

## ***Unidad 4. El Procesador II: Diseño y control de la ruta de datos. Arquitectura unicycle***

---

Escuela Politécnica Superior - UAM

# Índice

- **Introducción**
- Ruta de datos unicycle
- Control unicycle
- Añadir más instrucciones
- Parámetros temporales en la ruta unicycle

# Introducción

APLICACIÓN SOFTWARE	PROGRAMAS
SISTEMAS OPERATIVOS	DRIVERS
ARQUITECTURA	INSTRUCCIONES REGISTROS
MICRO-ARQUITECTURA	CAMINO DE DATOS CONTROLADORES
LÓGICA	SUMADORES MEMORIA
CIRCUITOS DIGITALES	PUERTAS LÓGICAS
CIRCUITOS ANALÓGICOS	AMPLIFICADORES FILTROS
DISPOSITIVOS	TRANSISTORES DIODOS
FÍSICA	ELECTRONES

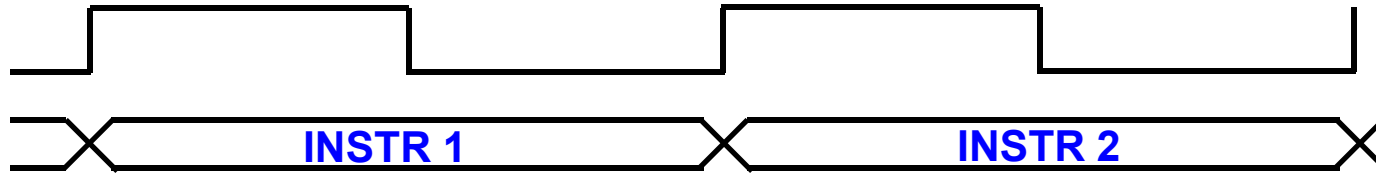
- **Arquitectura:**
  - Es la visión que desde el punto de vista del programador se tiene del sistema computador (U3).
- **Microarquitectura:**
  - Es la implementación en hardware del computador (U4 y U5).
  - Ruta de datos: bloques funcionales.
  - Ruta de control: señales de control internas.

# Microarquitectura

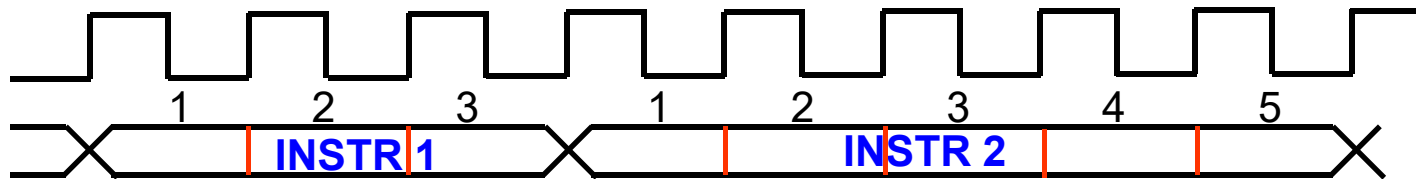
- Hay múltiples implementaciones de la misma arquitectura (juego de instrucciones):
  - Uniciclo (U4)
    - Cada instrucción se ejecuta en un ciclo de reloj
  - Multiciclo (U5)
    - Cada instrucción se divide en pasos cortos, cada uno de un ciclo de reloj mucho más rápido
  - Segmentado (*pipelined*)
    - Cada instrucción se divide en pasos cortos
    - Se ejecutan múltiples instrucciones a la vez, cada una en un paso distinto

# Microarquitectura

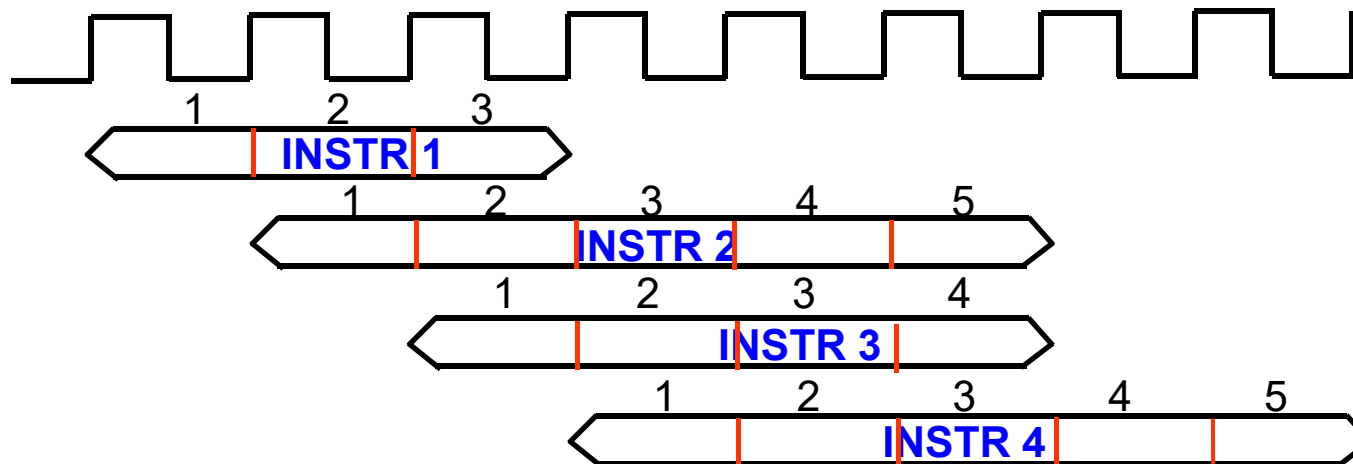
- Uniciclo (U4)



- Multiciclo (U5)



- Segmentado



# Subconjunto del MIPS

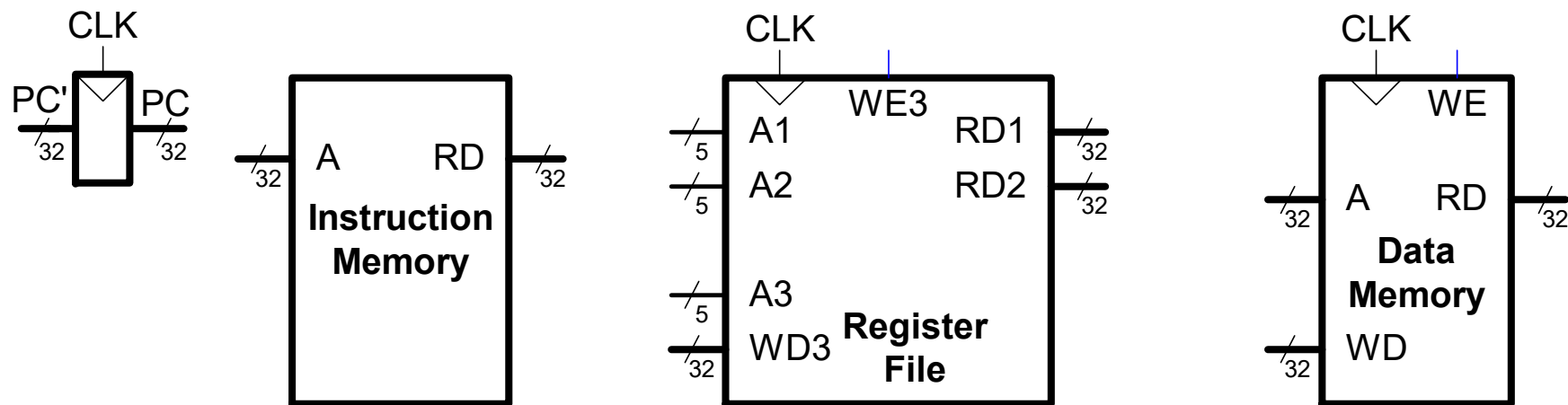
- Para estudiar la microarquitectura consideramos inicialmente sólo un subconjunto del juego de instrucciones:
  - ✓ R-Type: `and`, `or`, `add`, `sub`, `slt`
  - ✓ I-Type, de memoria: `lw`, `sw`
  - ✓ I-Type, de saltos: `beq`
- Luego añadiremos más instrucciones (`addi`, `j` (J-Type)).
- En el laboratorio de prácticas se añadirán algunas más.

# Índice

- Introducción
- **Ruta de datos unicycle**
- Control unicycle
- Añadir más instrucciones
- Parámetros temporales en la ruta unicycle

# Estado de la arquitectura

- Se puede conocer en qué situación se encuentra el micro conociendo los valores de:
  - PC
  - Banco de registros (los 32 registros)
  - Memoria (de código y de datos)
- Primeros elementos a considerar en la ruta de datos:





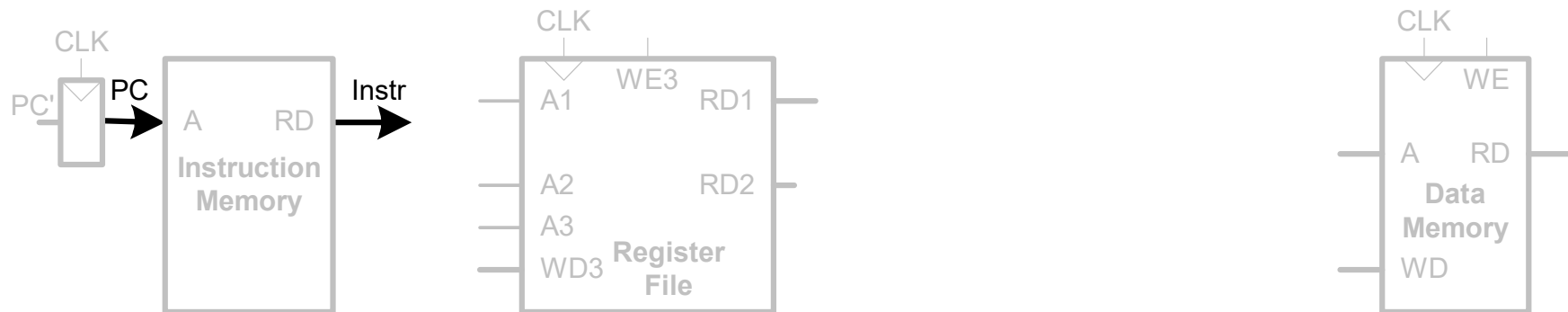
# Ruta de datos: captura `lw`

El análisis de la ruta de datos comienza con la instrucción:

**(0x8C112000) `lw $s1, 0x2000($0)`**

y los pasos para ejecutarla:

➤ **PASO 1:** captura de instrucción (*fetch*)

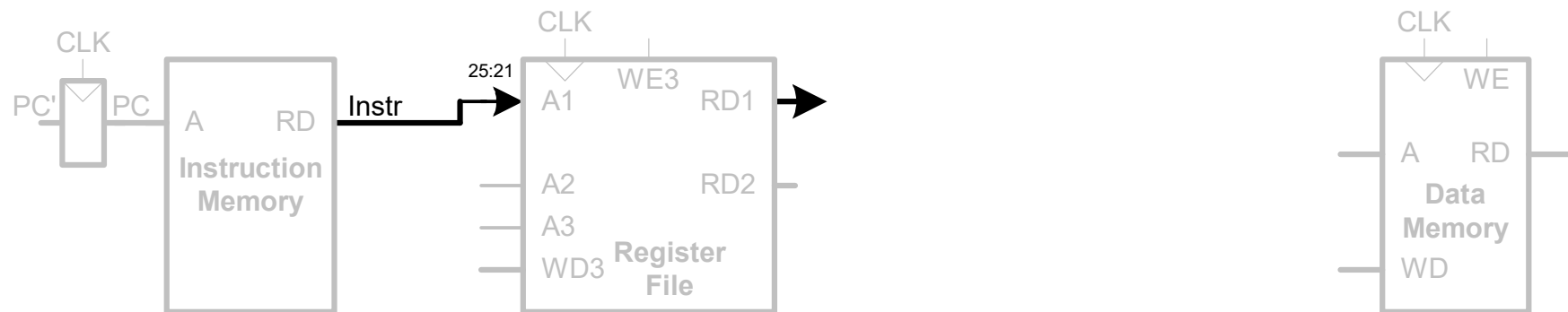


`Instr` <= "100011 00000 10001 0010000000000000"

op: Instr [31:26] = "100011"	=>	<b>lw</b>
rs: Instr [25:21] = "00000"	=>	<b>\$0</b>
rt: Instr [20:16] = "10001"	=>	<b>\$s1</b>
imm: Instr [15:0] = "0010000000000000"	=>	<b>0x2000</b>

# Ruta de datos: lectura de registros lw

- **PASO 2:** lectura de los operandos fuente del banco de registros

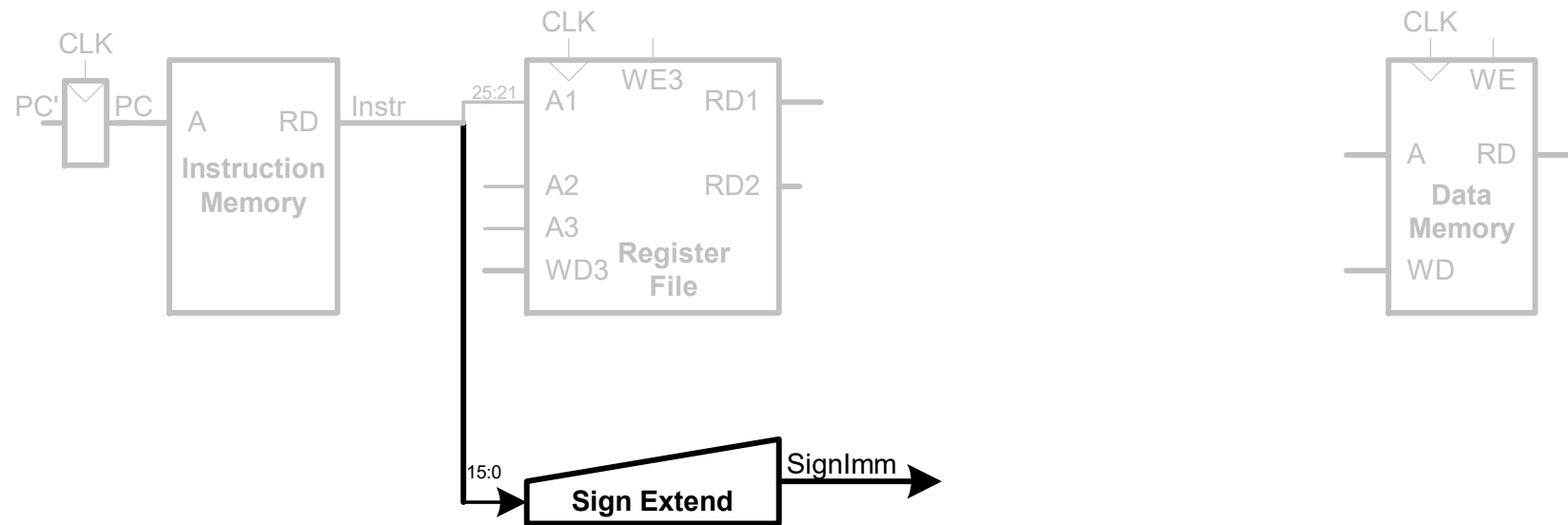


A1: Instr [25:21] = "00000"; RD1 <= "0x00000000" = (\$0)

```
lw $s1, 0x2000($0)
```

# Ruta de datos: dato inmediato lw

## ➤ PASO 3: extensión en signo del dato inmediato

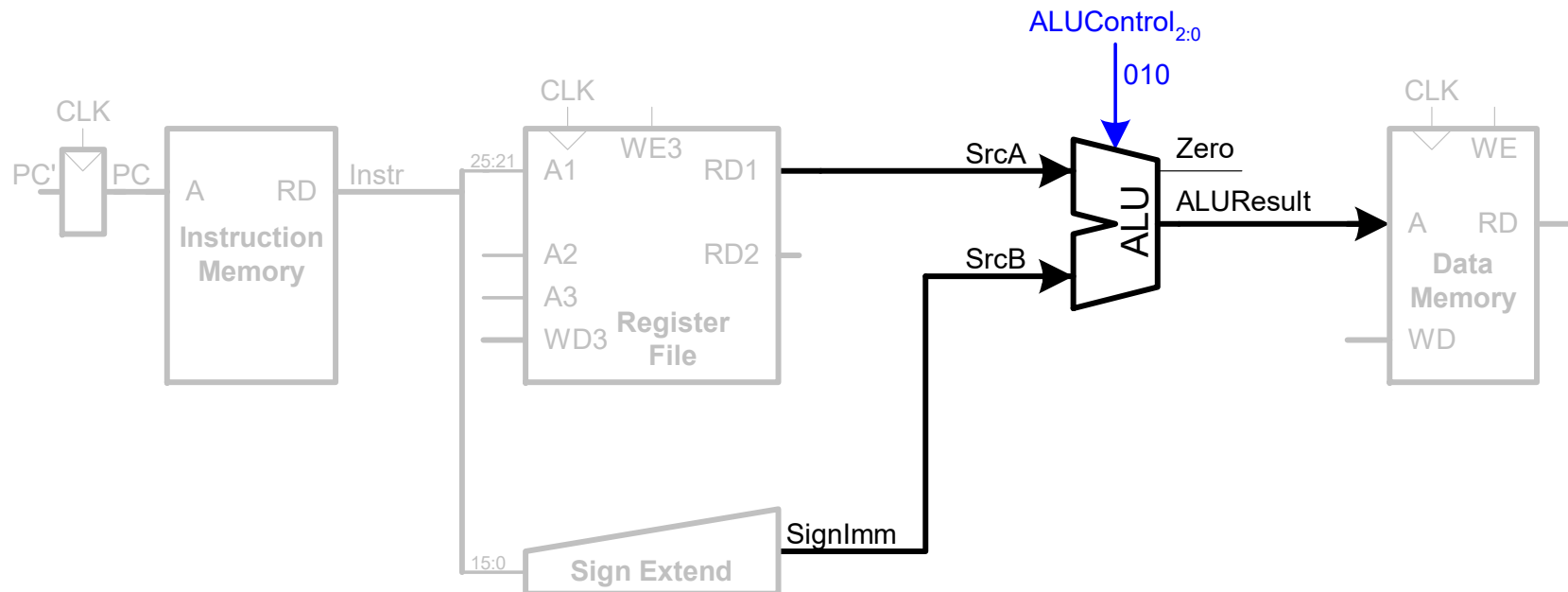


`SignImm <= 0x00002000 = Sign Extend (0x2000)`

```
lw $s1, 0x2000($0)
```

# Ruta de datos: dirección lw

- **PASO 4:** calcular la dirección para acceso a memoria de datos sumando ( $[rs] + \text{SignImm}$ ) en la ALU

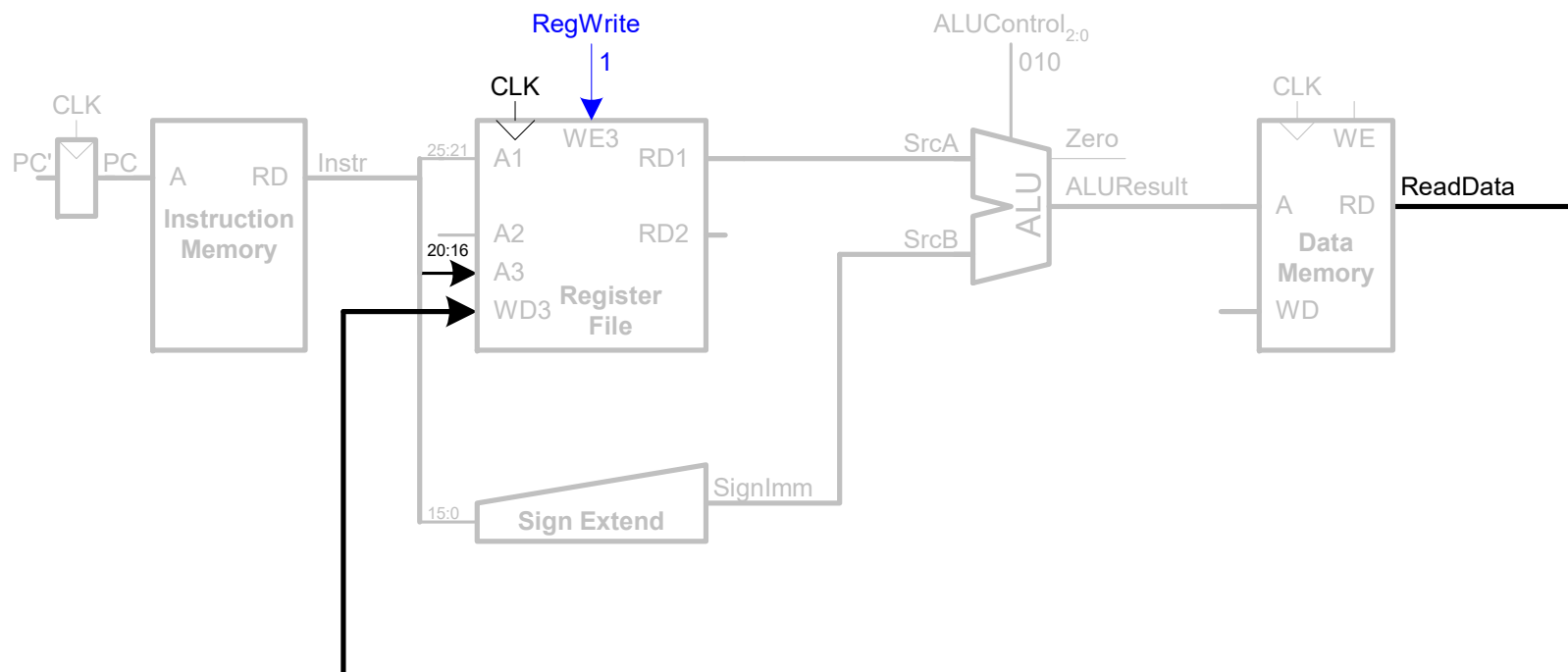


$\text{ALUResult} \leq 0x00002000 = 0x00000000 + 0x00002000$

`lw $s1, 0x2000($0)`

# Ruta de datos: leer memoria `lw`

- **PASO 5:** leer el dato buscado de memoria y escribirlo en el registro destino, `rt`



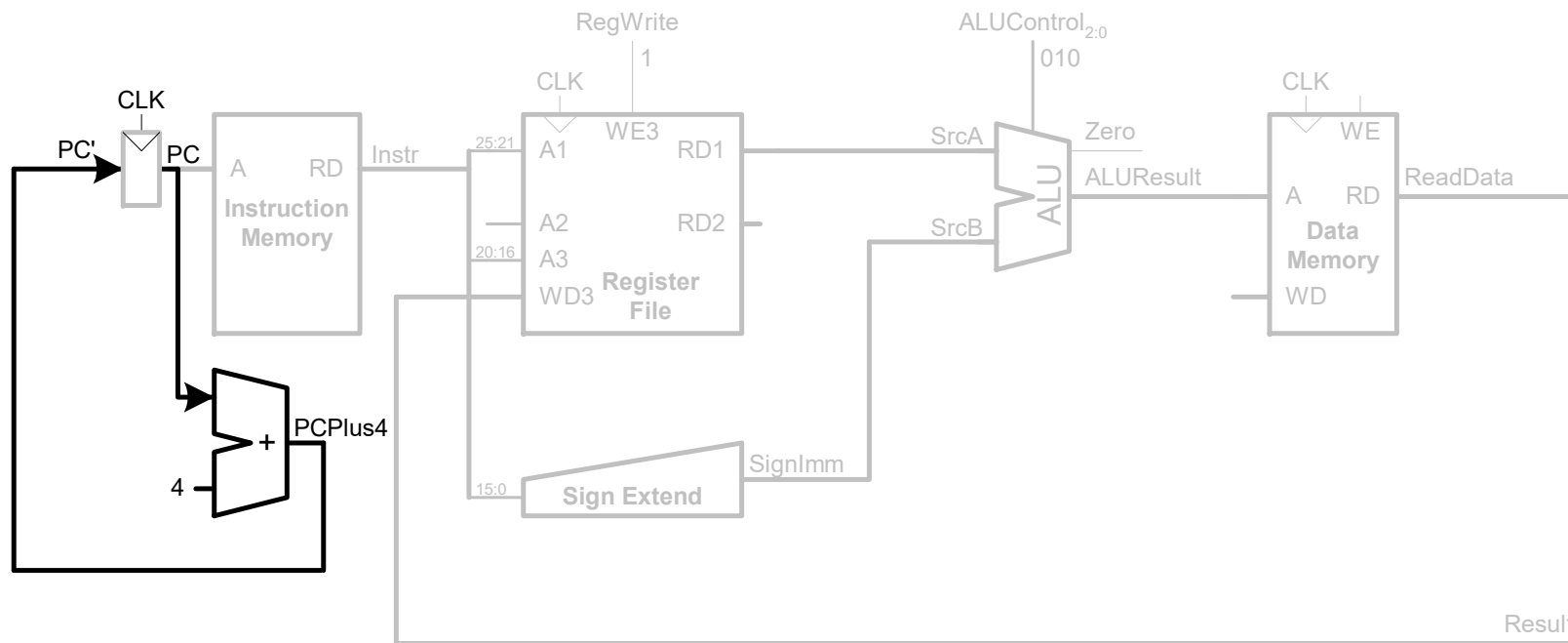
A3: Instr [20:16] = "10001" (`$s1`)

`$s1 <= WD3 = MEM[0x00002000]`

`lw $s1, 0x2000($0)`

# Ruta de datos: incrementar PC

- **PASO 6:** incrementar el PC en 4 para tener la dirección de la próxima instrucción



$$\$PC \leftarrow \$PC + 4$$

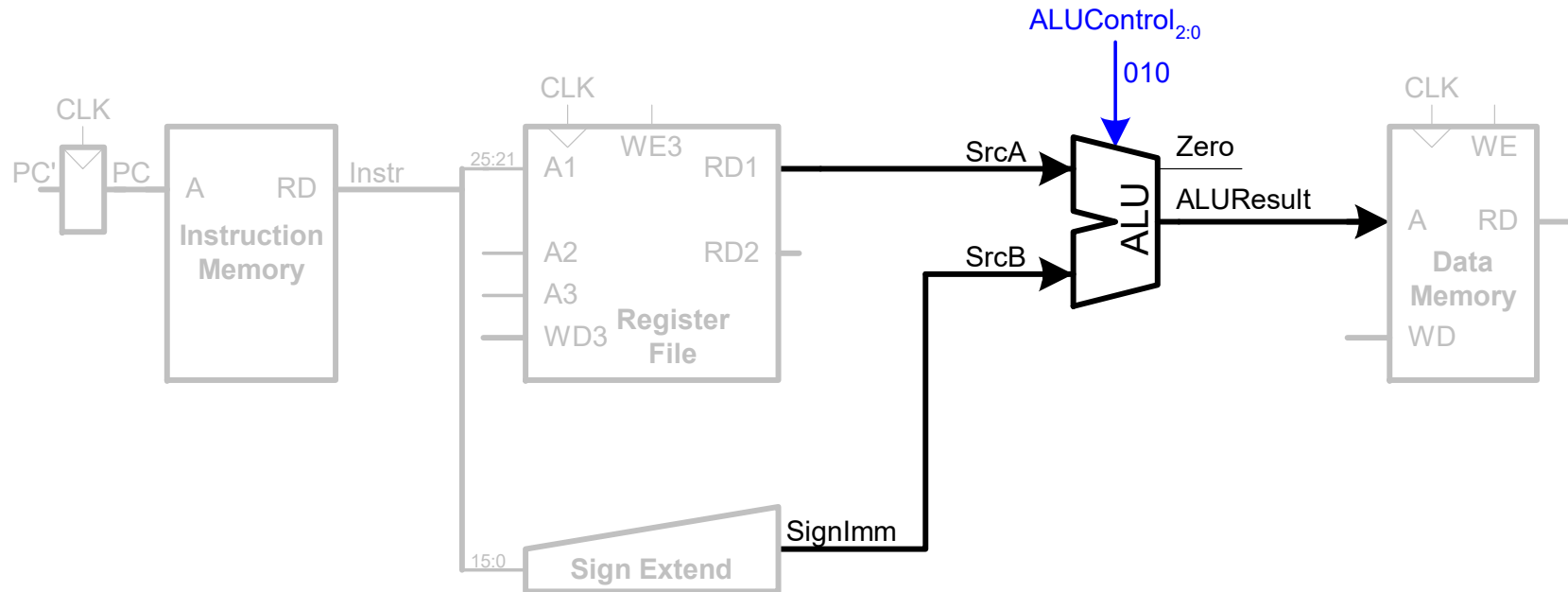
```
lw $s1, 0x2000($0)
```

# Ruta de datos: instrucción SW

Sobre esta ruta de datos, vemos qué falta para otras instrucciones:

Empezamos con `sw $s1, 0x2000($0)` (**0xAC112000**)

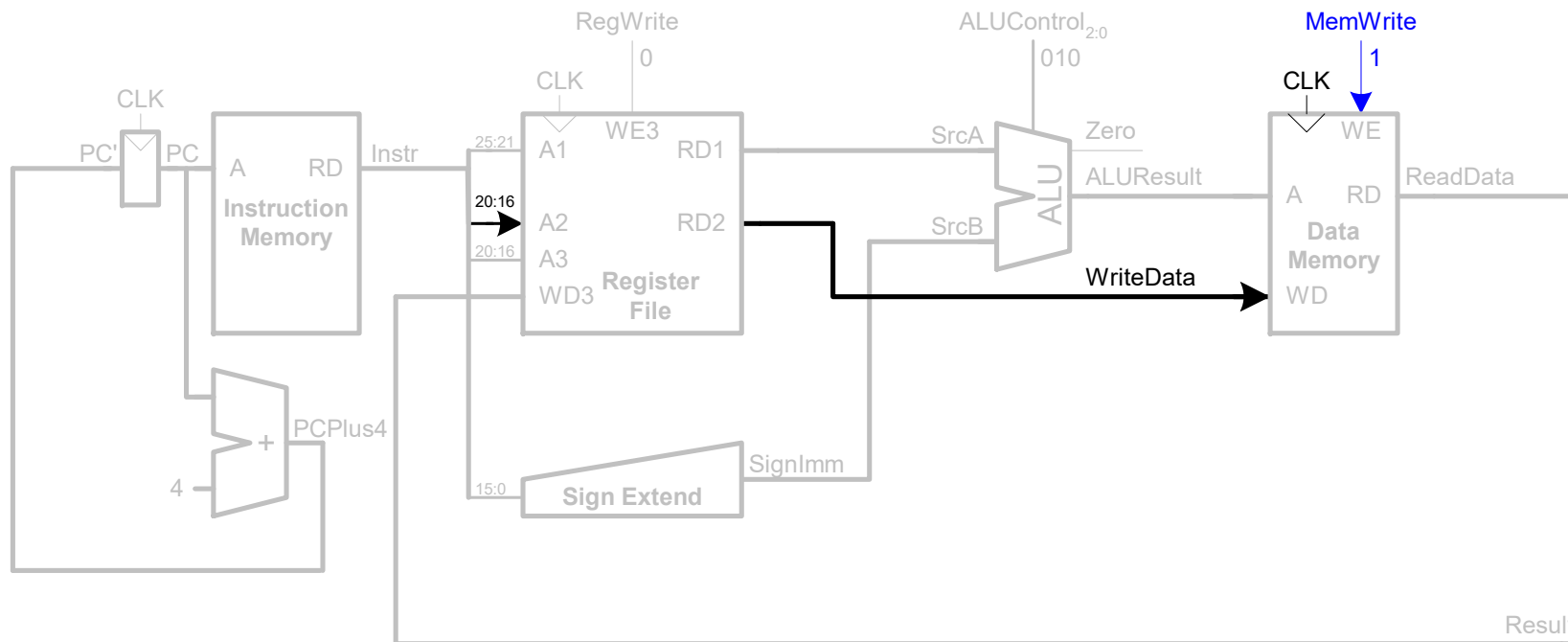
- ✓ La dirección de escritura se calculará igual que en el caso de lw



$ALUResult \leq 0x00002000 = 0x00000000 + 0x00002000$

# Ruta de datos: instrucción SW

- ✓ Falta una forma de leer un segundo registro y escribirlo en memoria



A2: Instr [20:16] = "10001" (\$s1)

MEM[0x00002000] <= RD2 = (\$s1)

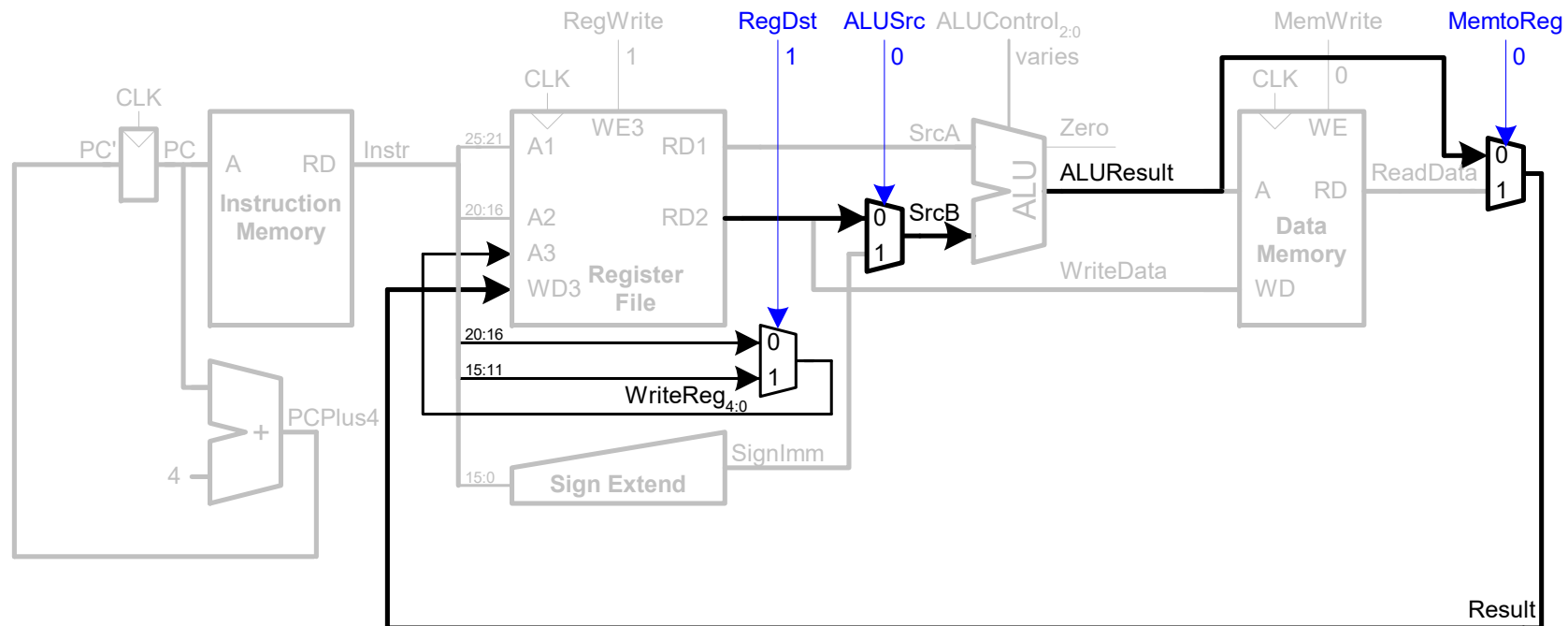
sw \$s1, 0x2000(\$0)



# Ruta de datos: Instrucciones tipo-R

Ejemplo: `add $s1, $t1, $t2`      **(0x012A8820)**

- ✓ Leer dos registros fuente, rs y rt. Ambos son entradas de la ALU
- ✓ Lo que se escribe en registro es ALUResult y no lo que viene de memoria
- ✓ Se escribe en rd (en lugar de rt).



A3: Instr [15:11] = "10001" ( $\$s1$ )

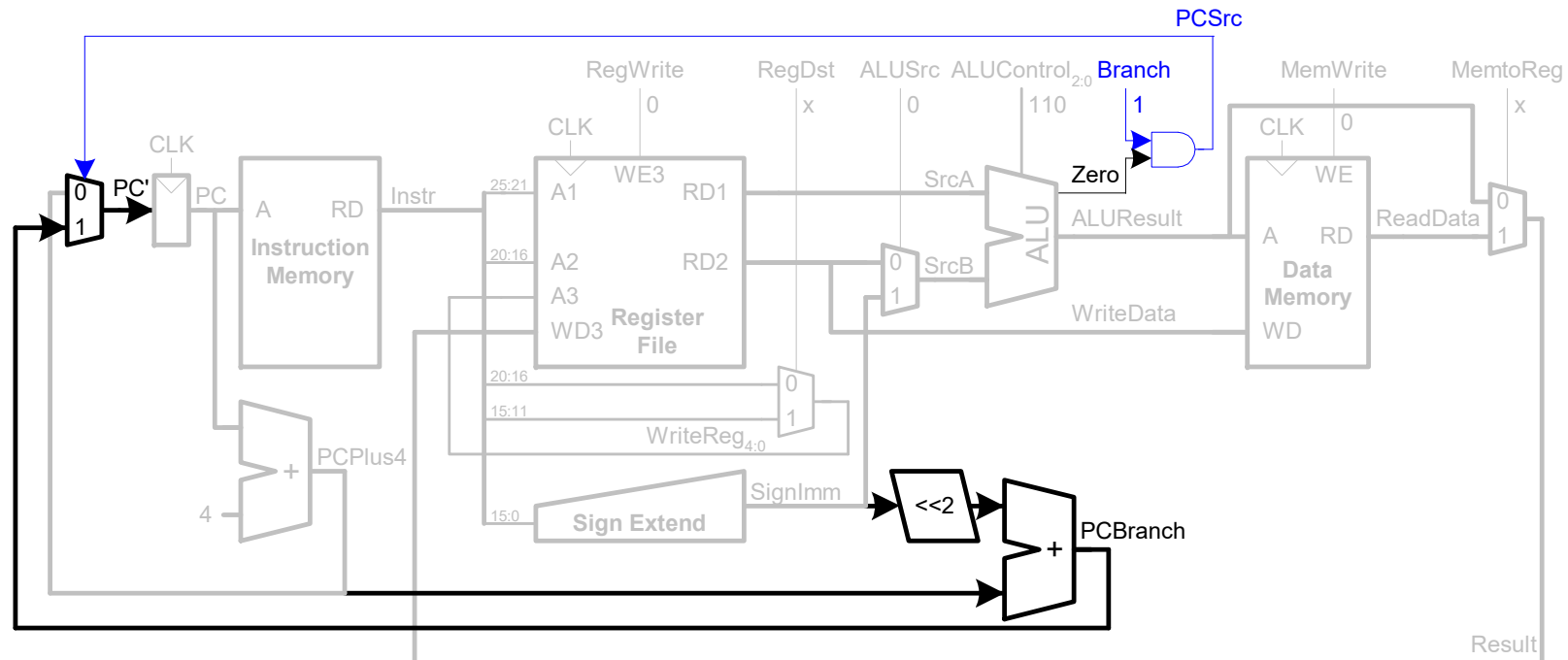
$\$s1 \leftarrow WD3 = ALUResult (\$t1 + \$t2)$

# Ruta de datos: instrucción beq

Se decide si se salta o no con la bandera Z

✓ Cálculo de la dirección de salto (*Branch Target Address*):

$$\text{BTA} = (\text{PC}+4) + (\text{Sign Extend } \{\text{imm} \ll 2\})$$



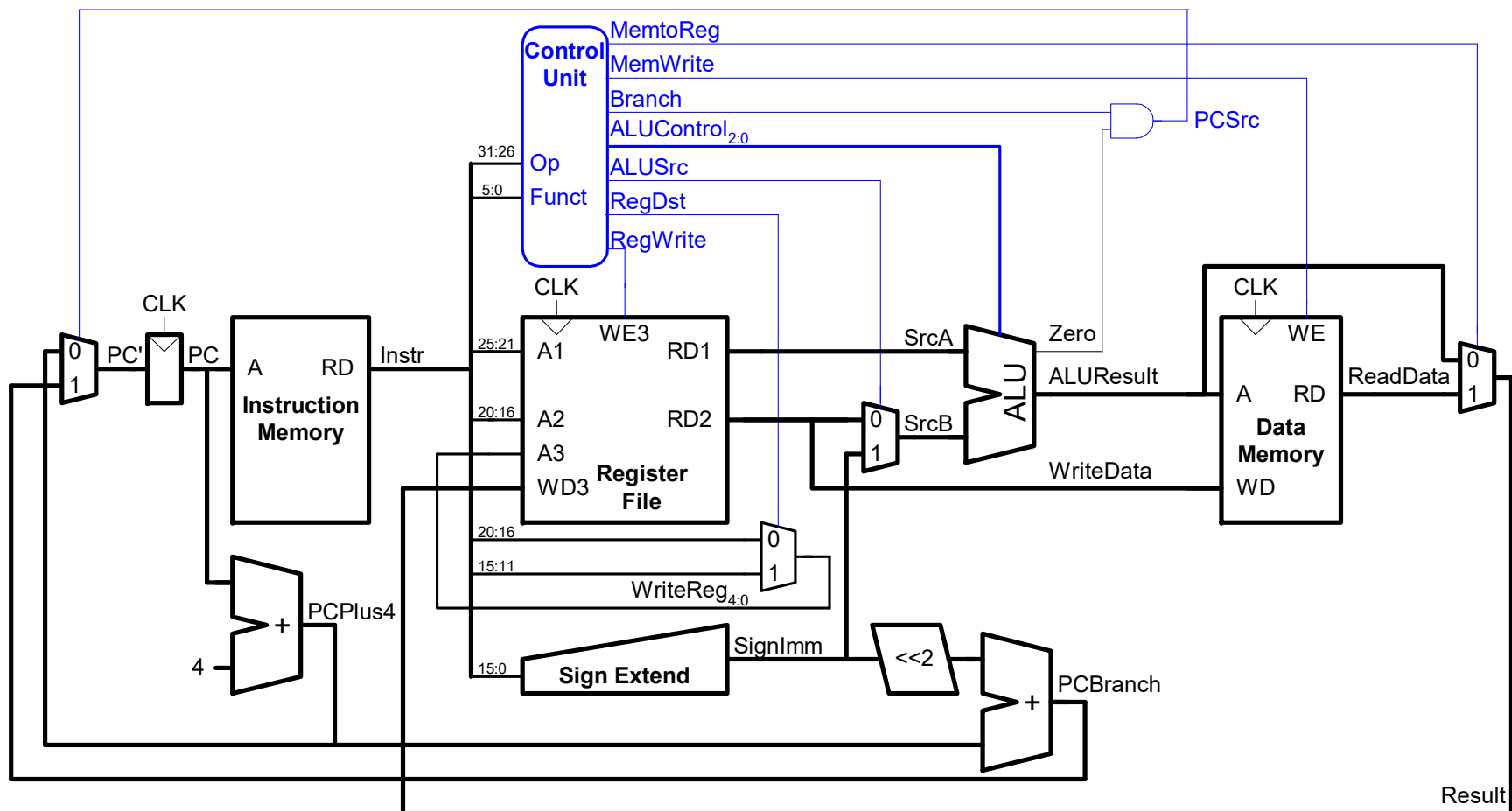
Branch se pondrá a 1 si se está ejecutando un beq

PCSrc se pondrá a 1 si Branch=1 y la condición de salto se cumple (Z=1)

# Ruta de datos y de control

Las señales de control se generan según cada instrucción

**(Hay que decodificar opcode y funct)**

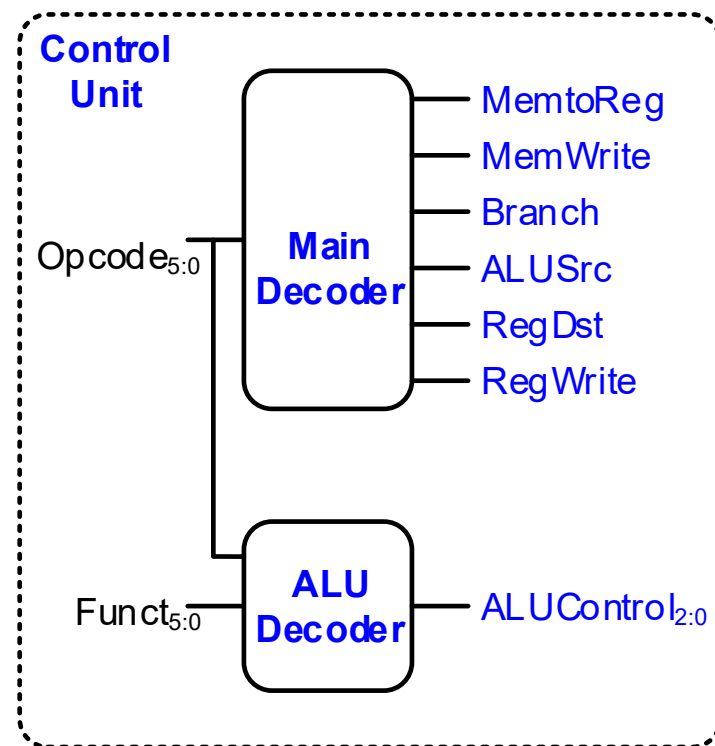


# Índice

- Introducción
- Ruta de datos unicity
- **Control unicity**
- Añadir más instrucciones
- Parámetros temporales en la ruta unicity

# Unidad de Control

- La UC consite en la generación de dos buses de control:
  - ALUControl (3 bits): depende de **opcode** y **funct**
  - Resto (6 bits): sólo depende de **opcode**, no depende de **funct**

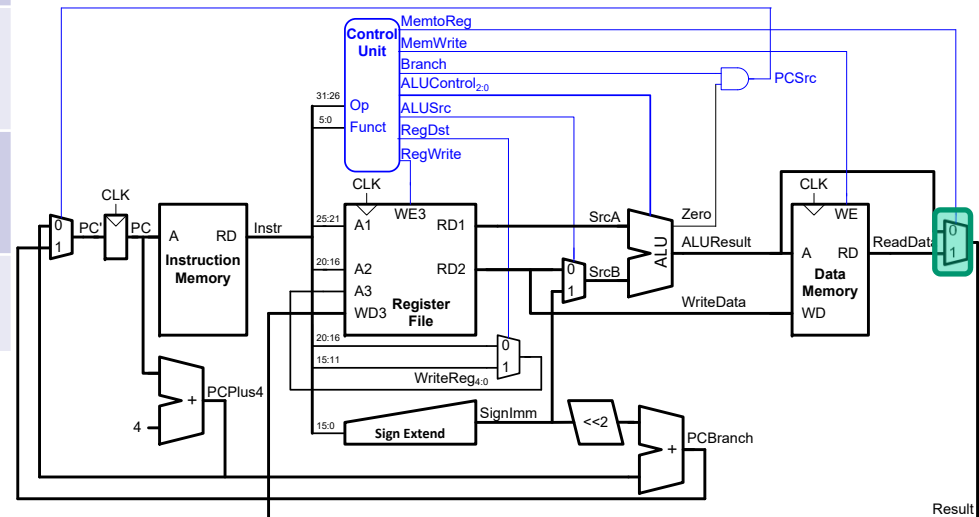


# Decodificador de la ALU

<b>OPCODE</b>	<b>Funct</b>	<b>ALUControl<sub>2:0</sub></b>
100011 (lw) 101011 (sw)	X	010 (Sumar)
000100 (beq)	X	110 (Restar)
000000	100000 (add)	010 (Sumar)
000000	100010 (sub)	110 (Restar)
000000	100100 (and)	000 (Y lógico)
000000	100101 (or)	001 (O lógico)
000000	101010 (slt)	111 ( <b>SetLessThan</b> )

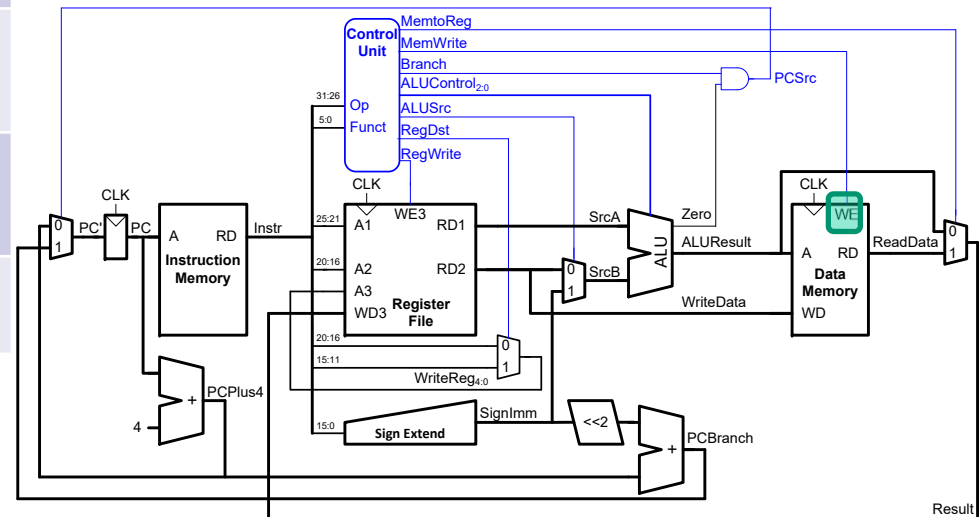
# Ruta de control

Señal	Significado	Significado de valores	
		0	1
<b>MemToReg</b>	Decide si al banco de registros le llega un dato de la memoria de datos o si le llega el resultado de la ALU	Resultado de la ALU	Dato de la memoria de datos
MemWrite	Decide si se va a escribir un registro en la memoria de datos.	No	Sí
Branch	Indica si se está ejecutando una instrucción beq	No	Sí
PcSrc	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1	No	Sí
ALUSrc	Decide cuál será el segundo operando de la ALU		
RegDst	Indica la dirección de registro destino		
RegWrite	Indica si el banco de registros debe guardar un resultado		



# Ruta de control

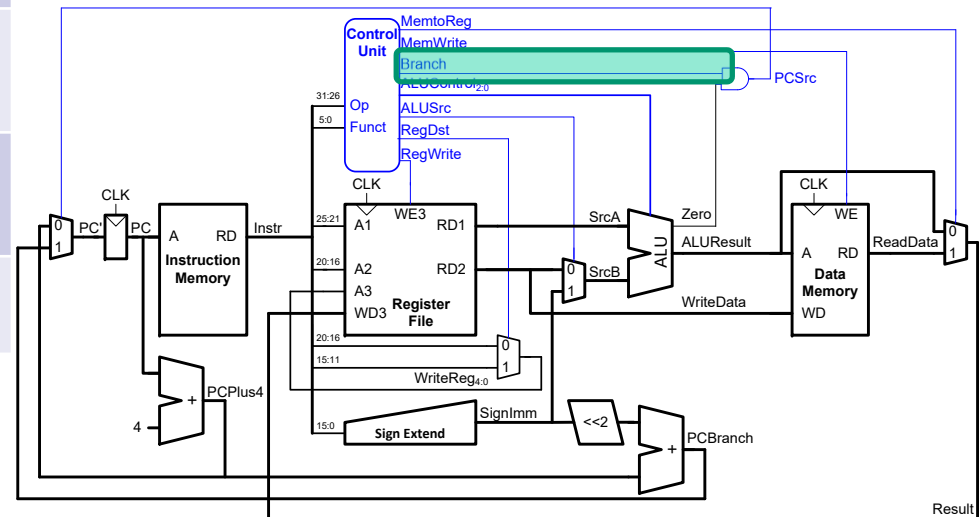
Señal	Significado	Significado de valores	
		0	1
<b>MemToReg</b>	Decide si al banco de registros le llega un dato de la memoria de datos o si le llega el resultado de la ALU	Resultado de la ALU	Dato de la memoria de datos
<b>MemWrite</b>	Decide si se va a escribir un registro en la memoria de datos.	No	Sí
Branch	Indica si se está ejecutando una instrucción beq	No	Sí
PcSrc	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1	No	Sí
ALUSrc	Decide cuál será el segundo operando de la ALU		
RegDst	Indica la dirección de registro destino		
RegWrite	Indica si el banco de registros debe guardar un resultado		





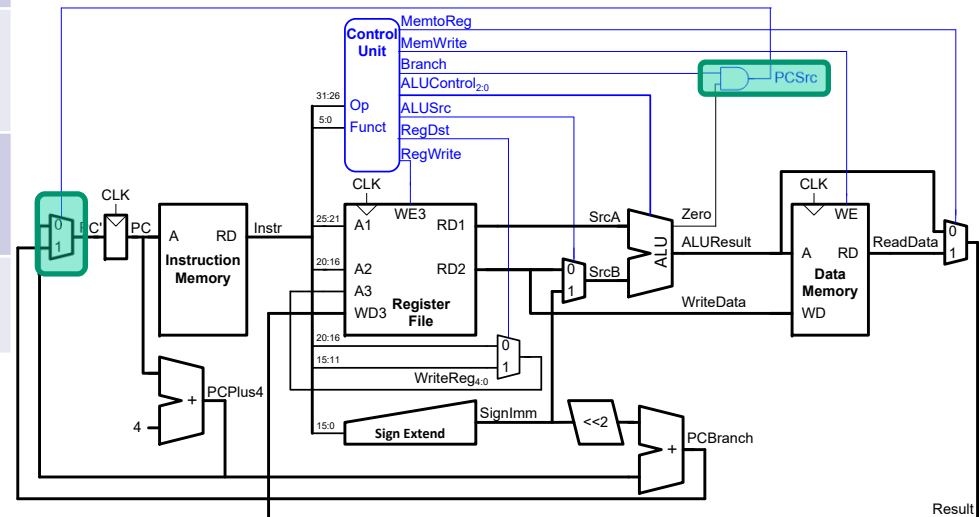
# Ruta de control

Señal	Significado	Significado de valores	
		0	1
<b>MemToReg</b>	Decide si al banco de registros le llega un dato de la memoria de datos o si le llega el resultado de la ALU	Resultado de la ALU	Dato de la memoria de datos
<b>MemWrite</b>	Decide si se va a escribir un registro en la memoria de datos.	No	Sí
<b>Branch</b>	Indica si se está ejecutando una instrucción beq	No	Sí
<b>PcSrc</b>	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1	No	Sí
<b>ALUSrc</b>	Decide cuál será el segundo operando de la ALU		
<b>RegDst</b>	Indica la dirección de registro destino		
<b>RegWrite</b>	Indica si el banco de registros debe guardar un resultado		



# Ruta de control

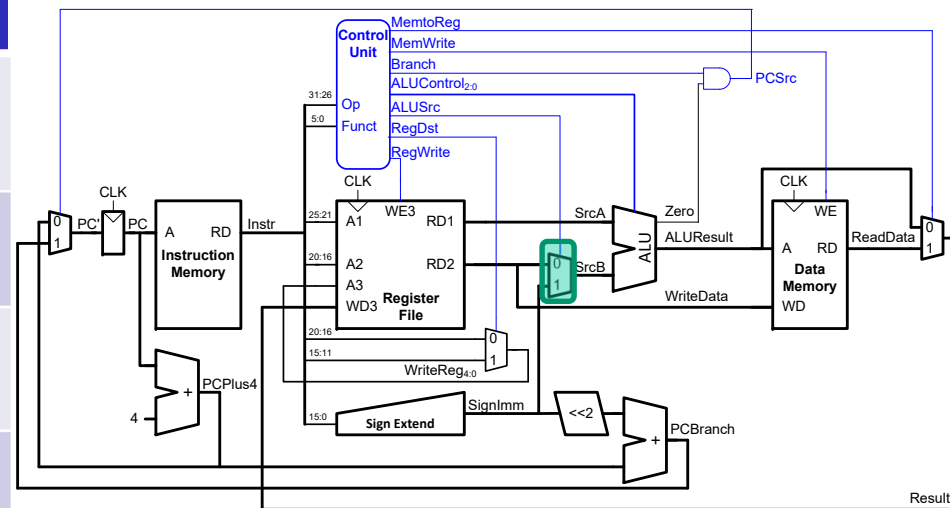
Señal	Significado	Significado de valores	
		0	1
<b>MemToReg</b>	Decide si al banco de registros le llega un dato de la memoria de datos o si le llega el resultado de la ALU	Resultado de la ALU	Dato de la memoria de datos
<b>MemWrite</b>	Decide si se va a escribir un registro en la memoria de datos.	No	Sí
<b>Branch</b>	Indica si se está ejecutando una instrucción beq	No	Sí
<b>PcSrc</b>	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1	No	Sí
ALUSrc	Decide cuál será el segundo operando de la ALU		
RegDst	Indica la dirección de registro destino		
RegWrite	Indica si el banco de registros debe guardar un resultado		



# Ruta de control

Señal	Significado
<b>MemToReg</b>	Decide si al banco de registros le llega un dato de la memoria de datos o si le llega el resultado de la ALU
<b>MemWrite</b>	Decide si se va a escribir un registro en la memoria de datos.
<b>Branch</b>	Indica si se está ejecutando una instrucción beq
<b>PcSrc</b>	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1
<b>ALUSrc</b>	Decide cuál será el segundo operando de la ALU
RegDst	Indica la dirección de registro destino
RegWrite	Indica si el banco de registros debe guardar un resultado

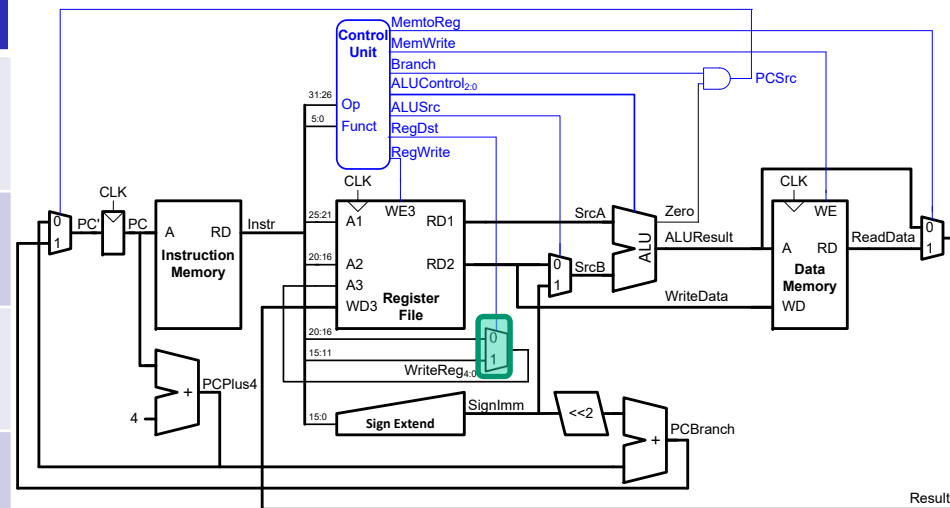
Significado de valores	
Registro RT [20:16]	Dato Inm
Registro RT [20:16]	Registro RD [15:11]
No	Sí



# Ruta de control

Señal	Significado
<b>MemToReg</b>	Decide si al banco de registros le llega un dato de la memoria de datos o si le llega el resultado de la ALU
<b>MemWrite</b>	Decide si se va a escribir un registro en la memoria de datos.
<b>Branch</b>	Indica si se está ejecutando una instrucción beq
<b>PcSrc</b>	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1
<b>ALUSrc</b>	Decide cuál será el segundo operando de la ALU
<b>RegDst</b>	Indica la dirección de registro destino
RegWrite	Indica si el banco de registros debe guardar un resultado

## Significado de valores

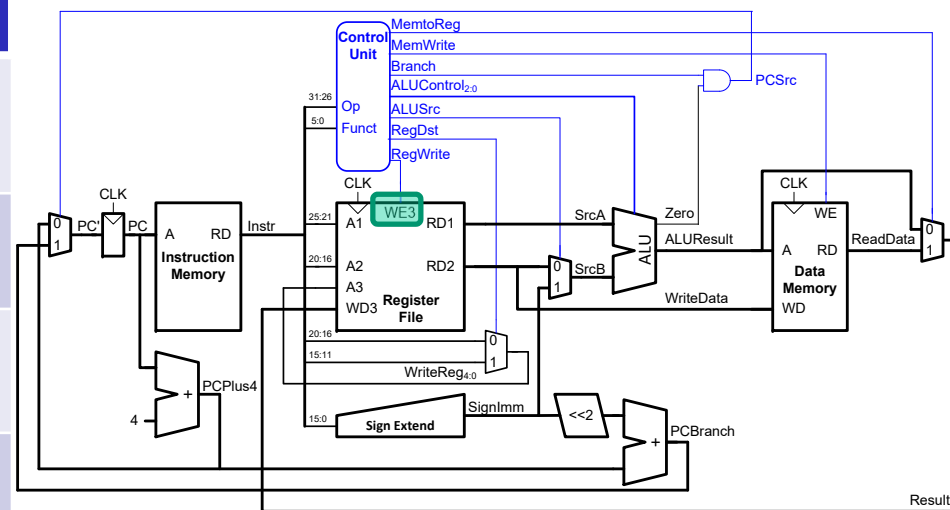


Registro RT [20:16]	Dato Inm
Registro RT [20:16]	Registro RD [15:11]
No	Sí

# Ruta de control

Señal	Significado
<b>MemToReg</b>	Decide si al banco de registros le llega un dato de la memoria de datos o si le llega el resultado de la ALU
<b>MemWrite</b>	Decide si se va a escribir un registro en la memoria de datos.
<b>Branch</b>	Indica si se está ejecutando una instrucción beq
<b>PcSrc</b>	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1
<b>ALUSrc</b>	Decide cuál será el segundo operando de la ALU
<b>RegDst</b>	Indica la dirección de registro destino
<b>RegWrite</b>	Indica si el banco de registros debe guardar un resultado

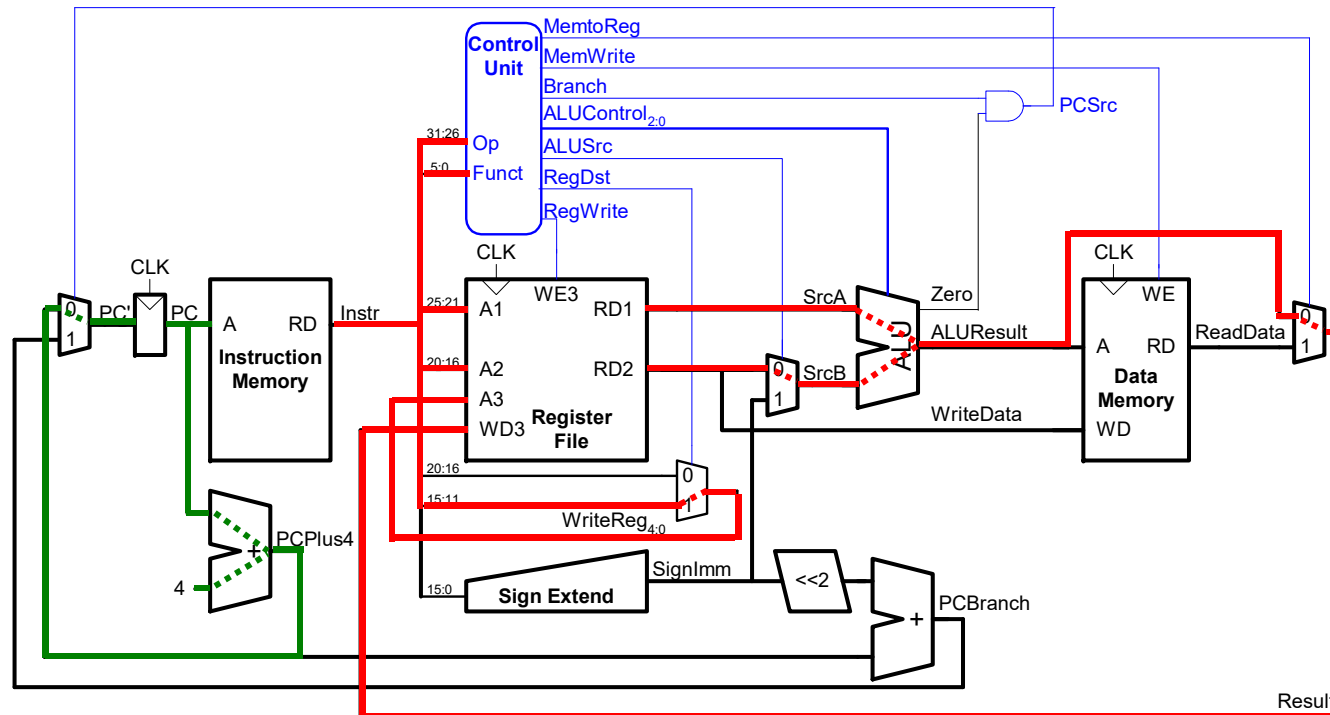
## Significado de valores



<b>ALUSrc</b>	Registro RT [20:16]	Dato Inm
<b>RegDst</b>	Registro RT [20:16]	Registro RD [15:11]
<b>RegWrite</b>	No	Sí

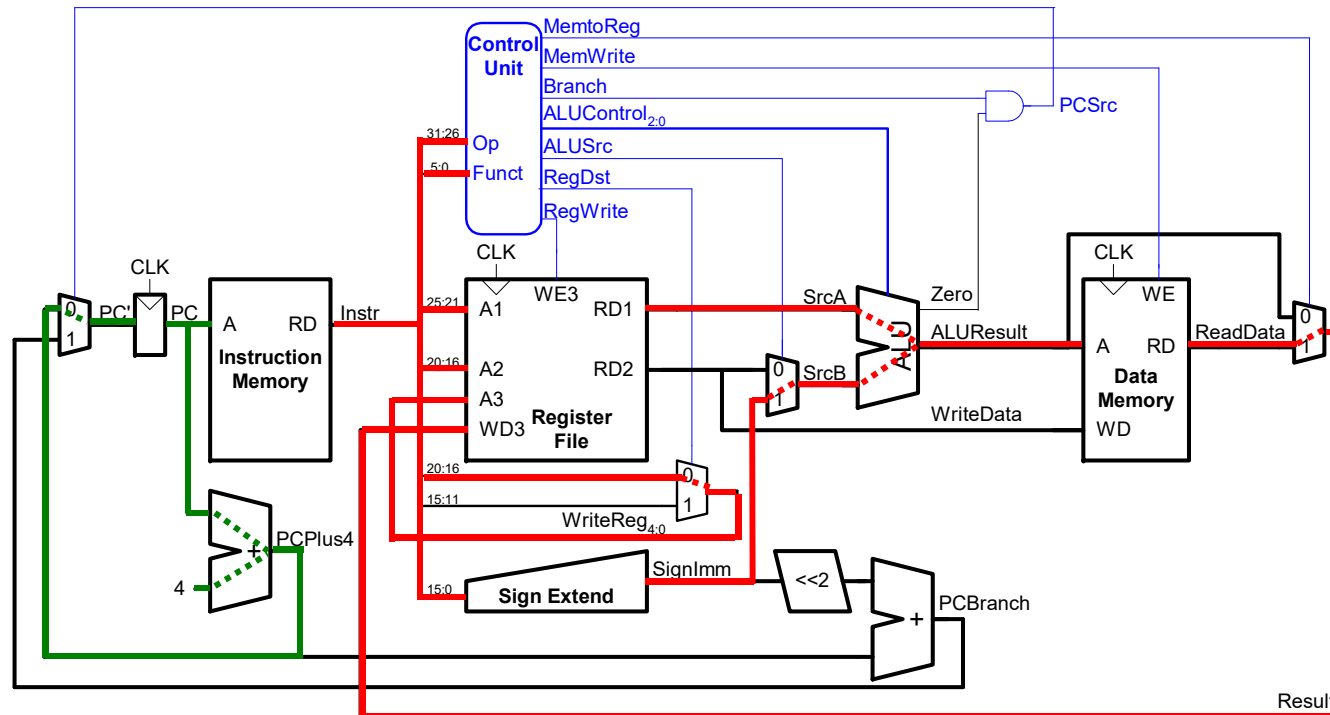
# Decodificador principal (R-type)

Instrucción	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
R-type	000000	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	
lw	100011							
sw	101011							
beq	000100							



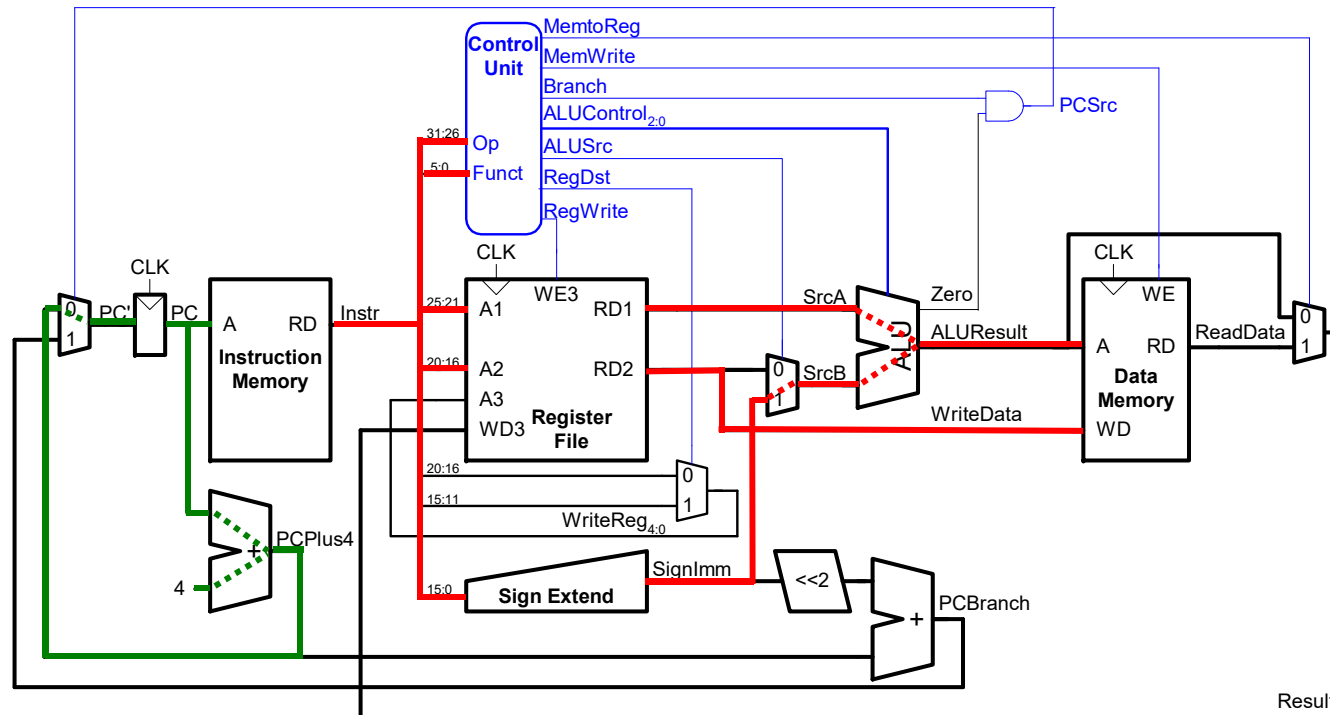
# Decodificador principal (lw)

Instrucción	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
R-type	000000	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	
lw	100011	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>010</b>
sw	101011							
beq	000100							



# Decodificador principal (sw)

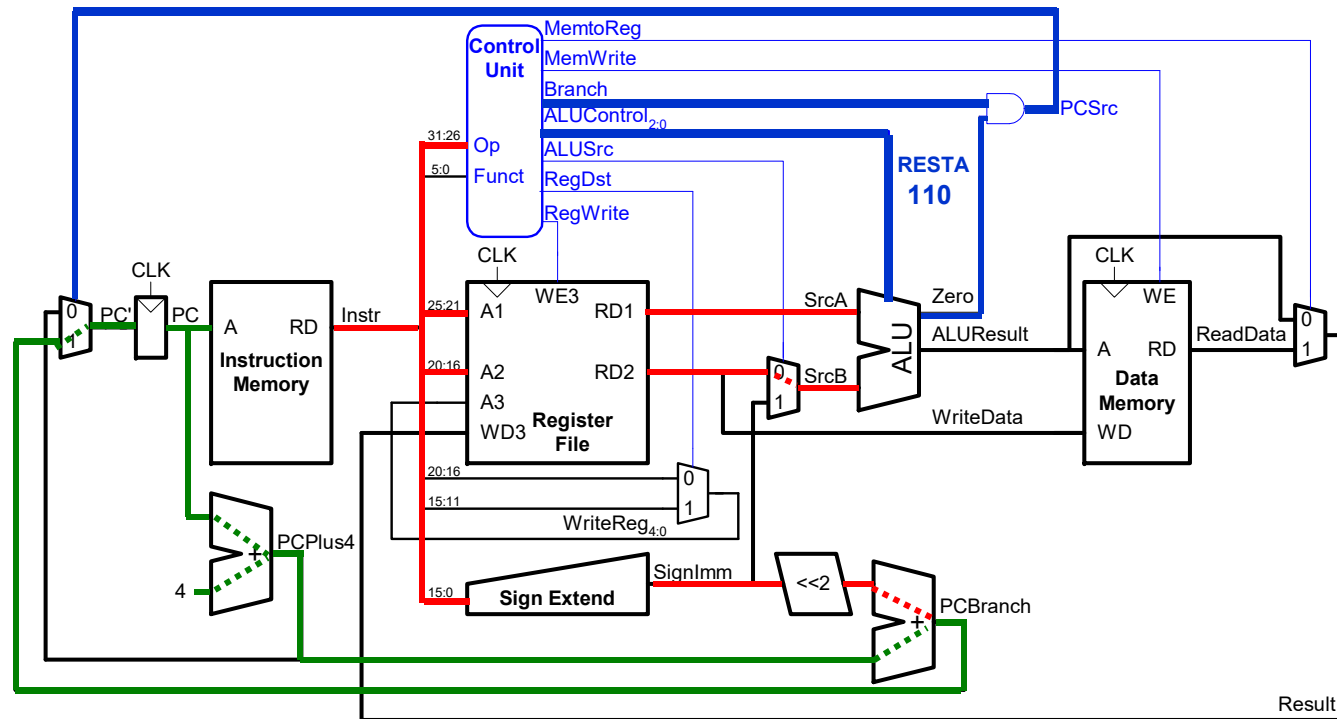
Instrucción	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUctrl
R-type	000000	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	
lw	100011	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>010</b>
sw	101011	<b>0</b>	<b>X</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>X</b>	<b>010</b>
beq	000100							





# Decodificador principal (beq)

Instrucción	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
R-type	000000	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	
lw	100011	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>010</b>
sw	101011	<b>0</b>	<b>X</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>X</b>	<b>010</b>
beq	000100	<b>0</b>	<b>X</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>X</b>	<b>110</b>

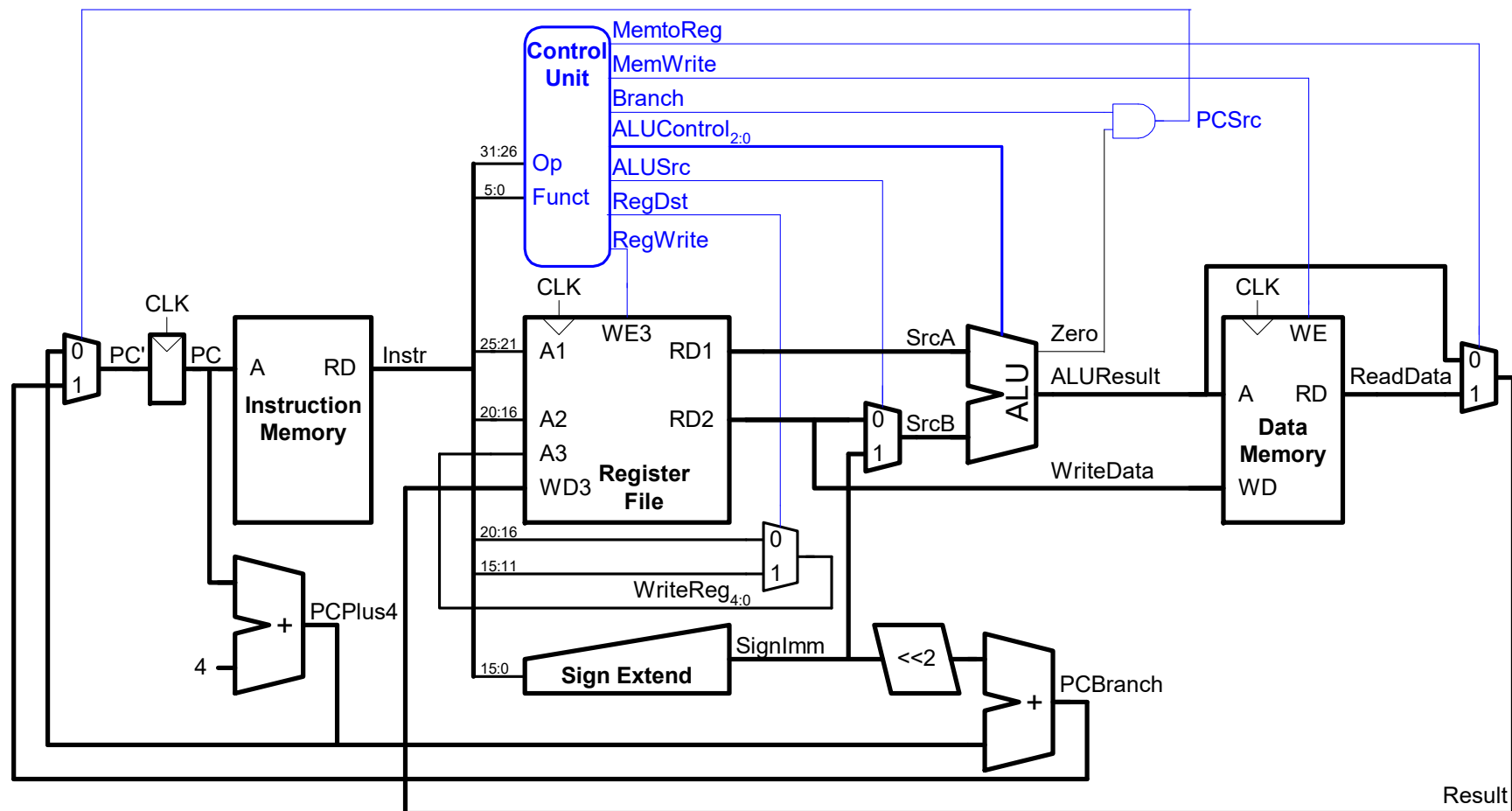


# Índice

- Introducción
- Ruta de datos unicycle
- Control unicycle
- **Añadir más instrucciones**
- Parámetros temporales en la ruta unicycle

# Añadimos instrucciones: addi

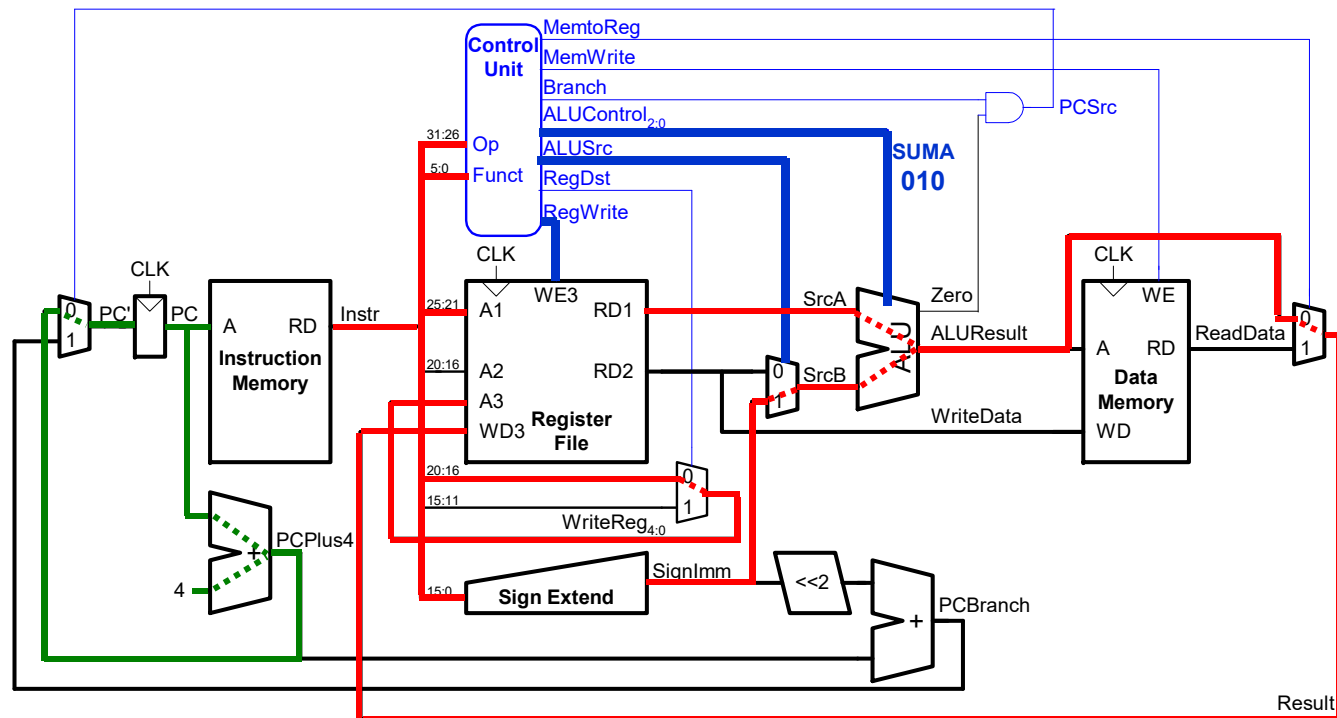
- No hacen falta cambios en la ruta de datos



# Cambios en el control: addi

➤ No hacen falta cambios en la ruta de datos

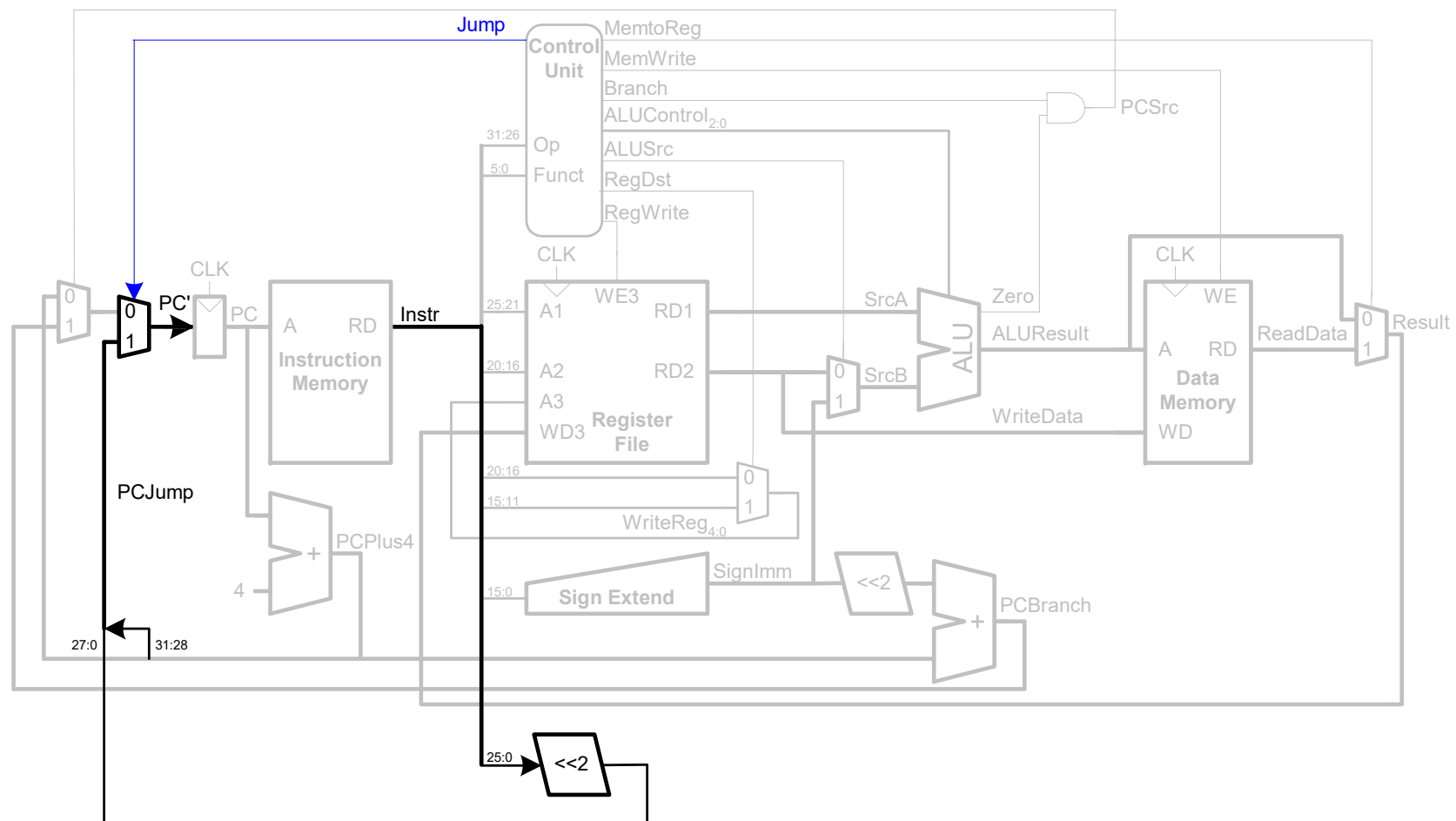
Instrucción	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
addi	001000	1	0	1	0	0	0	010



# Añadimos instrucciones: j

- Cálculo de la dirección de salto (*jump target address*):

$$JTA = (PC+4)[31:28] \& \text{addr} \& \text{"00"}$$



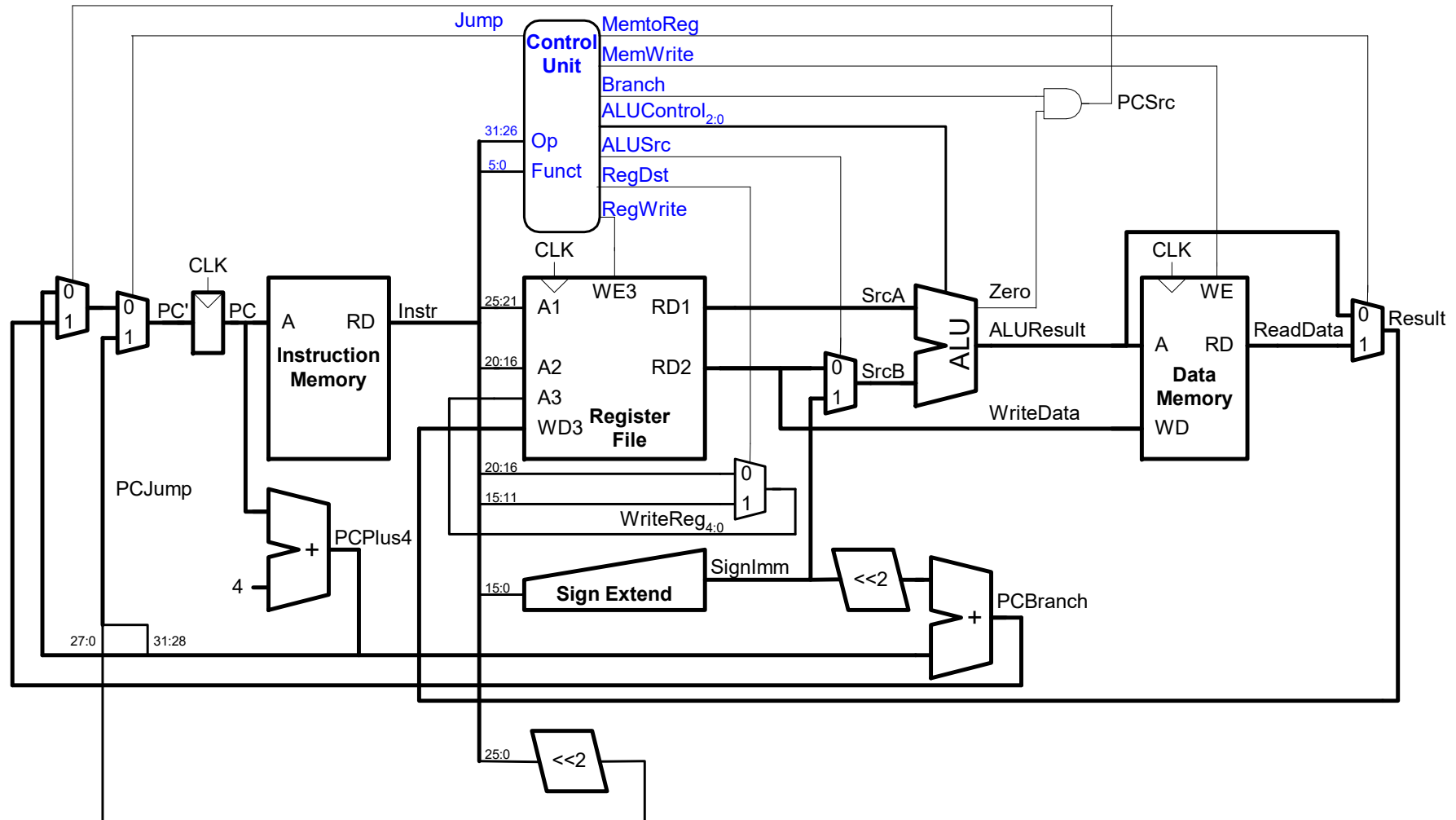
# Cambios en el control: j

Instrucción	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUctrl	Jump
R-type	000000	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>		<b>0</b>
lw	100011	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>010</b>	<b>0</b>
sw	101011	<b>0</b>	<b>X</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>X</b>	<b>010</b>	<b>0</b>
beq	000100	<b>0</b>	<b>X</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>X</b>	<b>110</b>	<b>0</b>
addi	001000	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>010</b>	<b>0</b>
<b>j</b>	<b>000010</b>	<b>0</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>0</b>	<b>X</b>	<b>XX</b>	<b>1</b>

# Ruta de datos y de control

Señal	Significado	Significado de valores		Uso
		0	1	
<b>MemToReg</b>	Decide si al banco de registros le llega un dato de la memoria de datos o si le llega el resultado de la ALU	Resultado de la ALU	Dato de la memoria de datos	1: lw X: sw, beq, jump 0: Resto de operaciones
<b>MemWrite</b>	Decide si se va a escribir un registro en la memoria de datos.	No	Sí	1: sw 0: Resto de operaciones
<b>Branch</b>	Indica si se está ejecutando una instrucción beq	No	Sí	1: beq X: jump 0: Resto de operaciones
<b>PcSrc</b>	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1	No	Sí	1: beq y condición cumplida 0: Resto de operaciones
<b>ALUControl</b>	Indica la operación a ejecutar en la ALU	-	-	-
<b>ALUSrc</b>	Decide cuál será el segundo operando de la ALU	Registro RT [20:16]	Dato Inm	1: I-type, excepto beq 0: R-type, beq X: jump
<b>RegDst</b>	Indica la dirección de registro destino	Registro RT [20:16]	Registro RD [15:11]	1: R-type 0: I-type, excepto sw y beq, jump X: sw, beq, jump
<b>RegWrite</b>	Indica si el banco de registros debe guardar un resultado	No	Sí	0: sw, beq, jump 1: Resto de operaciones
<b>Jump</b>	Indica si hay una instrucción de tipo jump	No	Sí	1: jump 0: Resto de operaciones

# Resumen: MIPS uniciclo





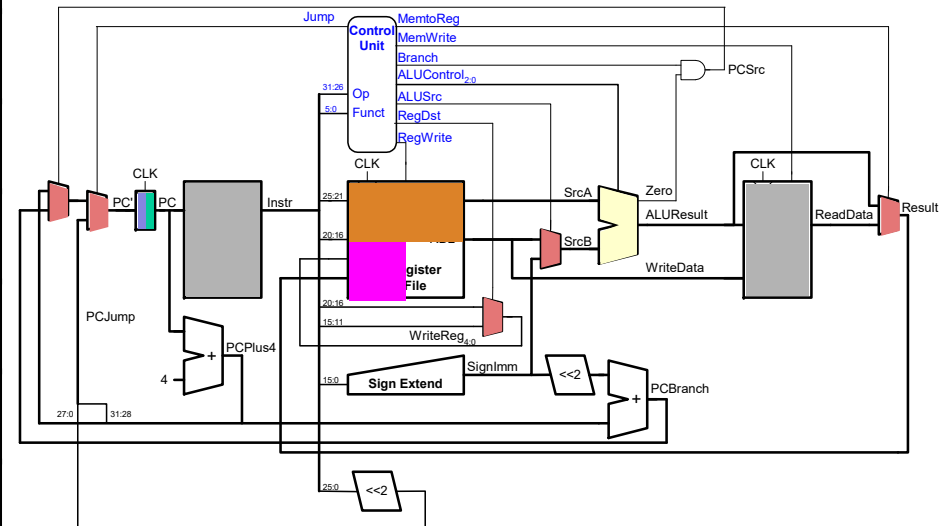
# Índice

- Introducción
- Ruta de datos unicycle
- Control unicycle
- Añadir más instrucciones
- **Parámetros temporales en la ruta unicycle**

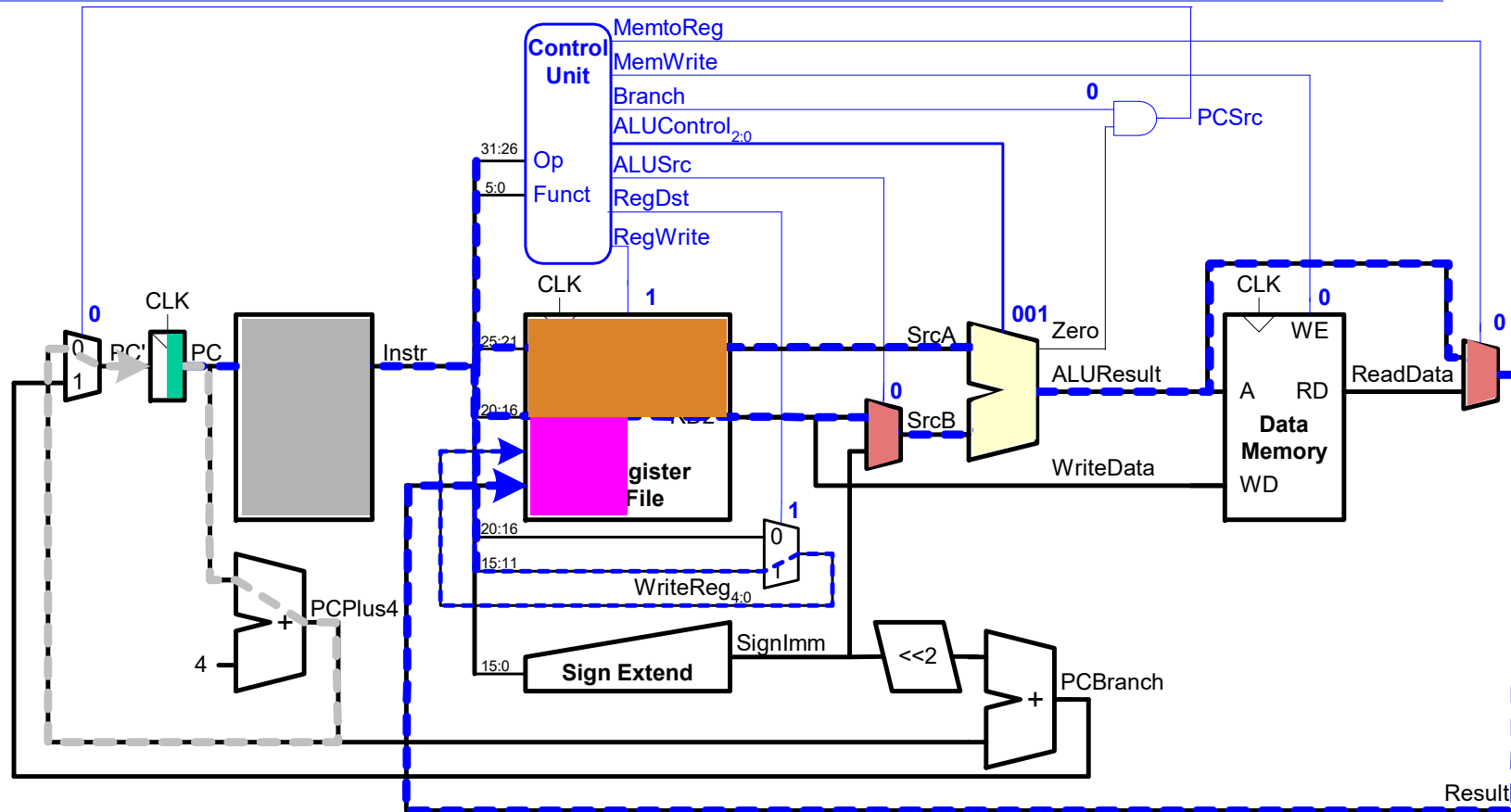
# Parámetros temporales: Ciclo de reloj en uniciclo

- Todas las instrucciones utilizan el mismo ciclo de reloj, que tendrá que ser suficiente para la más lenta de todas.

Elemento	Parámetro	Retardo (ps)
$T_{CLK \rightarrow Q}$ Registro	$t_{pcq\_PC}$	30
Leer de Memoria	$t_{mem}$	250
Leer del BancoReg	$t_{RFread}$	150
Multiplexor	$t_{mux}$	25
ALU	$t_{ALU}$	200
$T_{SETUP}$ BancoReg	$t_{RFsetup}$	20
$T_{SETUP}$ Registro	$t_{setup}$	20



# Parámetros temporales: Ciclo de reloj en un ciclo. Ejemplo: camino crítico para *add*



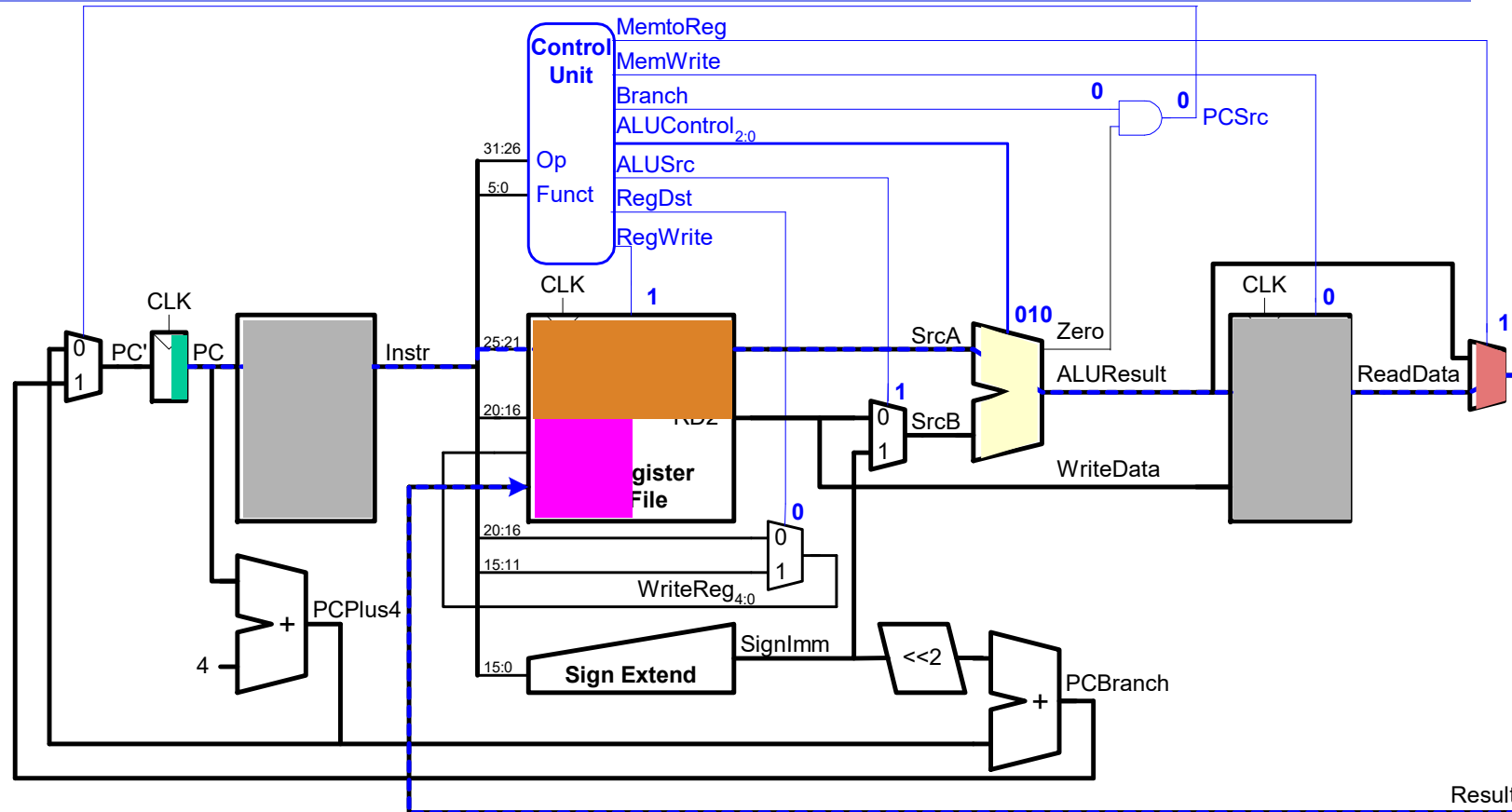
Retraso de la instrucción *add*:

$$T_c = t_{pcq\_PC} + t_{mem} + t_{RFread} + t_{mux} + t_{ALU} + t_{mux} + t_{RFsetup} =$$

$$= [30 + 250 + 150 + 25 + 200 + 25 + 20] \text{ ps} = 700 \text{ ps} (1,4 \text{ GHz})$$

“La escritura en el banco de registros se realiza en el flanco siguiente”

# Parámetros temporales: Ciclo de reloj en un ciclo. Ejemplo: camino crítico para *lw*



Retraso de la instrucción lw:

$$\begin{aligned}
 T_c &= t_{pcq\_PC} + t_{mem} + t_{RFread} + t_{ALU} + t_{mem} + t_{mux} + t_{RFsetup} = \\
 &= [30 + 250 + 150 + 200 + 250 + 25 + 20] \text{ ps} = 925 \text{ ps} (1,1 \text{ GHz})
 \end{aligned}$$

“La escritura en el banco de registros se realiza en el flanco siguiente”

## ***Unidad 4. El Procesador II: Diseño y control de la ruta de datos. Arquitectura unicycle***

---

Escuela Politécnica Superior - UAM