resto	—

Tabla de verdad

ALUop	op ₅	funct7 ₅	funct3	ALUctr
00 ^(sumar)	X	X	XXX	000 ^(A + B)
01 ^(restar)	X	X	XXX	001 ^(A - B)
10 ^(operar)	0	X	000 ^(addi)	000 ^(A + B)
10 ^(operar)	1	0	000 ^(add)	000 ^(A + B)
10 ^(operar)	1	1	000 ^(sub)	001 ^(A - B)
10 ^(operar)	X	X	010 ^(slt/slti)	101 ^(A < B)
10 ^(operar)	X	X	110 ^(or/ori)	011 ^(A B)
10 ^(operar)	X	X	111 ^(and/andi)	010 ^(A & B)



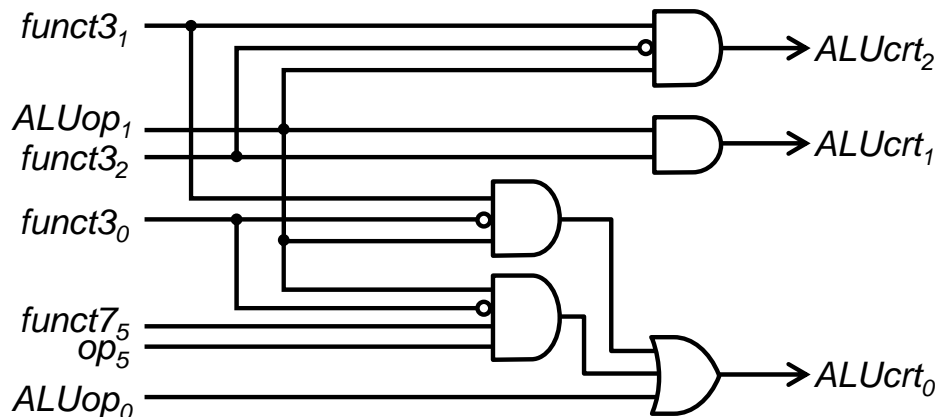
Diseño del controlador

Diseño del controlador: DEC local a la ALU

- Este subcircuito indica a la ALU el tipo de operación que deber realizar.
 - Adapta la codificación de la operación en la instrucción a la codificación de operaciones de la ALU.

Tabla de verdad

ALUop	op ₅	func7 ₅	func3	ALUctr
00 ^(sumar)	X	X	XXX	000 ^(A + B)
01 ^(restar)	X	X	XXX	001 ^(A - B)
10 ^(operar)	0	X	000 ^(addi)	000 ^(A + B)
10 ^(operar)	1	0	000 ^(add)	000 ^(A + B)
10 ^(operar)	1	1	000 ^(sub)	001 ^(A - B)
10 ^(operar)	X	X	010 ^(slt/slti)	101 ^(A < B)
10 ^(operar)	X	X	110 ^(or/ori)	011 ^(A B)
10 ^(operar)	X	X	111 ^(and/andi)	010 ^(A & B)





Diseño del controlador

Diseño del controlador: DEC local al Extensor de Signo

- Esté subcircuito indica al Extensor de Signo el tipo de extensión que deber realizar.

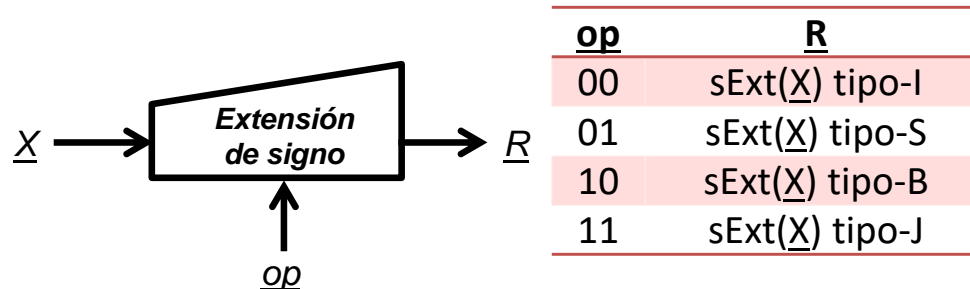
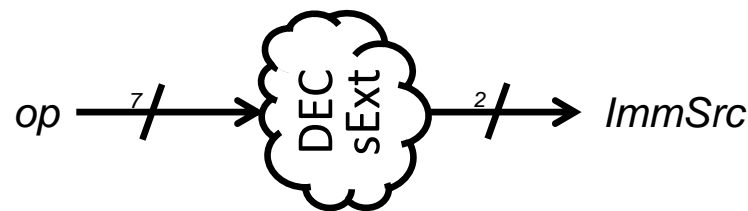


Tabla de verdad

Op	ImmSrc
0000011 ^(lw)	00 ^(tipo-I)
0100011 ^(sw)	01 ^(tipo-S)
0010011 ^(tipo-I)	00 ^(tipo-I)
0110011 ^(tipo-R)	—
1100011 ^(beq)	10 ^(tipo-B)
1101111 ^(jal)	11 ^(tipo-J)



Diseño del controlador

Diseño del controlador: DEC local al Extensor de Signo

- Esté subcircuito indica al Extensor de Signo el tipo de extensión que deber realizar.

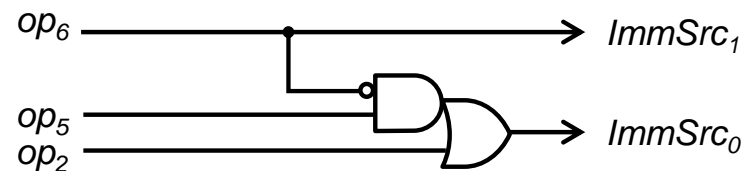


Tabla de verdad

Op	ImmSrc
0000011 ^(lw)	00 ^(tipo-I)
0100011 ^(sw)	01 ^(tipo-S)
0010011 ^(tipo-I)	00 ^(tipo-I)
0110011 ^(tipo-R)	—
1100011 ^(beq)	10 ^(tipo-B)
1101111 ^(jal)	11 ^(tipo-J)



Diseño del controlador

Diseño del controlador: DEC principal

- Este subcircuito gobierna el comportamiento general del procesador.

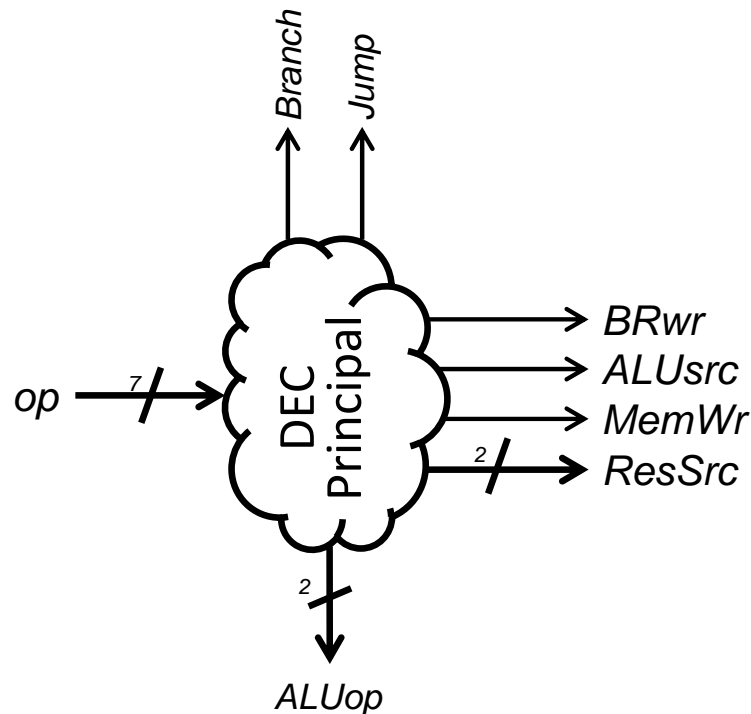


Tabla de verdad

op	Branch	Jump	BRwr	ALUsrc	ALUOp	MemWr	ResSrc
0000011 ^(lw)	0	0	1	1	00 ^(sumar)	0	00
0100011 ^(sw)	0	0	0	1	00 ^(sumar)	1	—
0010011 ^(tipo-I)	0	0	1	1	10 ^(operar)	0	01
0110011 ^(tipo-R)	0	0	1	0	10 ^(operar)	0	01
1100011 ^(beq)	1	0	0	0	01 ^(restar)	0	—
1101111 ^(jal)	0	1	1	—	—	0	10



Diseño del controlador

Diseño del controlador: DEC principal

- Este subcircuito gobierna el comportamiento general del procesador.

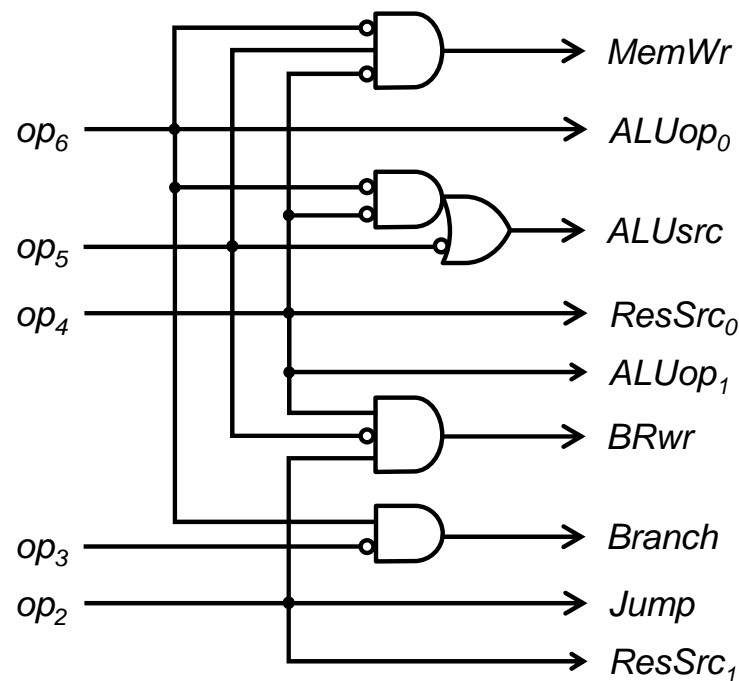


Tabla de verdad

op	Branch	Jump	BRwr	ALUsrc	ALUop	MemWr	ResSrc
0000011 ^(lw)	0	0	1	1	00 ^(sumar)	0	00
0100011 ^(sw)	0	0	0	1	00 ^(sumar)	1	—
0010011 ^(tipo-I)	0	0	1	1	10 ^(operar)	0	01
0110011 ^(tipo-R)	0	0	1	0	10 ^(operar)	0	01
1100011 ^(beq)	1	0	0	0	01 ^(restar)	0	—
1101111 ^(jal)	0	1	1	—	—	0	10



Procesador monociclo

Simulación (i)

0x1000 ...
 0x1004 **L7:** lw x6, -4(x9)
 0x1008 sw x6, 8(x9)
 0x100C or x4, x5, x6
 beq x4, x4, **L7**
 ...

-12

...
 0xFFC4A303
 0x0064A423
 0x0062E233
 0xFE420AE3
 ...

x5	6
x9	0x2004
Mem[0x2000]	10
PC	0x1000

imm _{11:0}	rs1	funct3	rd	op	
111111111100	01001	010	00110	0000011	0xFFC4A303
imm _{11:5}	rs2	rs1	funct3	imm _{4:0}	op
0000000	00110	01001	010	01000	0100011
					0x0064A423
funct7	rs2	rs1	funct3	rd	op
0000000	00110	00101	110	00100	0110011
					0x0062E233
imm _{12,10:5}	rs2	rs1	funct3	imm _{4:1,11}	op
1111111	00100	00100	000	10101	1100011
					0xFE420AE3

31 25 24 20 19 15 14 12 11 7 6 0



Procesador monociclo

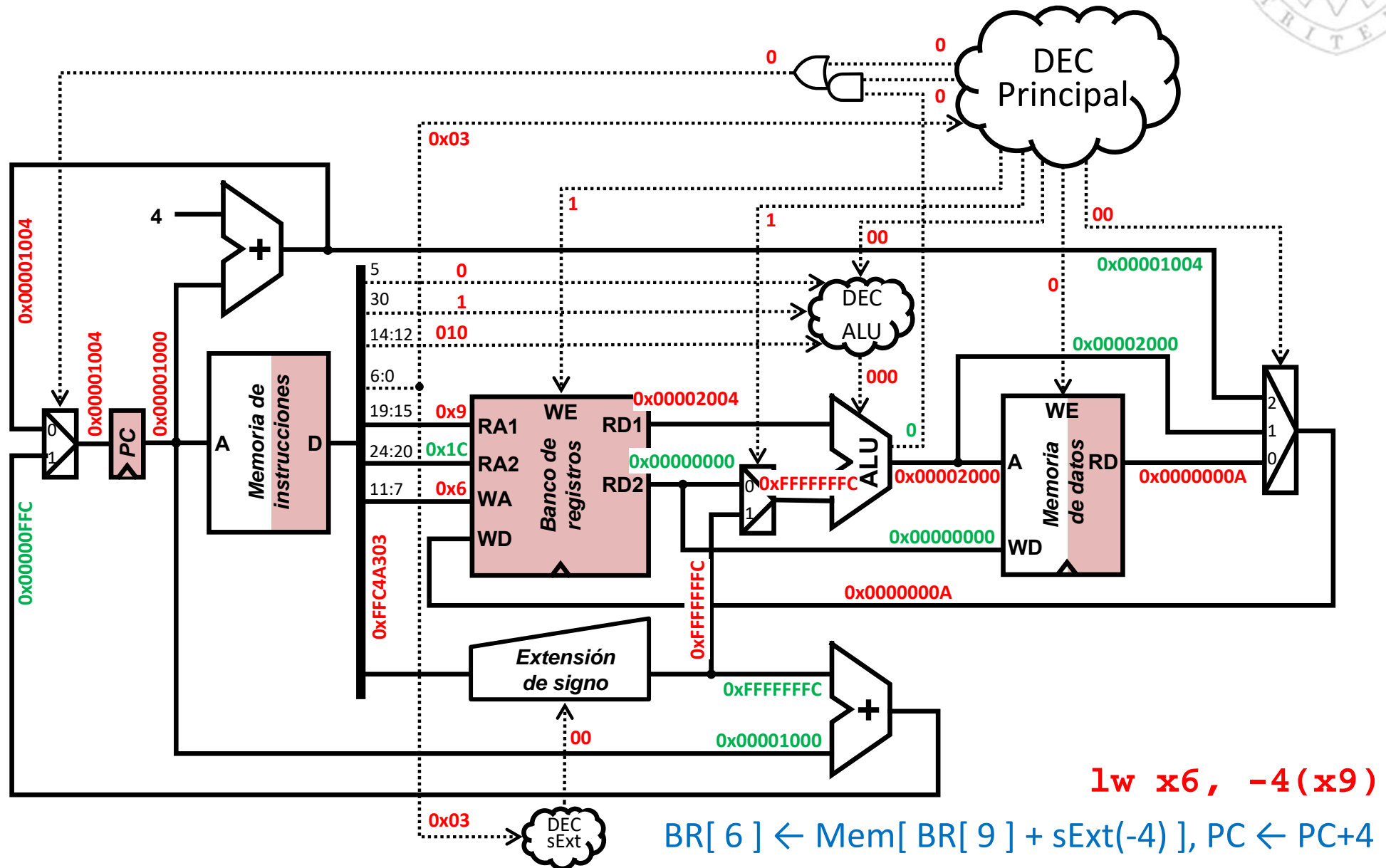
Simulación: 1er ciclo

versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

37

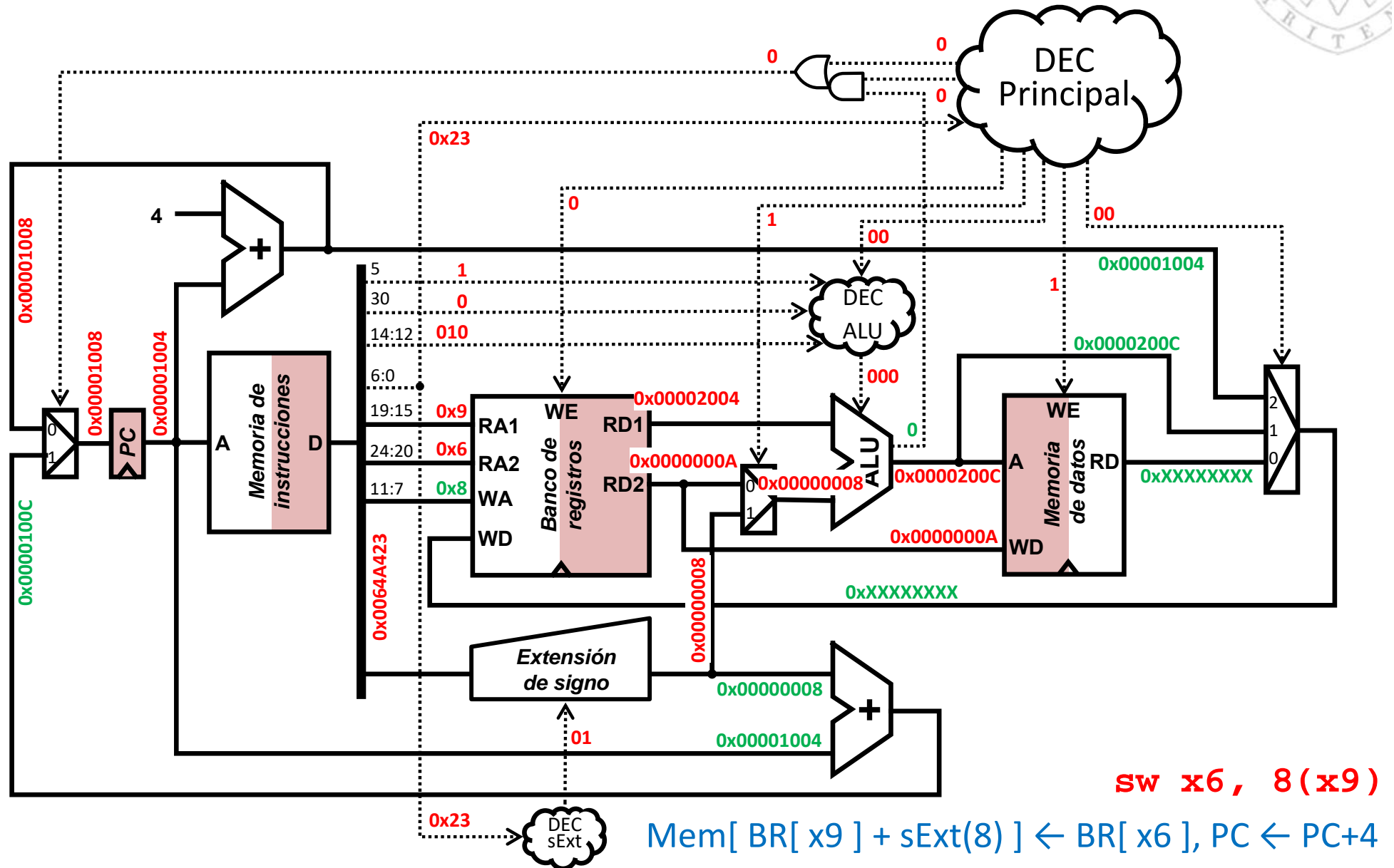


versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

38

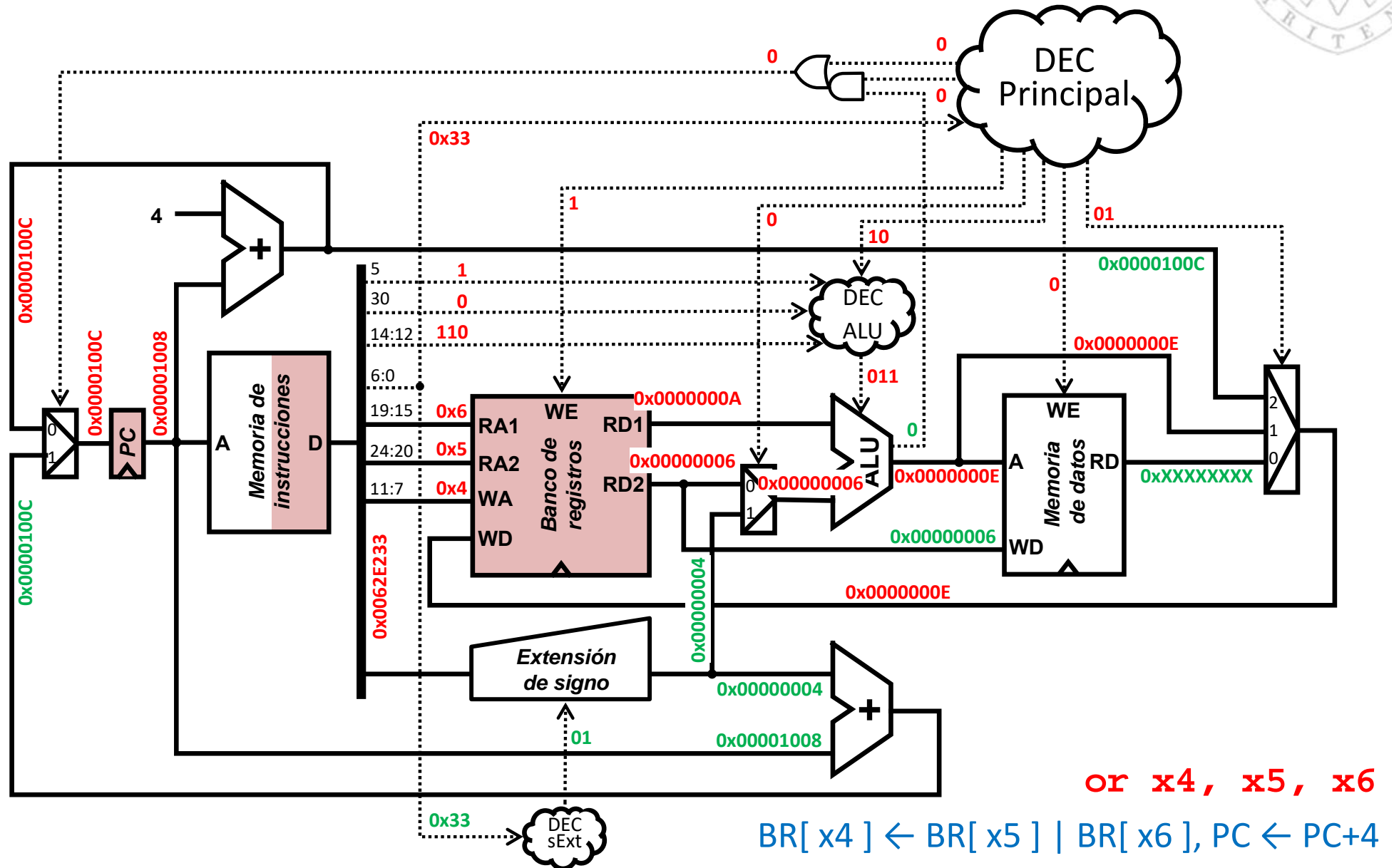


versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

39





Procesador monociclo

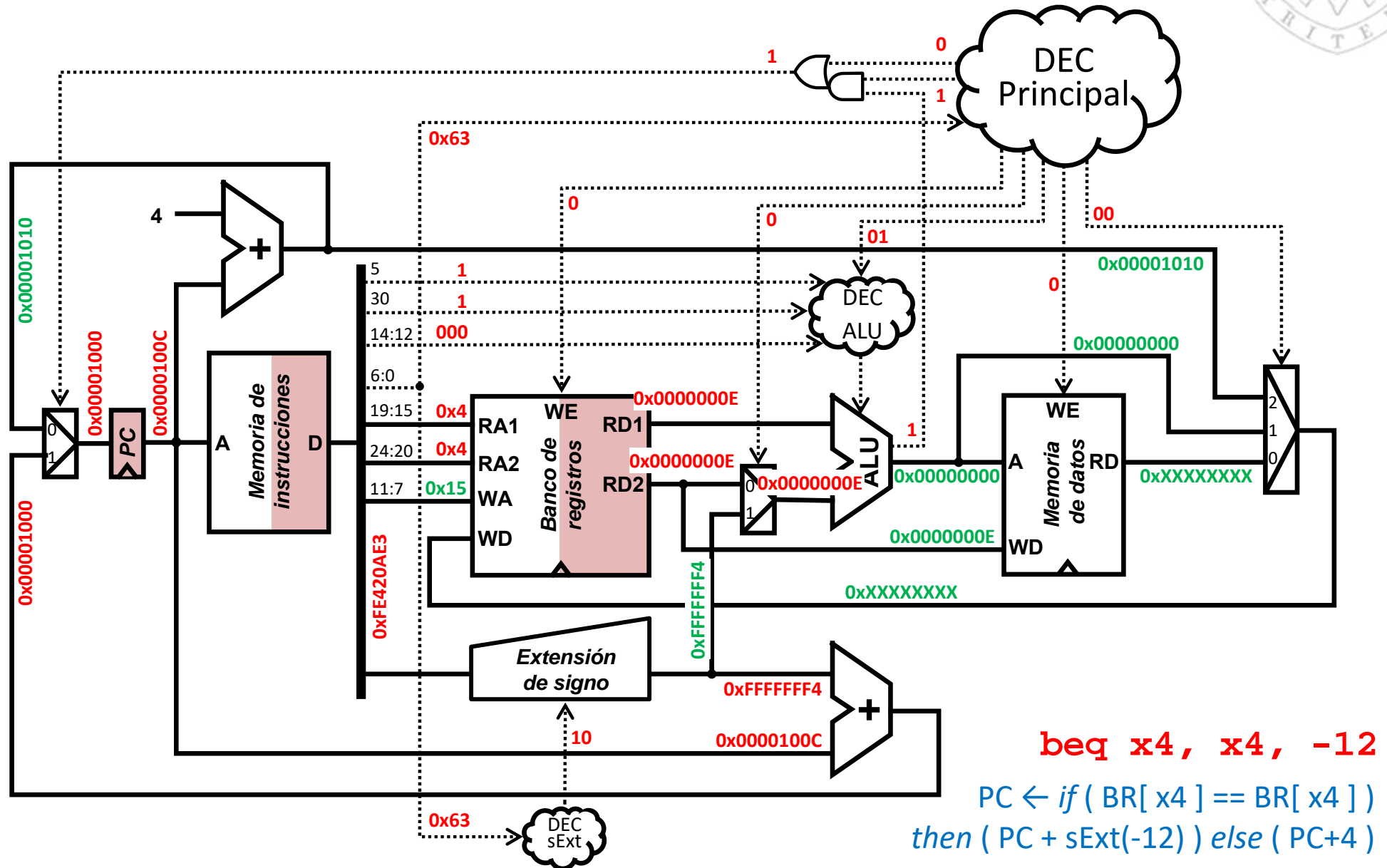
Simulación: 4o. ciclo

versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

40





Procesador monociclo

Coste y tiempo de ciclo (CMOS 90 nm)

versión 15/01/23

$$area = 59181 \mu m^2$$

$$t_{clk} = 27.6 ns$$

$$f_{clk} = \frac{1}{t_{clk}} = \frac{1}{27.6 \cdot 10^{-9}s} = 36.2 MHz$$

transferencia entre registros	instr.	camino crítico
PC \leftarrow PC+4	varias	9692 ps
BR[rd] \leftarrow Mem[BR[rs1] + sExt(imm)]	lw	27616 ps
Mem[BR[rs1] + sExt(imm)] \leftarrow BR[rs2]	sw	26661 ps
BR[rd] \leftarrow BR[rs1] op sExt(imm)	tipo I	19116 ps
BR[rd] \leftarrow BR[rs1] op BR[rs2]	tipo R	18928 ps
PC \leftarrow if (BR[rs1] = BR[rs2]) then (PC + sExt(imm)) else (PC+4)	beq	18547 ps
BR[rd] \leftarrow PC+4	jal	10073 ps
PC \leftarrow PC + sExt(imm)		17033 ps
	max.	27616 ps

...

L7:

lw	x6, -4(x9)	→	1 ciclo
sw	x6, 8(x9)	→	1 ciclo
or	x4, x5, x6	→	1 ciclo
beq	x4, x4, L7	→	+ 1 ciclo
...			

4 ciclos

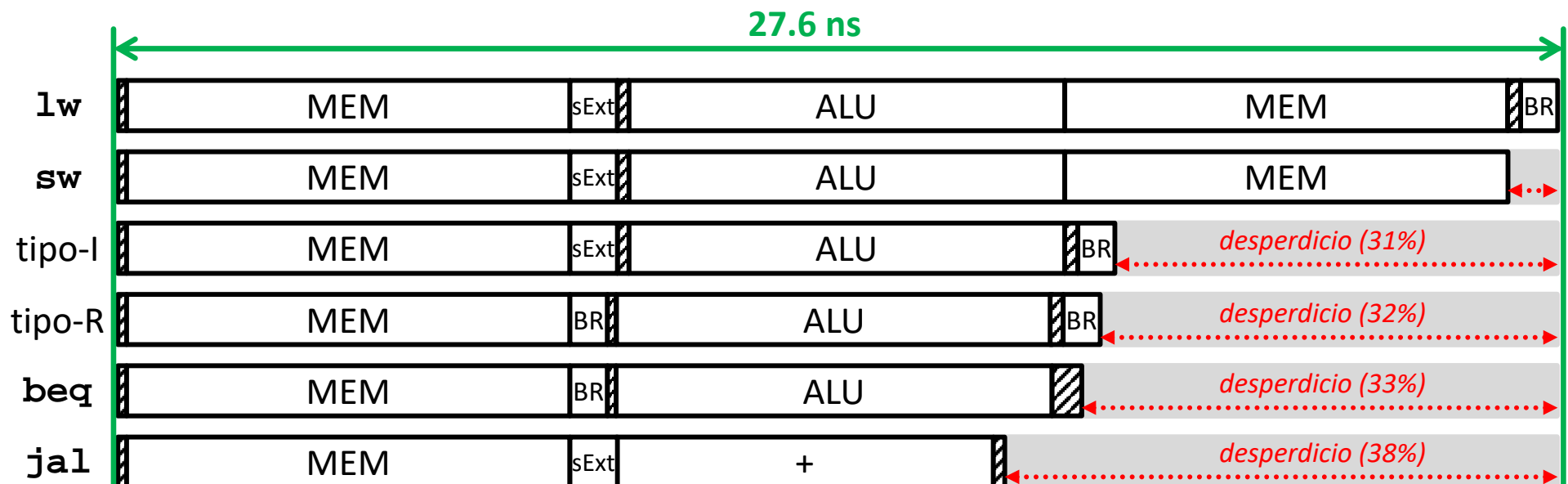
$$t_{ejec} = 4 \times 27.6 ns = 110.4 ns$$



Procesador monociclo

Conclusiones

- La **implementación monociclo** tiene algunos **problemas**:
 - El **tiempo de ciclo** viene dado por la **instrucción más lenta**
 - Todas las instrucciones tardan lo mismo en ejecutarse con independencia de su complejidad: se **desperdicia tiempo** en la ejecución de instrucciones rápidas.
 - En **repertorios reales** existen algunas **instrucciones muy largas**: memorias lentas, operaciones aritméticas complejas, modos de direccionamiento complejos...
 - **No es posible reutilizar hardware**:
 - Requiere una ALU y 2 sumadores, memoria de instrucciones y datos separada...

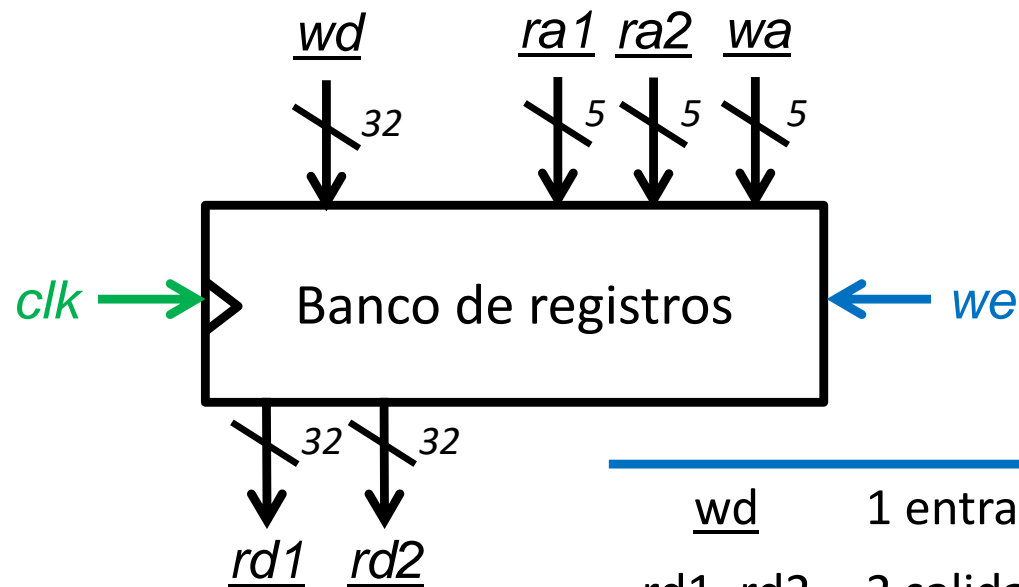




Aspectos tecnológicos

Diseño del Banco de Registros (i)

- El Banco de Registros contiene 32 registros de datos de 32 bits.



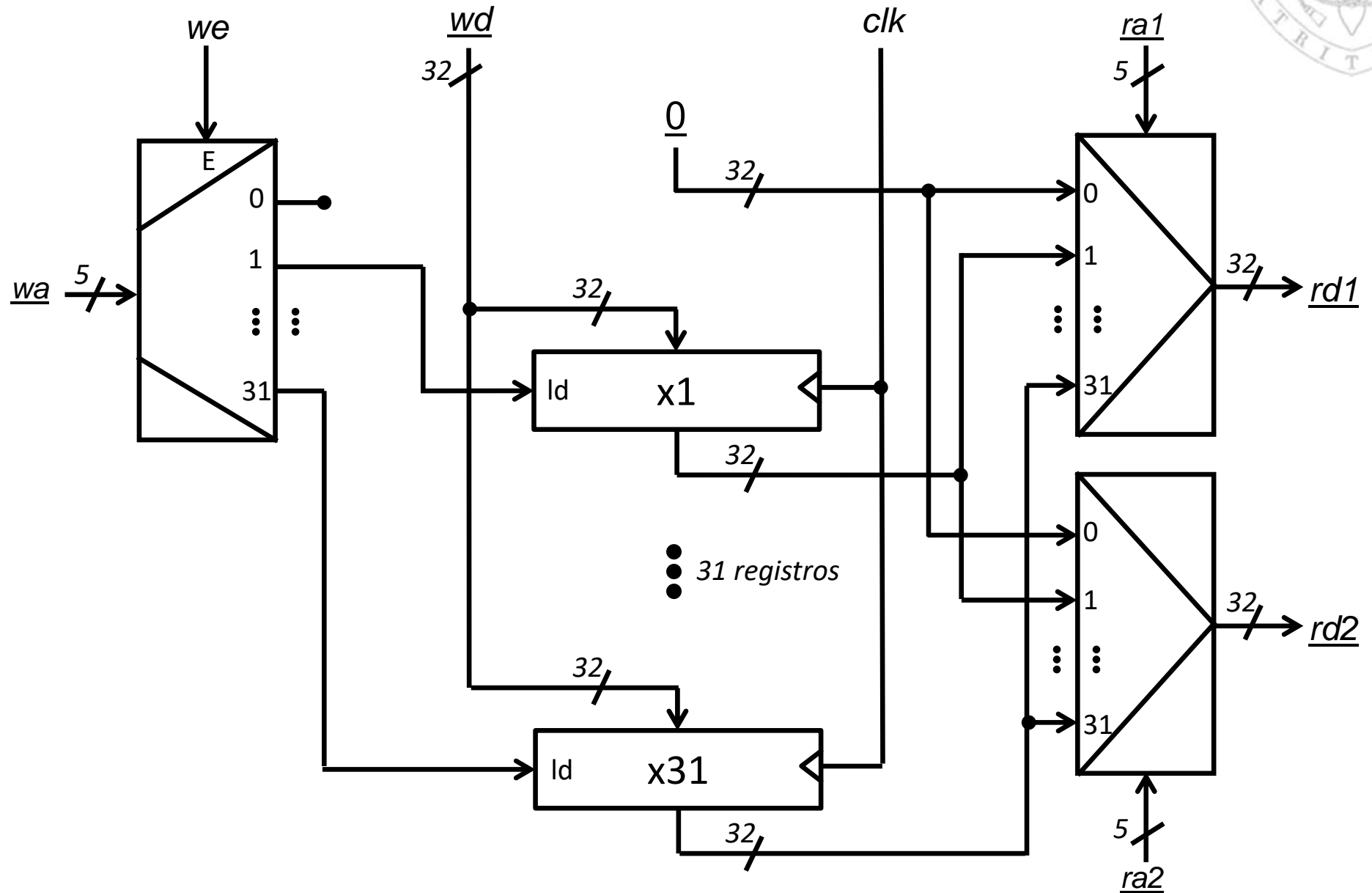
<u>wd</u>	1 entrada de datos de 32 bits
<u>rd1</u> , <u>rd2</u>	2 salidas de datos de 32 bits
<u>wa</u>	1 entrada de dirección de escritura de 5 bits
<u>ra1</u> , <u>ra2</u>	2 entradas de dirección de lectura de 5 bits
<u>we</u>	1 entrada de capacitación de escritura
<u>clk</u>	1 entrada de reloj





Aspectos tecnológicos

Diseño del Banco de Registros (ii)



Aspectos tecnológicos

Diseño de la Memoria (i)

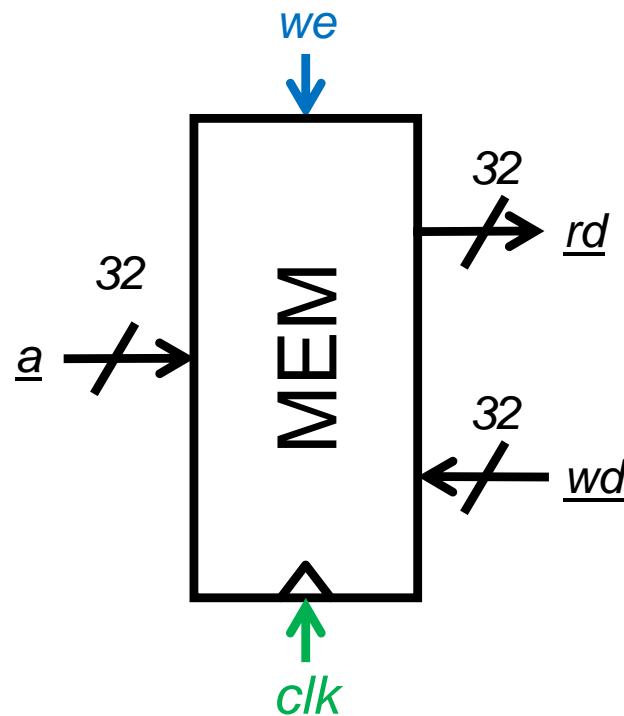


versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

45



RAM $2^{32} \times 8 = 2^{30} \times 32$
(2^{30} palabras de 32 bits)

-
- wd 1 entrada de datos de 32 bits
 - rd 1 salidas de datos de 32 bits
 - a 1 entrada de dirección de 32 bits
 - we 1 entrada de capacitación de escritura
 - clk 1 entrada de reloj
-

*Memoria con doble puerto de datos,
direccionable por bytes
y con ordenamiento little-endian*



Aspectos tecnológicos

Diseño de la Memoria (i)



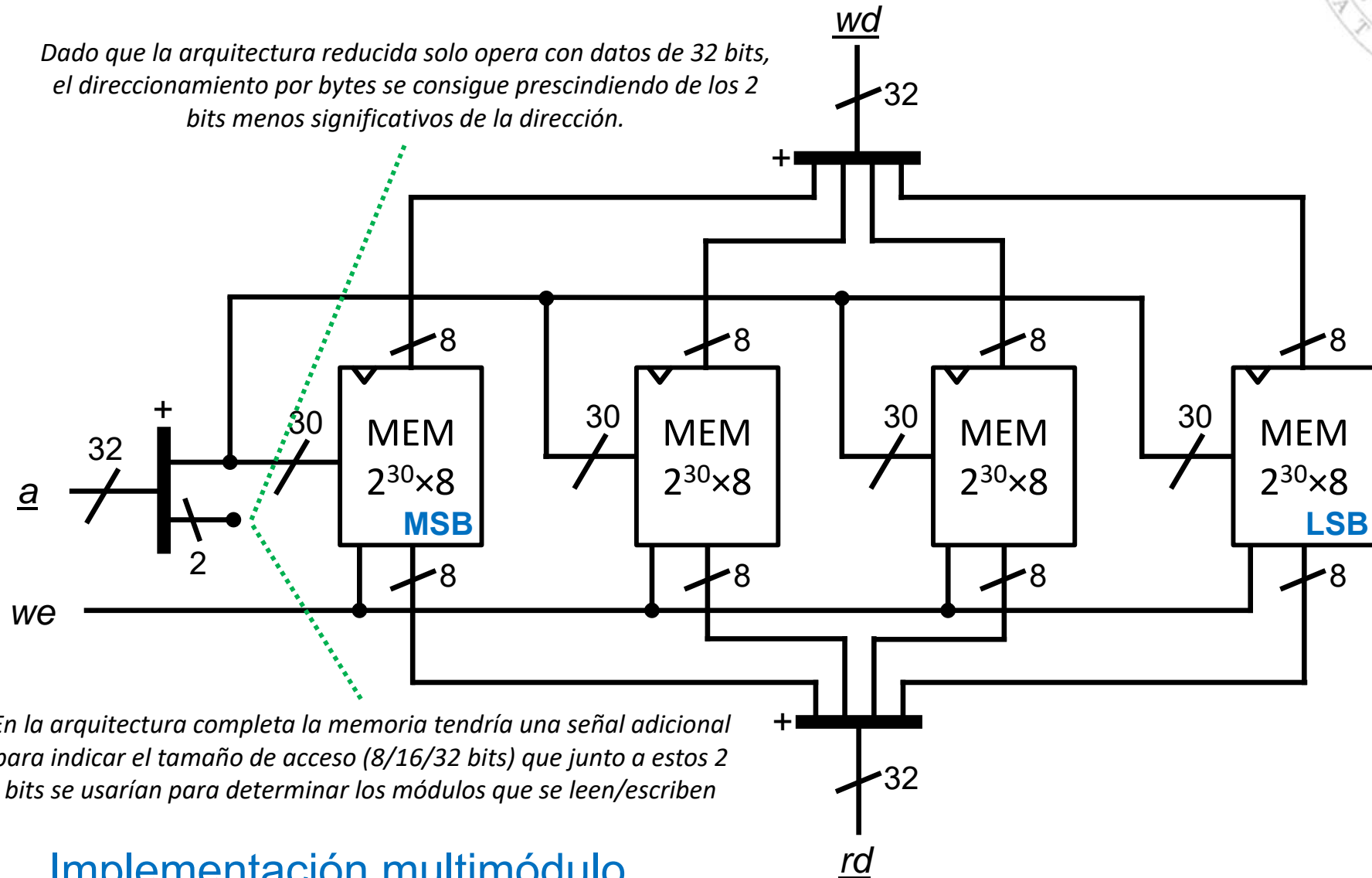
versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

46

Dado que la arquitectura reducida solo opera con datos de 32 bits, el direccionamiento por bytes se consigue prescindiendo de los 2 bits menos significativos de la dirección.



En la arquitectura completa la memoria tendría una señal adicional para indicar el tamaño de acceso (8/16/32 bits) que junto a estos 2 bits se usarían para determinar los módulos que se leen/escriben

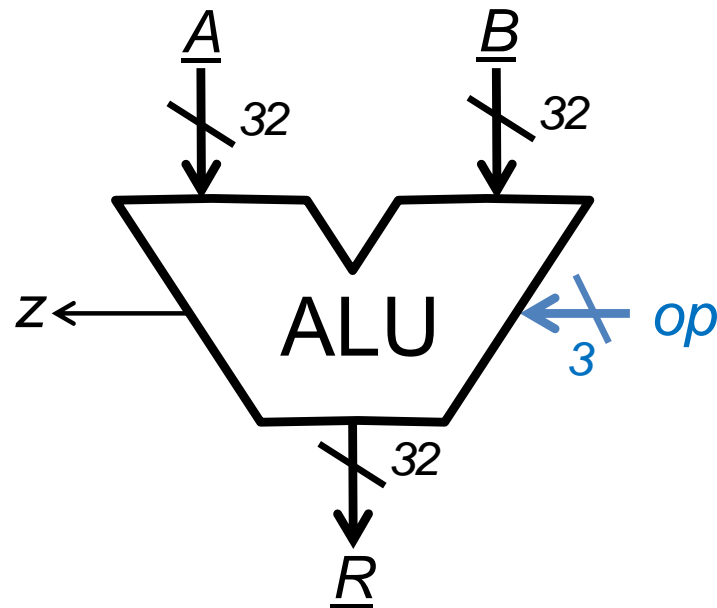
Implementación multimódulo

MEM $2^{30} \times 32$ (4 MiB) direccionable por bytes usando 4 MEM $2^{30} \times 8$ (1 MiB)



Aspectos tecnológicos

Diseño de la ALU (i)



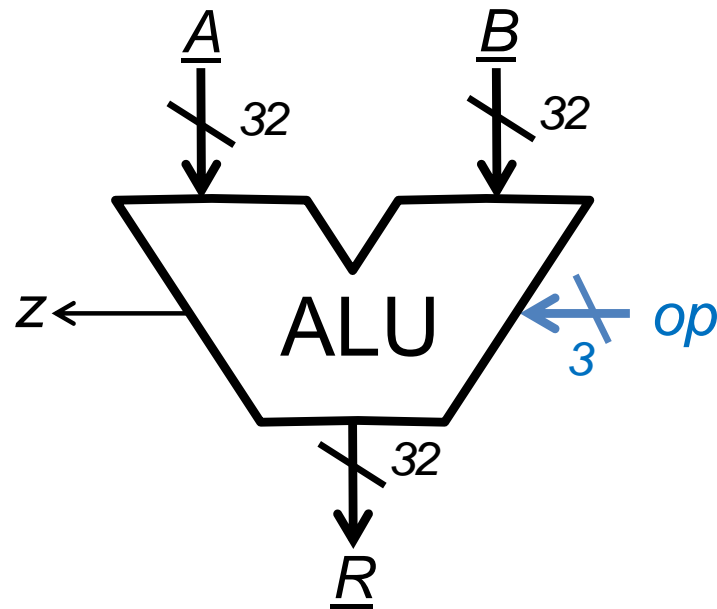
$\underline{A}, \underline{B}$	2 entradas de datos de 32 bits
\underline{op}	1 entrada de selección de operación
\underline{R}	1 salida de datos de 32 bits
z	1 salida de detección de cero

- La **ALU** es un módulo combinacional que realiza:
 - El **cálculo de direcciones efectivas** de memoria en instrucciones **lw/sw**.
 - Todas las **operaciones aritmético-lógicas** del instrucciones tipo I / tipo R.
 - La **comparación de operandos** en la instrucción **beq**.
 - En la **ruta de datos multiciclo**, también se usará para **incrementar el PC** y realizar el **cálculo de direcciones de salto** en instrucciones **beq/jal**



Aspectos tecnológicos

Diseño de la ALU (ii)



- A, B 2 entradas de datos de 32 bits
- op 1 entrada de selección de operación
- R 1 salida de datos de 32 bits
- z 1 salida de detección de cero

operaciones aritméticas

op ₂	op ₁	op ₀	<u>R</u>
0	0	0	$\underline{A} + \underline{B}$
0	0	1	$\underline{A} - \underline{B}$
1	0	0	—
1	0	1	if ($\underline{A} < \underline{B}$) then 1 else 0

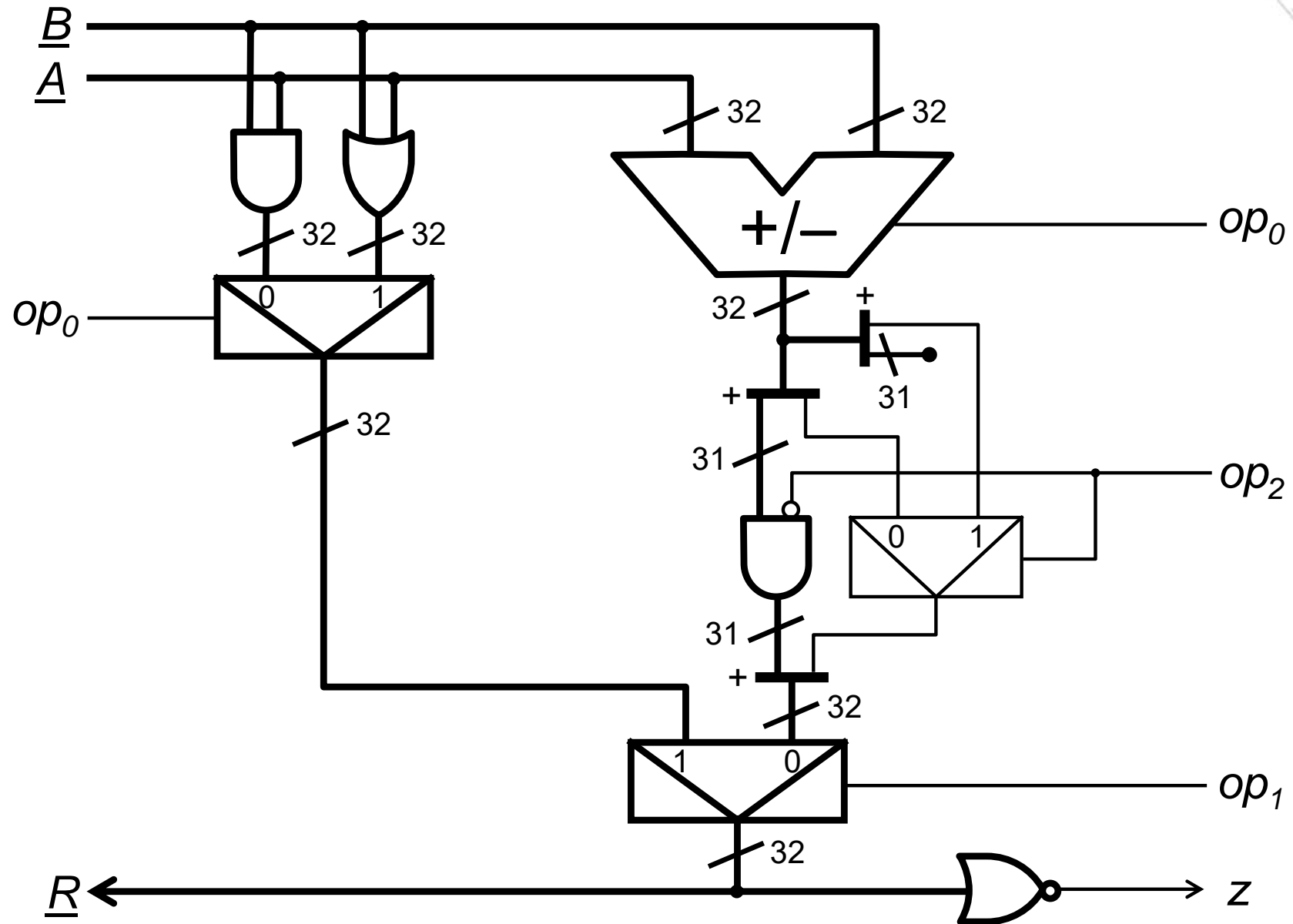
operaciones lógicas

op ₂	op ₁	op ₀	<u>R</u>
0	1	0	$\underline{A} \& \underline{B}$
0	1	1	$\underline{A} \underline{B}$
1	1	0	—
1	1	1	—



Aspectos tecnológicos

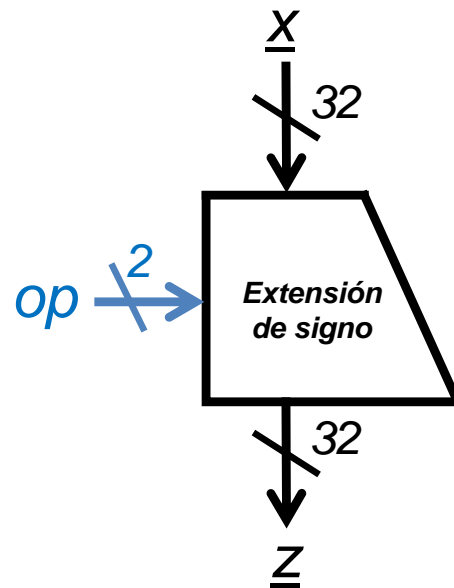
Diseño de la ALU (iii)





Aspectos tecnológicos

Diseño del Extensor de Signo (i)



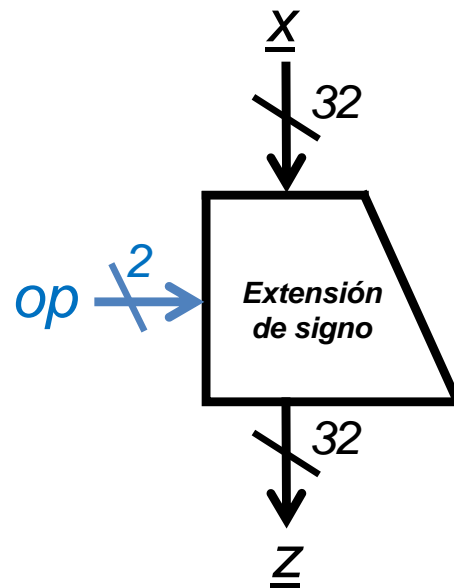
<u>X</u>	1 entrada de datos de 32 bits (instrucción)
<u>op</u>	1 entrada de selección de operación
<u>Z</u>	1 salida de datos de 32 bits (operando inmediato)

- El **Extensor de Signo** es un módulo combinacional que **construye el operando inmediato** de 32 bits a partir de información contenida en los campos imm de la instrucción:
 - Para cada **tipo de instrucción** el **campo imm** tiene **diferente tamaño y posición** dentro de la instrucción.
 - Por ello, aparte de **extender el signo**, debe **reordenar bits** y **completar con 0** en el caso de direcciones de salto.



Aspectos tecnológicos

Diseño del Extensor de Signo (iii)



<u>X</u>	1 entrada de datos de 32 bits (instrucción)
<u>op</u>	1 entrada de selección de operación
<u>Z</u>	1 salida de datos de 32 bits (operando inmediato)

<u>op</u>	z ₃₁	z ₃₀	z ₂₉	z ₂₈	z ₂₇	z ₂₆	z ₂₅	z ₂₄	z ₂₃	z ₂₂	z ₂₁	z ₂₀	z ₁₉	z ₁₈	z ₁₇	z ₁₆	z ₁₅	z ₁₄	z ₁₃	z ₁₂	z ₁₁	z ₁₀	z ₉	z ₈	z ₇	z ₆	z ₅	z ₄	z ₃	z ₂	z ₁	z ₀
00	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₀	x ₂₉	x ₂₈	x ₂₇	x ₂₆	x ₂₅	x ₂₄	x ₂₃	x ₂₂	x ₂₁	x ₂₀
01	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₀	x ₂₉	x ₂₈	x ₂₇	x ₂₆	x ₂₅	x ₁₁	x ₁₀	x ₉	x ₈	x ₇
10	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₇	x ₃₀	x ₂₉	x ₂₈	x ₂₇	x ₂₆	x ₂₅	x ₁₁	x ₁₀	x ₉	x ₈	0
11	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₃₁	x ₁₉	x ₁₈	x ₁₇	x ₁₆	x ₁₅	x ₁₄	x ₁₃	x ₁₂	x ₂₀	x ₃₀	x ₂₉	x ₂₈	x ₂₇	x ₂₆	x ₂₅	x ₂₄	x ₂₃	x ₂₂	x ₂₁	0



Aspectos tecnológicos

Cálculo del coste y tiempo de ciclo

- Para calcular el **coste del procesador** es la **suma del coste** de cada uno de los módulos que lo componen.
 - El **coste de cada módulo** se calcula sumando el **coste de sus celdas**.
- El **tiempo de ciclo del procesador** es el **máximo** de los **caminos críticos** de las **transferencias entre registros** que realiza el procesador.
 - El **camino crítico** de una transferencia entre registros es **el camino de datos de mayor retardo** de todos los implicados en dicha transferencia.
 - En el **procesador monociclo**, en cada ciclo se ejecuta una instrucción que implica **1 ó 2 transferencias entre registros**: la de actualización del PC y la operativa propia de cada instrucción.
- Para todos los cálculos se utilizará la **misma biblioteca** de celdas (CMOS 90nm) usada en **FC-1**



Aspectos tecnológicos

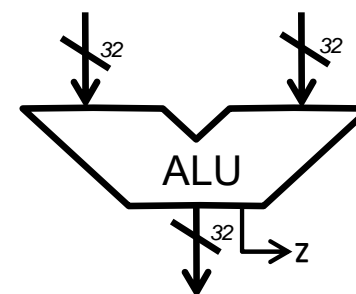
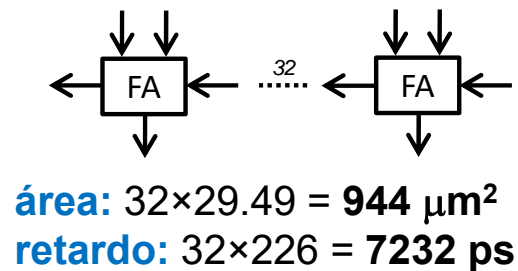
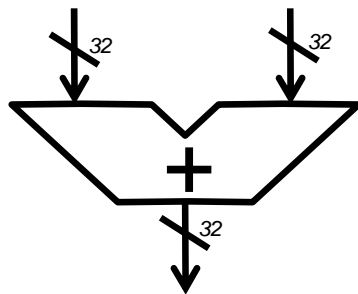
Cálculo del coste y tiempo de ciclo

versión 15/01/23

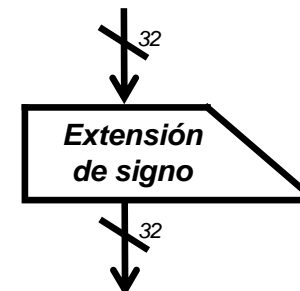
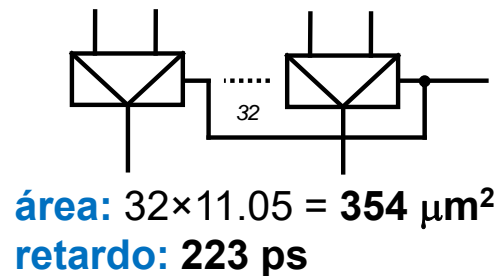
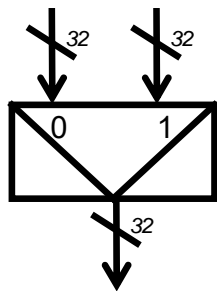
tema 5:
Diseño monociclo del procesador

FC-2

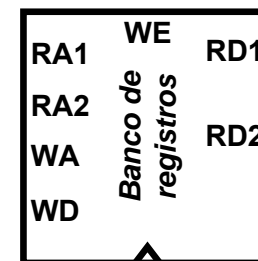
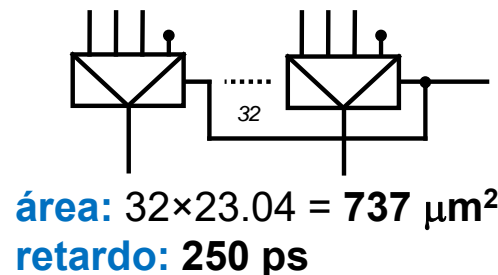
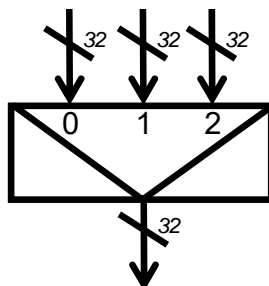
54



área: $3052 \mu\text{m}^2$
retardo: 8360 ps



área: $202 \mu\text{m}^2$
retardo: 460 ps



área: $51405 \mu\text{m}^2$
retardo lectura: 723 ps
setup escritura: 705 ps
(debido al DEC de dirección)



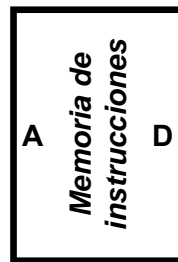
Aspectos tecnológicos

Cálculo del coste y tiempo de ciclo

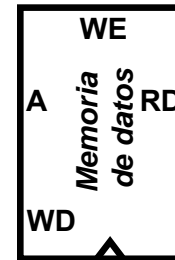


versión 15/01/23

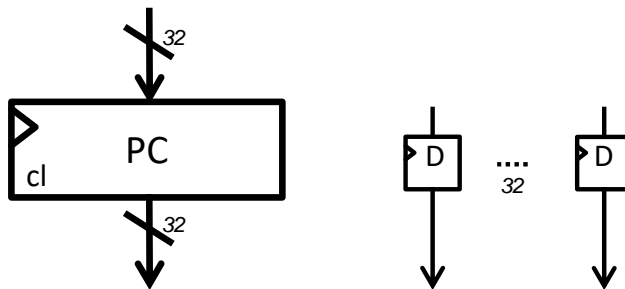
Comportamiento idealizado: retardo comparable al de la ALU
(para que pueda leerse en un ciclo de reloj)



área: -
tiempo de acceso: 8500 ps



área: -
tiempo de acceso: 8500 ps



área: $32 \times 32 \times 26 = 1032 \mu\text{m}^2$
retardo CLK \rightarrow Q: $1 \times 167 = 167 \text{ ps}$
setup: 0 ps



área: $56 \mu\text{m}^2$
retardo: 490 ps



área: $65 \mu\text{m}^2$
retardo: 451 ps



área: $21 \mu\text{m}^2$
retardo: 451 ps

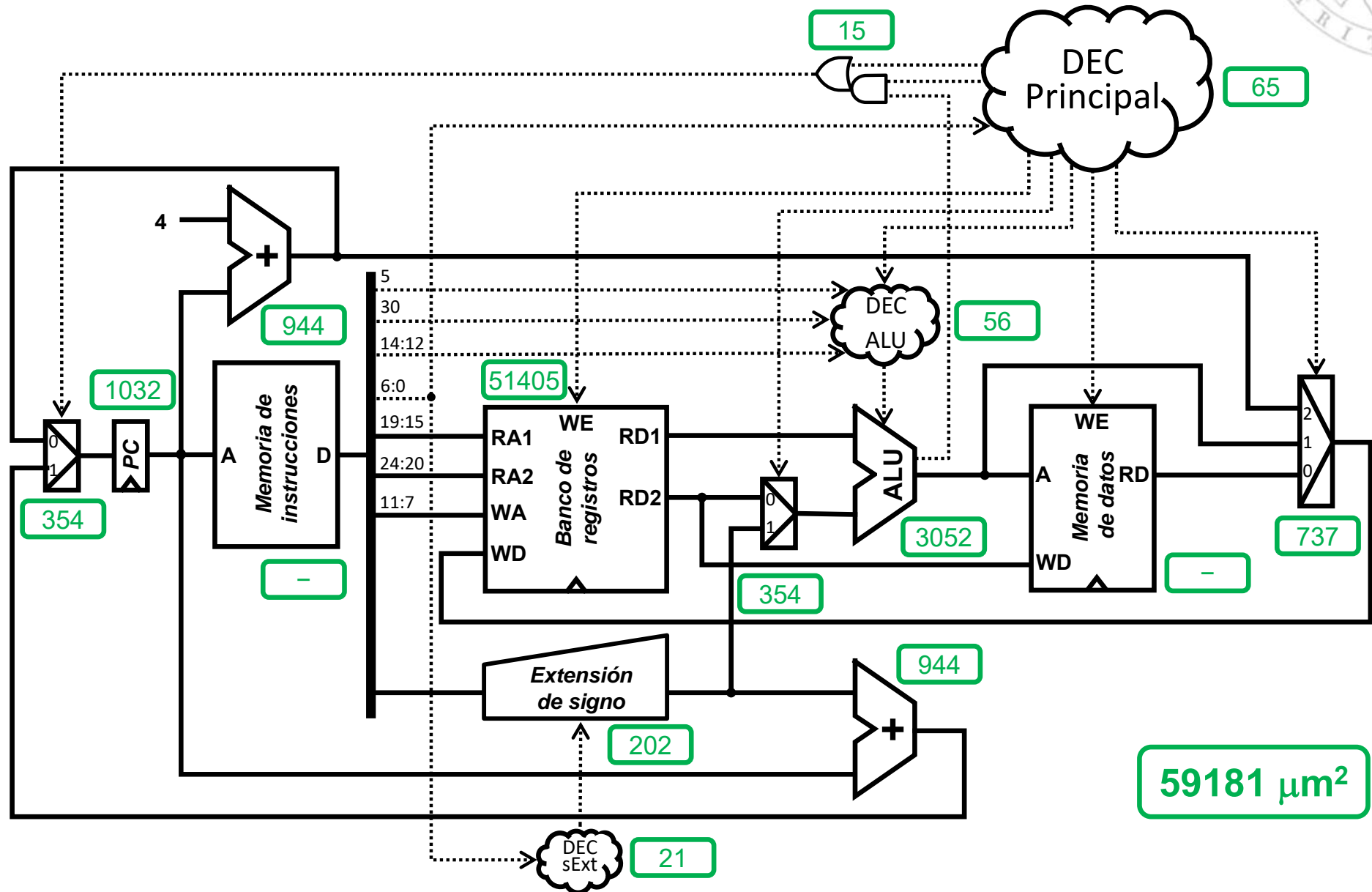


área: $15 \mu\text{m}^2$
retardo: 351 ps



Aspectos tecnológicos

Cálculo del coste (CMOS 90 nm)



versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

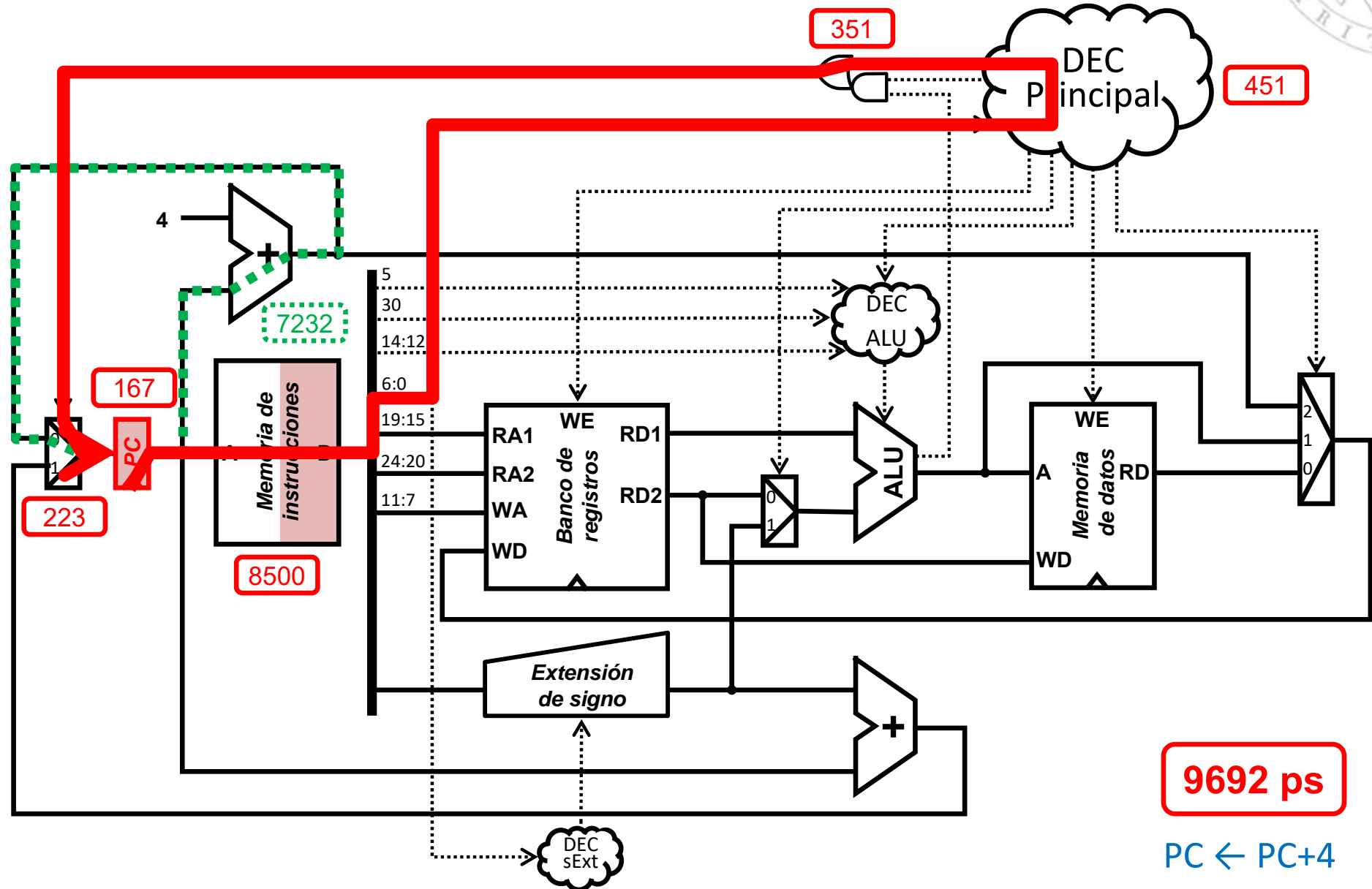
57


$$PC \leftarrow PC+4$$



Aspectos tecnológicos

Incremento del PC: camino crítico





Aspectos tecnológicos

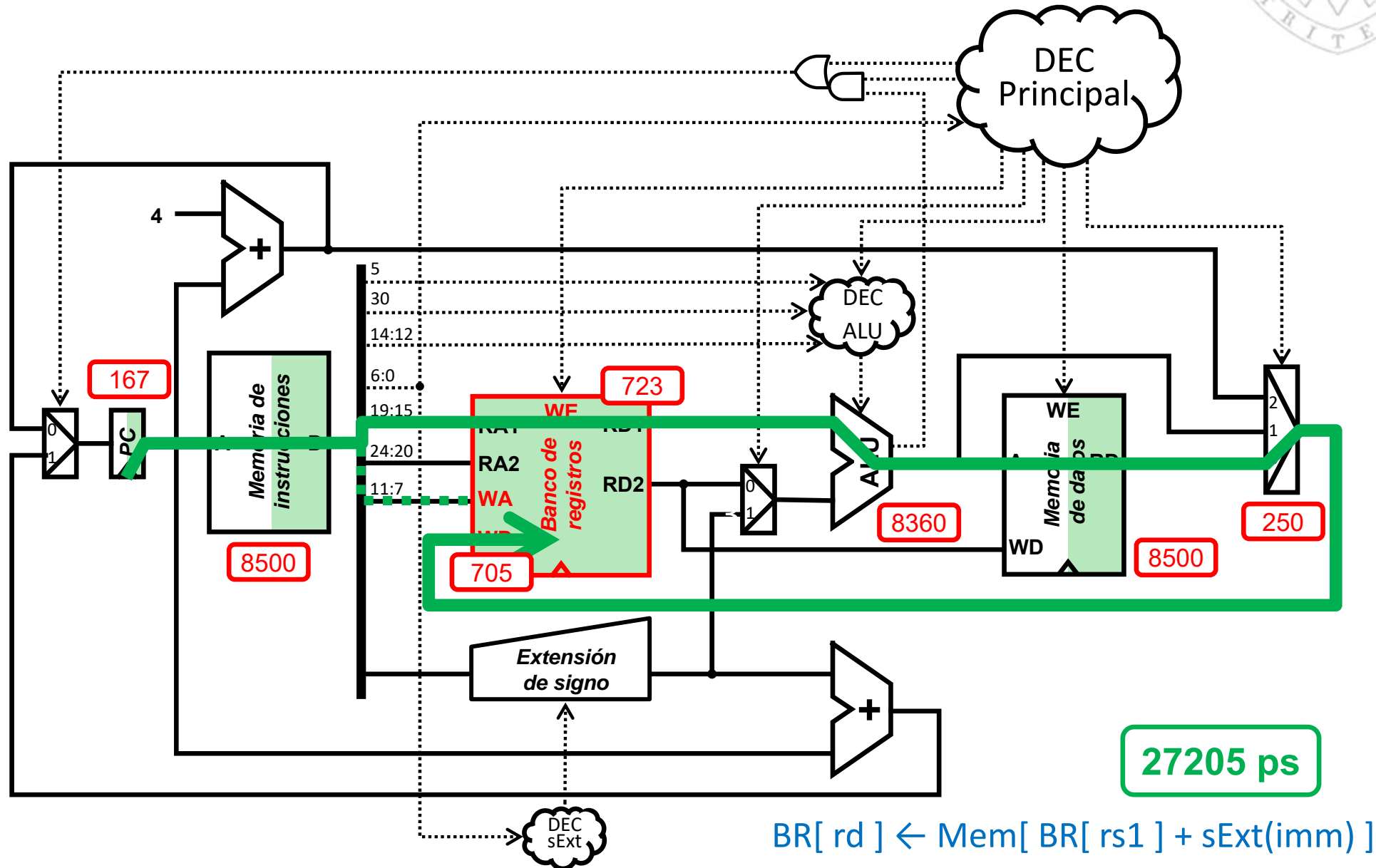
Instrucción lw (i)

versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

59





Aspectos tecnológicos

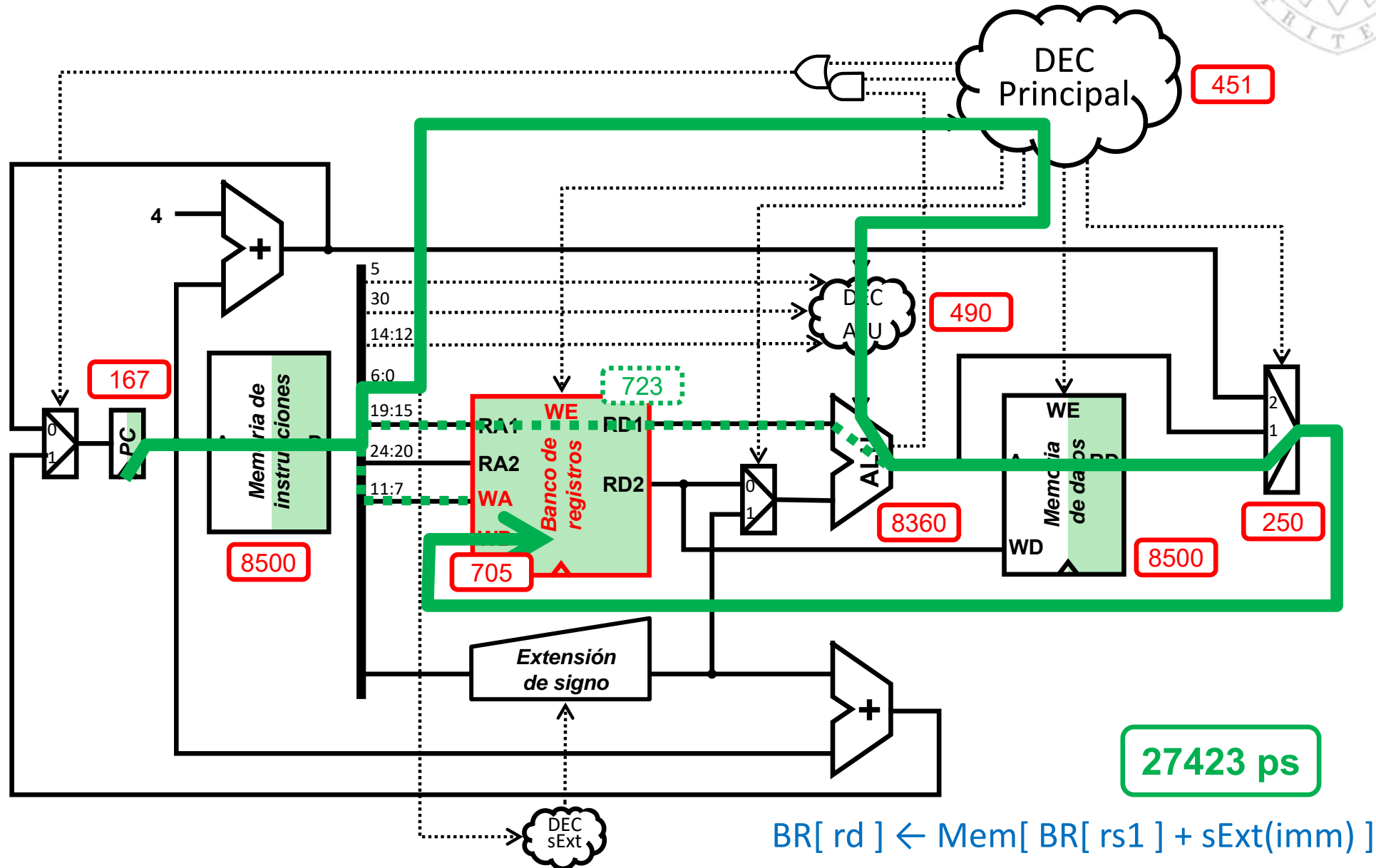
Instrucción lw (ii)

versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

60





Aspectos tecnológicos

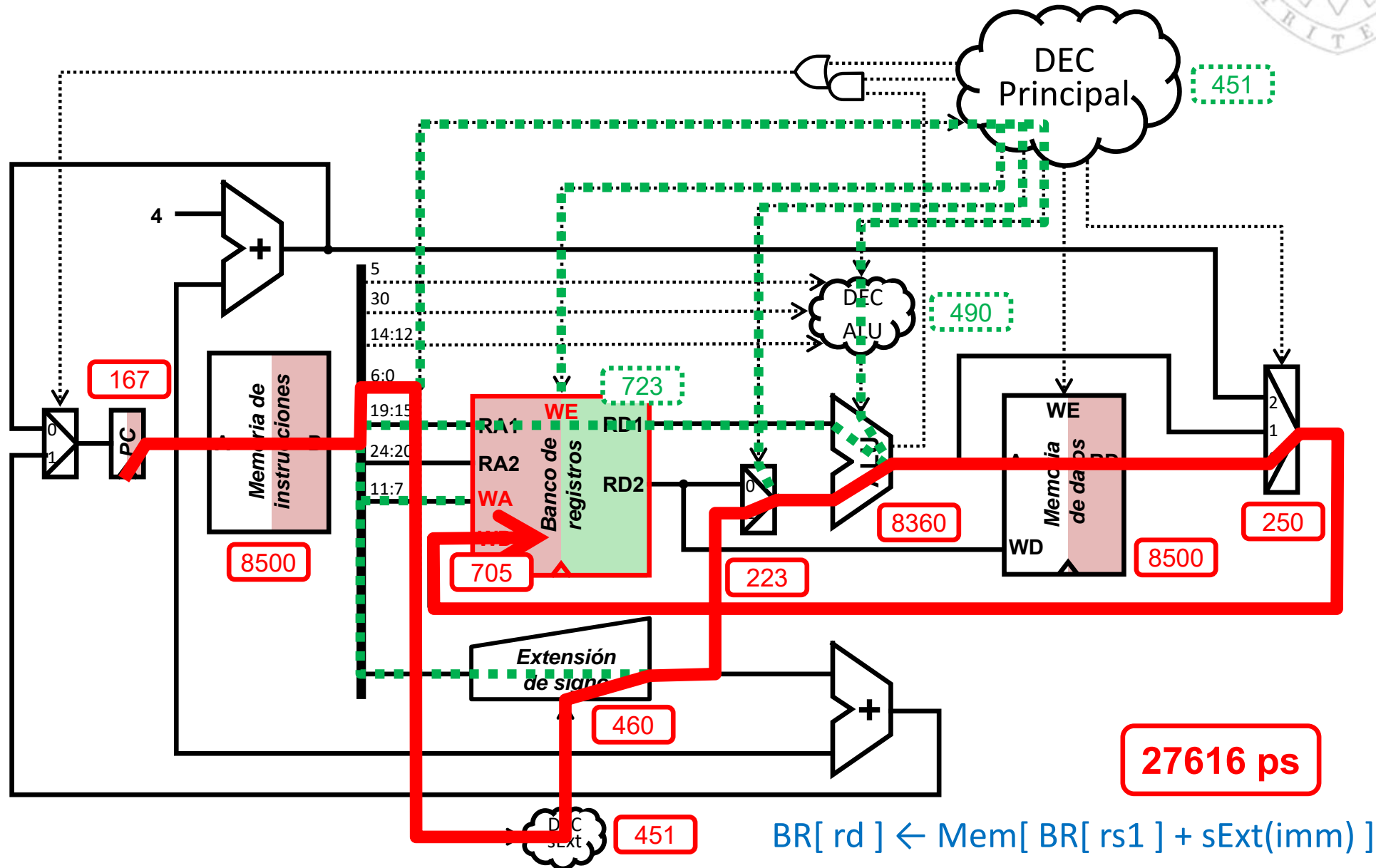
Instrucción lw: camino crítico

versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

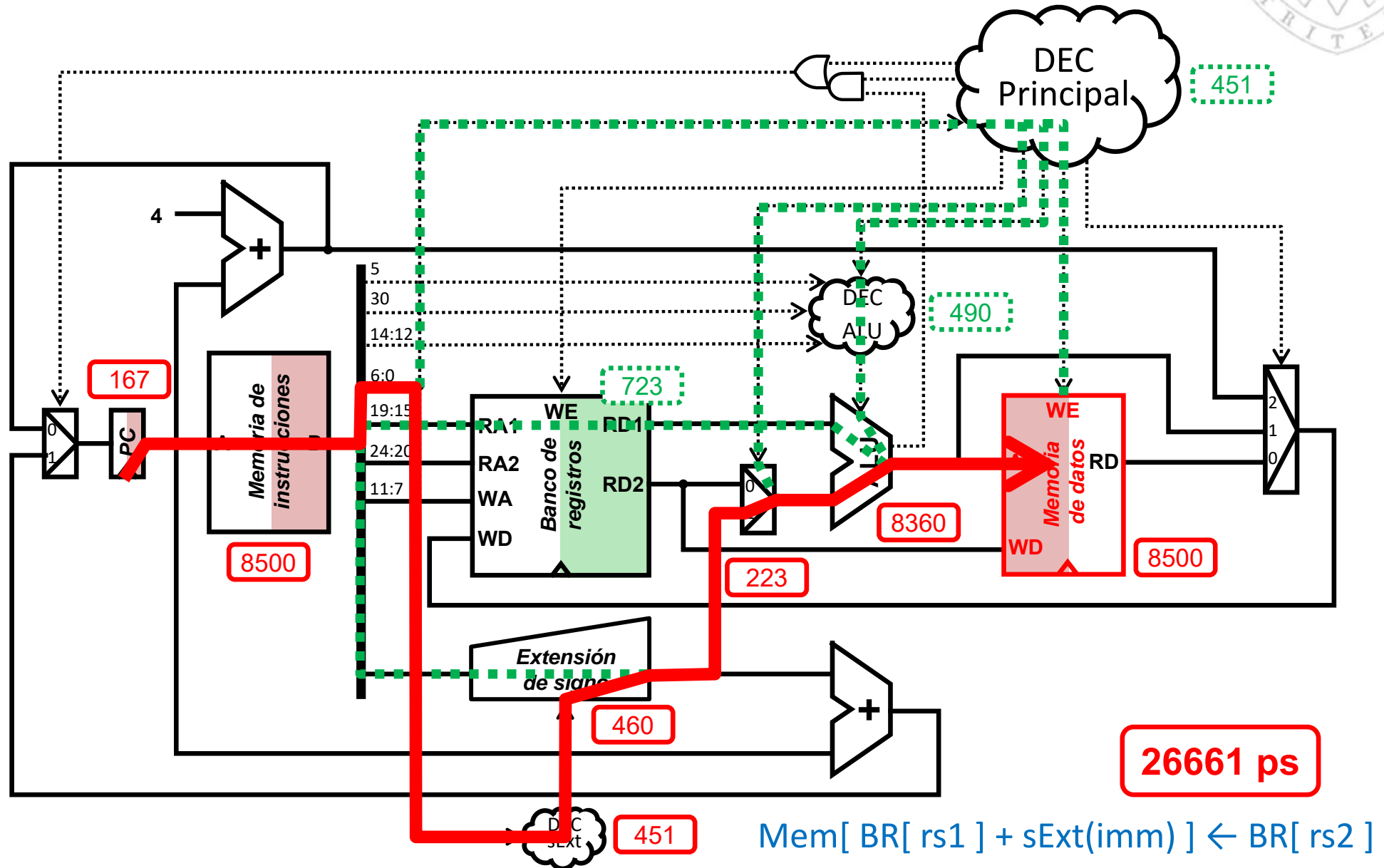
61

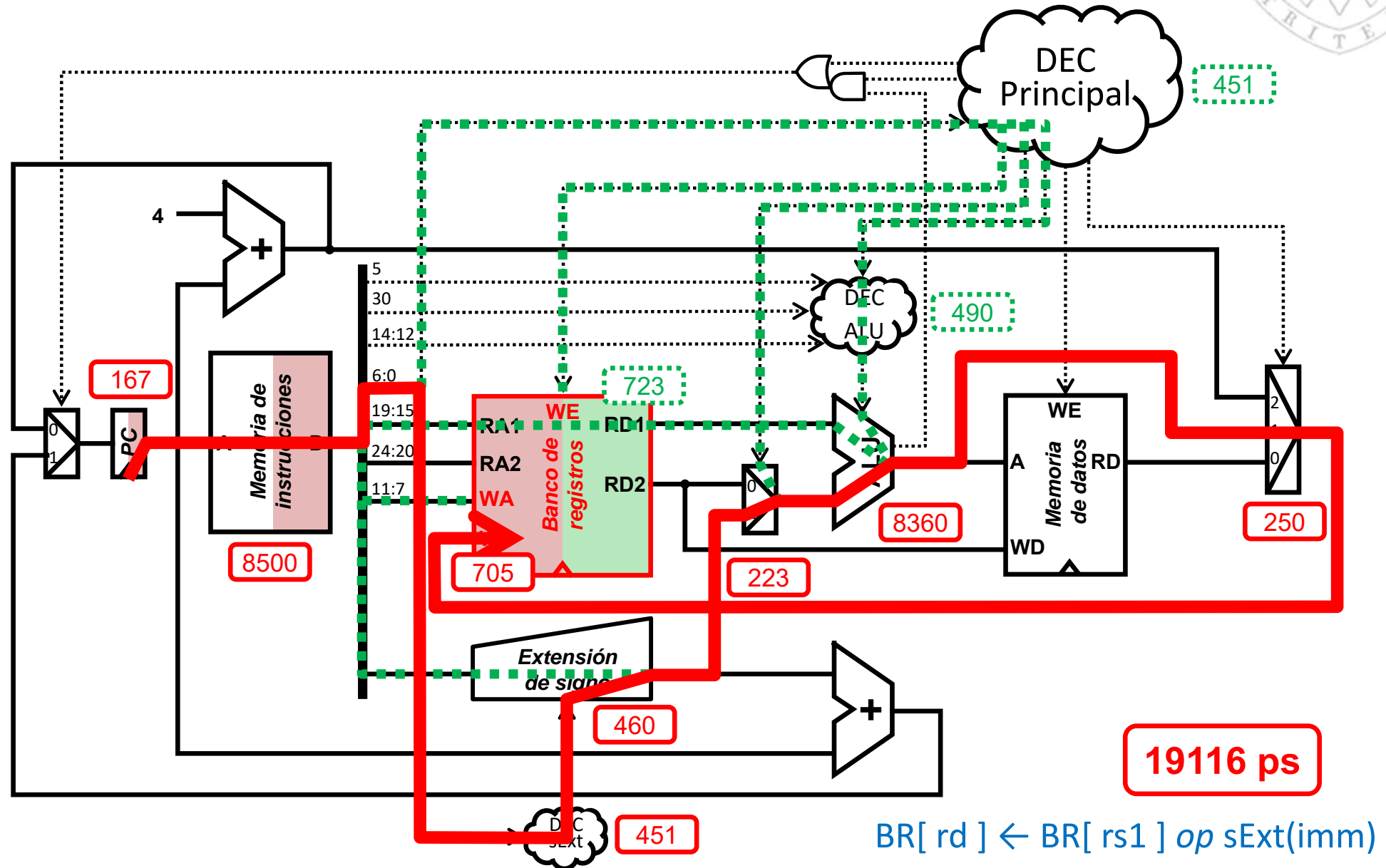




Aspectos tecnológicos

Instrucción **sw**: camino crítico

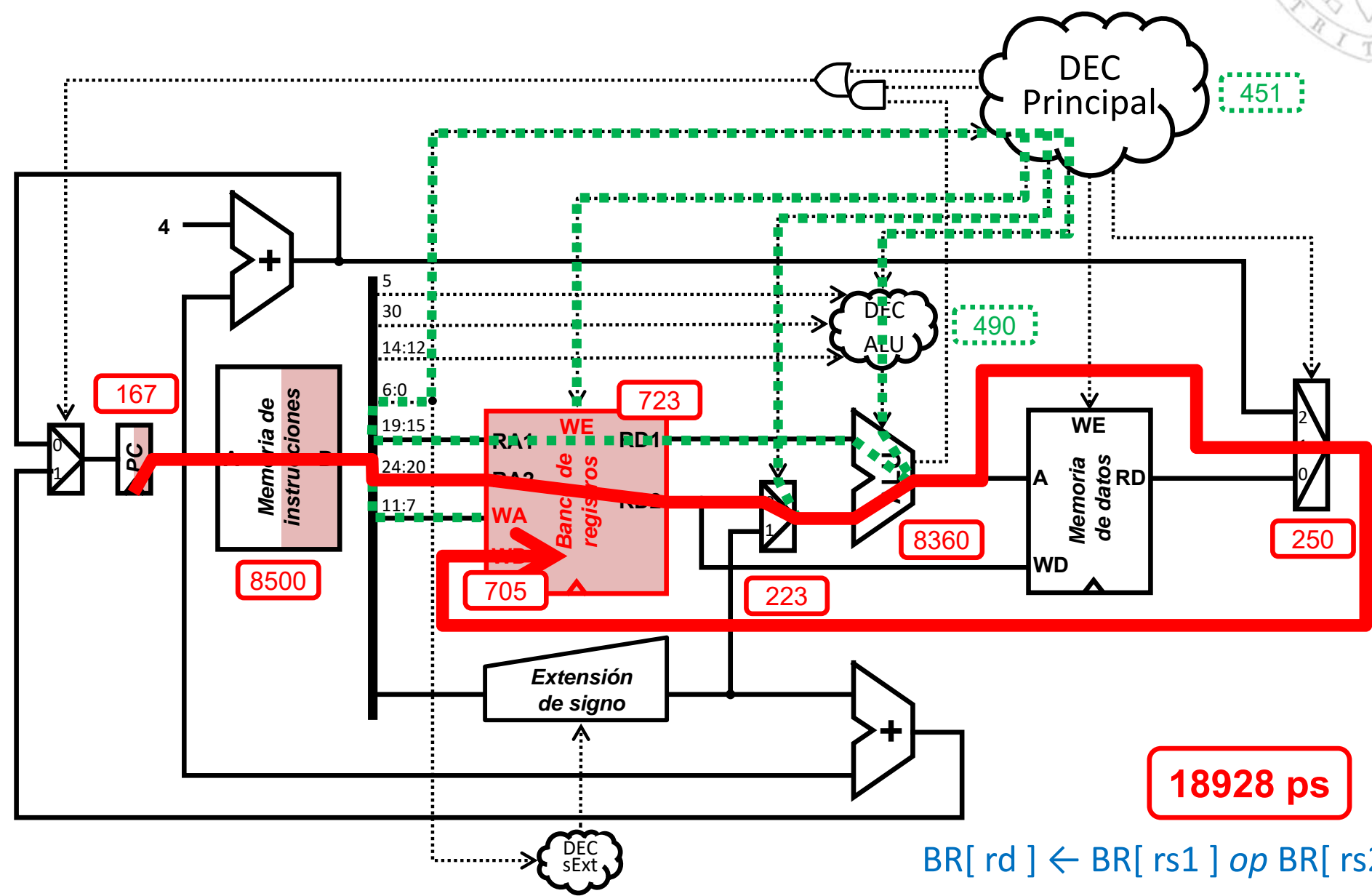






Aspectos tecnológicos

Instrucciones tipo **add**: camino crítico



$$BR[rd] \leftarrow BR[rs1] \text{ op } BR[rs2]$$



Aspectos tecnológicos

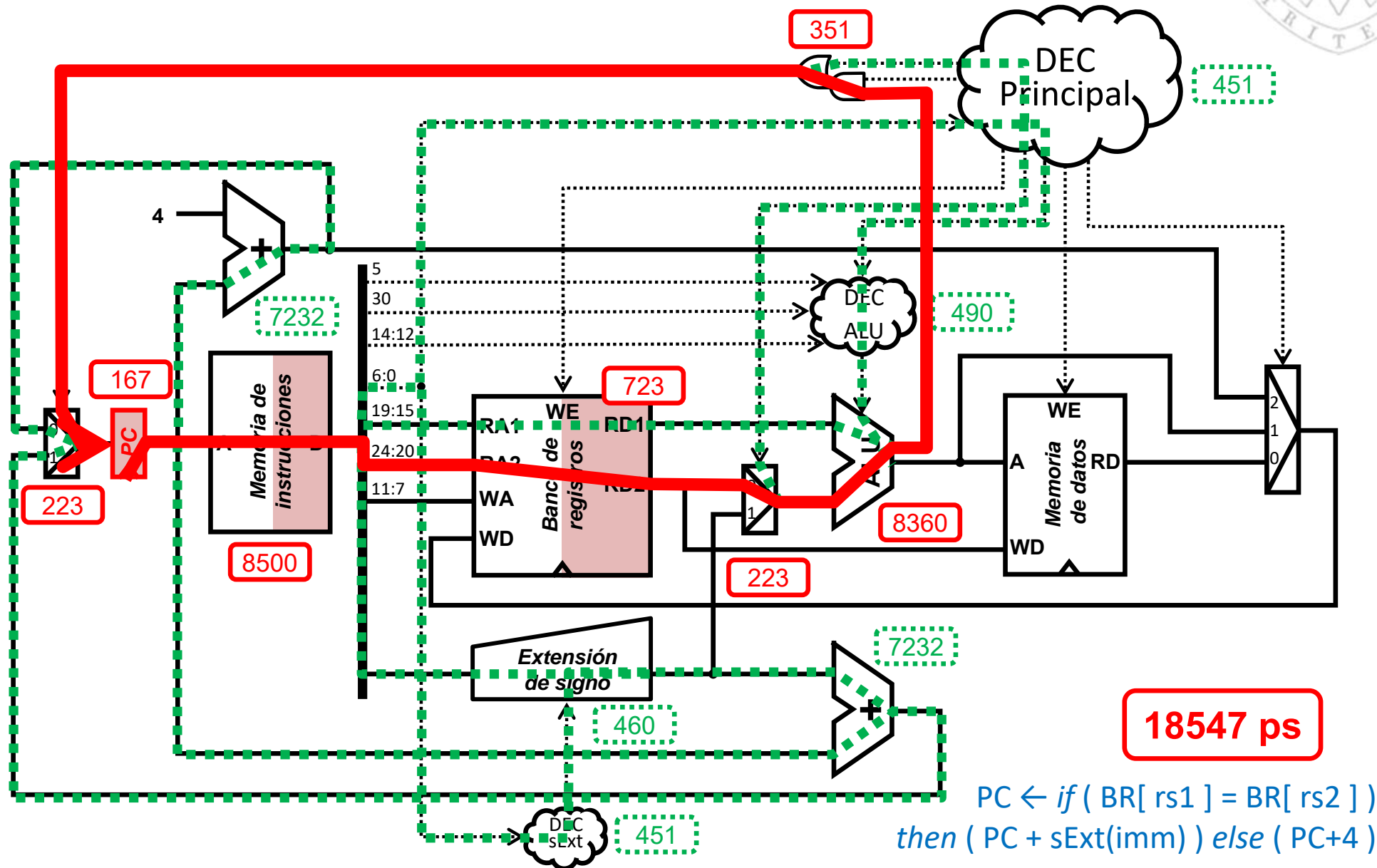
Instrucción **beq**: camino crítico

versión 15/01/23

tema 5:
Diseño monociclo del procesador

FC-2

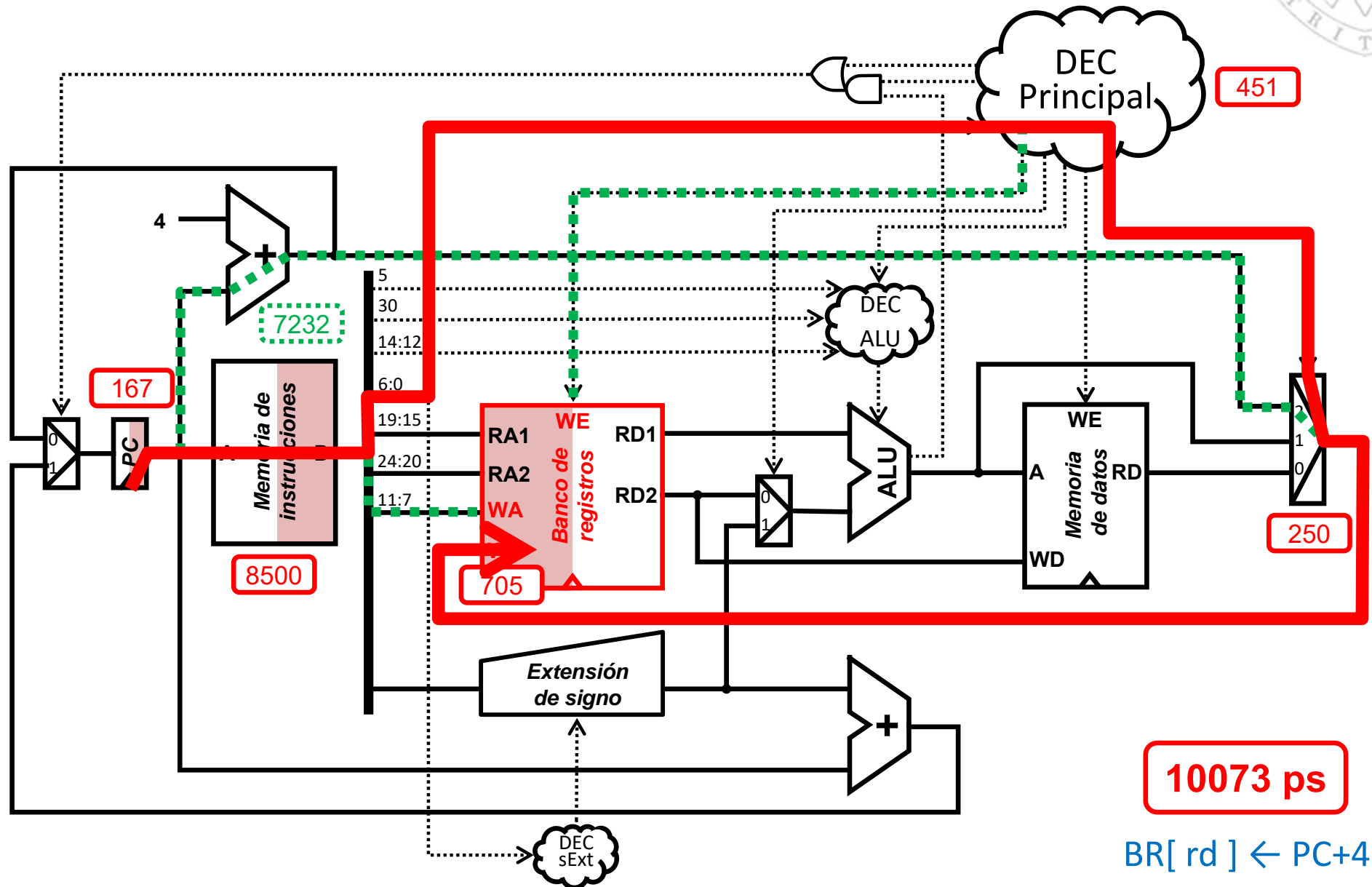
65





Aspectos tecnológicos

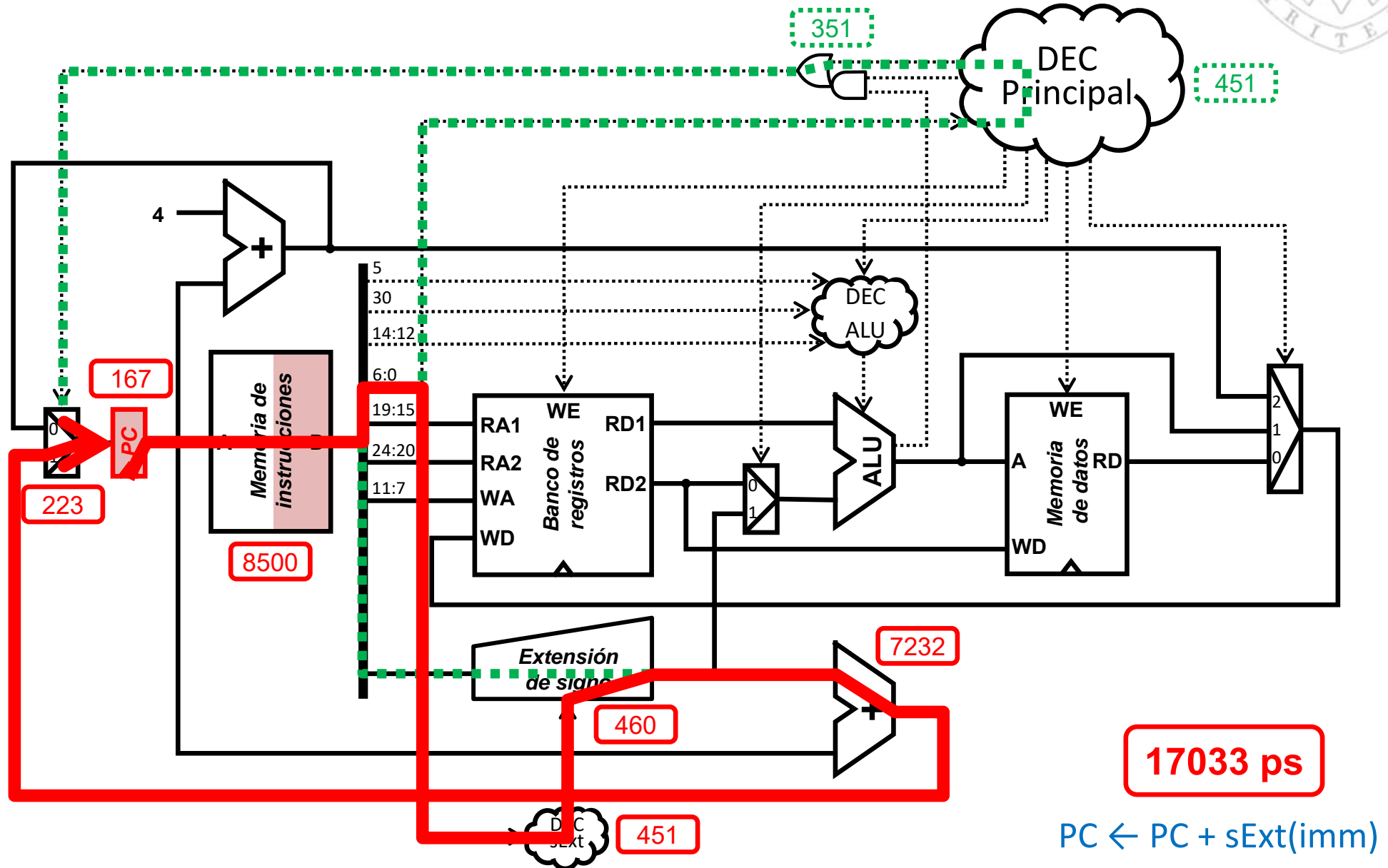
Instrucción **jal** (almacenaje dir. retorno): camino crítico





Aspectos tecnológicos

Instrucción **jal** (actualización PC): camino crítico



Acerca de *Creative Commons*



■ Licencia CC (*Creative Commons*)

- Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:



Reconocimiento (*Attribution*):

En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.



No comercial (*Non commercial*):

La explotación de la obra queda limitada a usos no comerciales.



Compartir igual (*Share alike*):

La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: <https://creativecommons.org/licenses/by-nc-sa/4.0/>