

Práctica 2.

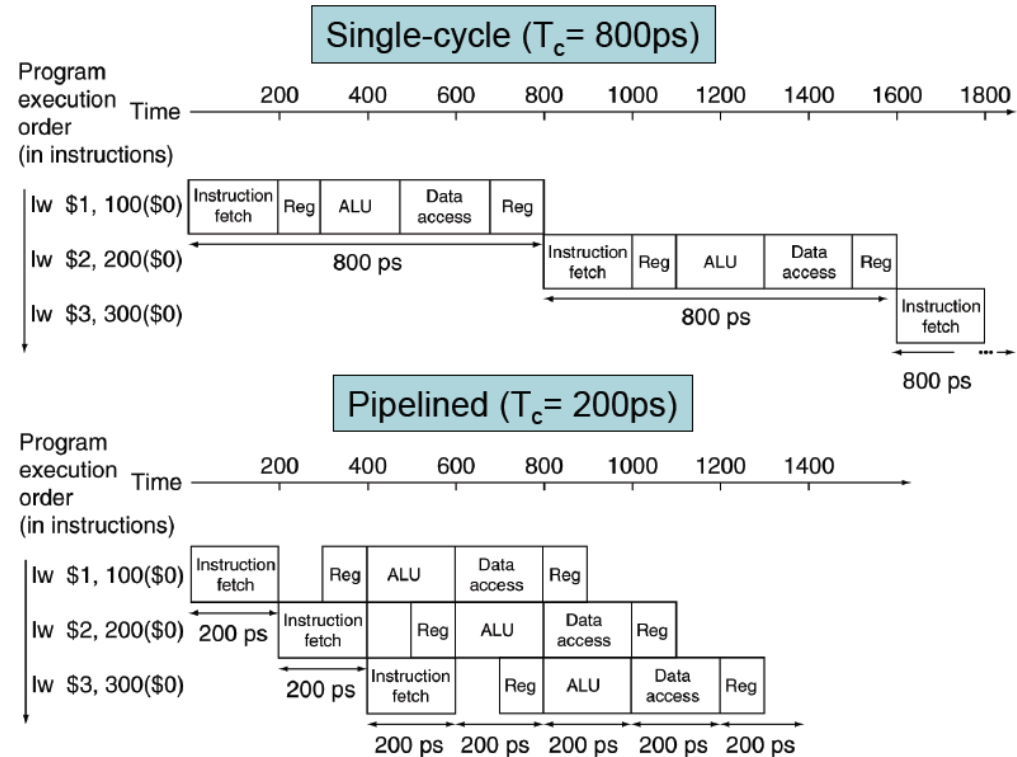
Mejoras al Microprocesador segmentado.

Arquitectura de Ordenadores

VÍCTOR MORENO, ALBERTO SÁNCHEZ
ESCUELA POLITÉCNICA SUPERIOR. UAM.

Segmentación en MIPS

IF	Captura instrucción.
ID	Decodificación instrucción. Lectura GPR. Cálculos saltos beq y jump.
EX	Ejecución. Cálculo dirección sw y lw.
MEM	Acceso memoria de datos.
WB	Escritura GPR.



*D.A. Patterson y J.L. Hennessy "Computer Organization and Design: The Hardware/Software Interface" 4th edition.

Tipos de riesgos por dependencia de datos

- ❑ Dependencias que se presentan para 2 instrucciones i y j, con i ejecutándose antes que j.
 - ❑ **RAW** (Read After Write): la instrucción posterior j intenta leer una fuente antes de que la instrucción anterior i la haya modificado.
 - ❑ **WAR** (Write After Read): la instrucción j intenta modificar un destino antes de que la instrucción i lo haya leído como fuente.
 - ❑ **WAW** (Write After Write): la instrucción j intenta modificar un destino antes de que la instrucción i lo haya hecho (se modifica el orden normal de escritura).

✓ Ejemplos:

RAW

```
ADD r1, r2, r3
SUB r5, r1, r6
AND r6, r5, r1
ADD r4, r1, r3
SW  r10, 100(r1)
```

WAR

```
ADD r1, r2, r3
OR  r3, r4, r5
```

WAW

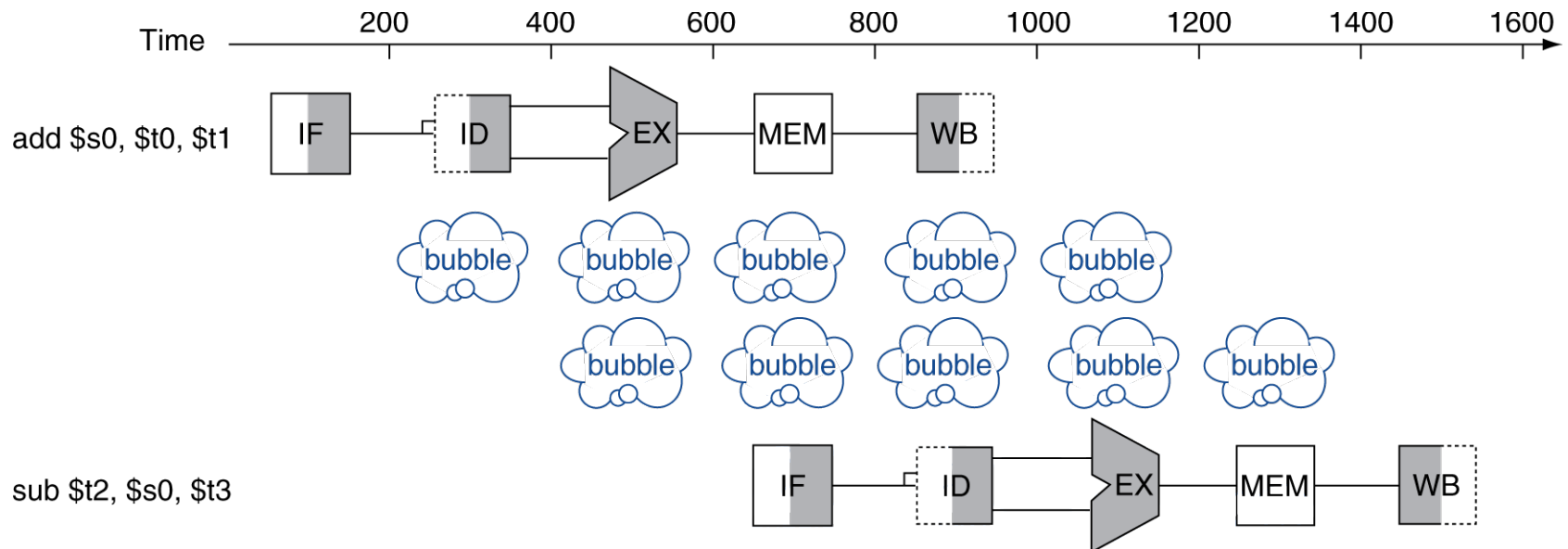
```
DIV r1, r2, r3
AND r1, r4, r5
```

En micros segmentados con ejecución en orden SÓLO son problema los RAW

E1: parar el *pipeline*


An instruction depends on completion of data access by a previous instruction

- add **\$s0**, \$t0, \$t1
sub \$t2, **\$s0**, \$t3



Banco de registros

- **32 registros**
- **Lectura asíncrona**
- **Escritura síncrona**
 - Flanco de bajada
 - El resto de registros del procesador en flanco de subida
- **Registro 0**
 - Siempre vale 0
 - Escrituras sin efecto



¡¡Si no un ciclo
más de parada!!

Segmentación en MIPS. Riesgos

0: add \$2, \$1, \$0
4: add \$5, \$2, \$1

0: add \$3, \$1, \$2
4: xor \$6, \$4, \$5
8: sub \$8, \$7, \$3

0: add \$2, \$1, \$0
4: or \$5, \$6, \$9
8: xor \$9, \$3, \$6
C: add \$5, \$2, \$1

0: add \$2, \$1, \$0
4: or \$2, \$6, \$9
8: add \$5, \$2, \$1

0: add \$2, \$1, \$0
4: sw \$2, etiq

0: add \$2, \$1, \$0
4: or \$5, \$6, \$9
8: sw \$2, etiq

0: add \$2, \$1, \$0
4: or \$5, \$6, \$9
8: and \$3, \$4, \$6
C: sw \$2, etiq

0: lw \$2, etiq1
4: sw \$2, etiq2

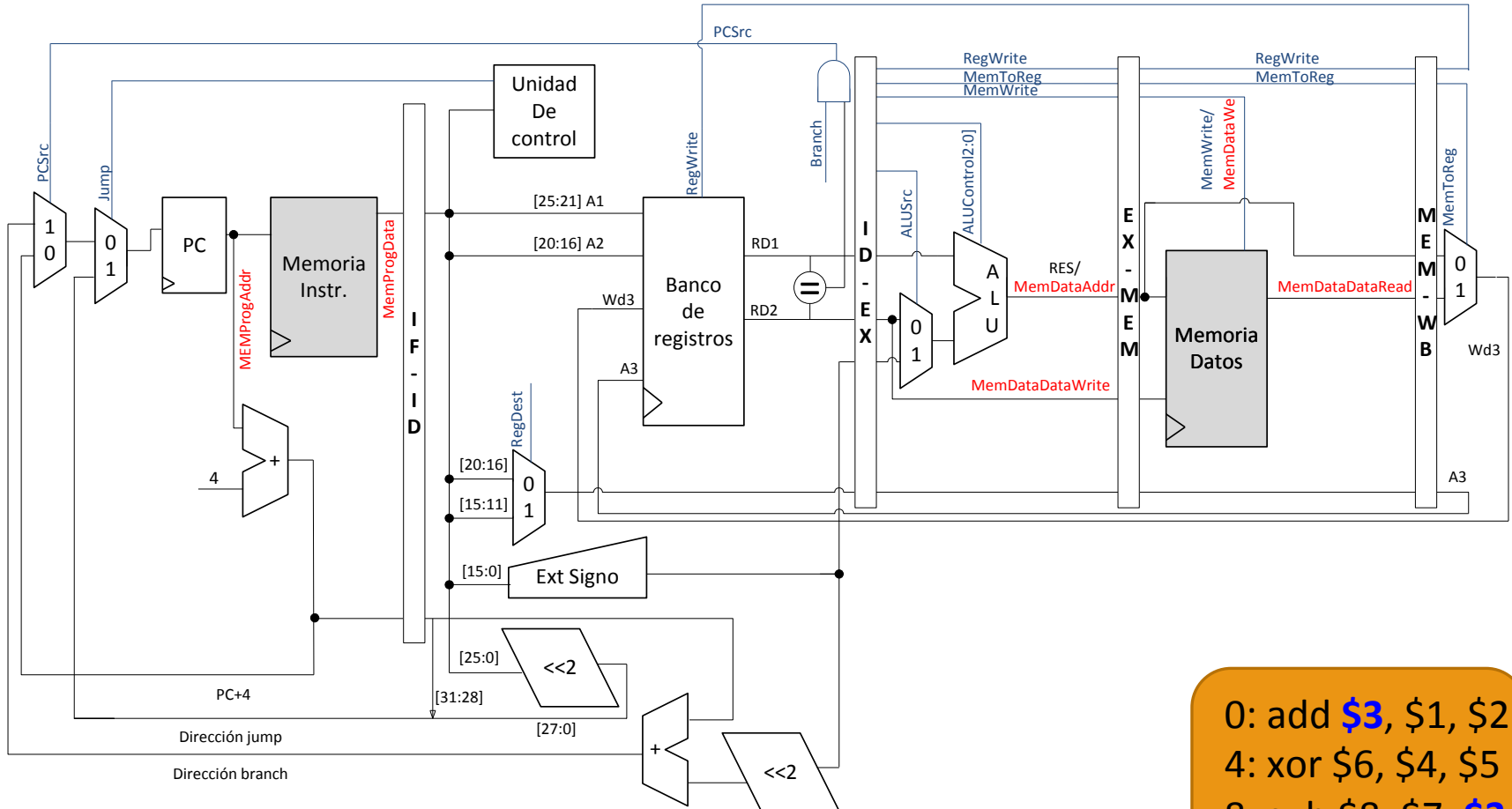
0: lw \$2, etiq
4: add \$5, \$1, \$2,

0: lw \$2, etiq
4: or \$5, \$6, \$9
8: add \$5, \$1, \$2

0: lw \$2, etiq
4: or \$5, \$6, \$9
8: and \$3, \$4, \$6
C: add \$5, \$1, \$2

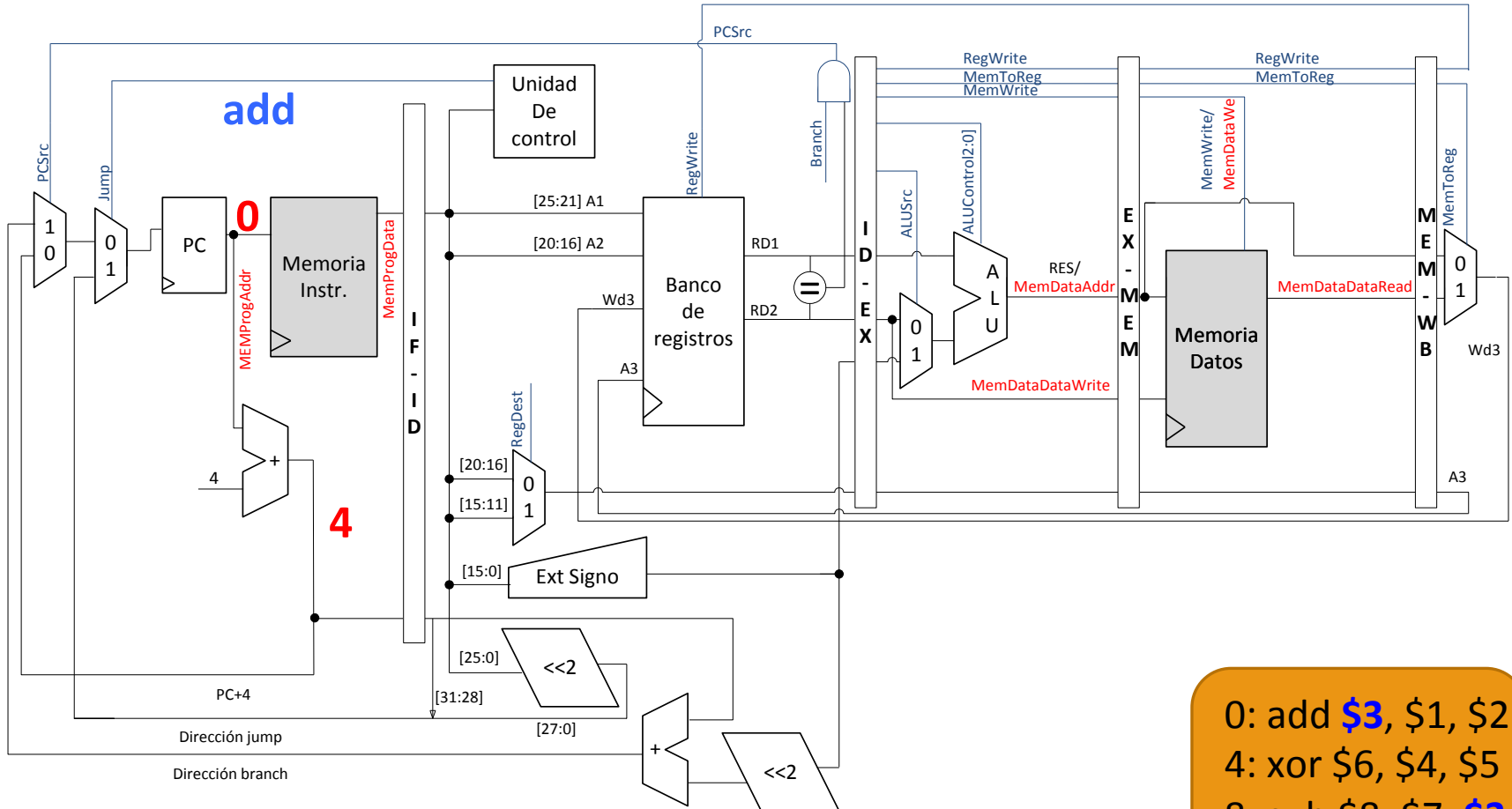
0: sw \$2, etiq1
4: lw \$2, etiq2

Segmentación en MIPS



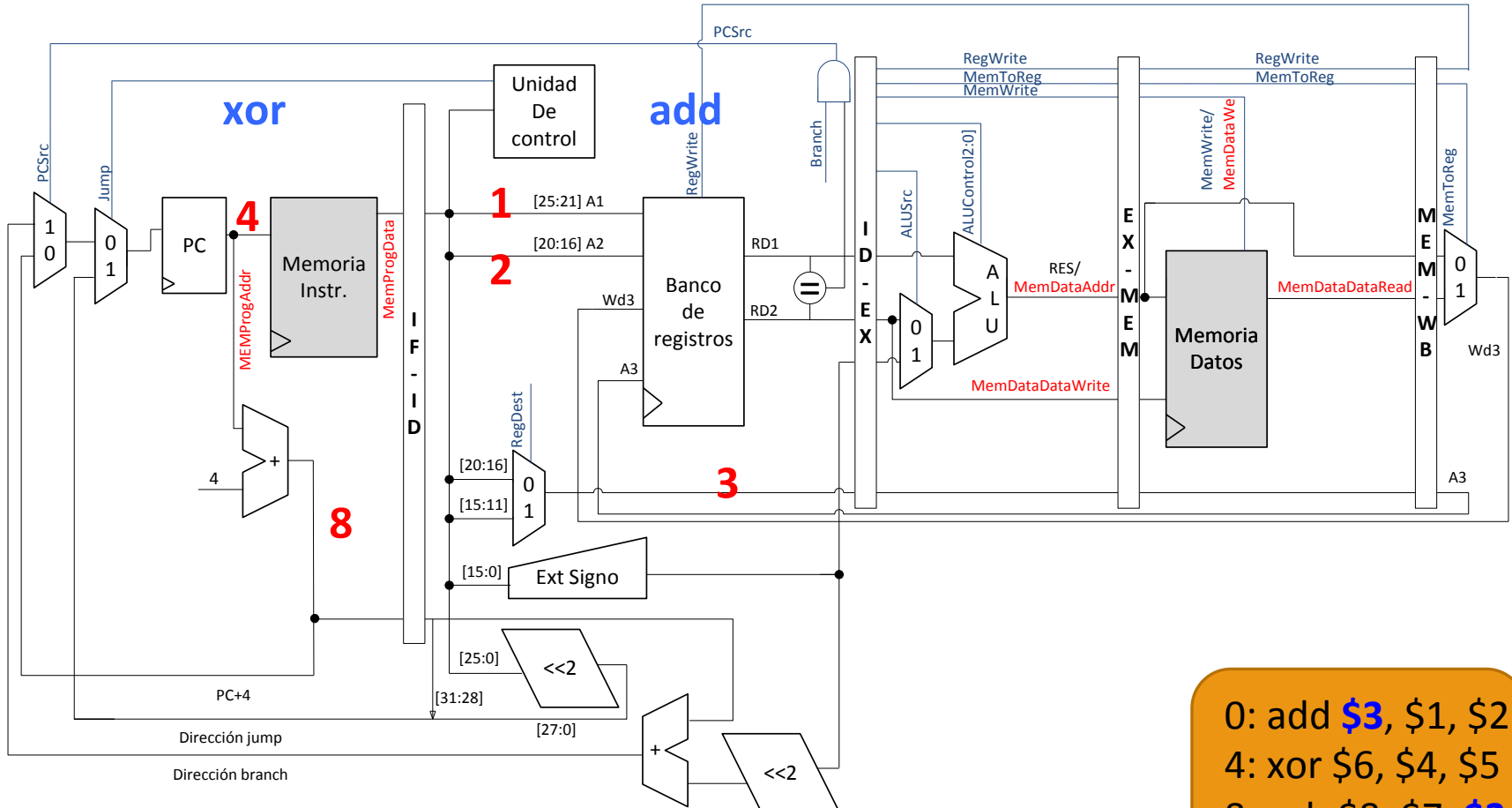
0: add \$3, \$1, \$2
 4: xor \$6, \$4, \$5
 8: sub \$8, \$7, \$3
 C: or \$11,\$9,\$10

Segmentación en MIPS. Ciclo 1



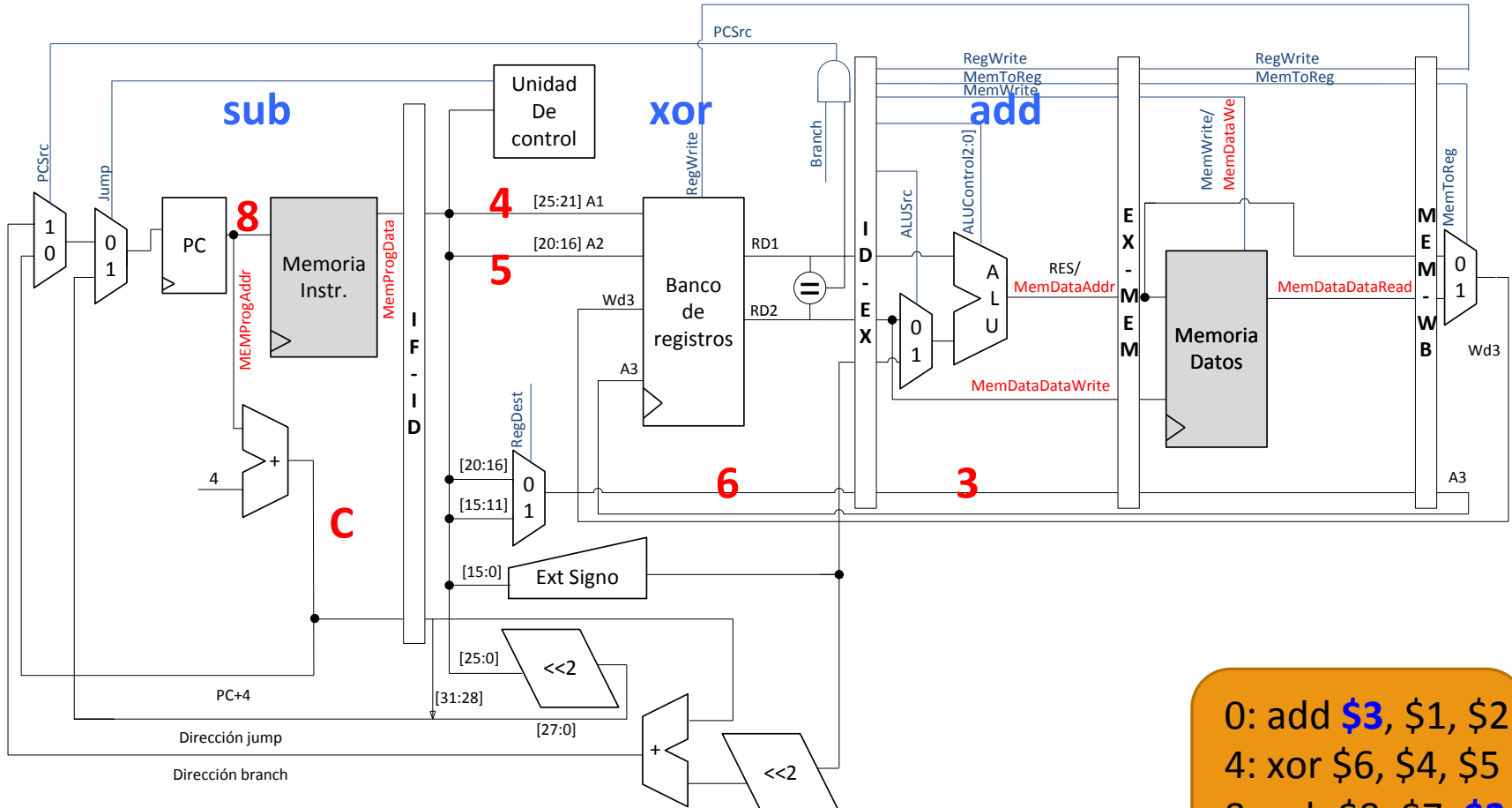
```
0: add $3, $1, $2
4: xor $6, $4, $5
8: sub $8, $7, $3
C: or $11, $9, $10
```


Segmentación en MIPS. Ciclo 2



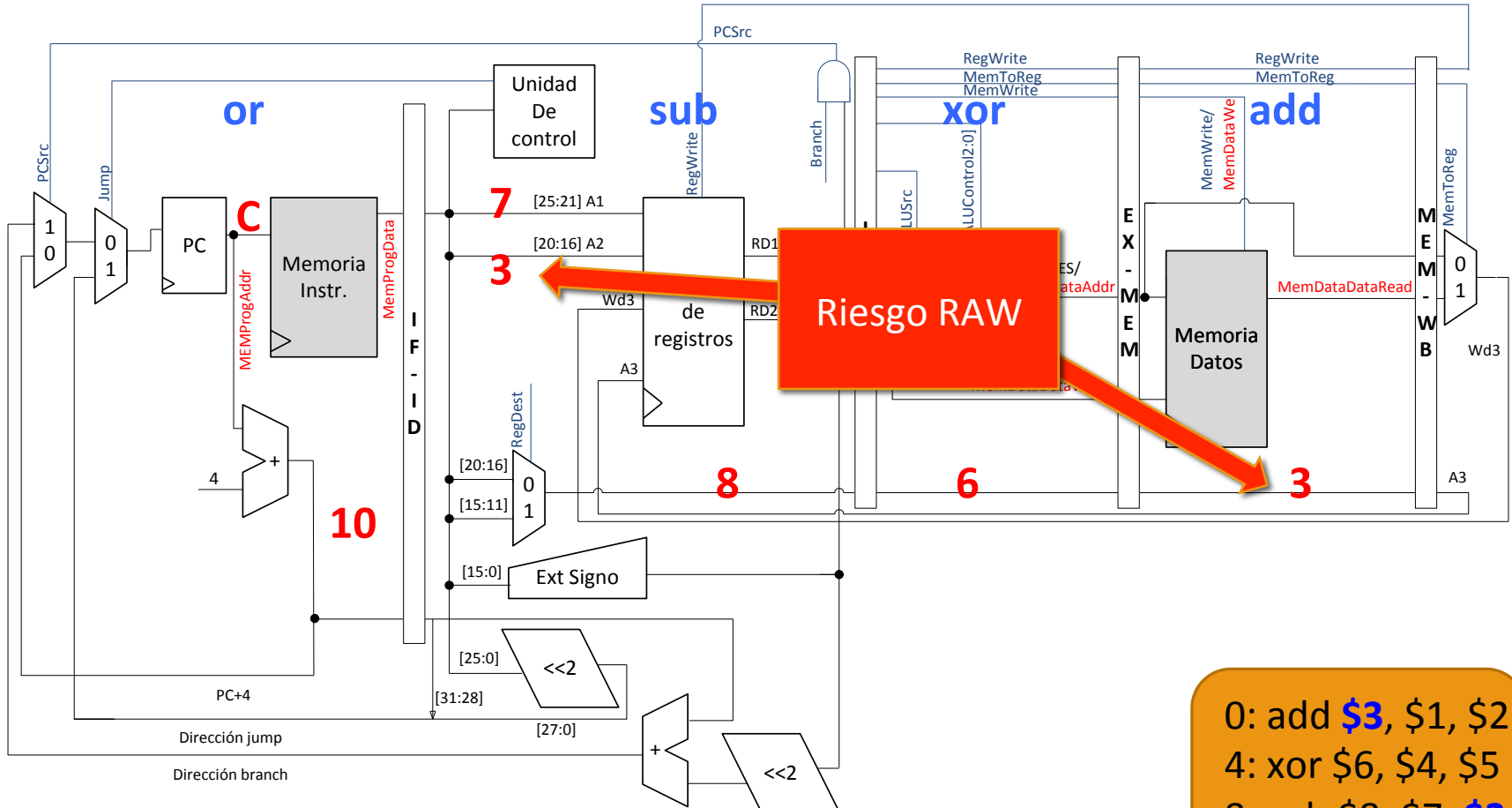
```
0: add $3, $1, $2
4: xor $6, $4, $5
8: sub $8, $7, $3
C: or $11, $9, $10
```

Segmentación en MIPS. Ciclo 3



```
0: add $3, $1, $2
4: xor $6, $4, $5
8: sub $8, $7, $3
C: or $11, $9, $10
```

Segmentación en MIPS. Ciclo 4



0: add **\$3**, \$1, \$2
 4: xor \$6, \$4, \$5
 8: sub \$8, \$7, **\$3**
 C: or \$11,\$9,\$10

Segmentación en MIPS. Resolución de riesgos

Alternativa 1 (E1):

Detección en etapa ID.

Parar la segmentación (Añadir burbuja) hasta que desaparezca el riesgo.

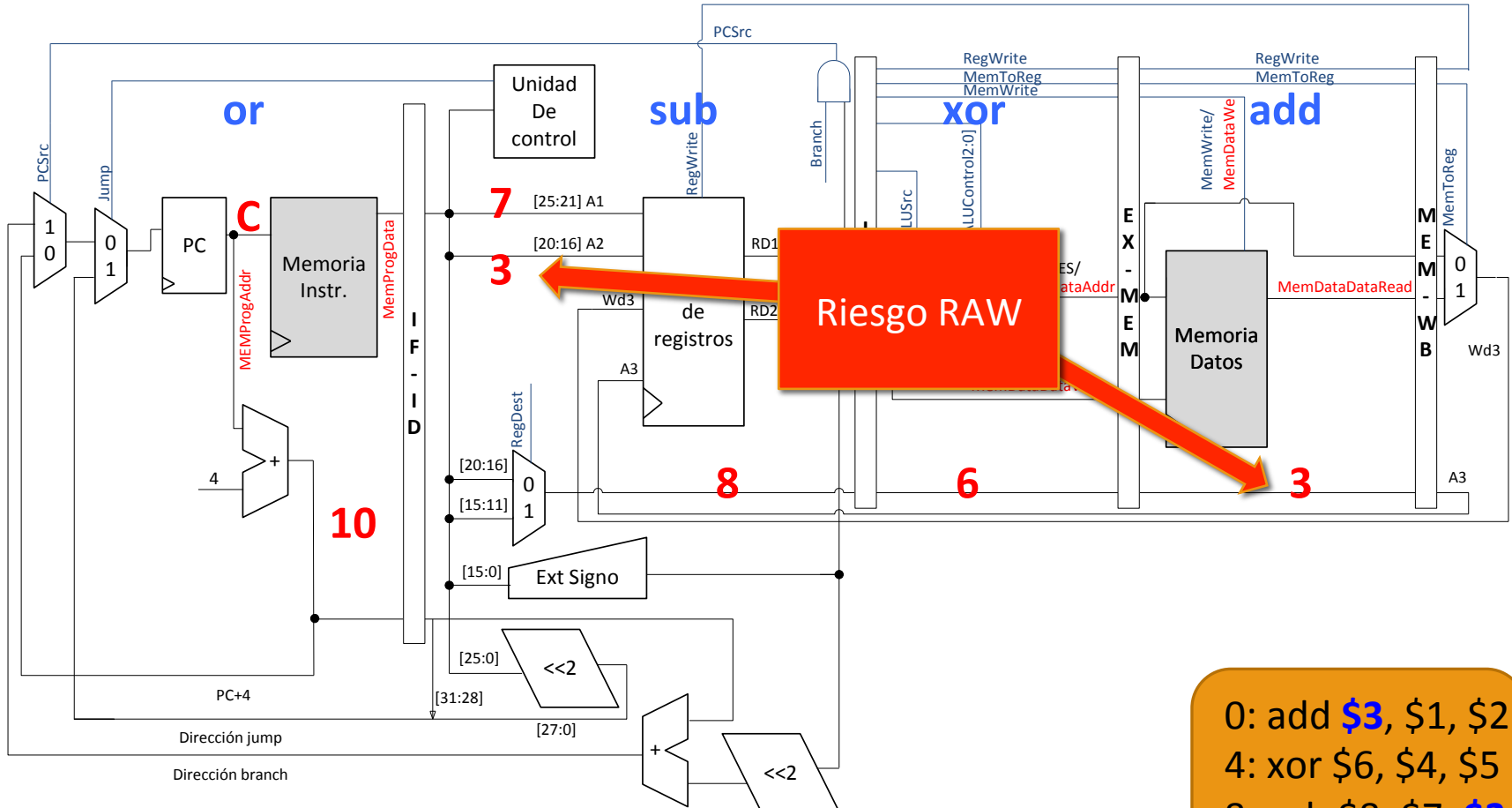
Alternativa 2 (E2):

Detección en etapa EX.

Adelantar datos si es posible.

Si no es posible, parar la segmentación (Añadir burbuja).

Segmentación en MIPS. Ciclo 4

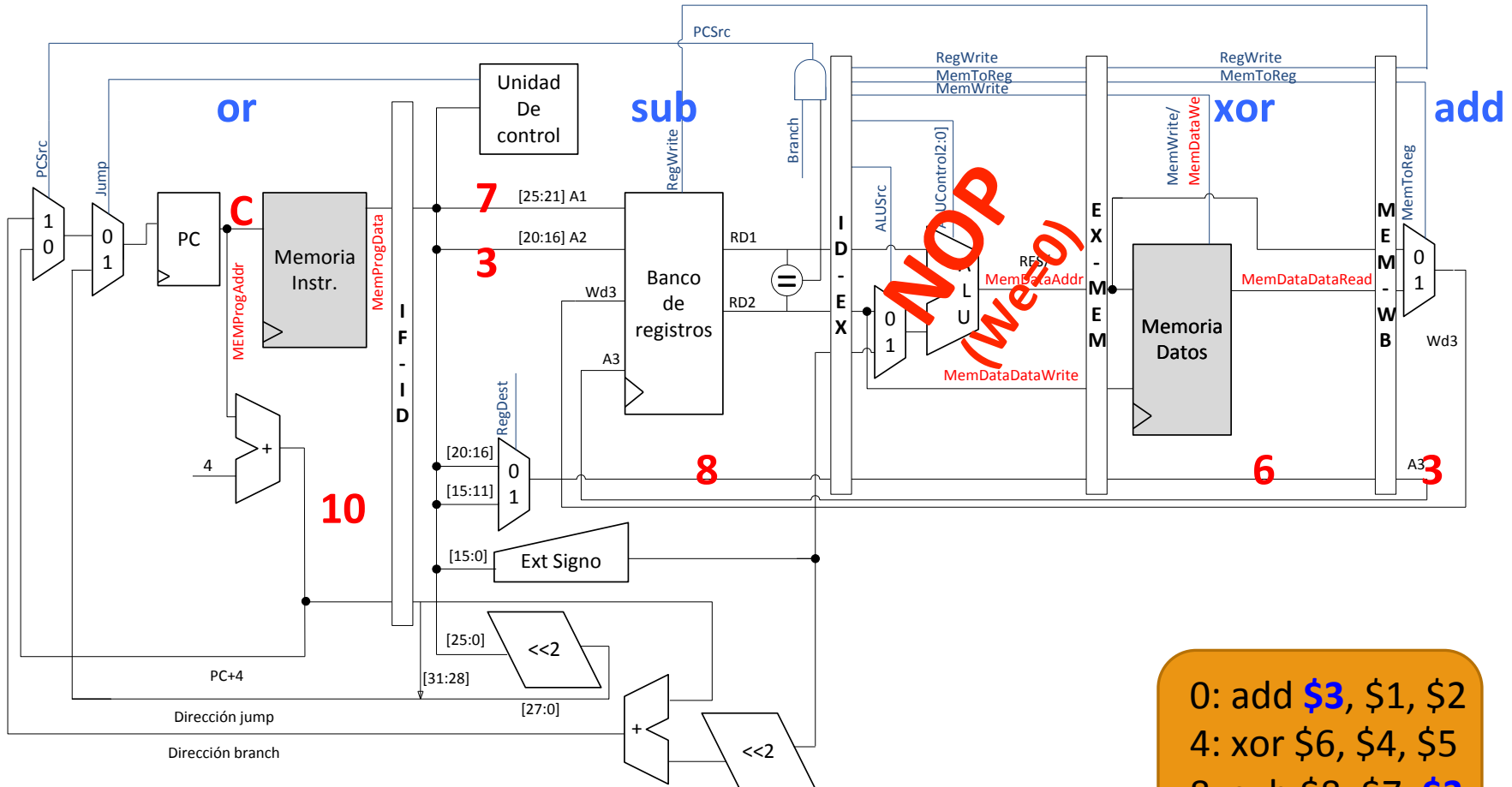


0: add \$3, \$1, \$2
 4: xor \$6, \$4, \$5
 8: sub \$8, \$7, \$3
 C: or \$11, \$9, \$10

14

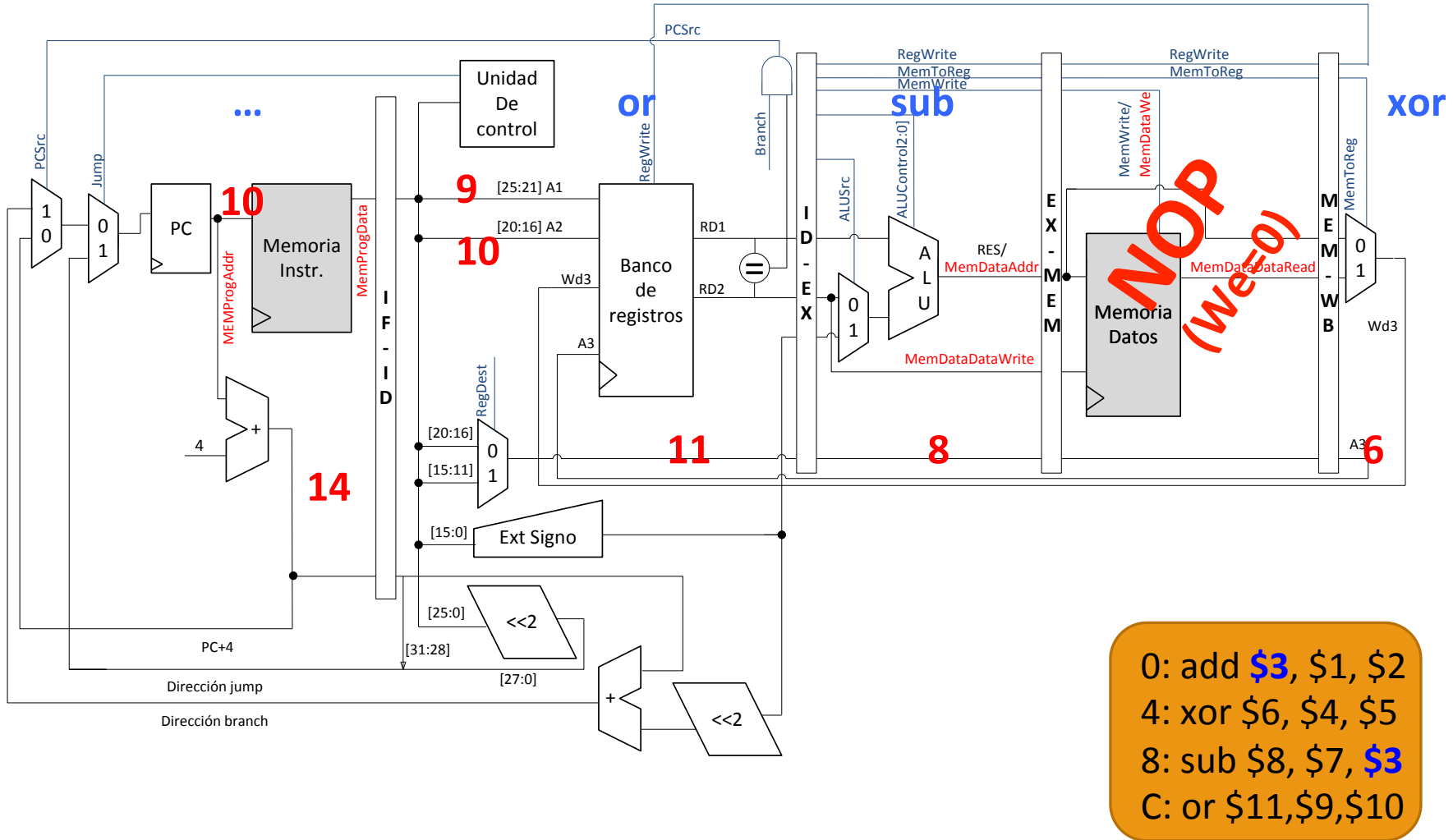


Segmentación en MIPS. Ciclo 5 (E1)



0: add \$3, \$1, \$2
 4: xor \$6, \$4, \$5
 8: sub \$8, \$7, \$3
 C: or \$11, \$9, \$10

Segmentación en MIPS. Ciclo 6 (E1)

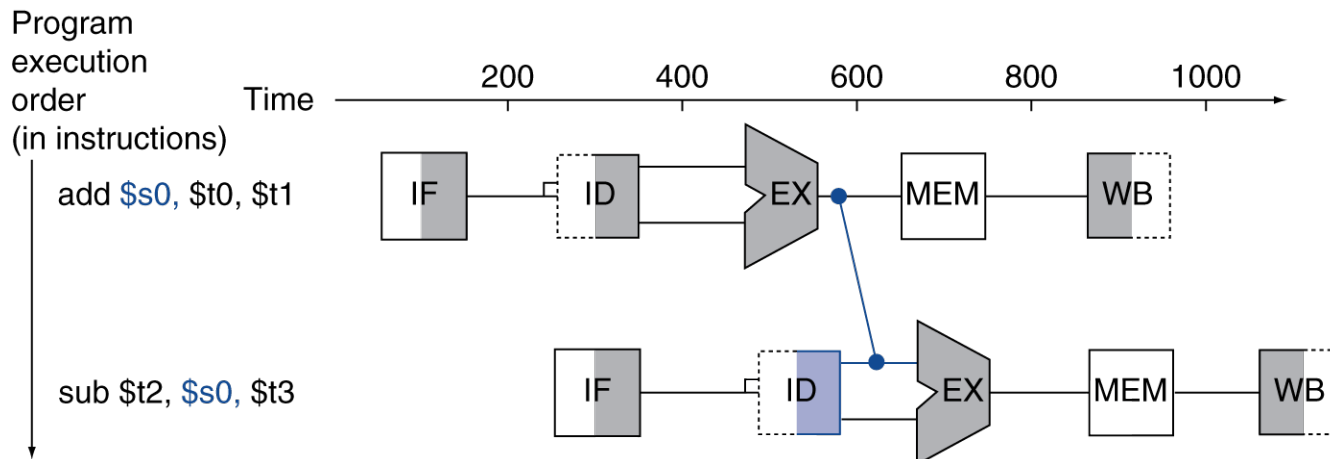


0: add \$3, \$1, \$2
 4: xor \$6, \$4, \$5
 8: sub \$8, \$7, \$3
 C: or \$11, \$9, \$10

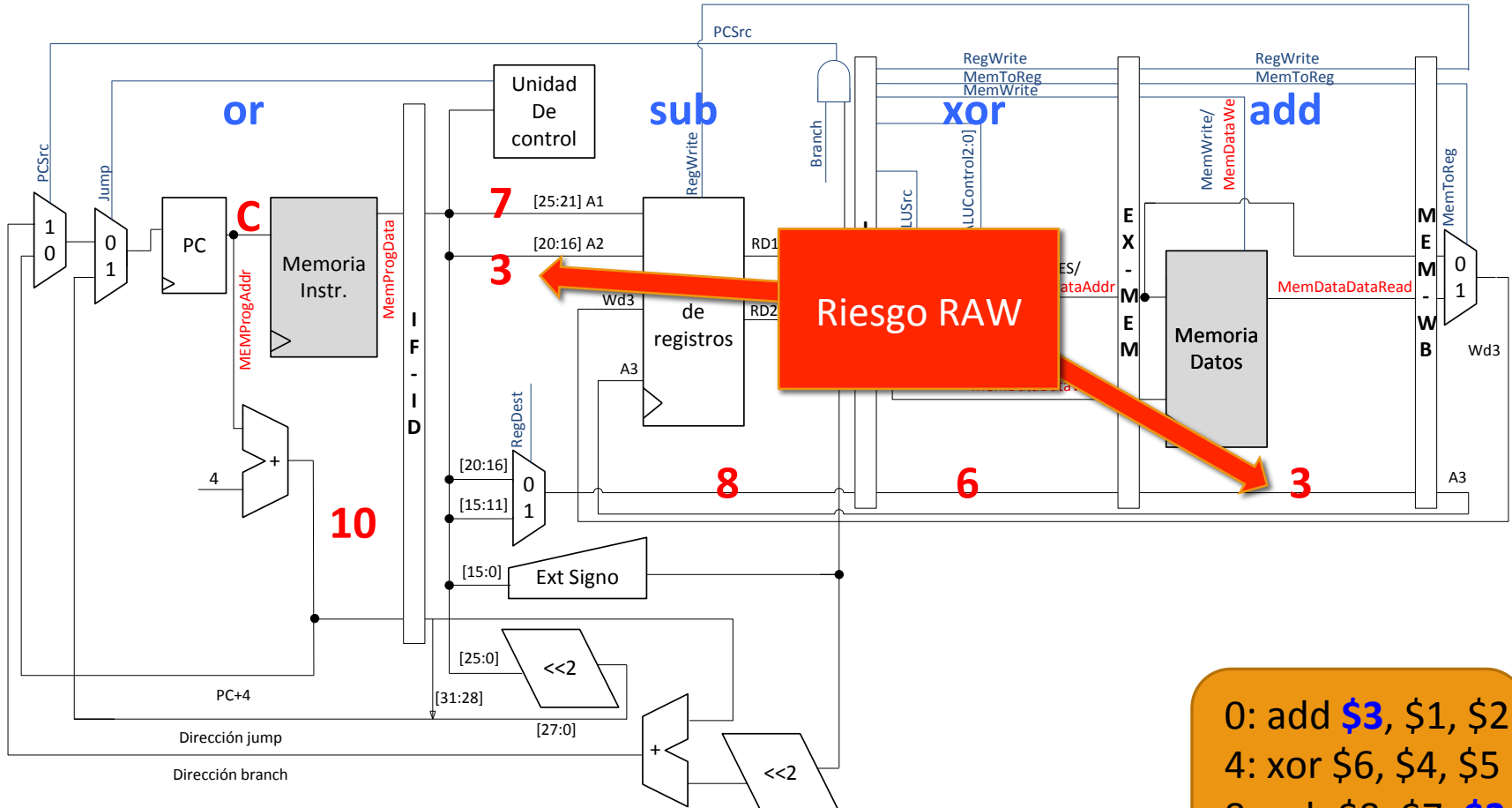
E2: adelantamiento de datos

Use result when it is computed

- Don't wait for it to be stored in a register
- Requires extra connections in the datapath

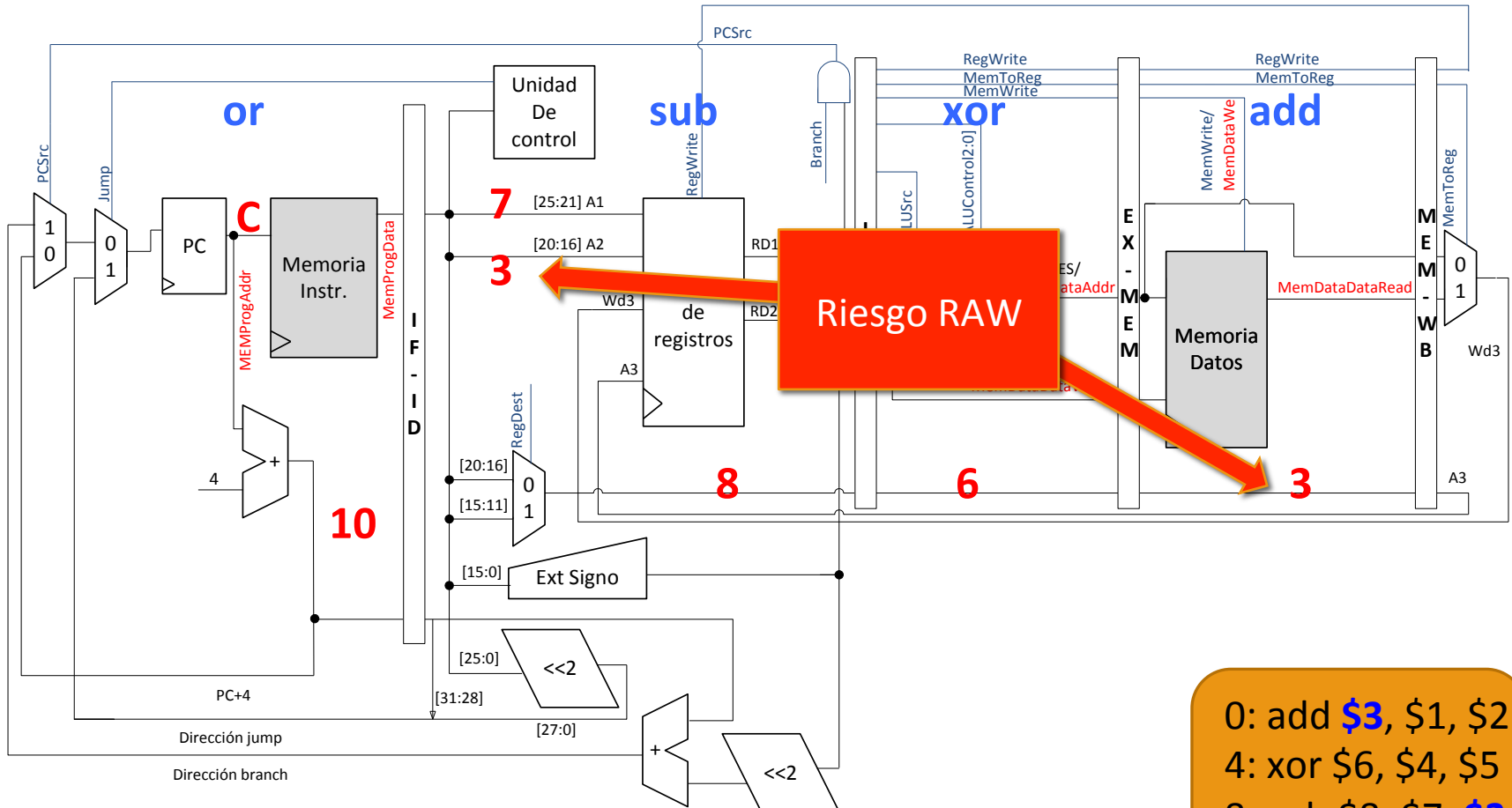


Segmentación en MIPS. Ciclo 4



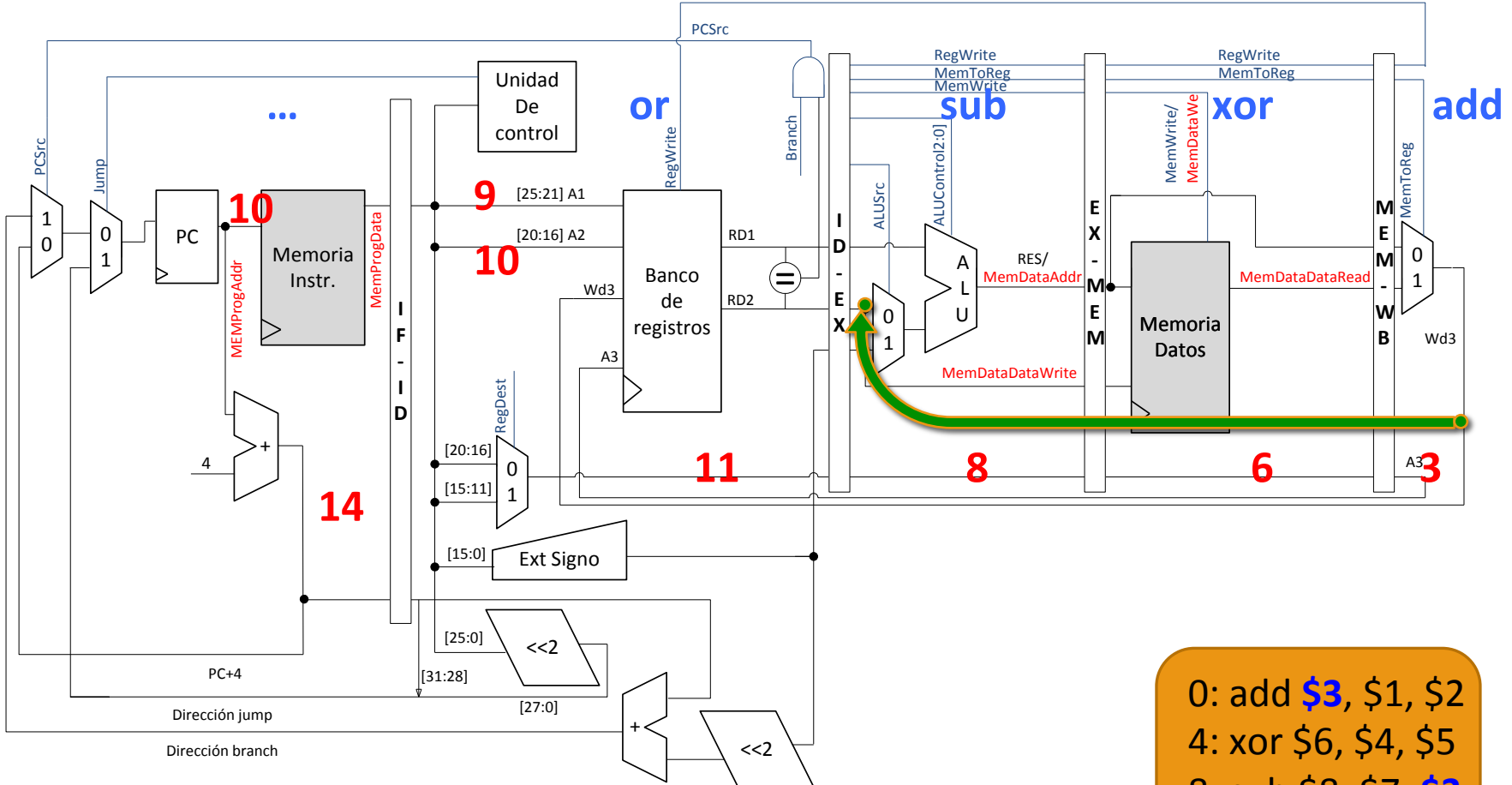
0: add \$3, \$1, \$2
 4: xor \$6, \$4, \$5
 8: sub \$8, \$7, \$3
 C: or \$11, \$9, \$10

Segmentación en MIPS. Ciclo 4 (E2)



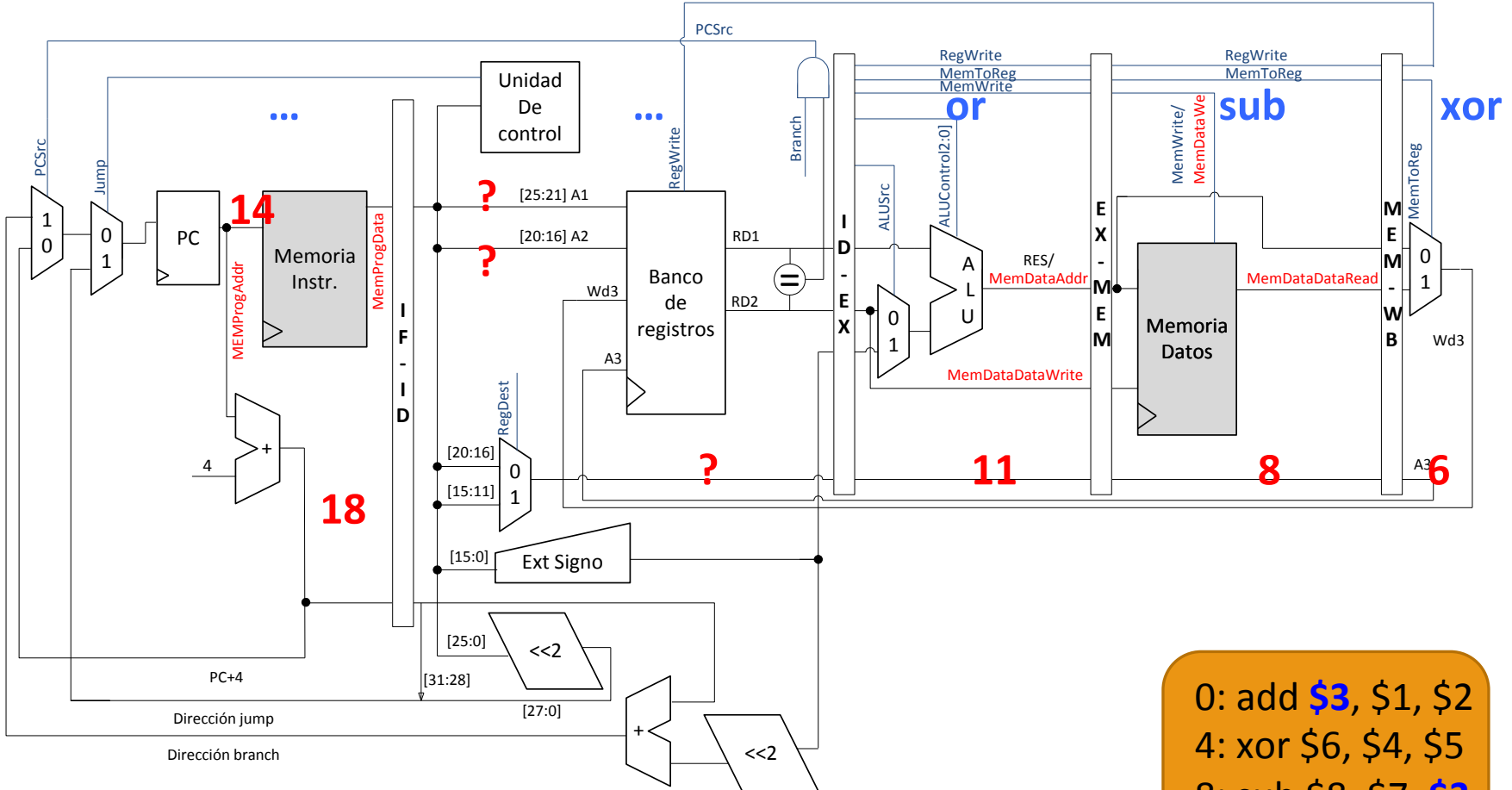
0: add \$3, \$1, \$2
 4: xor \$6, \$4, \$5
 8: sub \$8, \$7, \$3
 C: or \$11, \$9, \$10

Segmentación en MIPS. Ciclo 5 (E2)



* El procesador pedido no implementa la operación addi.

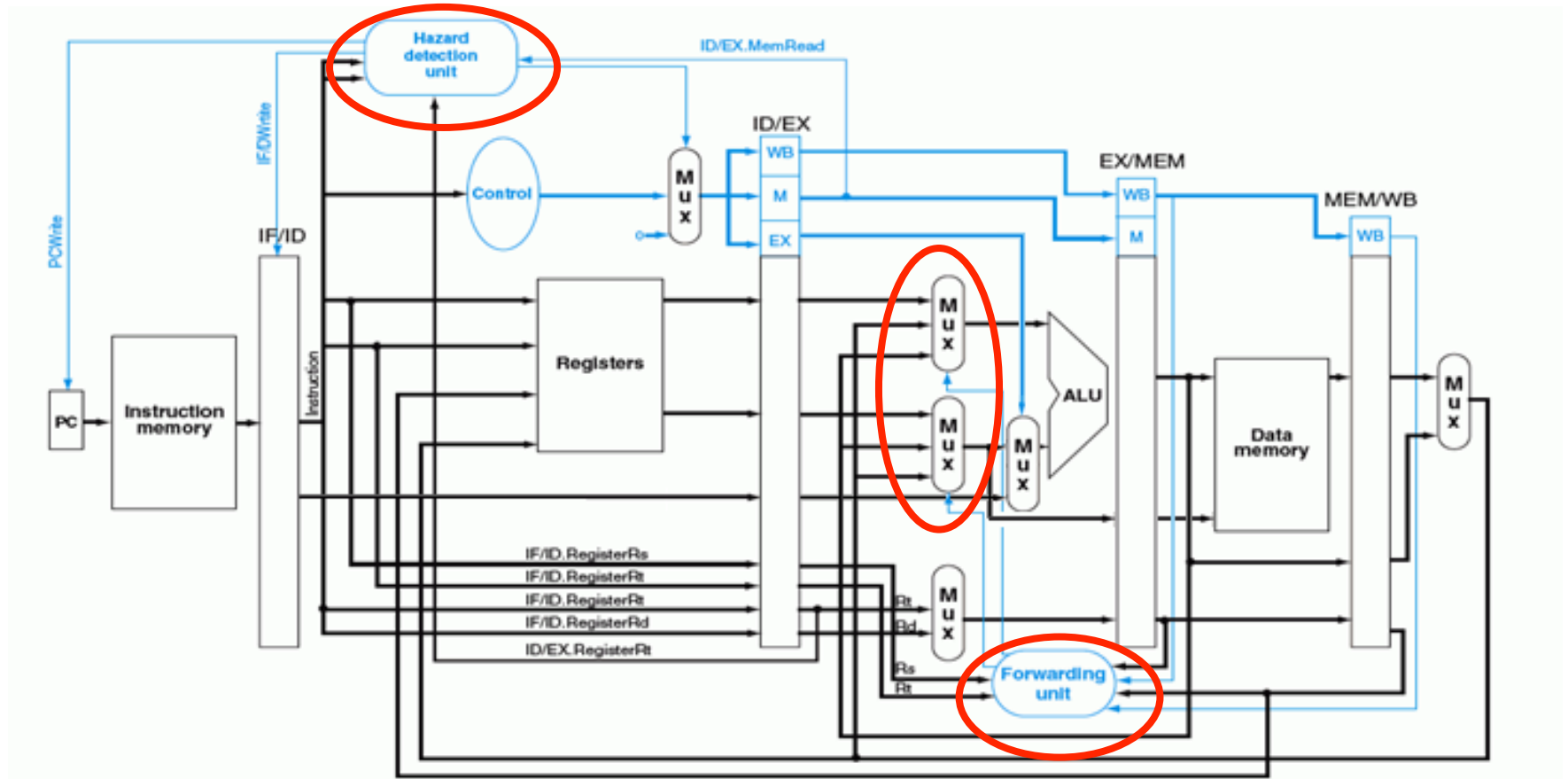
Segmentación en MIPS. Ciclo 6 (E2)



* El procesador pedido no implementa la operación addi.

```
0: add $3, $1, $2
4: xor $6, $4, $5
8: sub $8, $7, $3
C: or $11, $9, $10
```

Modificaciones al MIPS



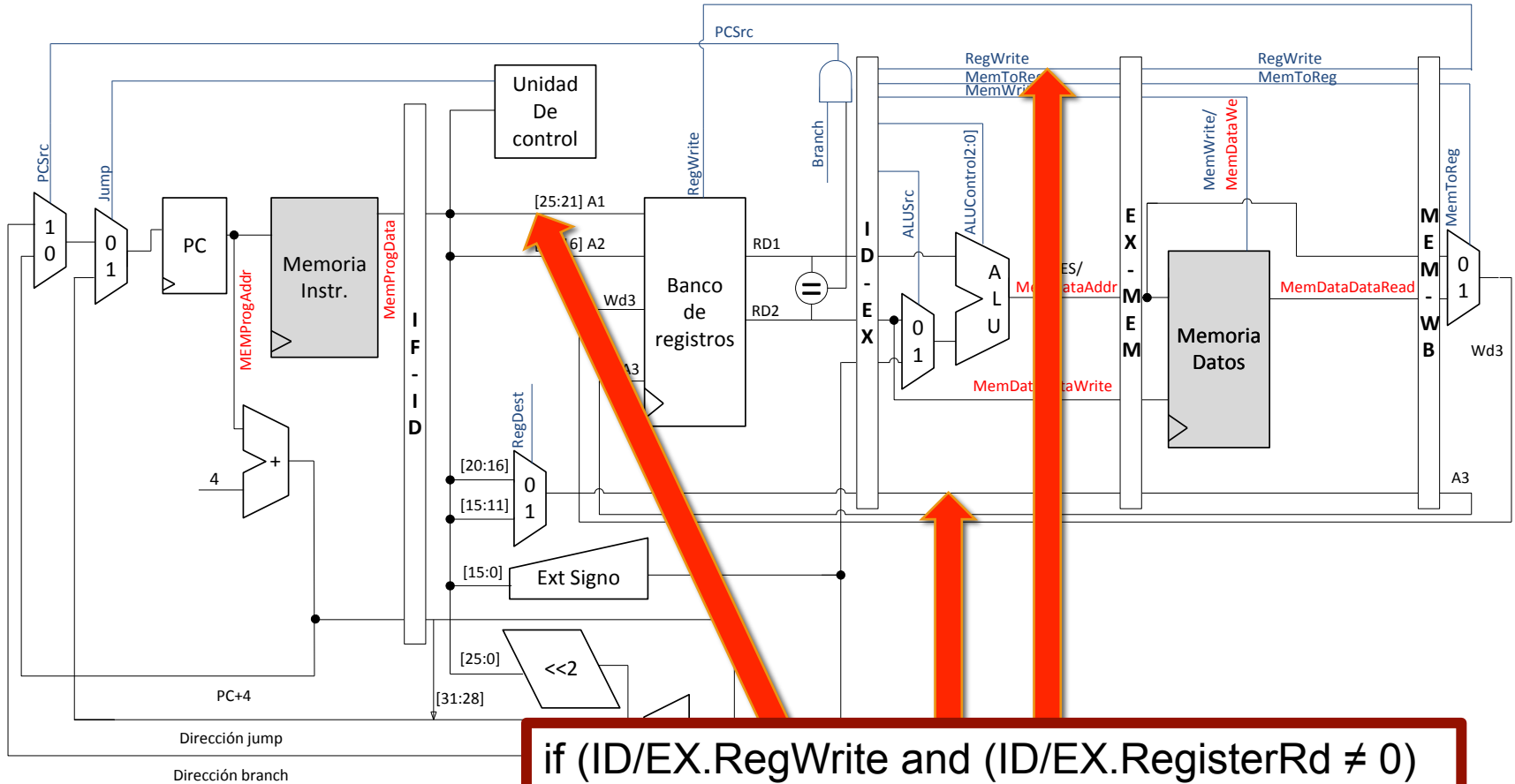
Detección de riesgos

- La primera alternativa (E1) requiere que la detección de riesgos se realice en la etapa ID.
- La detección de riesgos para la segunda alternativa (E2) se puede calcular en la etapa ID (igual que E1) o en la etapa EX (teoría).

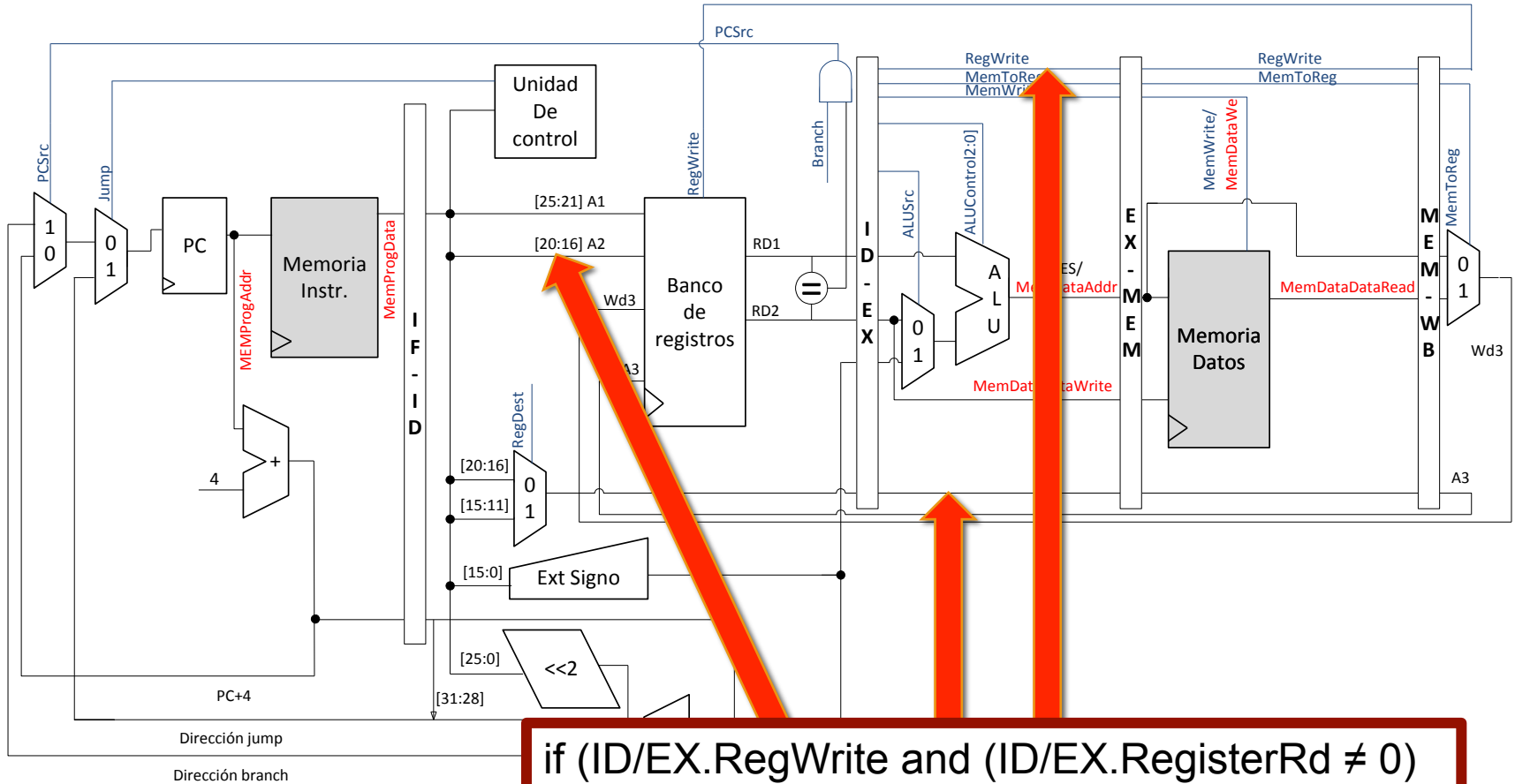
Detección de riesgos (E1)

- Para el ejercicio 1 **NO** nos sirven las condiciones vistas en teoría
 - Necesitamos detectar el riesgo cuando la instrucción está en ID
 - Simplemente hay que adelantar una etapa las comprobaciones
- Si se detecta riesgo hay que parar el pipeline
 - Enable_PC_reg = '0'
 - Enable_IF_ID_reg = '0'
 - Bubble_ID_EX_reg = '1'

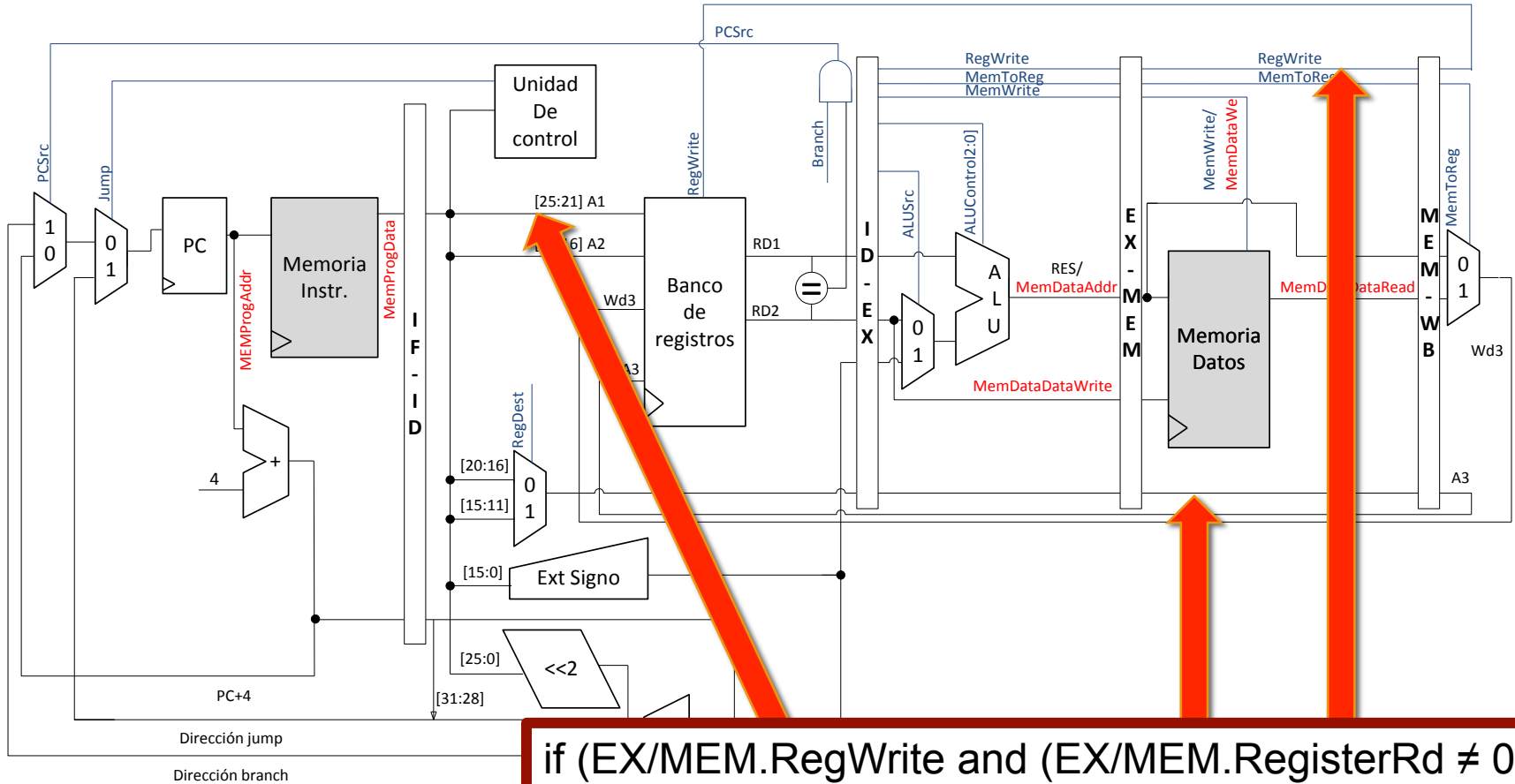
Detección de riesgos (E1): caso 1



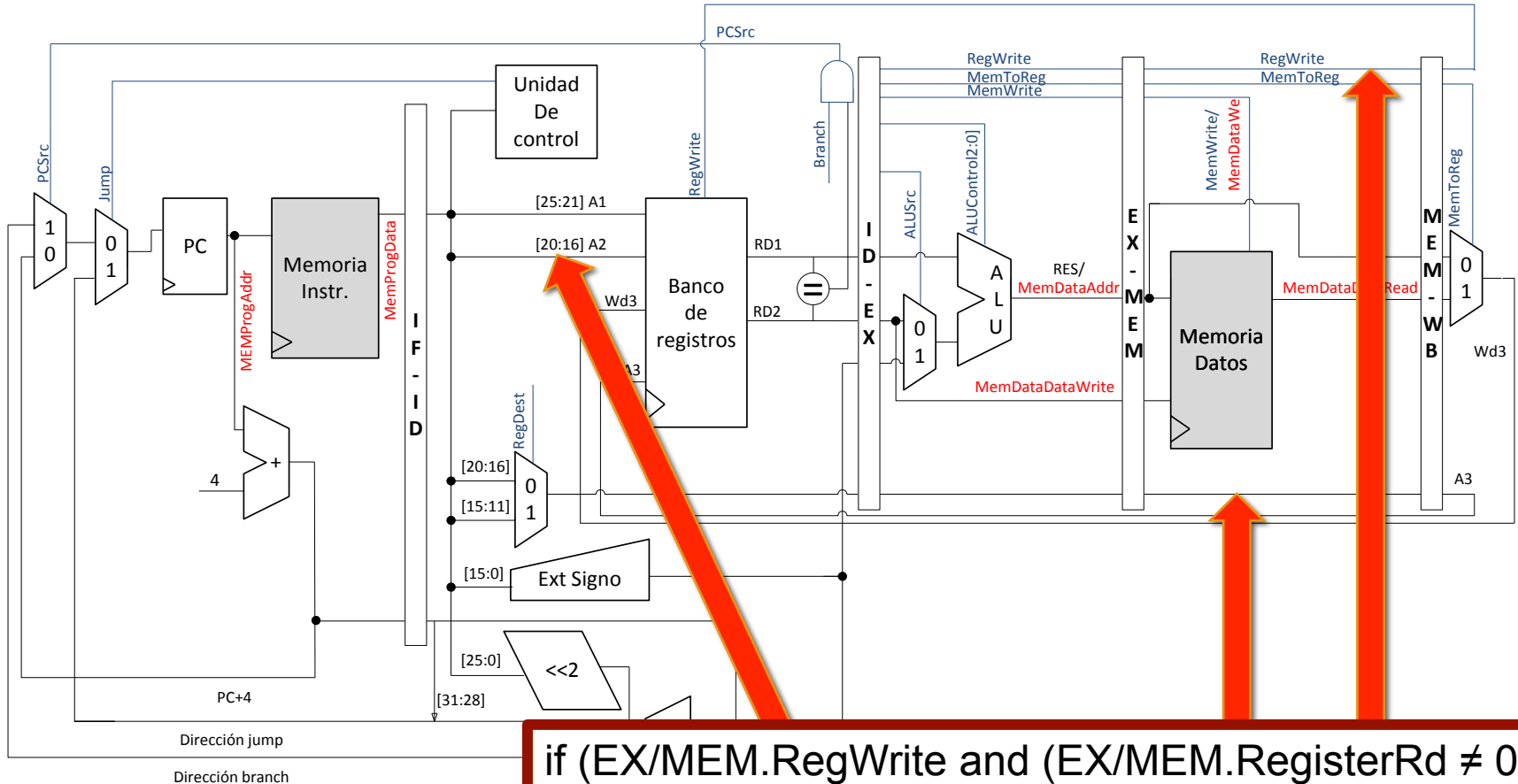
Detección de riesgos (E1): caso 2



Detección de riesgos (E1): caso 3



Detección de riesgos (E1): caso 4



if (EX/MEM.RegWrite and (EX/MEM.RegisterRd \neq 0)
and (EX/MEM.RegisterRd = IF/ID.RegisterRt))

Detección de riesgos (teoría)

Forwarding Conditions

■ EX hazard

- 1 ■ if (EX/MEM.RegWrite and (EX/MEM.RegisterRd \neq 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs))

ForwardA = 10

- 2 ■ if (EX/MEM.RegWrite and (EX/MEM.RegisterRd \neq 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt))

ForwardB = 10

■ MEM hazard

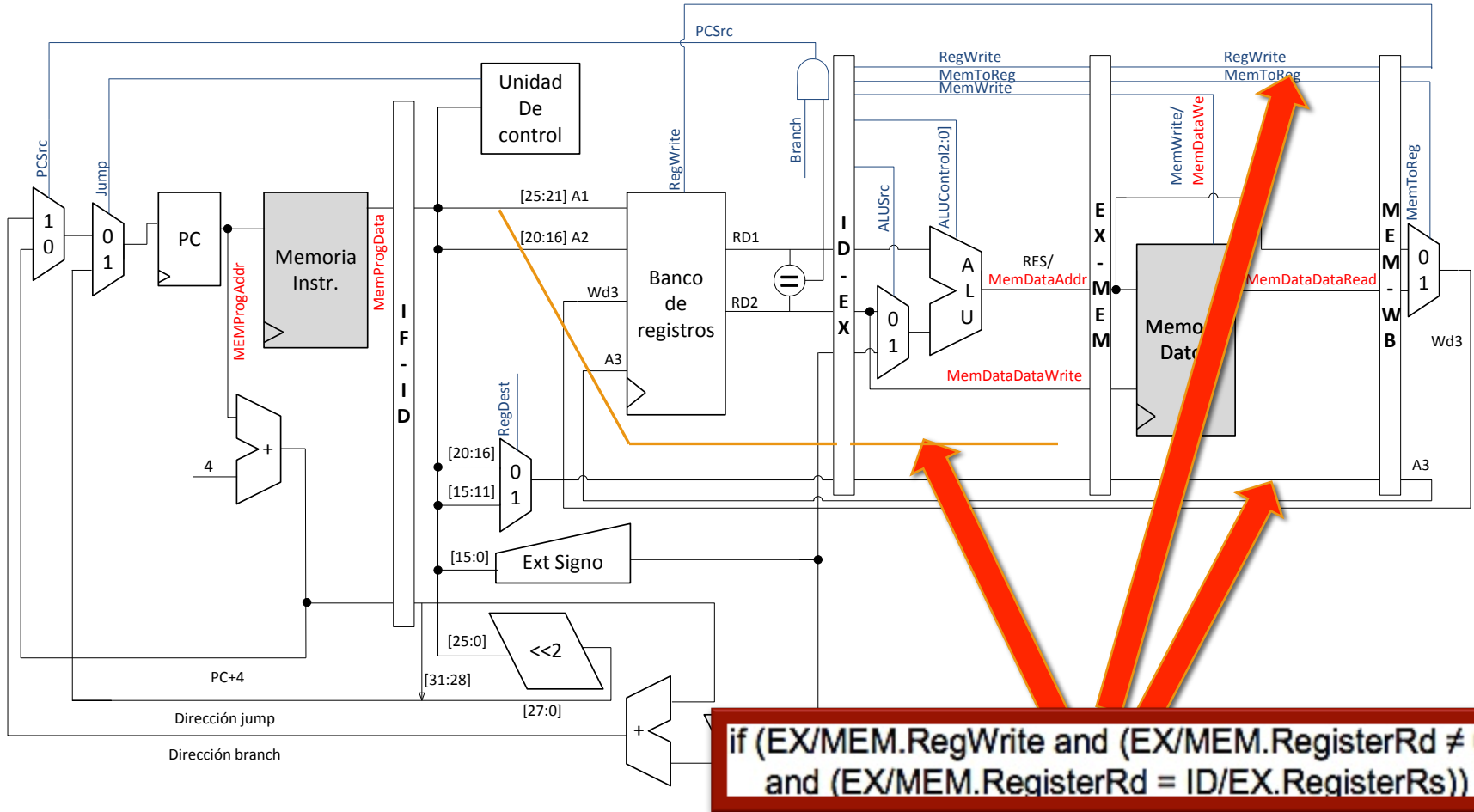
- 3 ■ if (MEM/WB.RegWrite and (MEM/WB.RegisterRd \neq 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs))

ForwardA = 01

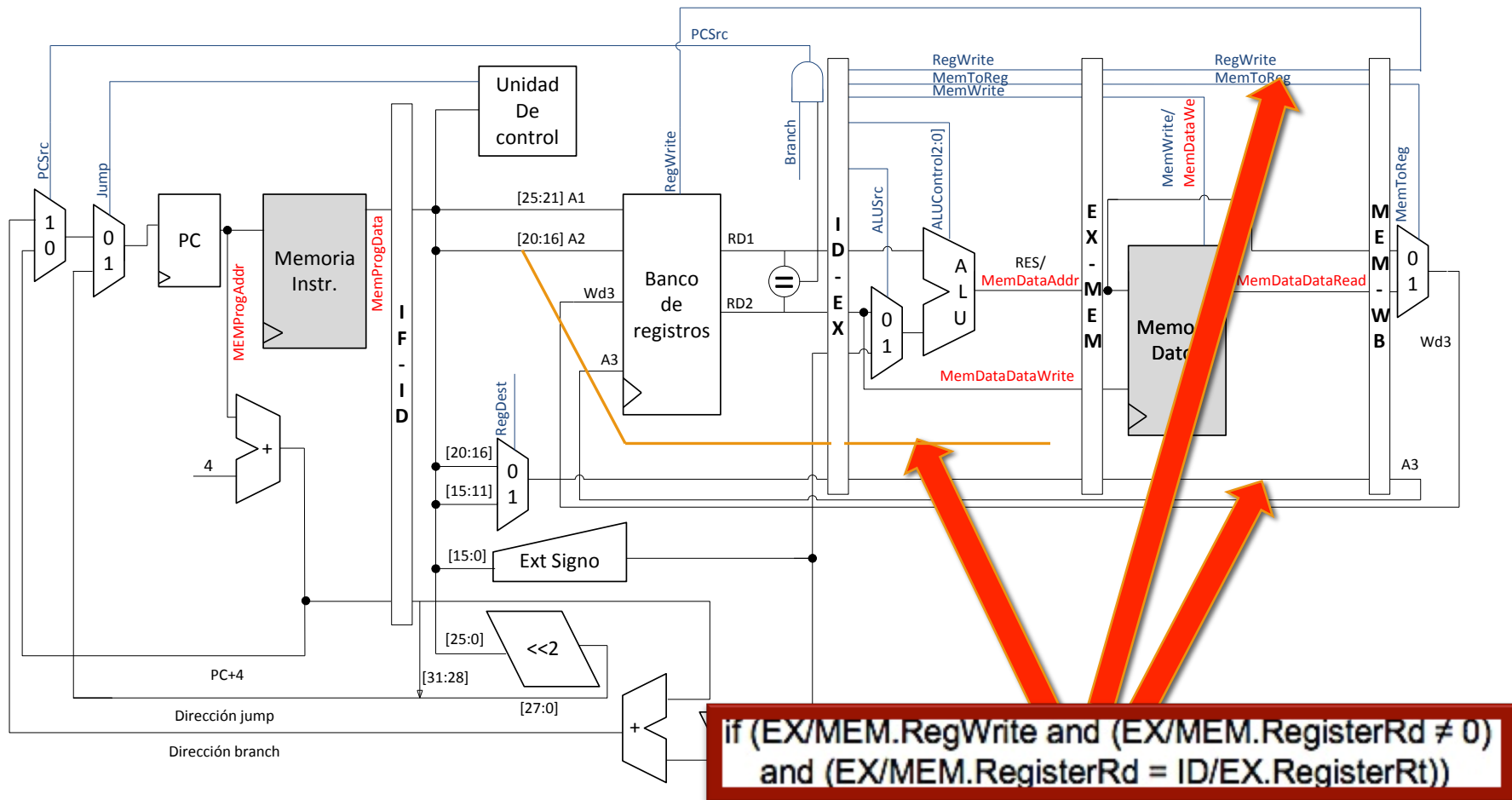
- 4 ■ if (MEM/WB.RegWrite and (MEM/WB.RegisterRd \neq 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt))

ForwardB = 01

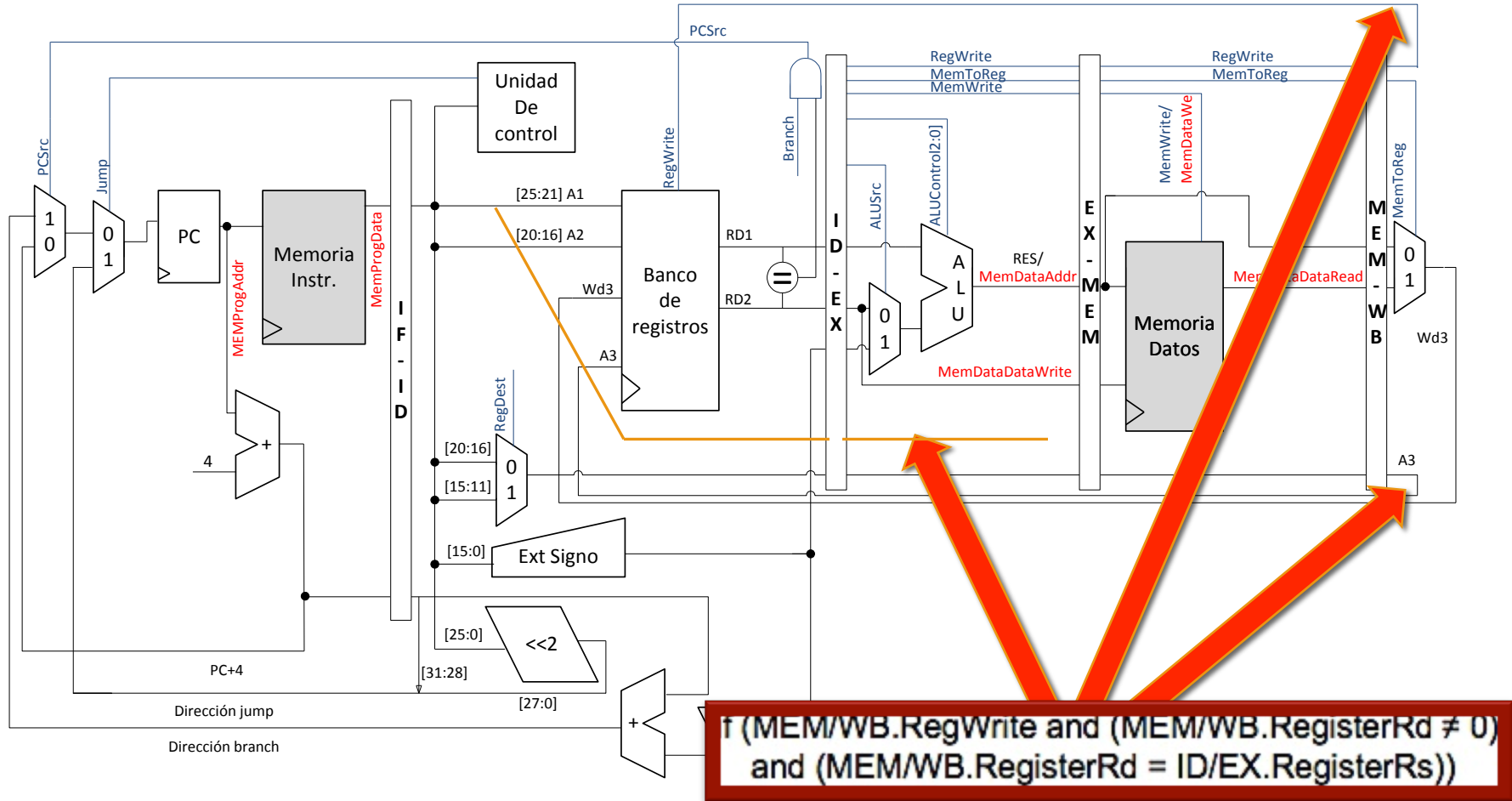
Detección de riesgos (teoría): caso 1



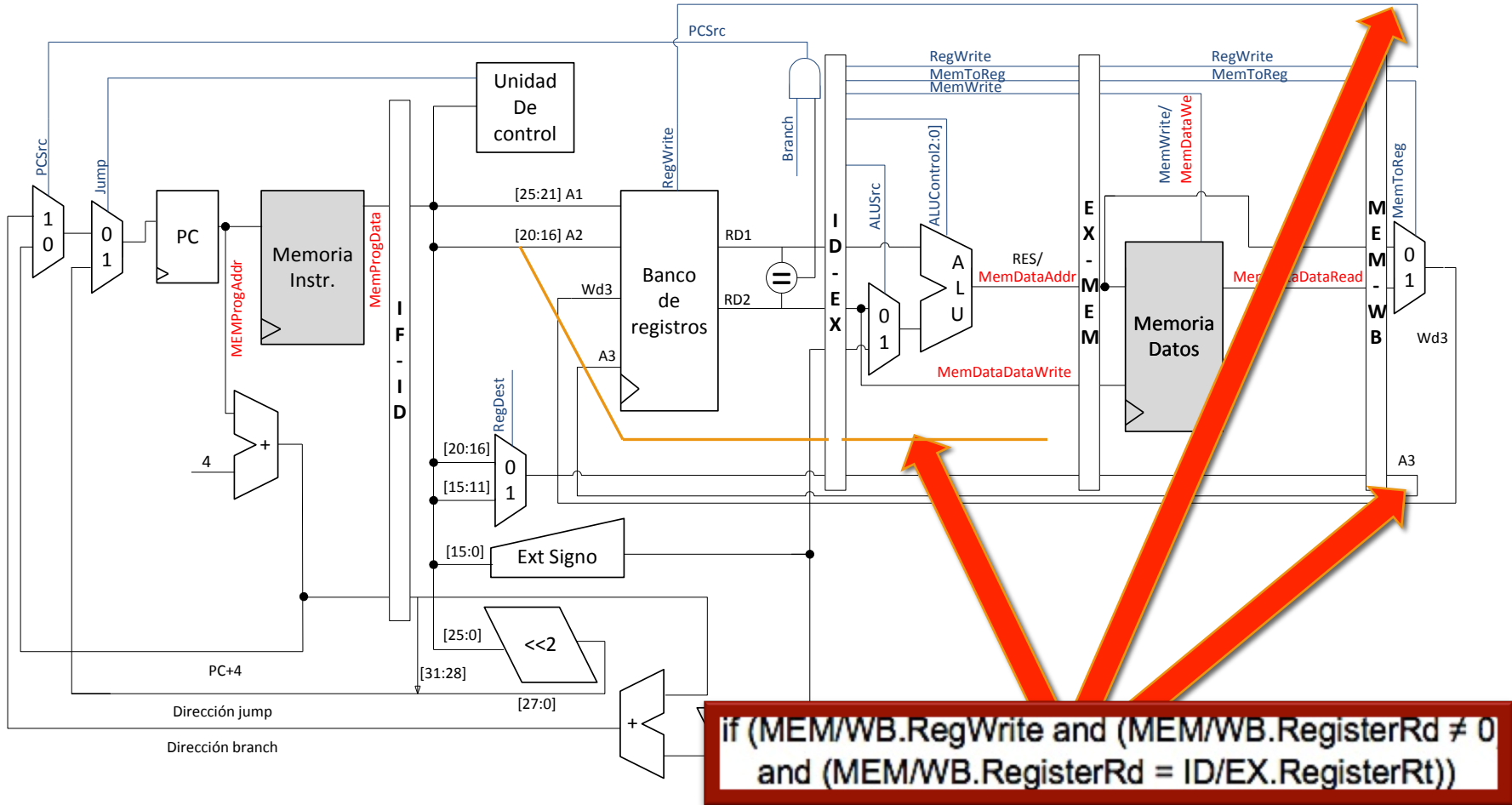
Detección de riesgos (teoría): caso 2



Detección de riesgos (teoría): caso 3



Detección de riesgos (teoría): caso 4



Detección de riesgos. Doble riesgo

0: add \$2, \$1, \$0
4: or \$2, \$6, \$9
8: add \$5, \$2, \$1

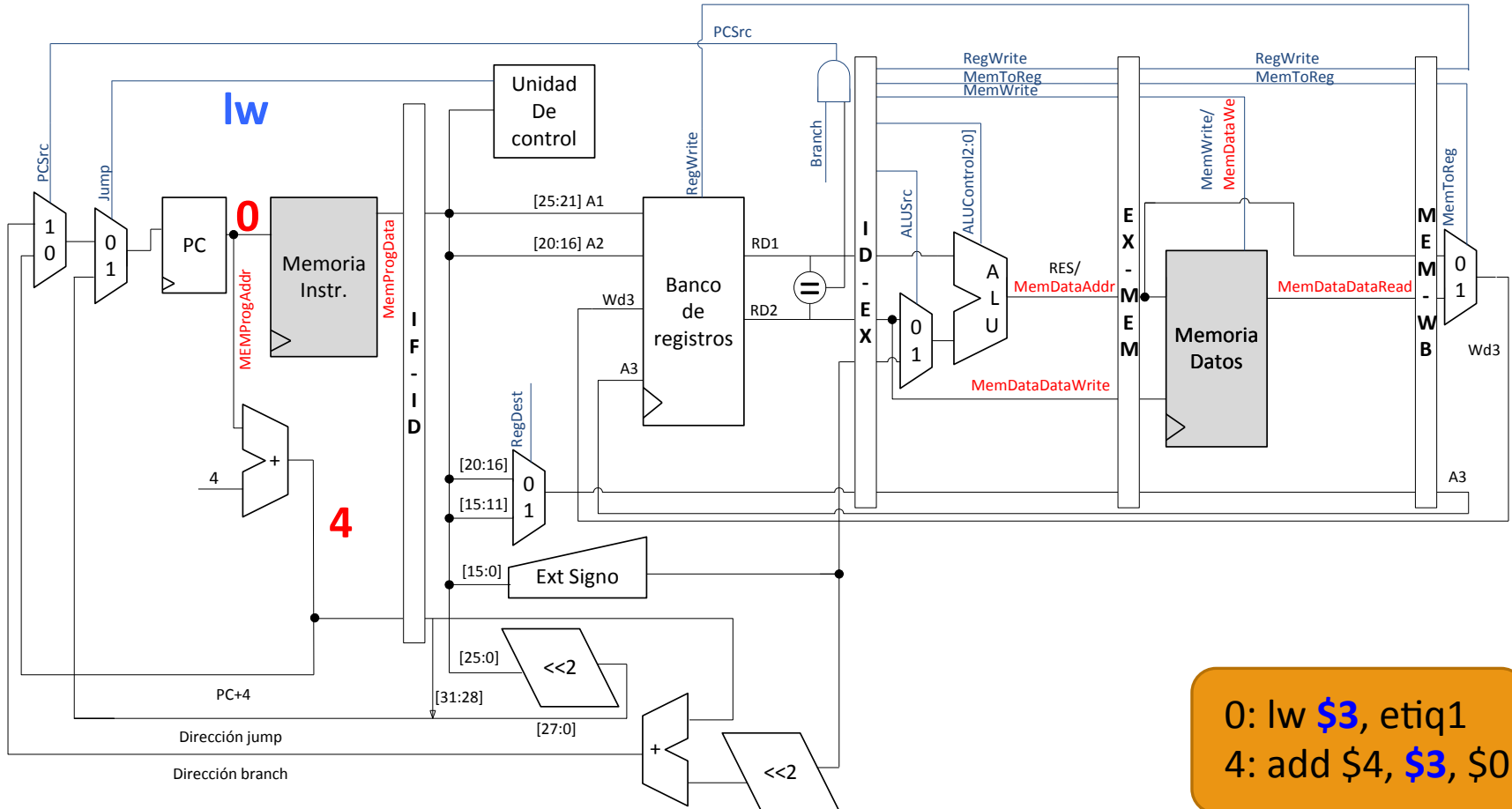
Si hay un riesgo doble, se debe adelantar el dato más actualizado.

Sólo se adelanta el resultado de la etapa MEM si no se puede adelantar el dato de la etapa EX.

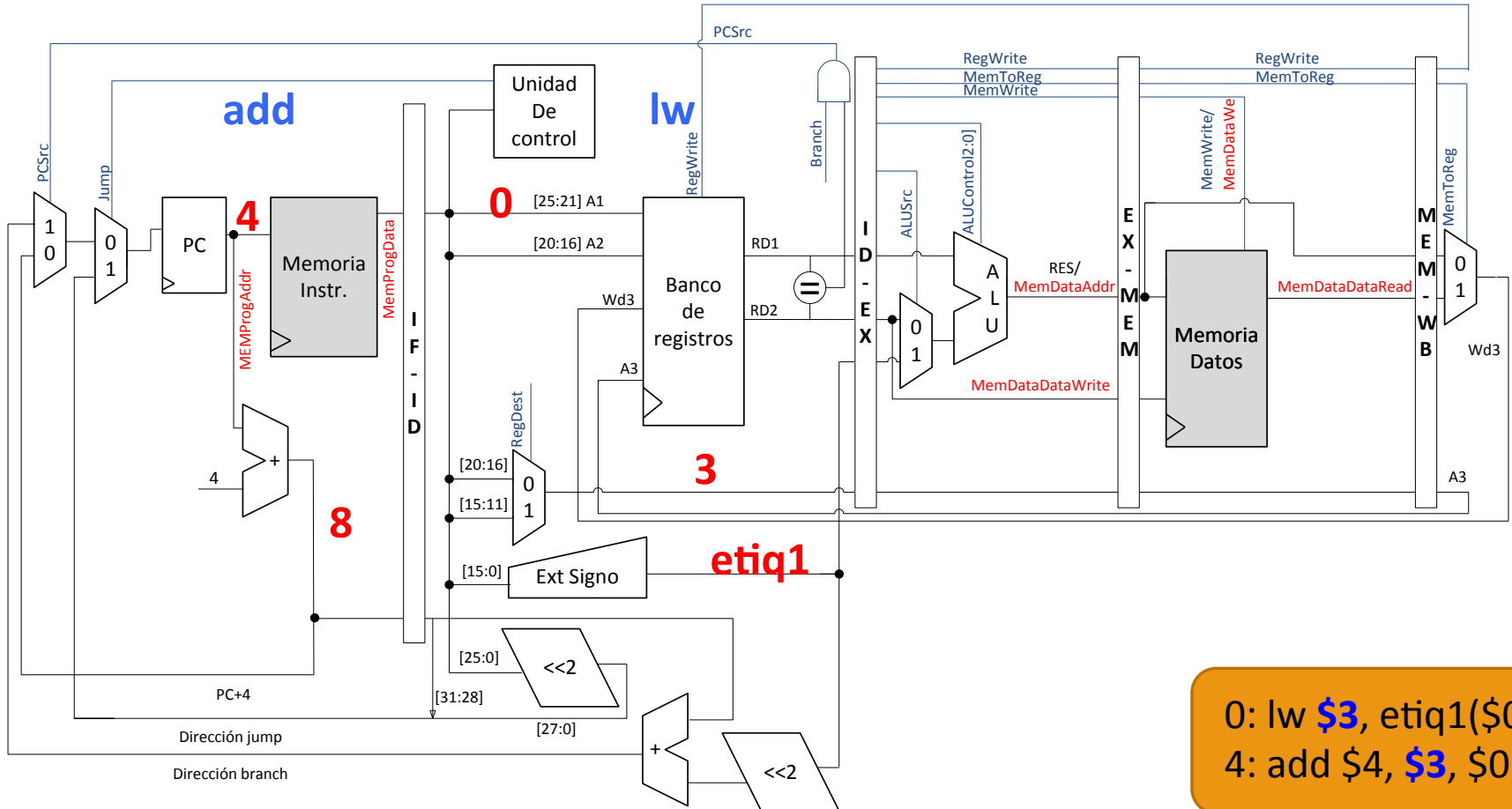
Adelantamiento de datos (E2)

- Para el ejercicio 2 **SÍ** nos sirven las condiciones vistas en teoría
- Dos opciones:
 1. Utilizar las comparaciones del libro
 - Las señales de control de los MUX de adelantamiento se generan en la etapa EX (la misma en la que se utilizan)
 2. Re-utilizar las comparaciones el ejercicio 1
 - Las señales de control de los MUX de adelantamiento se generan en la etapa ID, por lo que hay que pasarlas por un registro para utilizarlos en EX
- Parar pipeline: cuando hay riesgo con LW

Adelantamiento de datos. Lw. Ciclo 1



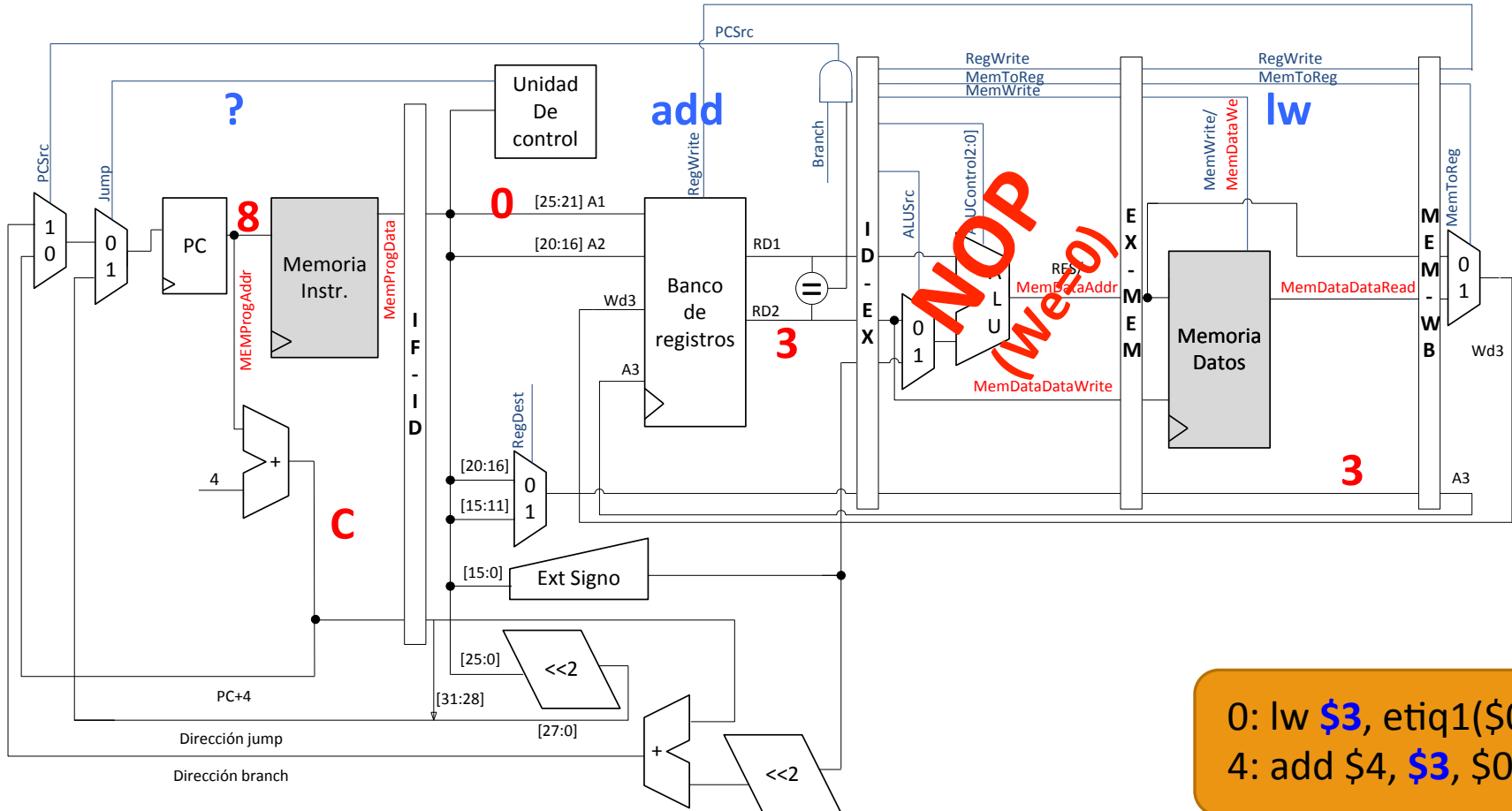
Adelantamiento de datos. Lw. Ciclo 2



38

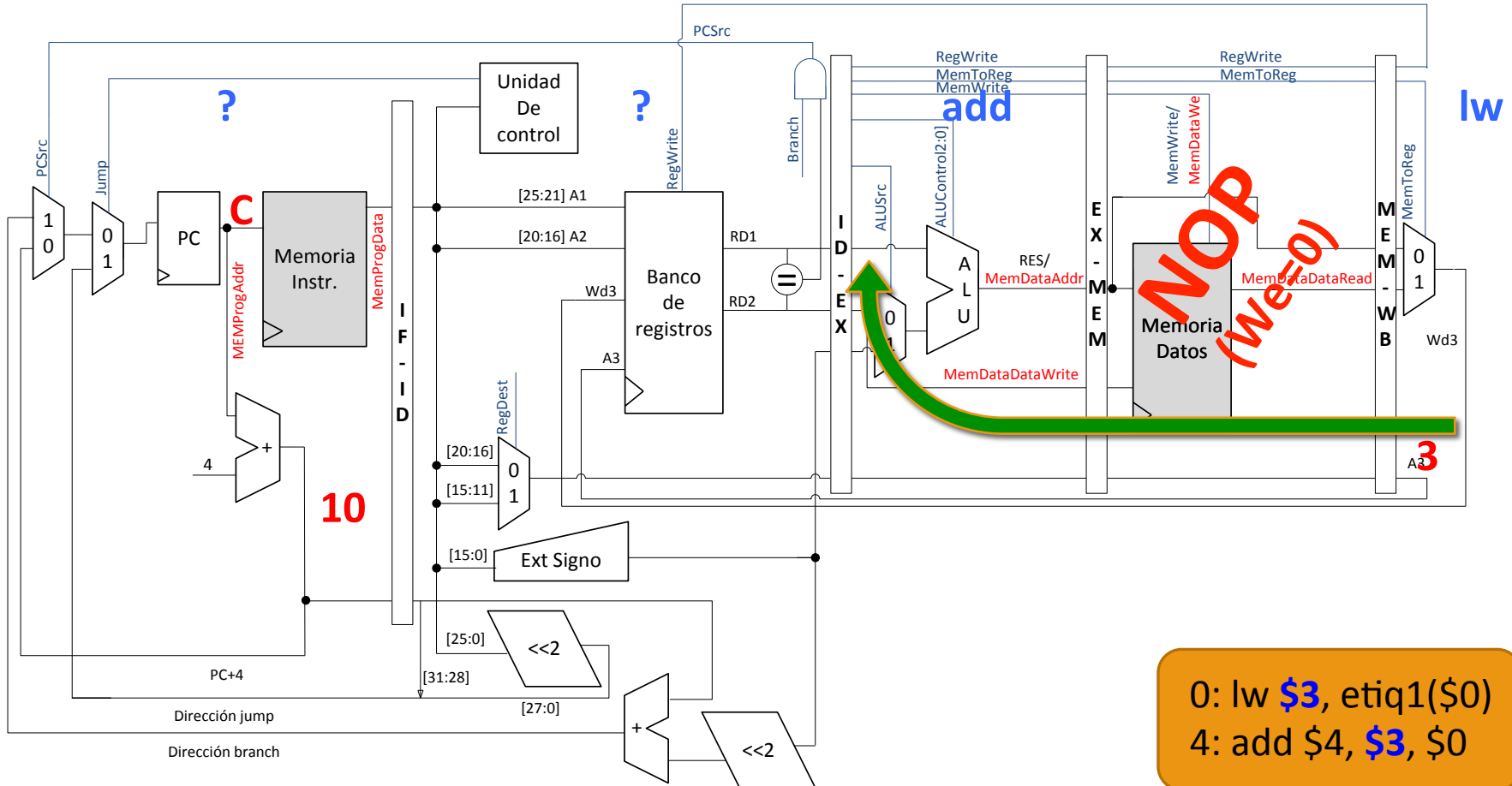


Adelantamiento de datos. Lw. Ciclo 4



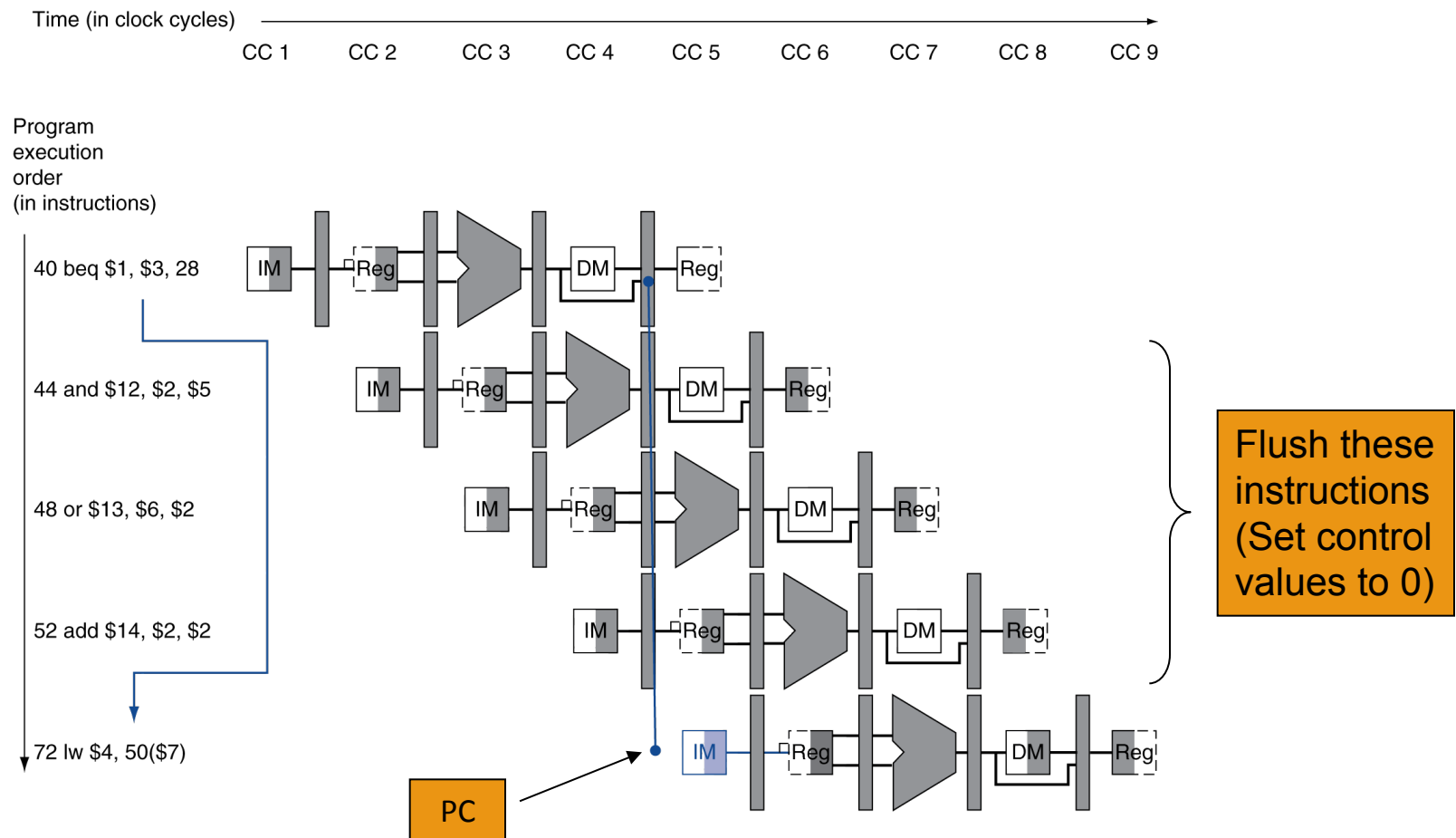
0: lw \$3, etiq1(\$0)
4: add \$4, \$3, \$0

Adelantamiento de datos. Lw. Ciclo 5



Branch Hazards

If branch outcome determined in MEM (Teoría)



Reducing Branch Delay

Move hardware to determine outcome **to ID** stage

- Target address adder
- Register comparator

Example: branch taken

```
36:  sub    $10, $4, $8
40:  beq    $1,  $3, 7
44:  and    $12, $2, $5
48:  or     $13, $2, $6
52:  add    $14, $4, $2
56:  slt    $15, $6, $7

    . . .
72:  lw     $4, 50($7)
```