Fundamentos de Microprocesadores

Unidad 4. El Procesador II: Diseño y control de la ruta de datos. Arquitectura uniciclo

Escuela Politécnica Superior - UAM

Índice

- **Introducción**
- Ruta de datos uniciclo
- Control uniciclo
- Añadir más instrucciones

Introducción

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

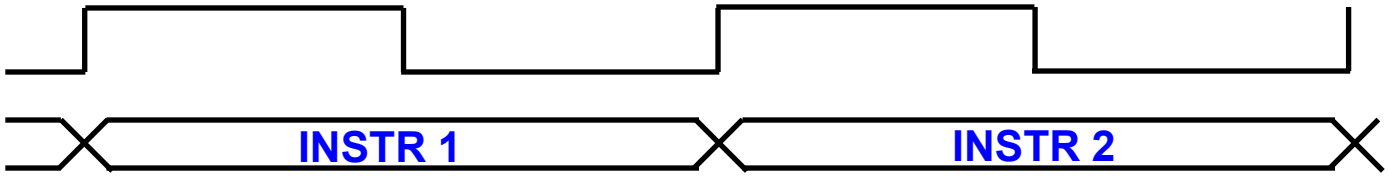
- **Arquitectura:**
  - Es la visión que desde el punto de vista del programador se tiene del sistema computador (U3).
- **Microarquitectura:**
  - Es la implementación en hardware del computador (U4 y U5).
  - Ruta de datos: bloques funcionales
  - Ruta de control: señales de control internas

Microarquitectura

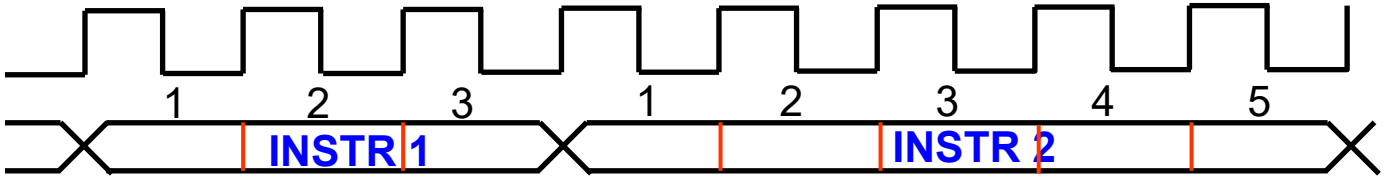
- Hay múltiples implementaciones de la misma arquitectura (juego de instrucciones):
  - Uniciclo (U4)
    - Cada instrucción se ejecuta en un ciclo de reloj
  - Multiciclo (U5)
    - Cada instrucción se divide en pasos cortos, cada uno de un ciclo de reloj mucho más rápido
  - Segmentado (*pipelined*)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

Microarquitectura

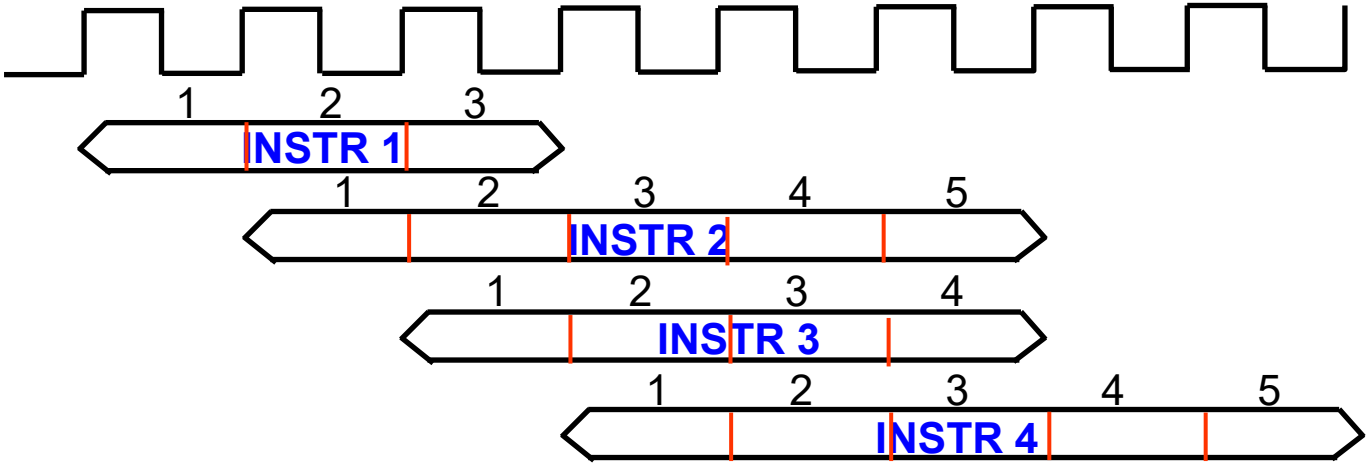
- Uniciclo (U4)



- Multiciclo (U5)



- Segmentado (ARQ 3º)



Subconjunto del MIPS

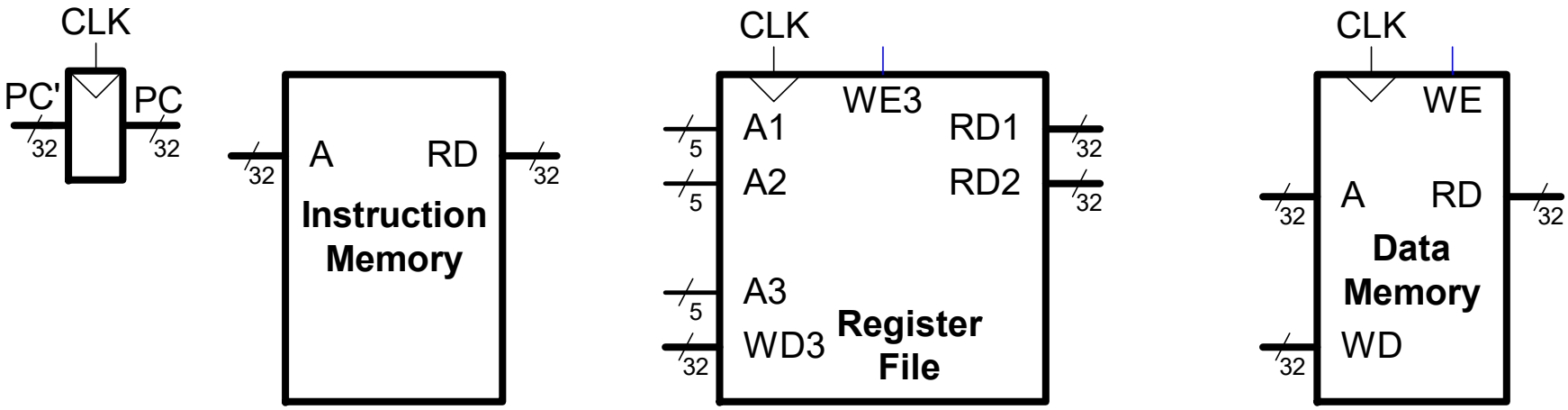
- Para estudiar la microarquitectura consideramos inicialmente sólo un subconjunto del juego de instrucciones:
  - ✓ R-Type: `and, or, add, sub, slt`
  - ✓ I-Type, de memoria : `lw, sw`
  - ✓ I-Type, de saltos: `beq`
- Luego añadiremos más instrucciones (`addi`, `j` (J-Type)).
- En el laboratorio de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

Índice

- Introducción
- **Ruta de datos uniciclo**
- Control uniciclo
- Añadir más instrucciones

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:



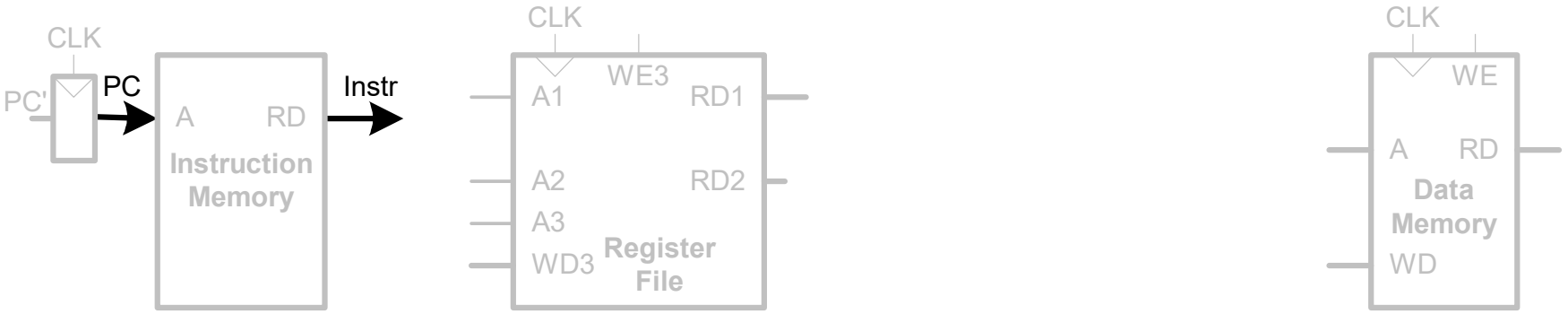
Ruta de datos: captura `lw`

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*)

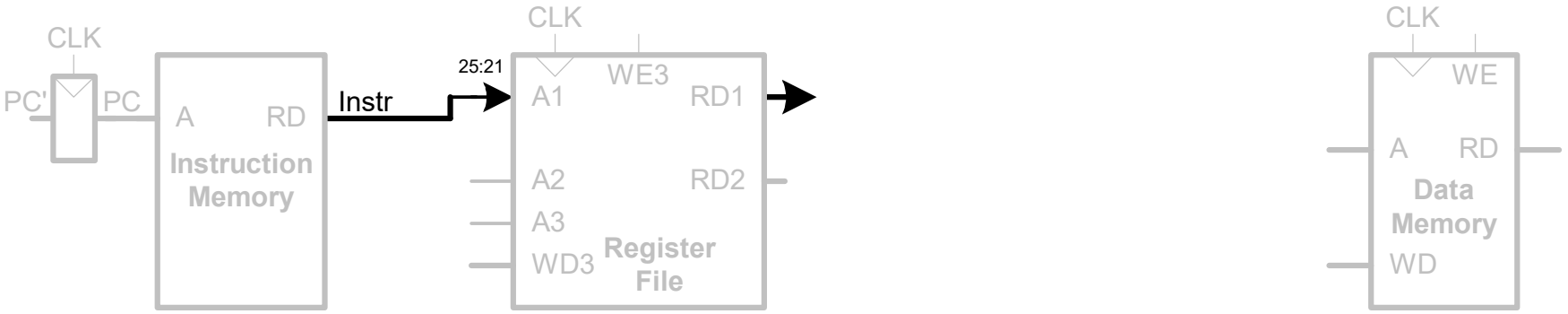


Instr <= “100011 00000 10001 0010000000000000”

op: Instr [31:26] =	“100011”	=>	**`lw`**
rs: Instr [25:21] =	“00000”	=>	**`$0`**
rt: Instr [20:16] =	“10001”	=>	**`$s1`**
imm: Instr [15:0] =	“0010000000000000”	=>	**`0x2000`**

Ruta de datos: lectura de registros `lw`

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

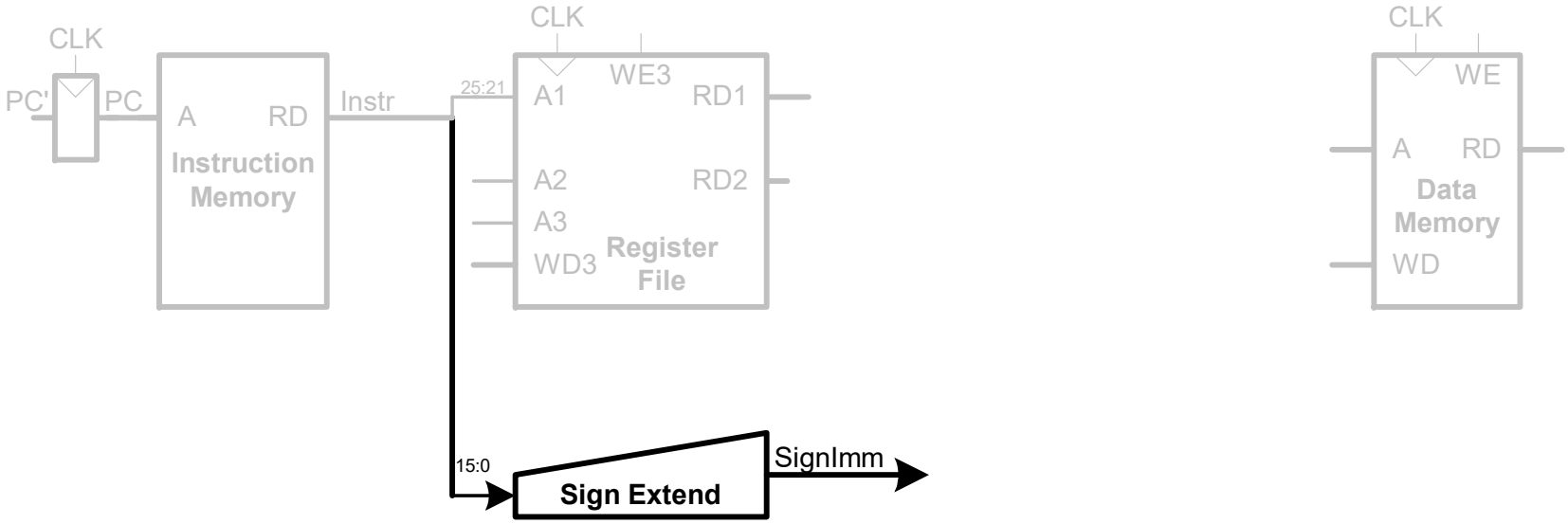


A1: Instr [25:21] = “00000”; RD1 <= “0x00000000” = ($0)

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

Ruta de datos: dato inmediato `lw`

- **PASO 3:** extensión en signo del dato inmediato

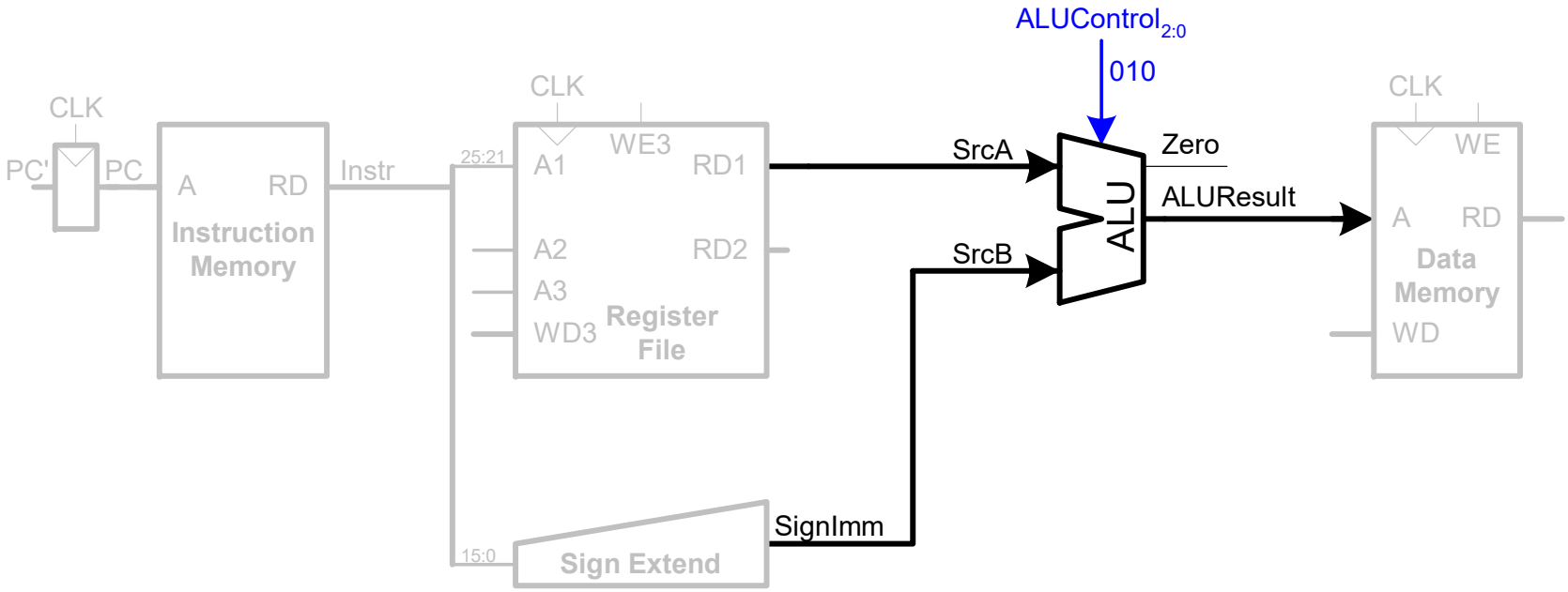


SignImm <= 0x00002000 = Sign Extend (0x2000)

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

Ruta de datos: dirección `lw`

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

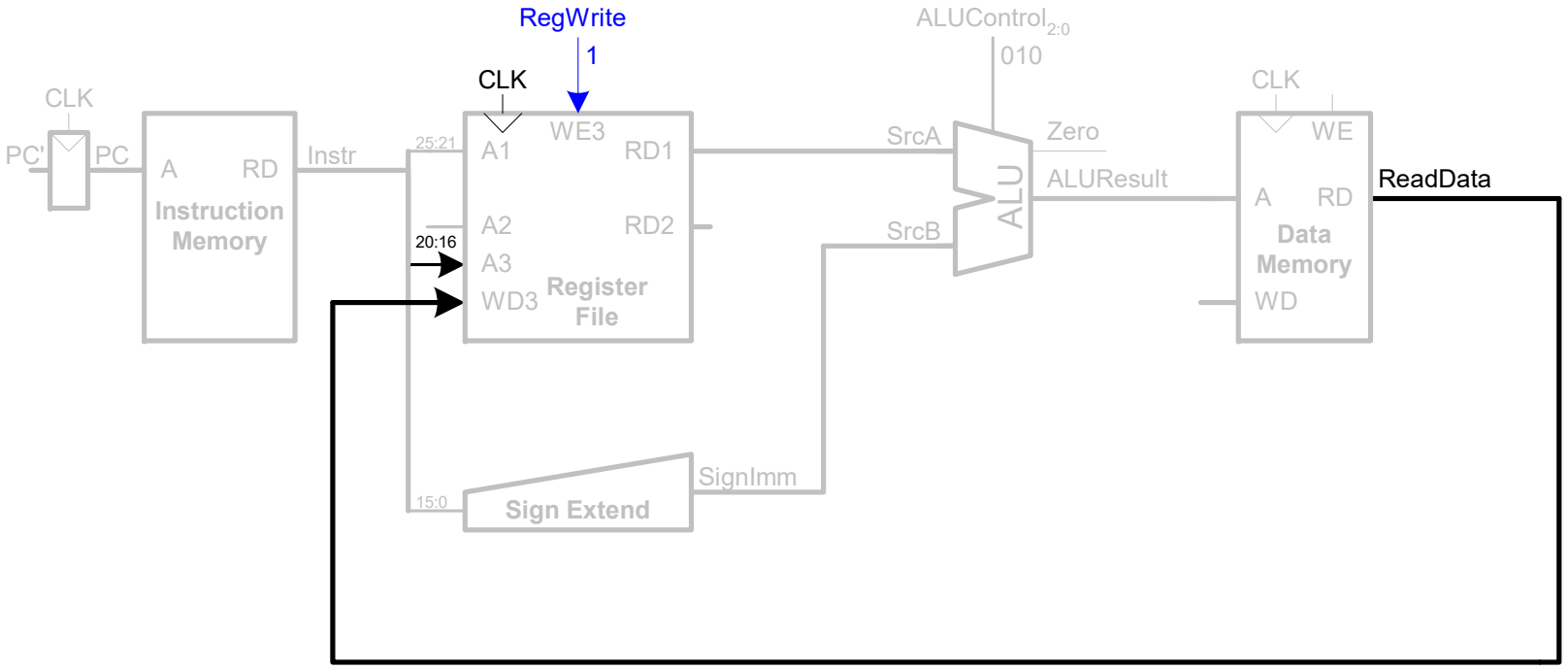


ALUResult <= 0x00002000 = 0x00000000 + 0x00002000

```
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



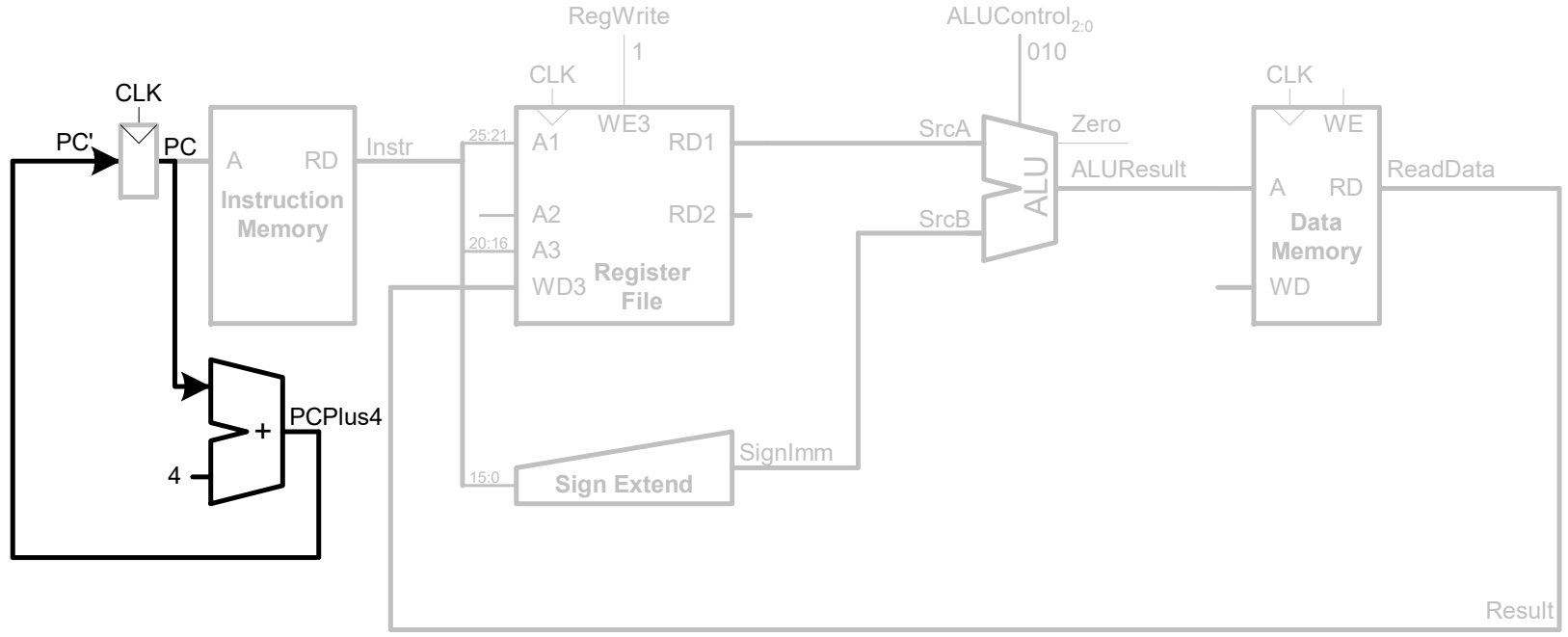
A3: Instr [20:16] = “10001” ($s1)

$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 <= $PC + 4

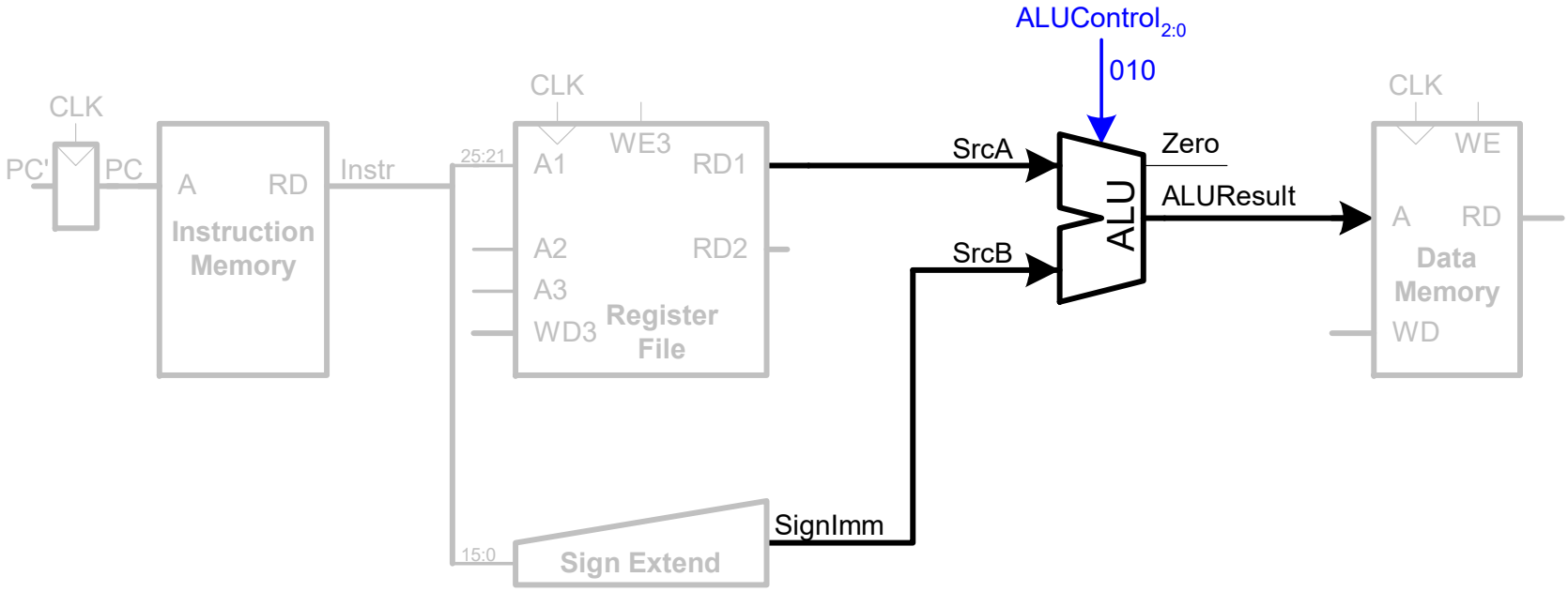
```
lw $s1,0x2000($0)
```

+ Ruta de datos: instrucción `SW`

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

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

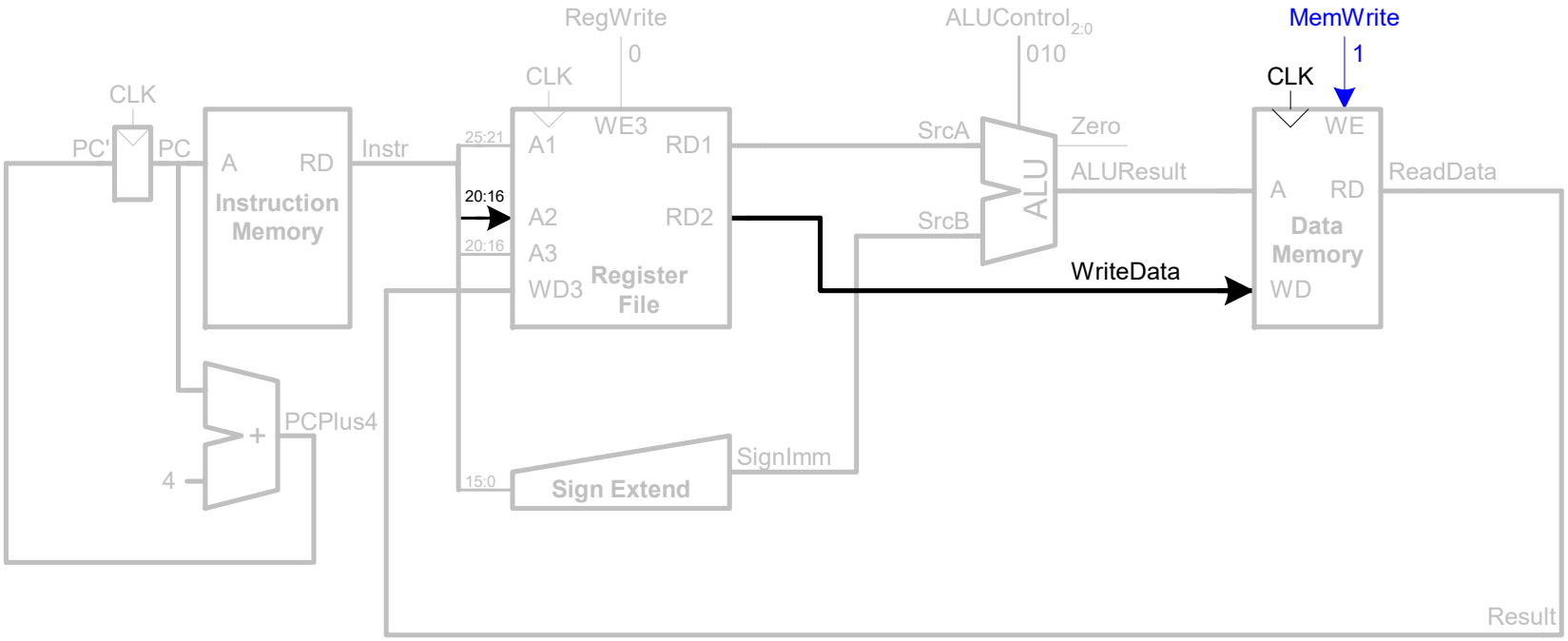
✓ Pasos 1, 2, 3 y 4 iguales que en el caso de lw



ALUResult <= 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)

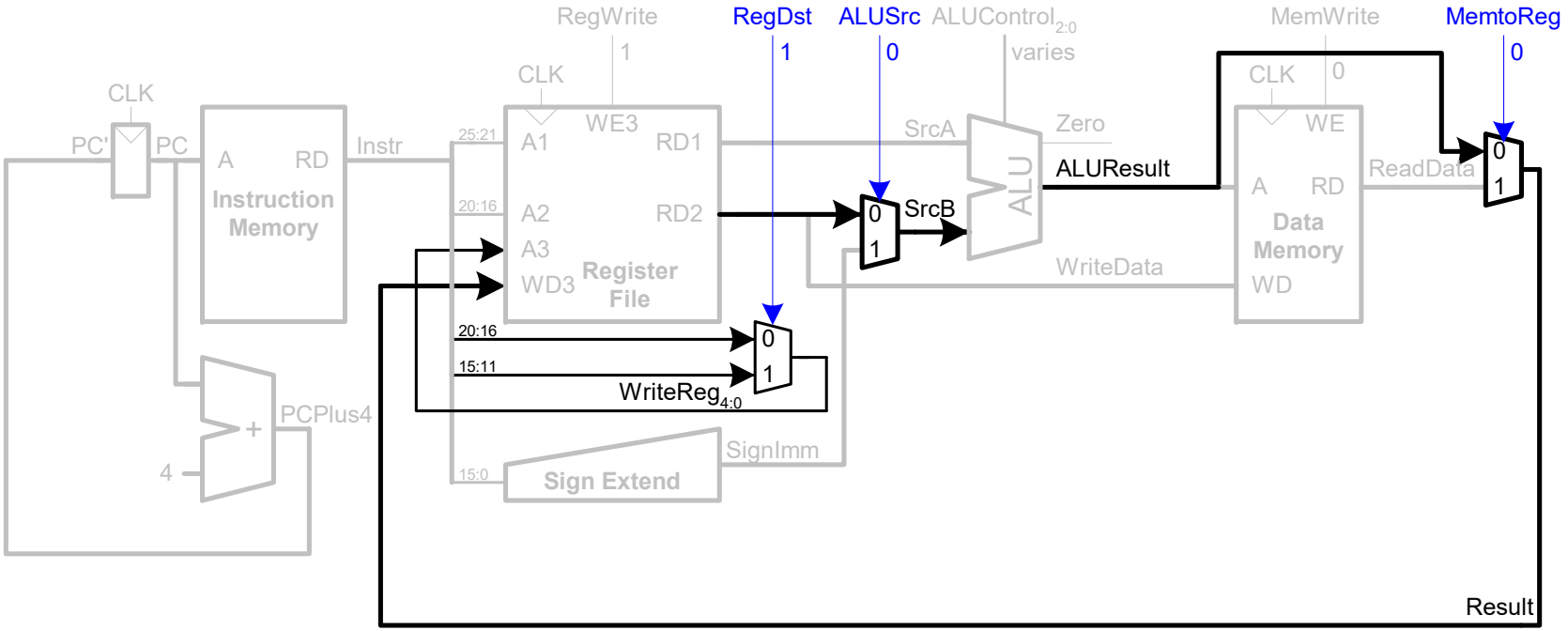
MEM[0x00002000] <= RD2 = ($s1)

```
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).



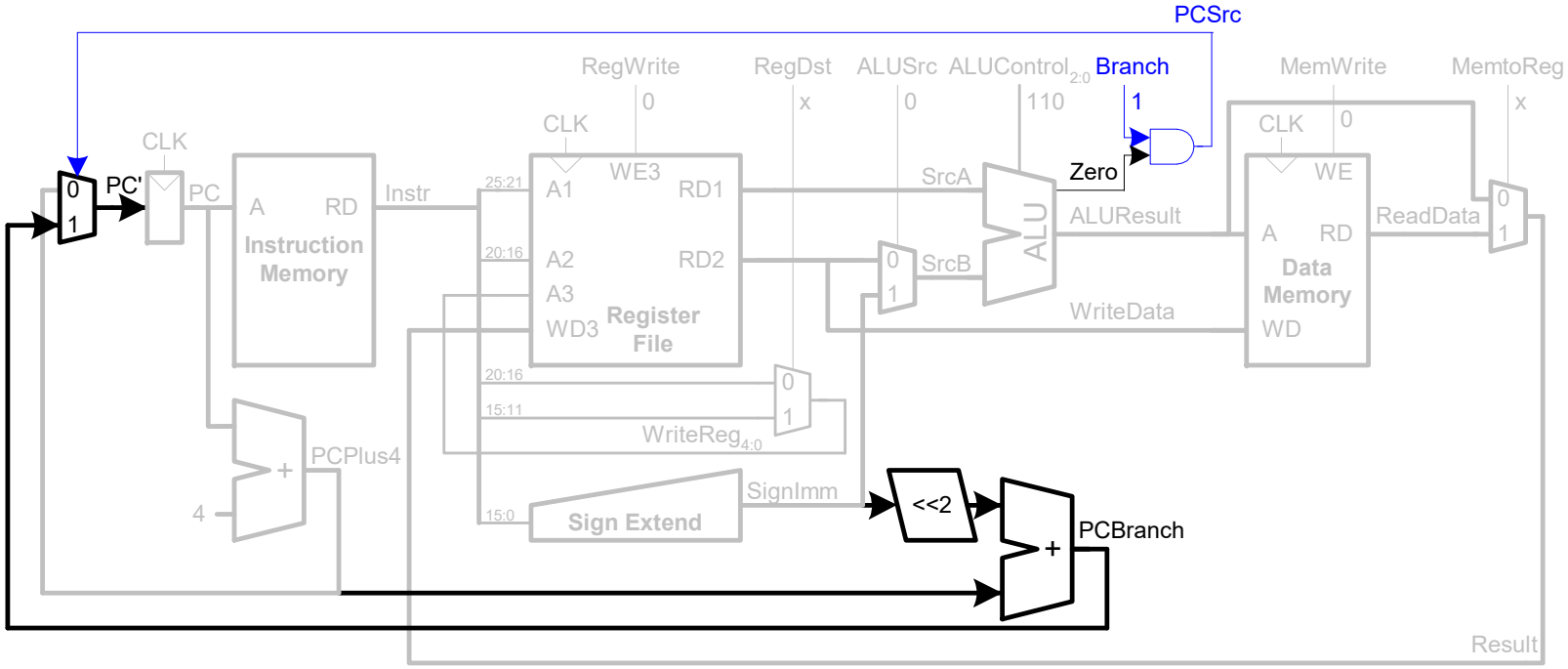
A3: Instr [15:11] = “10001” ($s1) $s1 <=WD3 = ALUResult ($t1+$t2)

+ 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*):

BTA = (PC+4) + (Sign Extend {imm << 2})



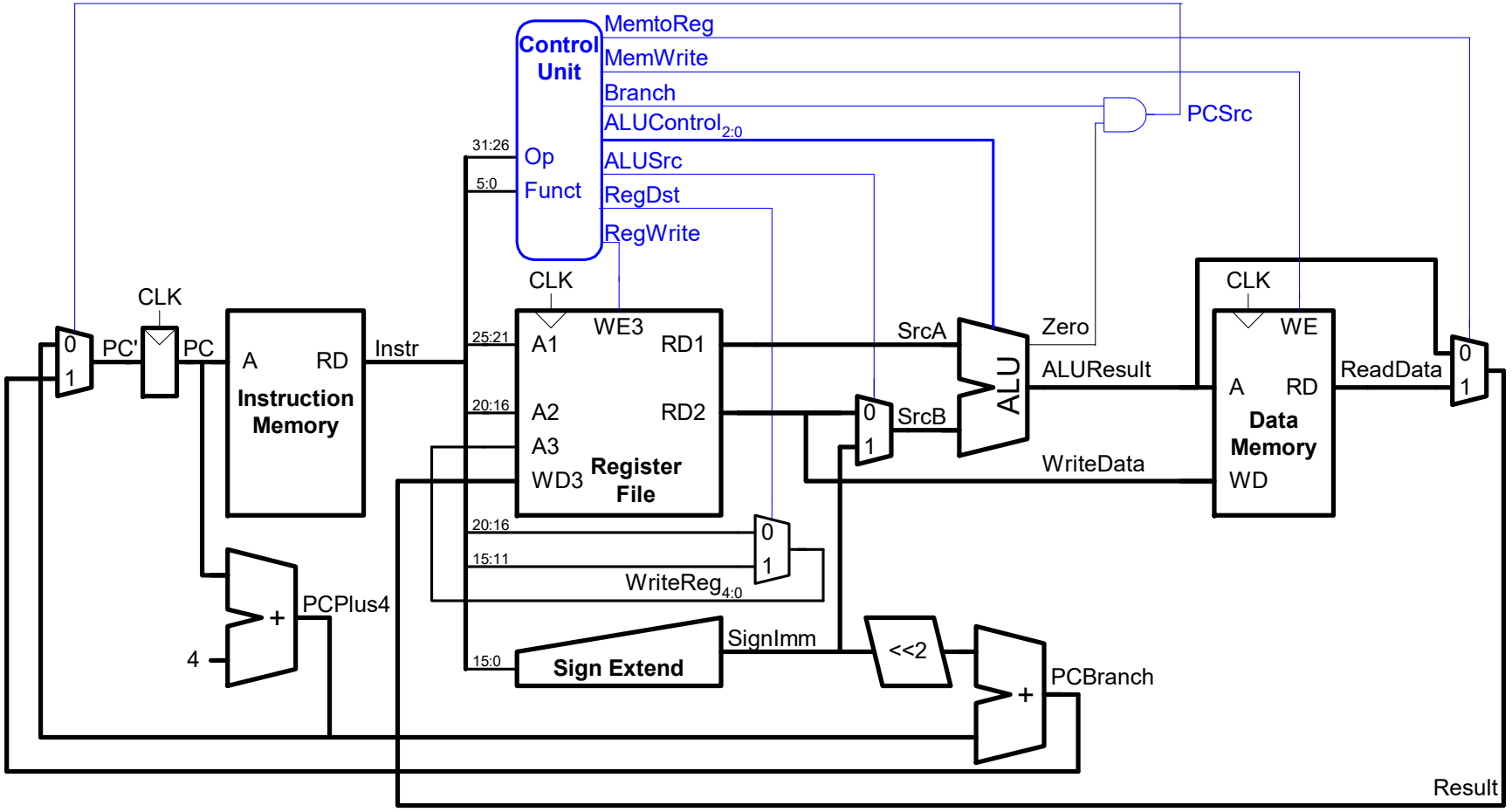
Branch se pondrá a 1 si se está ejecutando un beq

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)

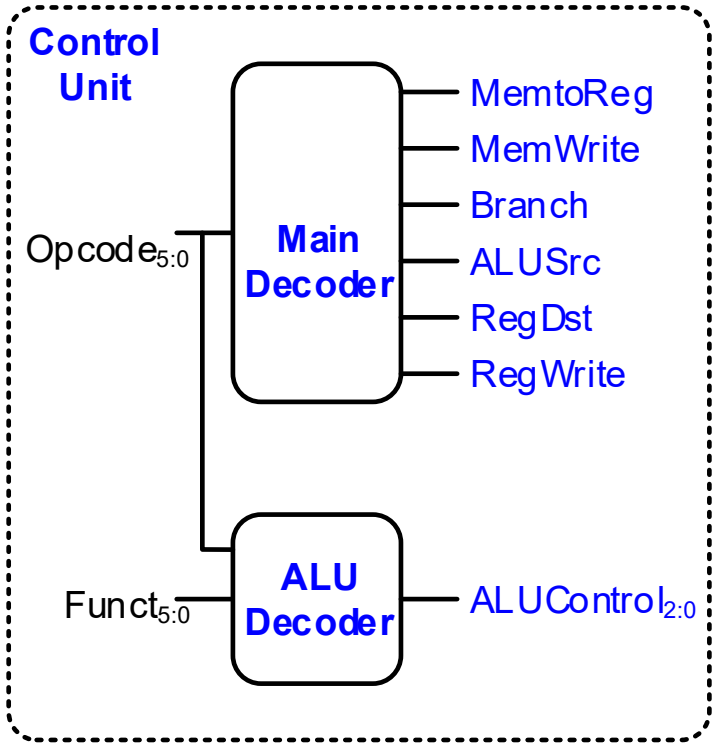


Índice

- Introducción
- Ruta de datos uniciclo
- **Control uniciclo**
- Añadir más instrucciones

Unidad de Control

- La Unidad de Control UC, genera dos buses de control:
  - ALUControl (3 bits): depende de **opcode** y **funct**
  - Resto (6 bits): sólo depende de **opcode**, no depende de **funct**

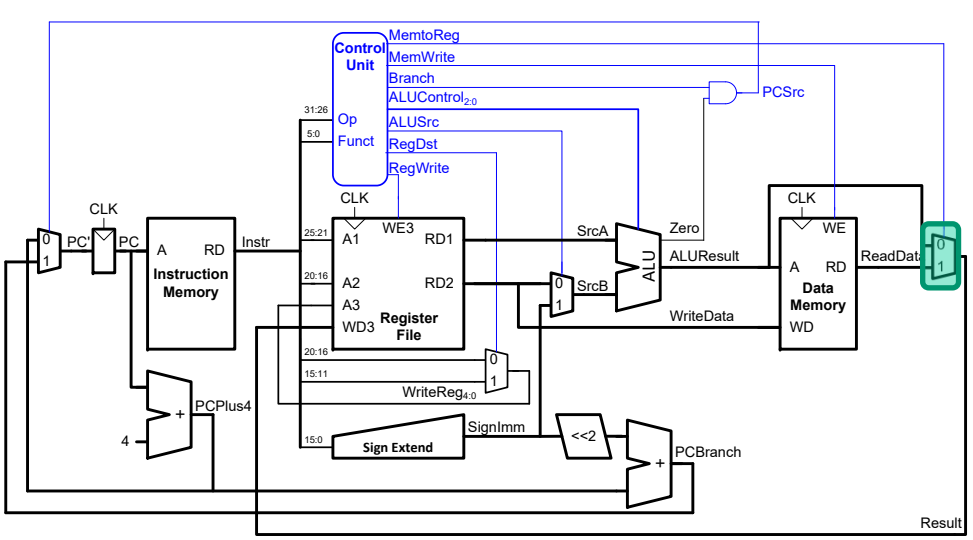


Decodificador de la ALU

OPCODE	Funct	$ALUControl_{2:0}$
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 (**S**et**L**ess**T**han)

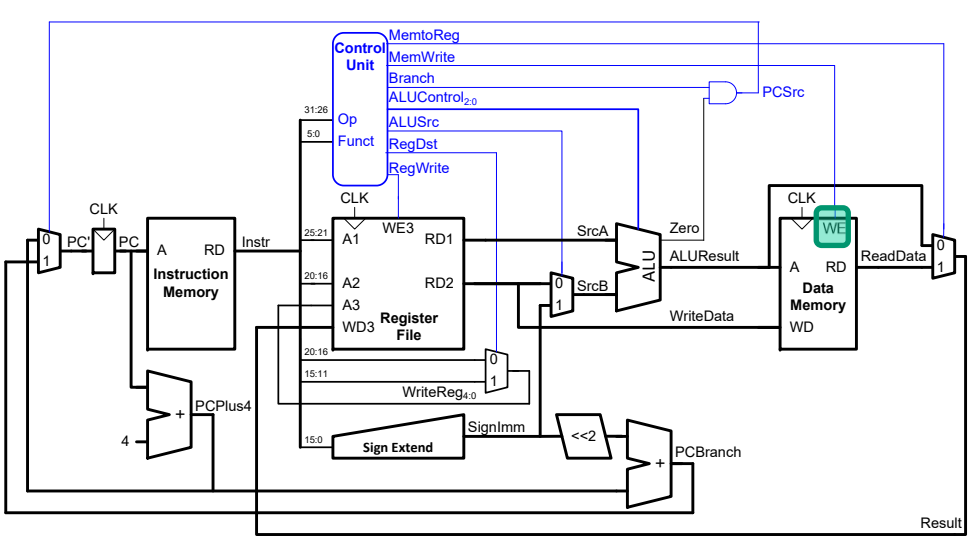
Ruta de control

Señal	Significado	Significado de valores	
		0	1
MemToReg	Decide si al banco de registros les 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	Registro RT [20:16]	Dato Inm
RegDst	Indica la dirección de registro destino		
RegWrite	Indica si el banco de registros debe guardar un		



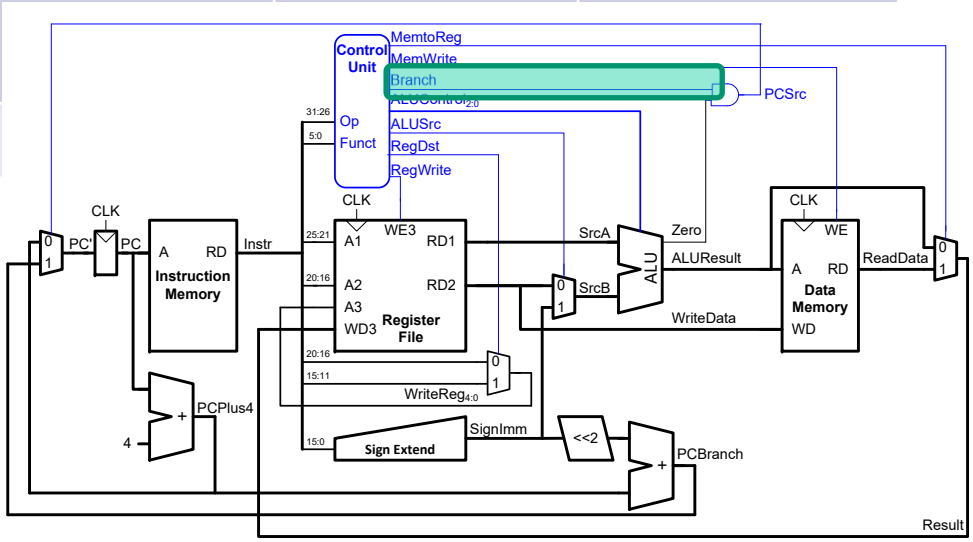
Ruta de control

Señal	Significado	Significado de valores	
		0	1
MemToReg	Decide si al banco de registros les 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	Registro RT [20:16]	Dato Inm
RegDst	Indica la dirección de registro destino		
RegWrite	Indica si el banco de registros debe guardar un		



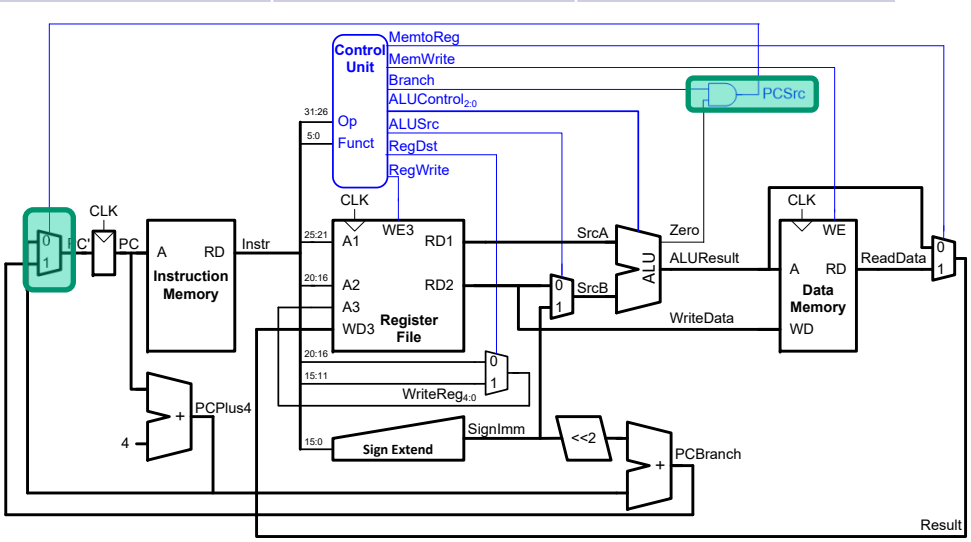
Ruta de control

Señal	Significado	Significado de valores	
		0	1
MemToReg	Decide si al banco de registros les 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	Registro RT [20:16]	Dato Inm
RegDst	Indica la dirección de registro destino		
RegWrite	Indica si el banco de registros debe guardar un		

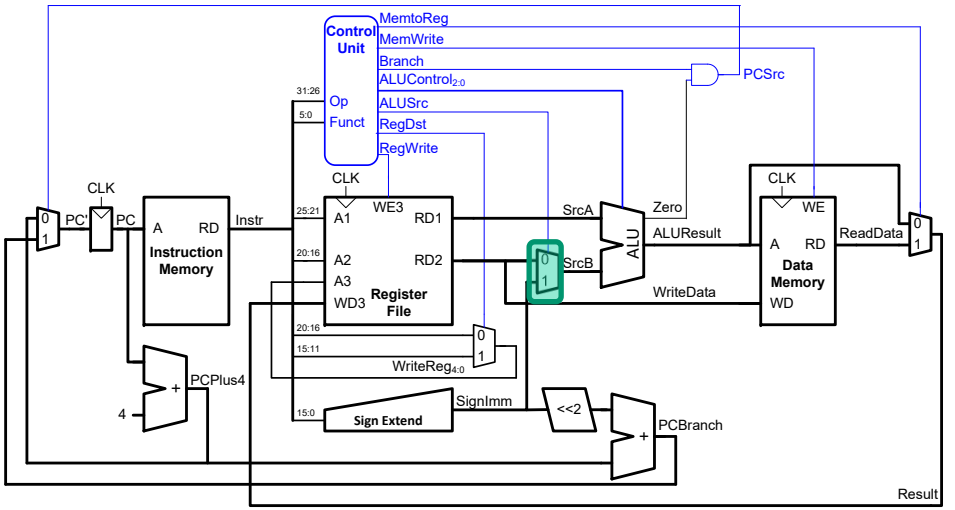


Ruta de control

Señal	Significado	Significado de valores	
		0	1
MemToReg	Decide si al banco de registros les 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	Registro RT [20:16]	Dato Inm
RegDst	Indica la dirección de registro destino		
RegWrite	Indica si el banco de registros debe guardar un		

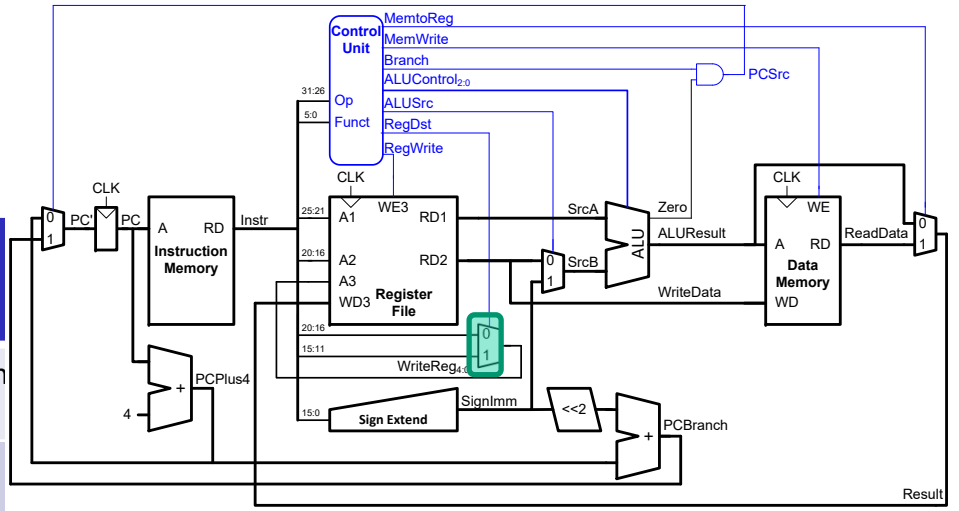


Ruta de control



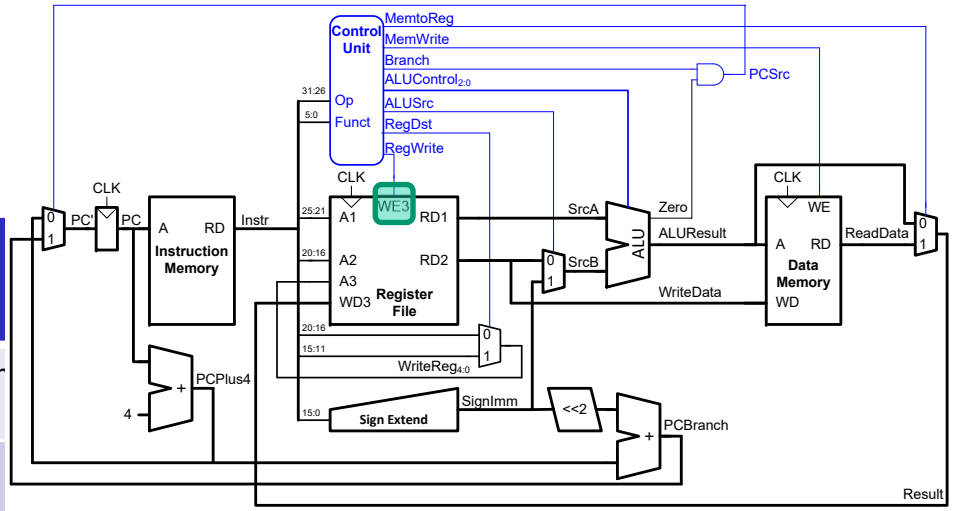
Señal	Significado		
MemToReg	Decide si al banco de registros les llega un datos o si le llega el resultado de la ALU		
MemWrite	Decide si se va a escribir un registro en la		
Branch	Indica si se está ejecutando una instrucción beq	No	Sí
PcSrc	Indica si hay que realizar el salto del beq. Sólo activa cuando Branch=1 y Z=1	No	Sí
ALUSrc	Decide cuál será el segundo operando de la ALU	Registro RT [20:16]	Dato Inm
RegDst	Indica la dirección de registro destino	Registro RT [20:16]	Registro RD [15:11]
RegWrite	Indica si el banco de registros debe guardar un resultado	No	Sí

Ruta de control



Señal	Significado		
MemToReg	Decide si al banco de registros les llega un datos o si le llega el resultado de la ALU		
MemWrite	Decide si se va a escribir un registro en la		
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	Registro RT [20:16]	Dato Inm
RegDst	Indica la dirección de registro destino	Registro RT [20:16]	Registro RD [15:11]
RegWrite	Indica si el banco de registros debe guardar un resultado	No	Sí

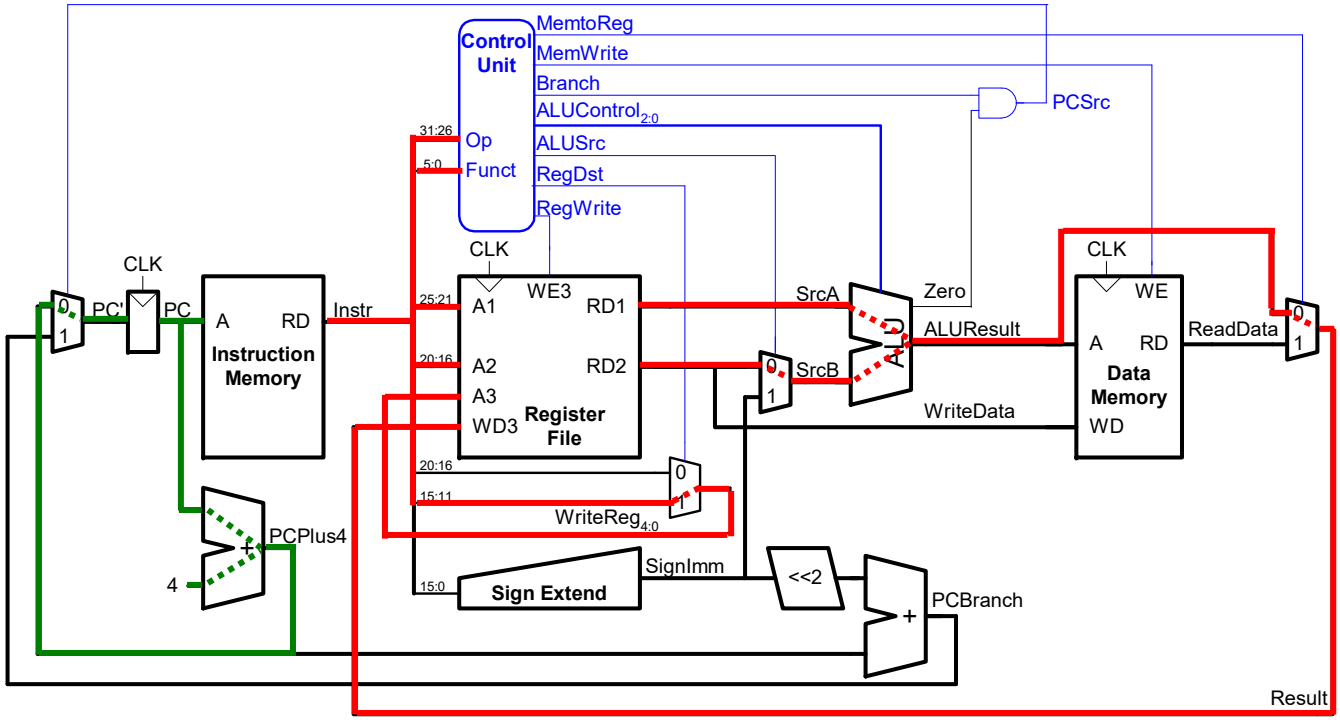
Ruta de control



Señal	Significado		
MemToReg	Decide si al banco de registros les llega ur datos o si le llega el resultado de la ALU		
MemWrite	Decide si se va a escribir un registro en la		
Branch	Indica si se está ejecutando una instrucción beq	No	Sí
PcSrc	Indica si hay que realizar el salto del beq. Sólo se activa cuando Branch = 1 y Z = 1	No	Sí
ALUSrc	Decide cuál será el segundo operando de la ALU	Registro RT [20:16]	Dato Inm
RegDst	Indica la dirección de registro destino	Registro RT [20:16]	Registro RD [15:11]
RegWrite	Indica si el banco de registros debe guardar un resultado	No	Sí

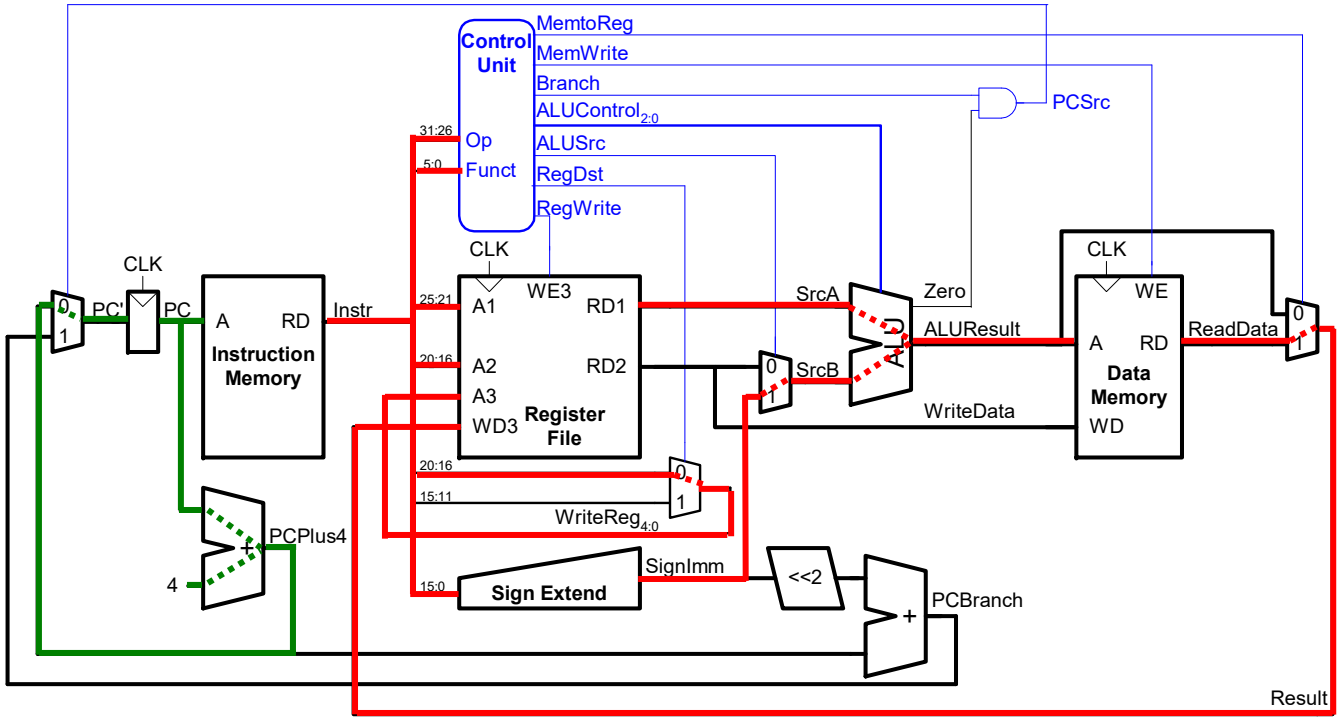
Decodificador principal (R-type)

Instrucción	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
R-type	000000	**1**	**1**	**0**	**0**	**0**	**0**	
lw	100011							
sw	101011							
beq	000100							



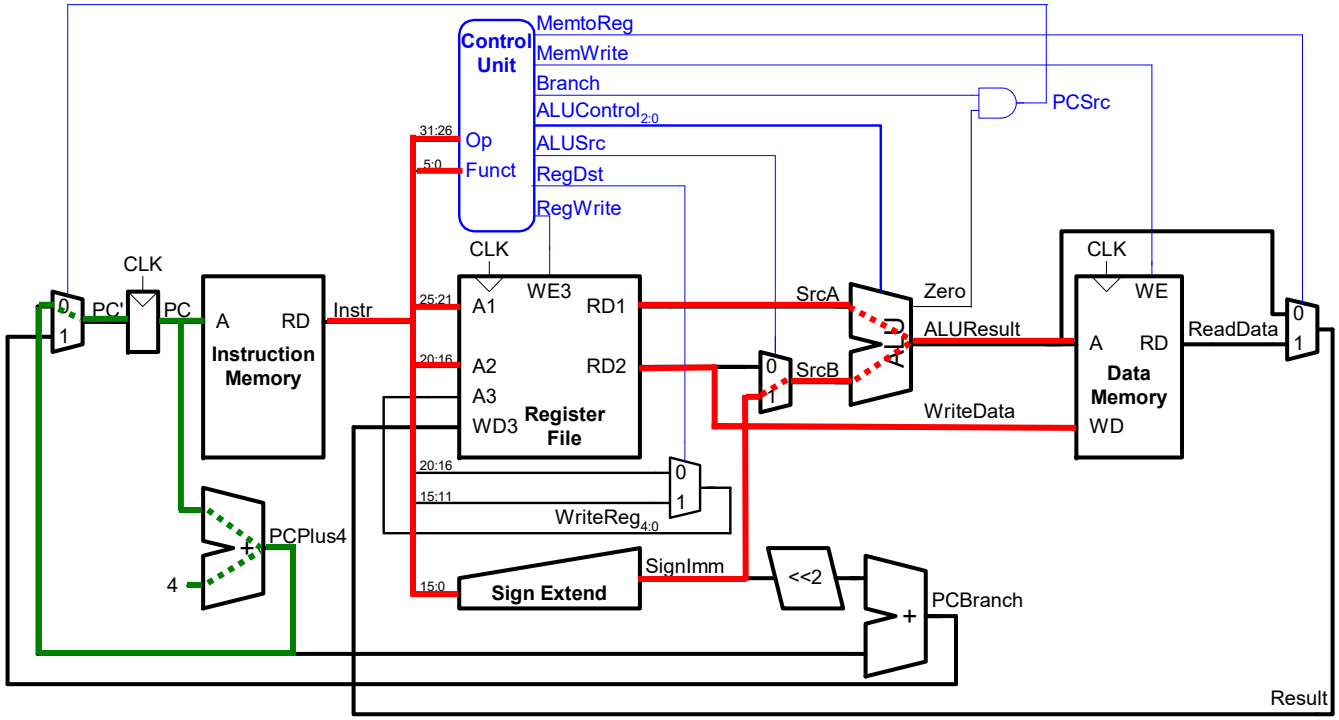
Decodificador principal (lw)

Instrucción	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
R-type	000000	**1**	**1**	**0**	**0**	**0**	**0**	
lw	100011	**1**	**0**	**1**	**0**	**0**	**1**	**010**
sw	101011							
beq	000100							



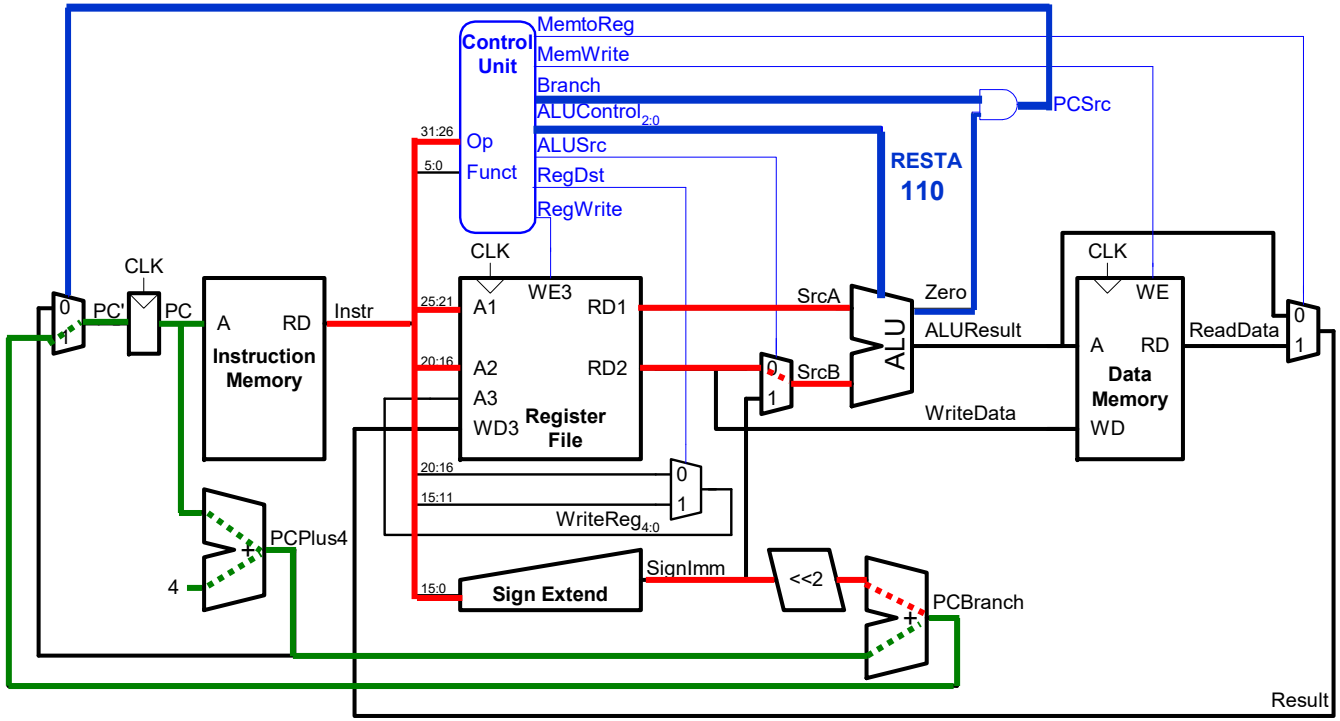
Decodificador principal (sw)

Instrucción	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
R-type	000000	1	1	0	0	0	0	
lw	100011	1	0	1	0	0	1	010
sw	101011	0	X	1	0	1	X	010
beq	000100							



Decodificador principal (beq)

Instrucción	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
R-type	000000	1	1	0	0	0	0	
lw	100011	1	0	1	0	0	1	010
sw	101011	0	X	1	0	1	X	010
beq	000100	0	X	0	1	0	X	110



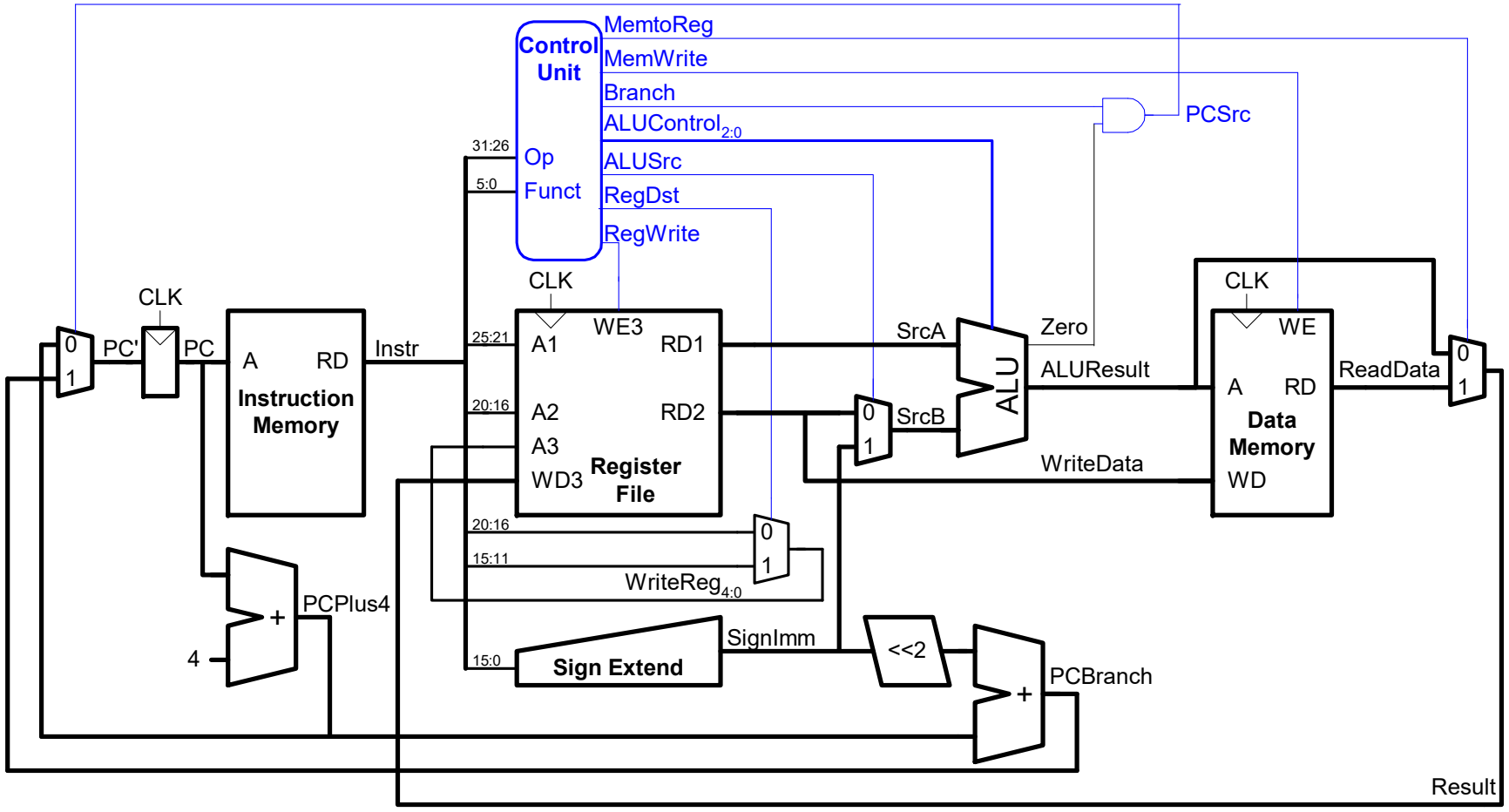
P4.7

Índice

- Introducción
- Ruta de datos uniciclo
- Control uniciclo
- **Añadir más instrucciones**

Añadimos instrucciones: addi

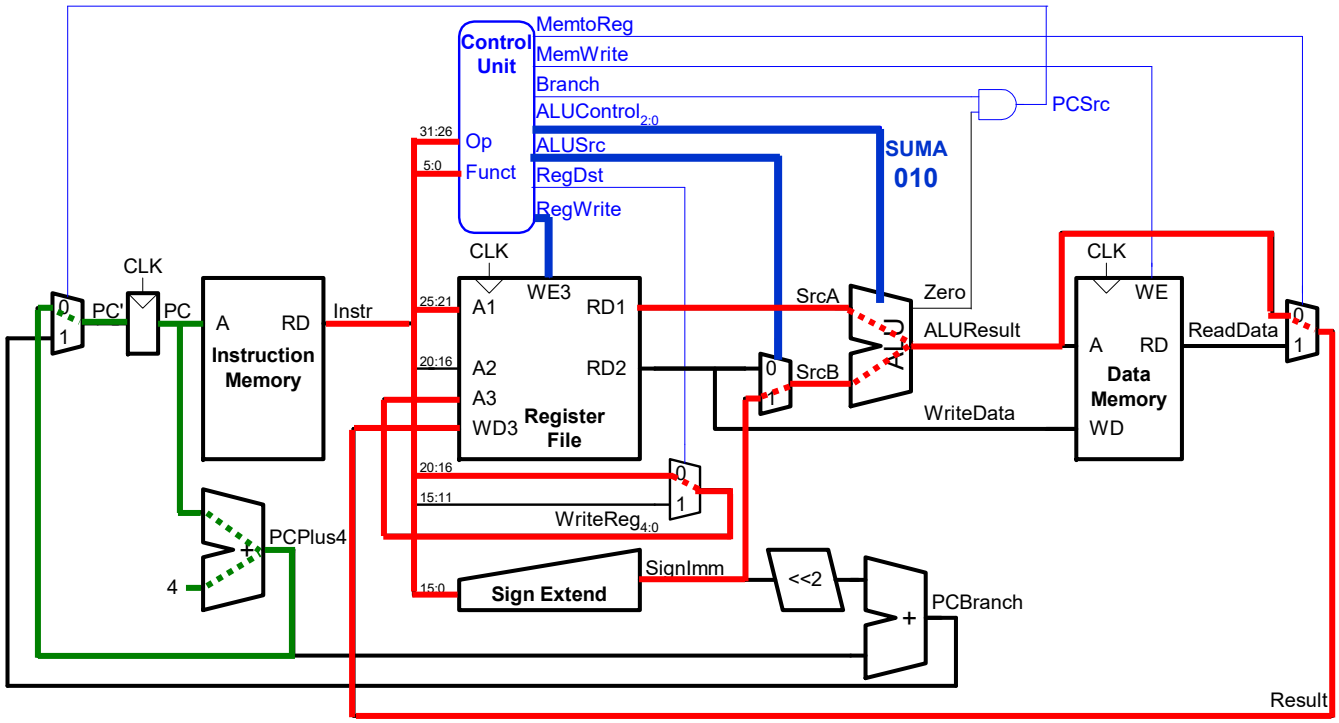
- No hacen falta cambios en la ruta de datos



Cambios en el control: addi

- No hacen falta cambios en la ruta de datos

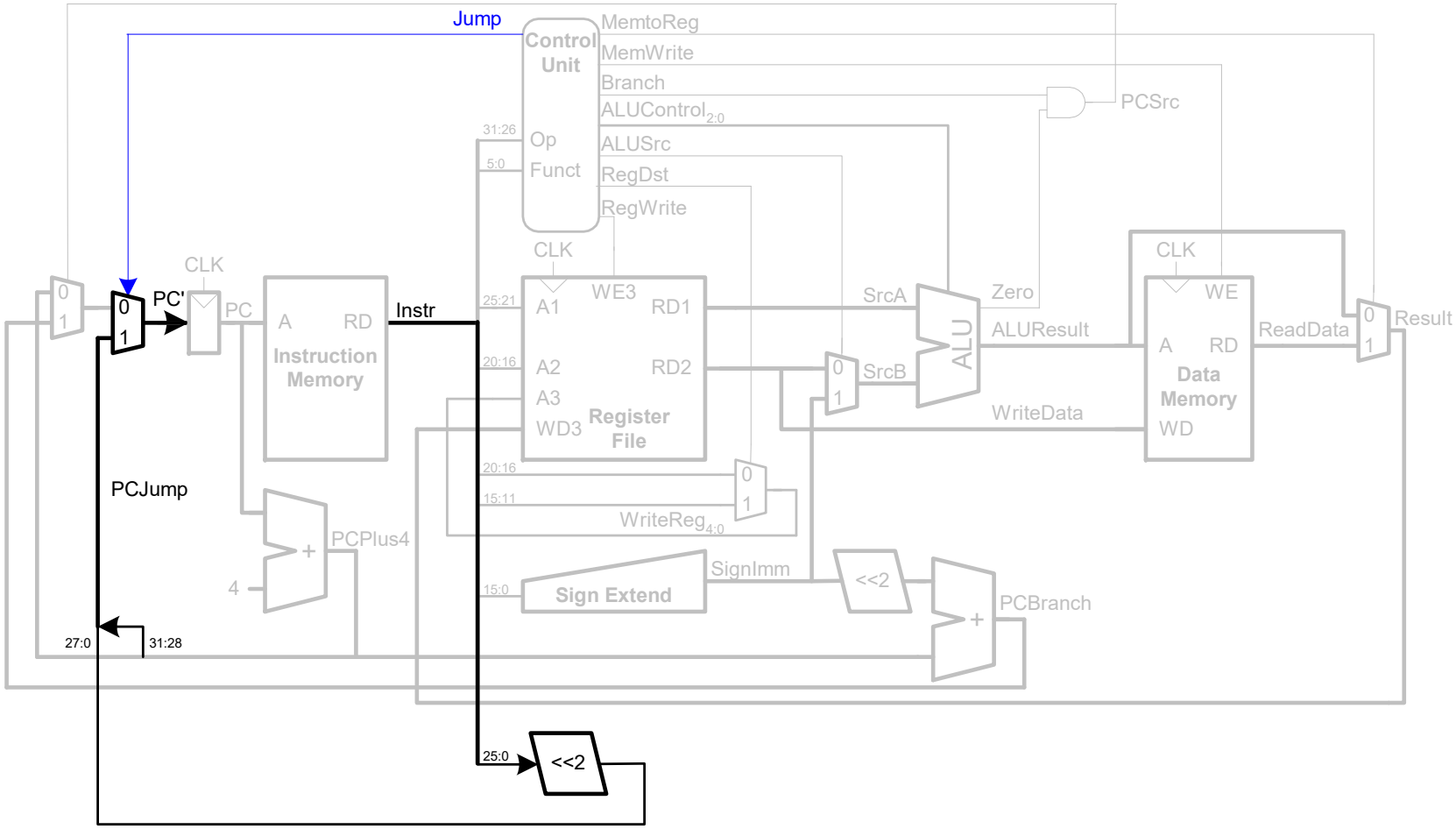
Instrucción	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl
addi	001000	**1**	**0**	**1**	**0**	**0**	**0**	**010**



Añadimos instrucciones: j

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

  JTA = (PC+4)[31:28] & addr & “00”



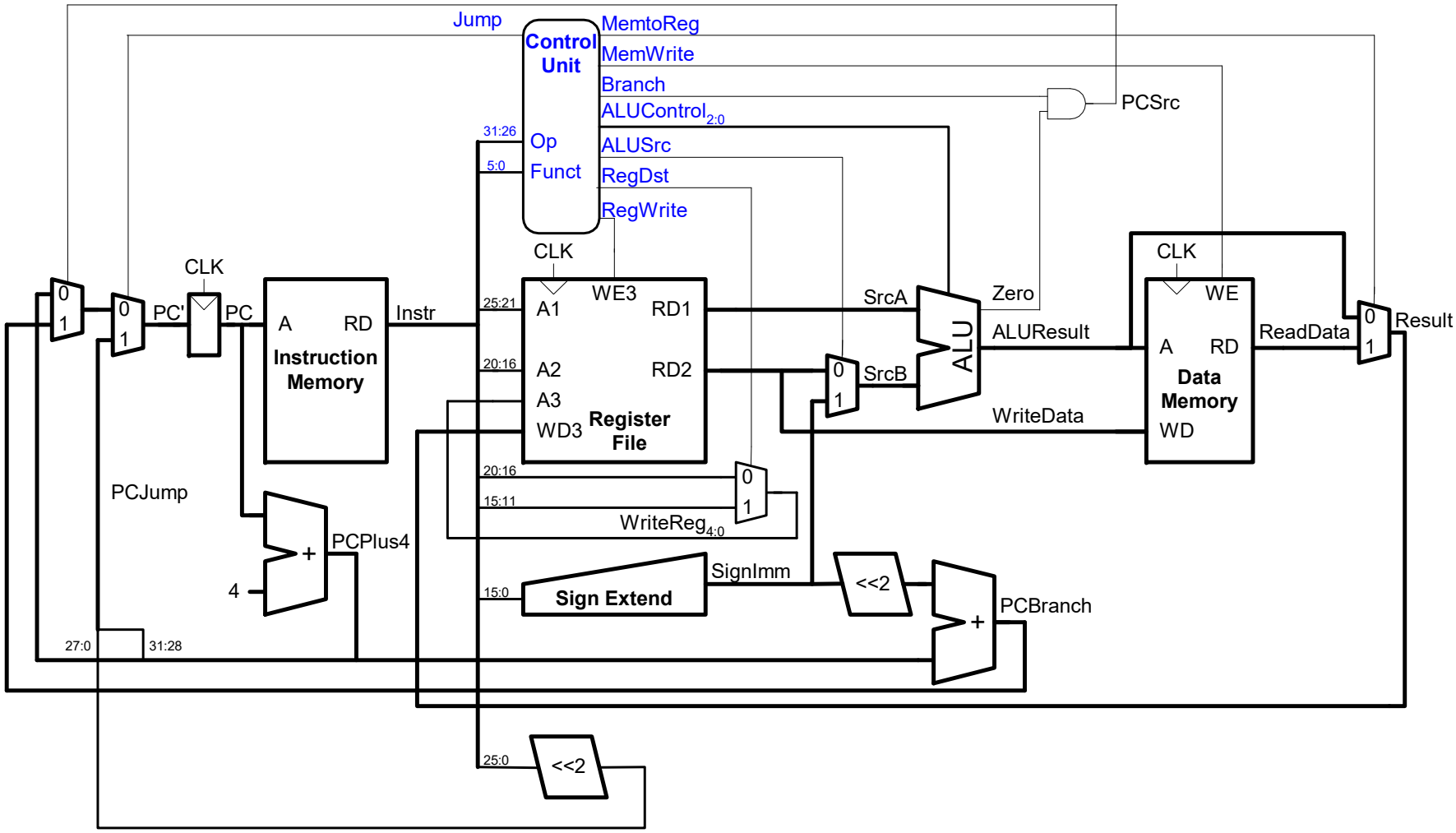
Cambios en el control: j

Instrucción	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUCtrl	Jump
R-type	000000	1	1	0	0	0	0		0
lw	100011	1	0	1	0	0	1	010	0
sw	101011	0	X	1	0	1	X	010	0
beq	000100	0	X	0	1	0	X	110	0
addi	001000	1	0	1	0	0	0	010	0
j	000010	0	X	X	X	0	X	XX	1

Ruta de datos y de control

Señal	Significado	Significado de valores		Uso
		0	1	
MemToReg	Decide si al banco de registros les 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
MemWrite	Decide si se va a escribir un registro en la memoria de datos.	No	Sí	1: sw 0:Resto de operaciones
Branch	Indica si se está ejecutando una instrucción beq	No	Sí	1: beq X: jump 0: Resto de operaciones
PcSrc	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
ALUControl	Indica la operación a ejecutar en la ALU	-	-	-
ALUSrc	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
RegDst	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
RegWrite	Indica si el banco de registros debe guardar un resultado	No	Sí	0: sw, beq, jump 1: Resto de operaciones
Jump	Indica si hay una instrucción de tipo jump	No	Sí	1: jump 0: Resto de operaciones

Resumen: MIPS uniciclo



Fundamentos de Microprocesadores

Unidad 4. El Procesador II: Diseño y control de la ruta de datos. Arquitectura uniciclo

Escuela Politécnica Superior - UAM

P4.5, P4.15, P4.17