

Estructura de Computadores

Grado de Ingeniería Informática
ETSINF

Tema I: El procesador



Objetivos

- Contextualizar la asignatura
 - ✓ Conocer el contexto de la arquitectura MIPS32
 - ✓ Conocer los aspectos más generales de la arquitectura MIPS32
 - ✓ Aprender el ciclo de ejecución del procesador
- Conocer el diseño de la ruta de datos basada en multiplexores
- Conocer el diseño de la unidad de control del procesador

Introductorio (Tema 0)

Contenido y Bibliografía

- I – Arquitectura MIPS32

- ✓ Introducción
- ✓ Carácterísticas básicas
- ✓ Ejemplo de ejecución

Introductorio
(Tema 0)



- 2 – La ruta de datos y la unidad de control

- ✓ Etapas de búsqueda y decodificación
- ✓ Diseño de la ruta para aritméticas de tipo R
- ✓ Diseño de la ruta para aritméticas de tipo R+I
- ✓ Diseño de la ruta para instrucciones lw/sw
- ✓ Diseño de la ruta para instrucciones de salto beq/bne



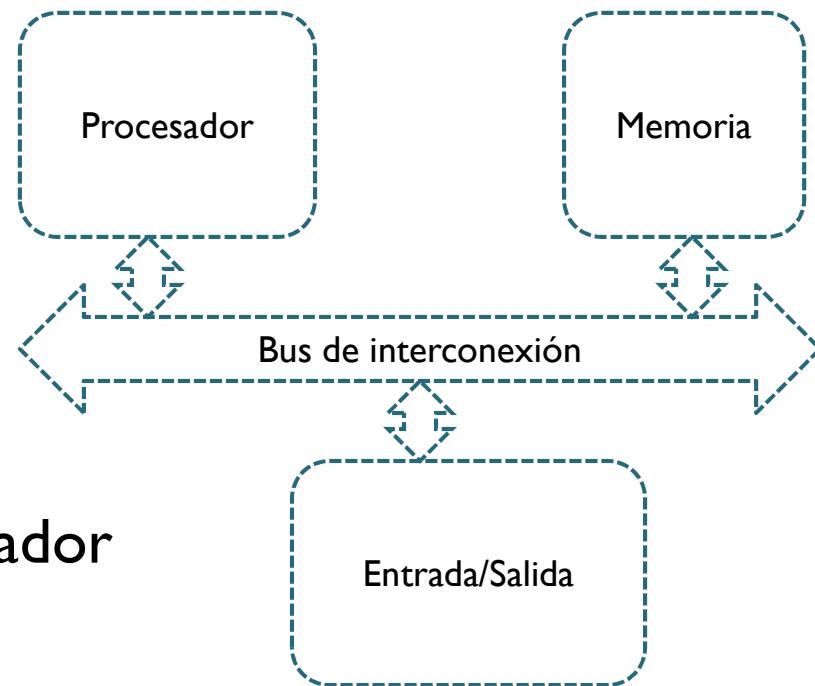
Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4)

Introducción

- I – Arquitectura MIPS32
 - ✓ Introducción
 - ✓ Carácterísticas básicas
 - ✓ Ejemplo de ejecución

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4)

Introducción



- Componentes básicos computador
 - ✓ Procesador
 - ✓ Memoria
 - ✓ Entrada/Salida
 - ✓ Interconexión

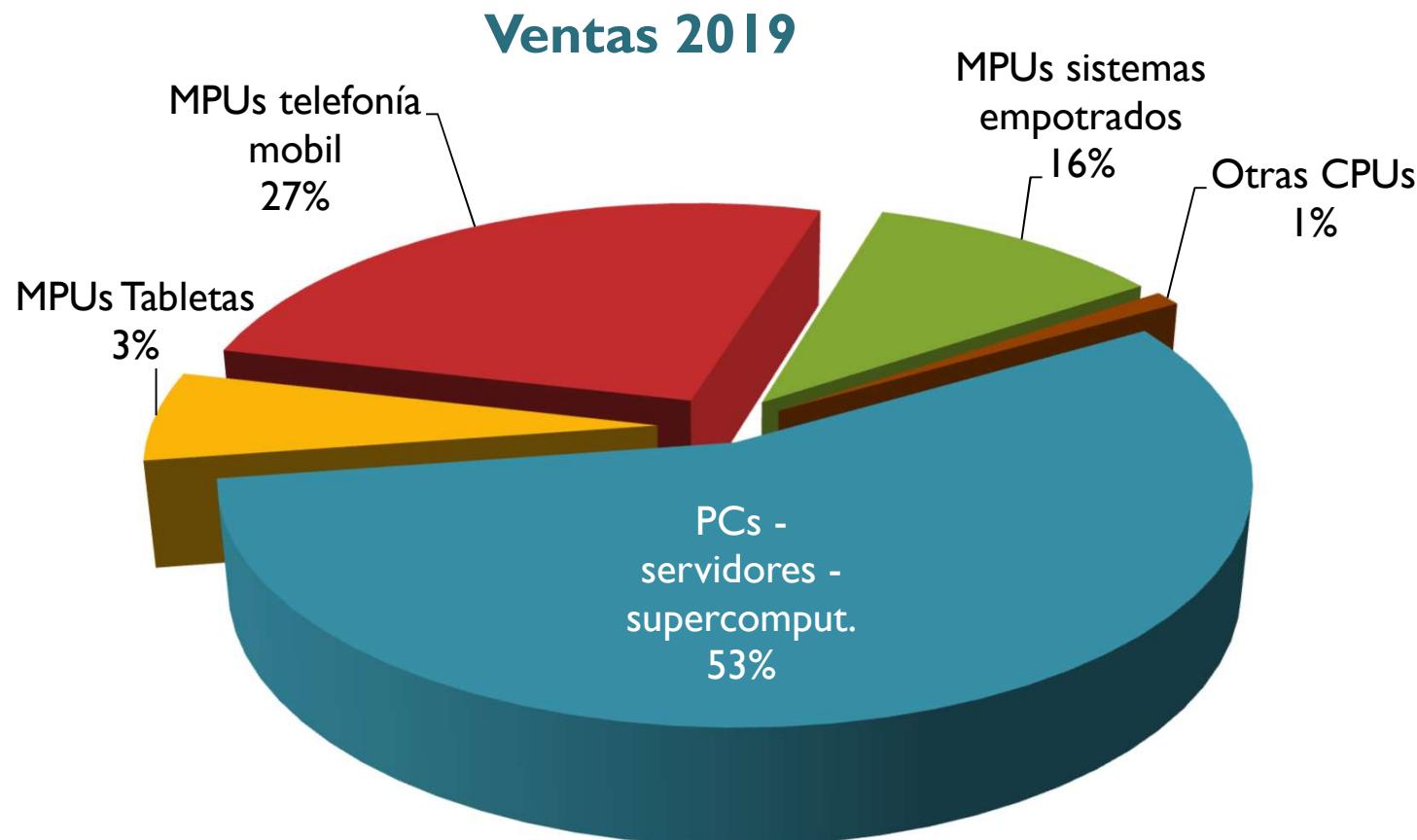
Concepto de programa almacenado:

- Las instrucciones se representan como números
- Los programas son almacenados en memoria para ser leídos o escritos también como números

Introducción: Tipos de computadores

Tipo	Potencia	Ancho Palabra (bits)	Consumo	Aplicaciones
Mainframes y Supercomputers	Muy alta potencia de cálculo.	64, 128	Extremadamente alto	Gestión en Corporaciones, Gobiernos. Aplicaciones computación masiva: centros de investigación.
Workstations	Alta	32, 64	Alto	Diseño, simulación, control: Empresas, Universidades
Desktops Laptops Netbooks	Alta-Media	32, 64	Medio Bajo	Informática en general Computadores personales
Tablet	Media	32,64	Muy bajo	Informática personal
Embedded	Media-Alta.	16, 32, 64	Medio Bajo	Sistemas empotrados para: Comunicaciones (routers, switches). Electrodomésticos, automoción, periféricos, sistemas industriales,...
Cellphones	Media-Baja	16, 32	Extremadamente bajo	Teléfonos móviles

Introducción: El mercado de los procesadores



MPU: Microprocessor Unit

Todo procesador se define de acuerdo con una **ARQUITECTURA**

Introducción: Arquitectura

- **Arquitectura (ISA: Instruction Set Architecture)**

- Hace referencia al repertorio de instrucciones, registros, modelo de excepciones, manejo de la memoria virtual, mapa de direcciones físicas y otras características comunes de un procesador
- Todo aquello que el programador debe saber acerca de la máquina

- **Ejemplos de arquitecturas:**

Arquitectura	Empresas
x86, x86-64, IA-32, INTEL®64	INTEL,AMD,..
MIPS-32, MIPS-64	Mips Technologies
DEC Alpha Architecture	Digital Equipment Corp.
Power , PowerPC	Apple, IBM, Motorola
ARM Architecture	Advanced RISC Machines Holdings)

- Las arquitecturas pueden evolucionar, dando lugar a versiones de 16, 32 o 64 bits, siempre compatibles entre sí.

Introducción: Implementación

- **Implementación:** Hace referencia a las características concretas de los circuitos que conforman el diseño de un procesador que ejecuta una arquitectura.
 - A veces, con una implementación concreta se crean familias de modelos, como las ‘microarquitecturas’, de INTEL

Arquitectura	Familias / Modelos		Fabricantes
MIPS	R2000, R3000, R4000,...., R16000		MIPS Tech.
IA-32, Intel 64	Microarquitectura: P5, P6	Modelos: Pentium II, III, Pro	INTEL
	NetBurst	Pentium 4, Pentium D, Celeron D	
	Pentium M	Core Duo, Core Solo, Pentium M, Celeron M	
	Intel Core	Core 2 Duo, Quad, Extreme, Xeon 5xxx, 7xxx, Celeron dual-core	
	Atom	Atom	
	Core i7	i3, i5, i7	
X86, x86-64	K5,K6,K8	Athlon 64, Phenom, Turion, Semprom, Opteron	AMD
ALPHA	22064, 21164, 22164, 23164		DEC

Arquitectura MIPS

- **MIPS: Microprocessor without Interlocked Pipeline Stages)**
 - ✓ Procesador RISC (**R**educed **I**nstruction **S**et) desarrollado en la Universidad de Stanford por John L. Hennessy que emplea la técnica de segmentación .
 - ✓ Comercializado por MIPS Technologies (Silicon Graphics International)
 - ✓ Arquitectura sencilla → Muy utilizada por las universidades para docencia
 - ✓ El diseño segmentado del MIPS es el precursor de la mayoría de los procesadores RISC posteriores

Versión juego de instrucciones (ISA)	Ancho palabra	Procesadores
MIPS I	32	R2000, R3000
MIPS II	32	R6000
MIPS III	64	R4000
MIPS IV	64	R5000, R10000
MIPS V	64	-
MIPS32	32	4K
MIPS64	64	5K

Introducción

- I – Arquitectura MIPS32

- ✓ Introducción y Definición
- ✓ Características básicas
- ✓ Ejemplo de ejecución

AYUDA EN: Recursos\ Grupos\F\Tema1\videos:

- Vídeo 1. “Repaso MIPS R2000”
- Presentación: Arquitectura del MIPS R2000.ppsx

AYUDA ADICIONAL:

- En la subcarpeta “vídeos de repaso MIPS” se pueden encontrar videos para cada uno de los tipos de instrucciones

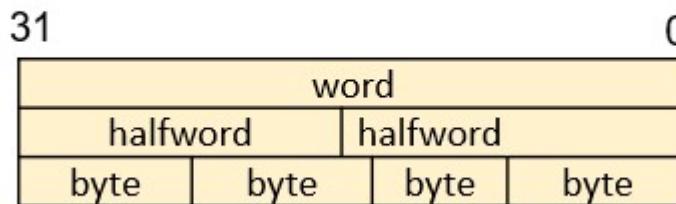
Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores.

La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4)

Arquitectura MIPS32: Características Básicas

- Ancho de palabra y tamaño de buses de 32 bits
- Principales tamaño de datos:

✓ **Byte (B), halfword (H), word (W)**



ES DECIR....



Instrucciones se codifican en una palabra: 32 bits

Bus de direcciones de 32 bits: luego 2^{32} direcciones diferentes = 2^{32} Bytes = 4GB

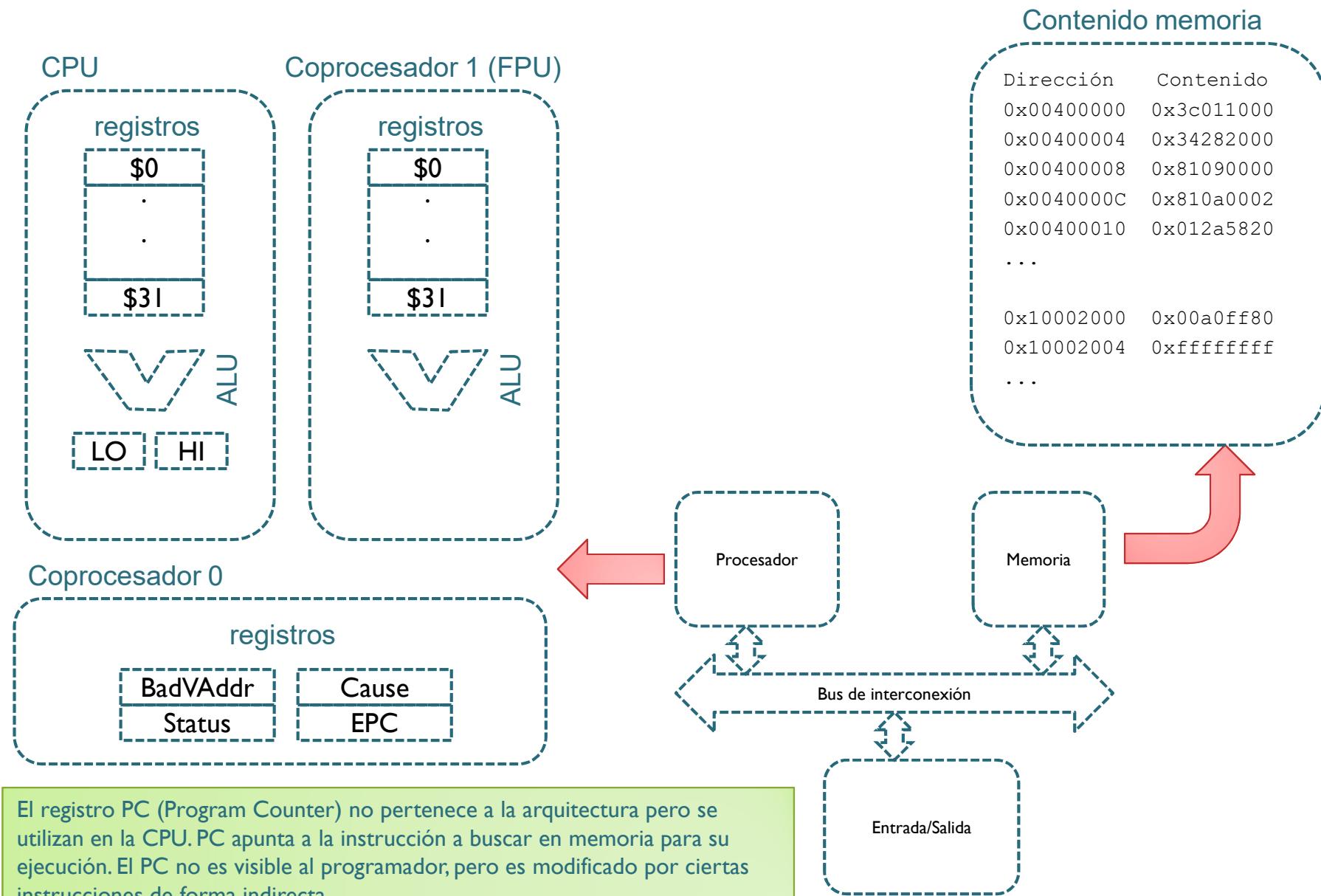
Arquitectura MIPS32: Características Básicas

- Ancho de palabra y tamaño de buses de 32 bits
- Principales tamaño de datos en las instrucciones:
 - ✓ Byte (B), halfword (H), word (W)
- Arquitectura de carga/almacenamiento
 - ✓ Instrucciones específicas de lectura (carga) y escritura (almacenamiento) en memoria
 - ✓ Todos los operandos de una instrucción aritmética se cargan inicialmente en registros, o están en la propia instrucción (constantes)
 - ✓ Instrucciones aritméticas con 3 operandos de 32 bits en registros

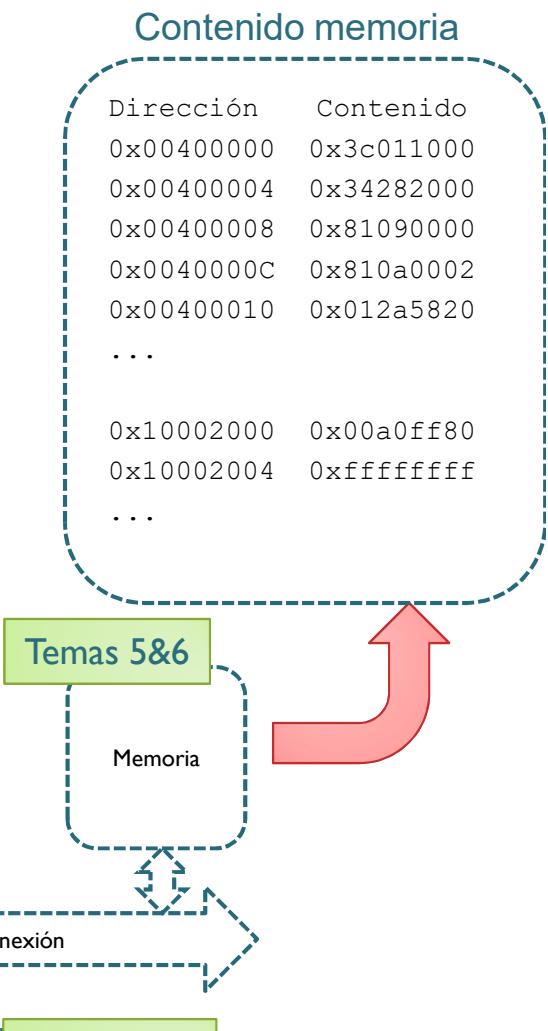
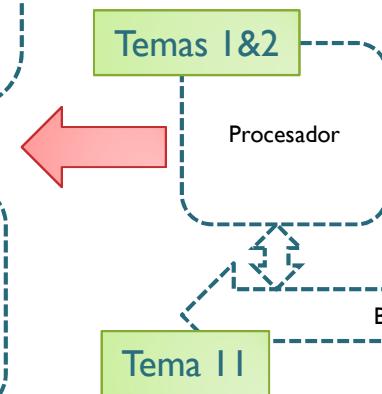
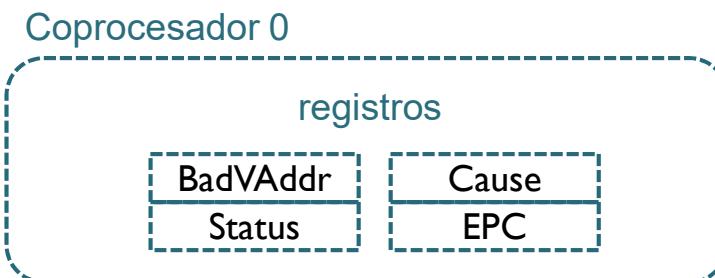
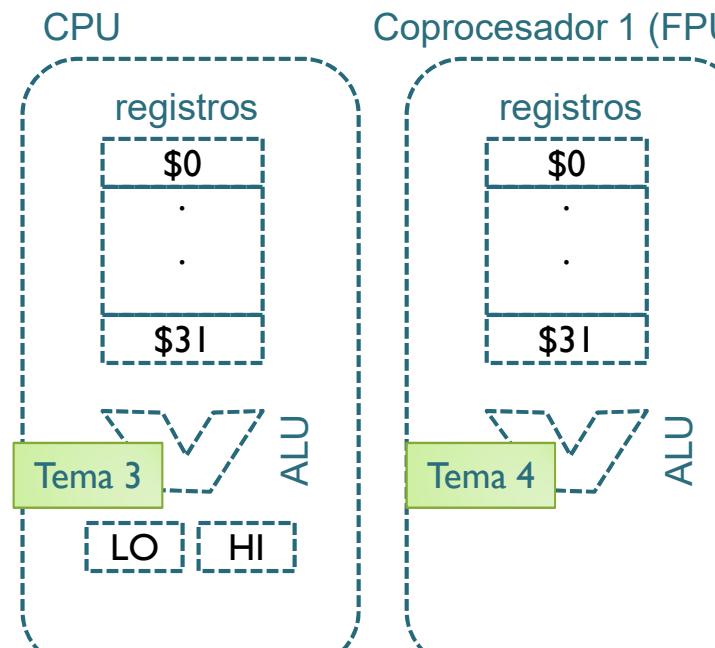
Arquitectura MIPS32: Características Básicas

- Ancho de palabra y tamaño de buses de 32 bits
- Principales tamaño de datos en las instrucciones:
 - ✓ Byte (B), halfword (H), word (W)
- Arquitectura de carga/almacenamiento
 - ✓ Instrucciones específicas de lectura (carga) y escritura (almacenamiento) en memoria
 - ✓ Todos los operandos de una instrucción aritmética se cargan inicialmente en registros, o están en la propia instrucción (constantes)
 - ✓ Instrucciones aritméticas con 3 operandos de 32 bits en registros
- **Modos de funcionamiento: usuario, núcleo (kernel) y depuración (Que veremos en Temas finales del curso)**

Arquitectura MIPS32: Características Básicas



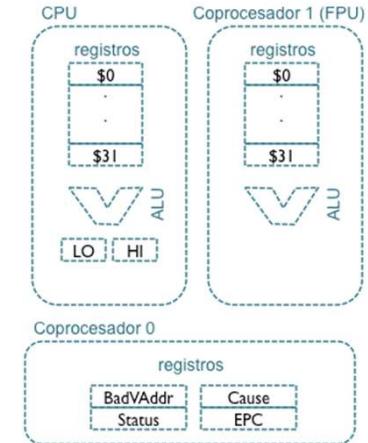
Arquitectura MIPS32 en la asignatura



La arquitectura MIPS32 será utilizada como ejemplo en toda la asignatura, utilizando siempre ejemplos de código en ensamblador del MIPS

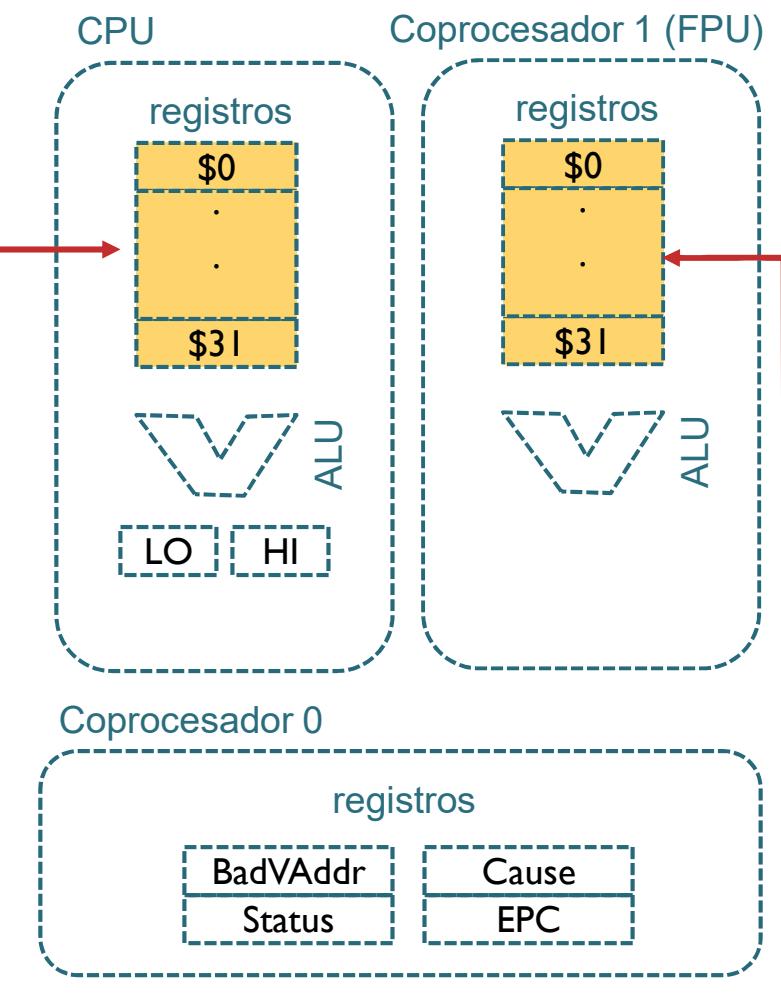
Arquitectura MIPS32: Modelo de programación

- CPU
 - ✓ 32 registros de 32 bits (\$0..\$31)
 - ✓ Registros HI-LO
- Coprocesador 0
 - ✓ Control del sistema (jerarquía, excepciones, modos de ejecución, ...)
 - ✓ 4 registros específicos (hay más)
- FPU (Coprocesador 1, opcional)
 - ✓ 32 registros de 32 bits (\$0..\$31)
 - ✓ Aritmética coma flotante de simple precisión (32 registros)
 - ✓ Aritmética coma flotante de doble precisión (16 parejas de regs)

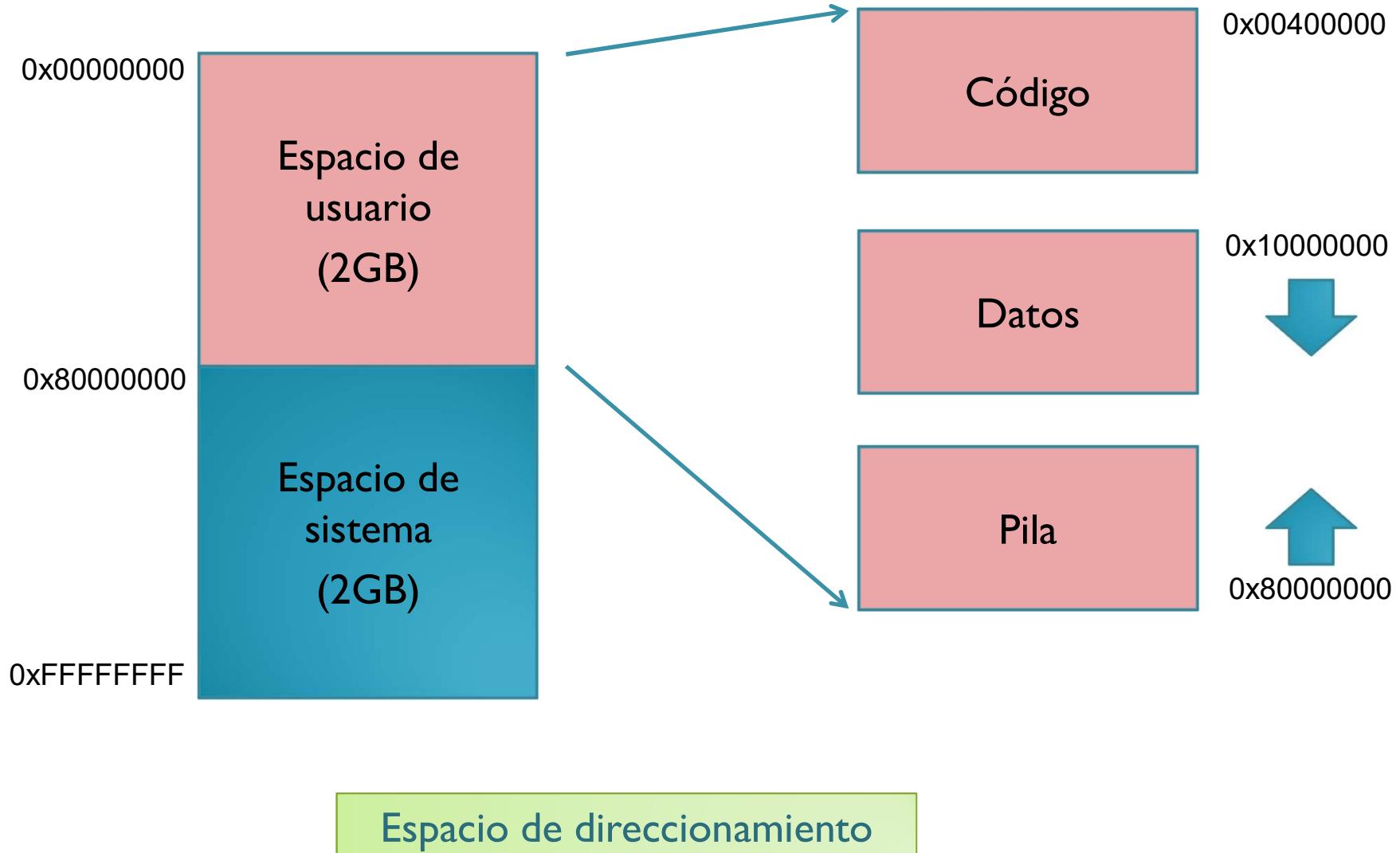


Arquitectura MIPS32: Registros

Nombre	Registro	Uso
\$zero	\$0	Constante 0
\$at	\$1	Registro temporal del ensamblador
\$v0-\$v1	\$2-\$3	Retorno de funciones
\$a0-\$a3	\$4-\$7	Argumentos de funciones
\$t0-\$t7	\$8-\$15	Registros temporales
\$s0-\$s7	\$16-\$23	Temporales salvados
\$t8-\$t9	\$24-\$25	Registros temporales
\$k0-\$k1	\$26-\$27	Utilizados por el SO
\$gp	\$28	Puntero global
\$sp	\$29	Puntero de pila
\$fp	\$30	Puntero de marco (frame)
\$ra	\$31	Dirección de retorno
\$f0-\$f31	\$0..\$31	Registros de coma flotante



Arquitectura MIPS32: Memoria



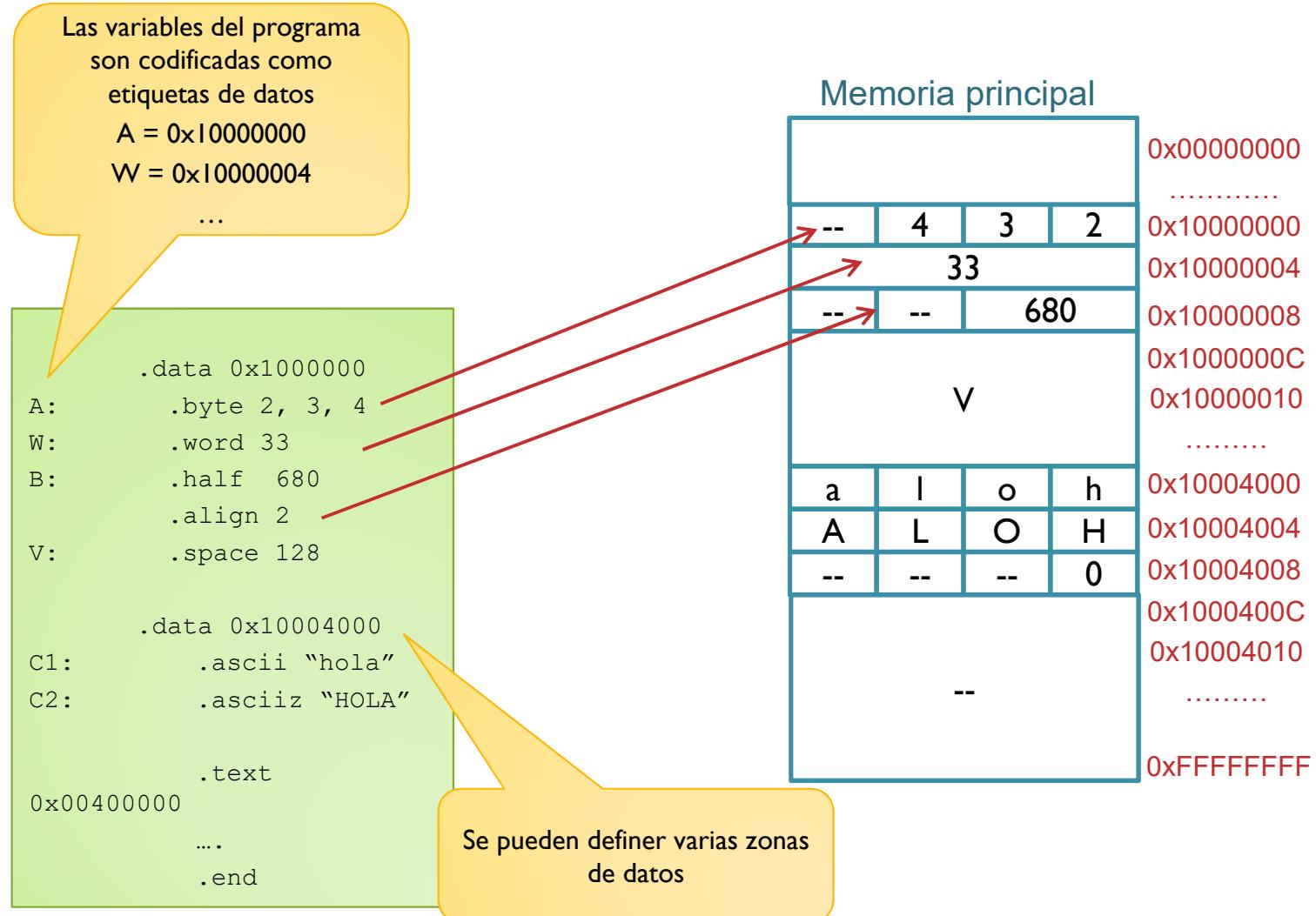
Arquitectura MIPS32: Memoria

- Direccionamiento
 - ✓ Direccionamiento a nivel de byte
 - ✓ Modos big-endian y little-endian soportados
- Alineamiento
 - ✓ Byte en cualquier dirección
 - ✓ Media palabra (half) en dirección par
 - ✓ Palabra (word) en dirección múltiplo de 4
 - ✓ Doble palabra (dword) en dirección múltiplo de 8
- Directiva
 - ✓ `.align N`
 - ✓ Alinea a partir de la siguiente dirección múltiplo de 2^N

Arquitectura MIPS32: Memoria

- Directivas de ensamblador
 - ✓ Directivas de segmentos de memoria
 - **.data [dirección]**
 - **.text [dirección]**
 - **.end**
 - ✓ Directivas de reserva de memoria de datos
 - **.space n**
 - **.byte b1 [, b2] ...**
 - **.half b1 [, b2] ...**
 - **.word b1 [, b2] ...**
 - **.ascii cadena1 [, cadena2] ...**
 - **.asciiz cadena1 [, cadena2]...**

Arquitectura MIPS32: Alineamiento en memoria



Arquitectura MIPS32: Instrucciones

- Grupos de instrucciones

- ✓ Transferencia entre registros
- ✓ Carga/almacenamiento
- ✓ Aritméticas
- ✓ Lógicas
- ✓ Comparación
- ✓ Salto
- ✓ Coma flotante
- ✓ Otras

Las instrucciones serán ejercitadas en las sesiones de laboratorio.

Instrucciones vs pseudo-instrucciones

Las instrucciones son soportadas por el hardware mientras que las pseudo-instrucciones son traducidas por el ensamblador en un conjunto de instrucciones equivalente

EJEMPLO:

```
la $t1, 0x10002000 -> lui $1, 0x1000  
                                ori $t1, $1, 0x2000
```

- Sintaxis y codificación

- ✓ <http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html>
- ✓ http://en.wikipedia.org/wiki/MIPS_architecture

Introducción

- I – Arquitectura MIPS32

- ✓ Introducción y Definición
- ✓ Carácterísticas básicas
- ✓ Ejemplo de ejecución

AYUDA EN: Recursos\ Grupos\F\Tema1\videos:

- Vídeo 2. “Ejemplo Carga/Ejecución MIPS”
- Vídeo 6: “Codificación, Ensamblado y Carga de un programa en ensamblador”

AYUDA EN: Recursos\ Temas \ T1 –La ruta de los datos \ Presentaciones de clases

- Tema 1.4. Codificacion, Ensamblado y Carga.ppsx
- Tema 1.5. Ejecución.ppsx

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores.

La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4)

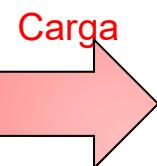
Arquitectura MIPS32: ejemplo de ejecución

```

.data 0x10002000
a: .byte 0x80, 0xFF, 0xA0, 0x00
b: .word -1

.text 0x00400000
lui $1, 4096
ori $8, $1, 8192
lb $9, 0($8)
lb $10, 2($8)
add $11, $9, $10
sb $11, 3($8)
add $9, $0, $0

```



Contenido memoria

Dirección	Contenido
0x00400000	0x3c011000
0x00400004	0x34282000
0x00400008	0x81090000
0x0040000C	0x810a0002
0x00400010	0x012a5820
0x00400014	0xa10b0003
0x00400018	0x00004820
...	
0x10002000	0x00a0: 0000 0001 0010 1010 0101 1000 0010 0000
0x10002004	0xffff: Instrucción con formato R (add \$11, \$9, \$10)
...	

Instrucción con formato R (add \$11, \$9, \$10)
 C.op: bits 31-26: 000000
 Rs: bits 25-21: 01001 -> \$9
 Rt: bits 20-16: 01010 -> \$10
 Rd: bits 15:11: 01011 -> \$11
 Func: bits 5-0: 100000 -> add

Pseudo instrucción

```

.data 0x10002000
a: .byte 0x80, 0xFF, 0xA0, 0x00
c: .word -1

.text 0x00400000
la $t0, a
lb $t1, 0($t0)
lb $t2, 2($t0)
add $t3, $t1, $t2
sb $t3, 3($t0)
add $t1, $0, $0
.end

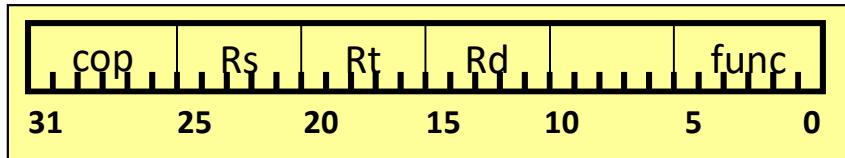
```

Alias en nombre de los registros

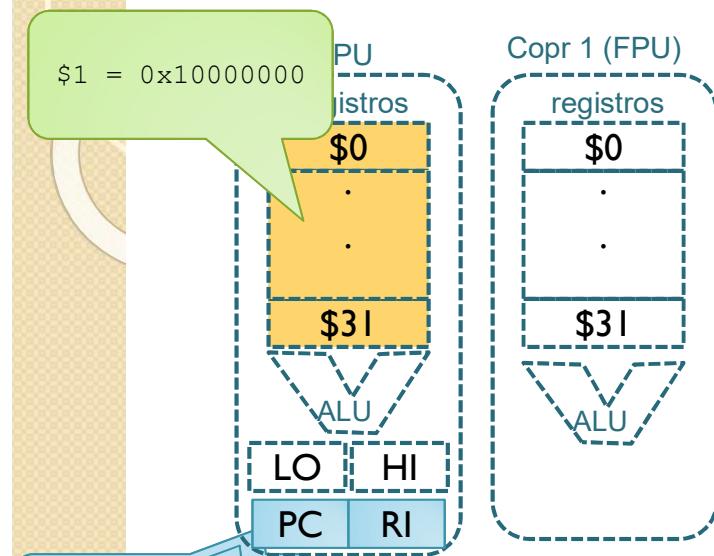
Obtenido con PCSpim

[00400000]	3c011000	lui \$1, 4096 [a]	; 6: la \$t0, a
[00400004]	34282000	ori \$8, \$1, 8192 [a]	
[00400008]	81090000	lb \$9, 0(\$8)	; 7: lb \$t1, 0(\$t0)
[0040000C]	810a0002	lb \$10, 2(\$8)	; 8: lb \$t2, 2(\$t0)
[00400010]	012a5820	add \$11, \$9, \$10	; 9: add \$t3, \$t1, \$t2
[00400014]	a10b0003	sb \$11, 3(\$8)	; 10: sb \$t3, 3(\$t0)
[00400018]	00004820	add \$9,\$0,\$0	; 189: add \$t1, \$0,\$0

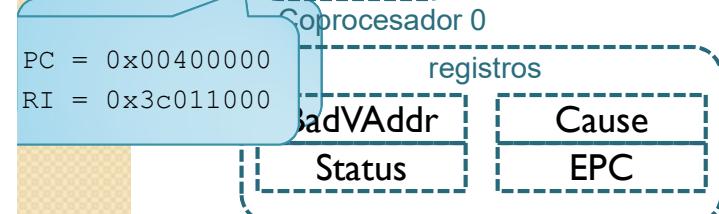
Codif.
Formato R



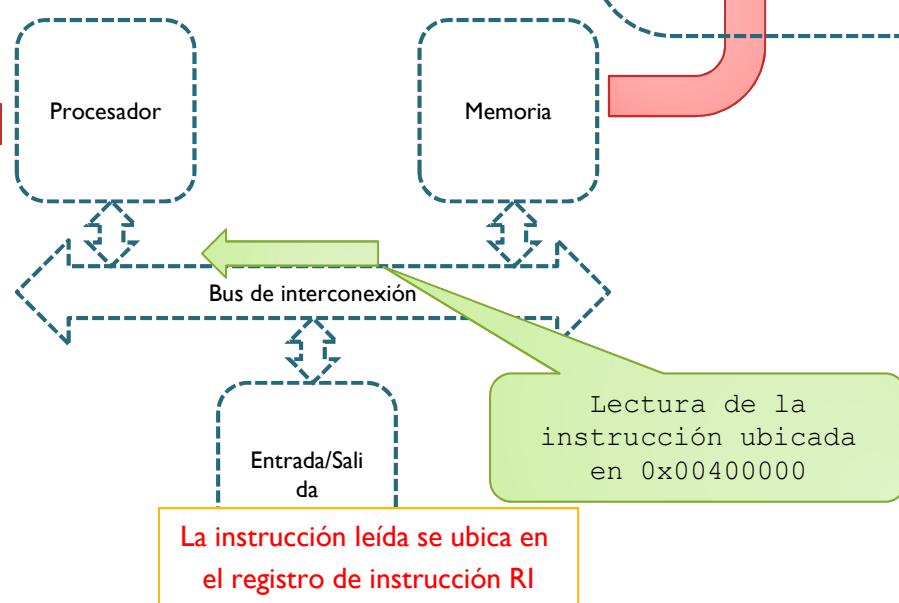
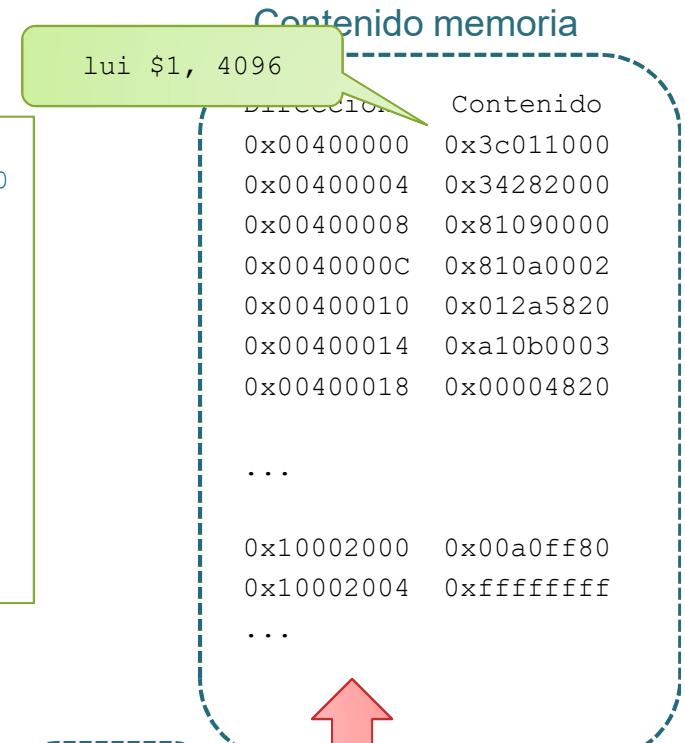
Ejemplo: Ejecución instrucción I



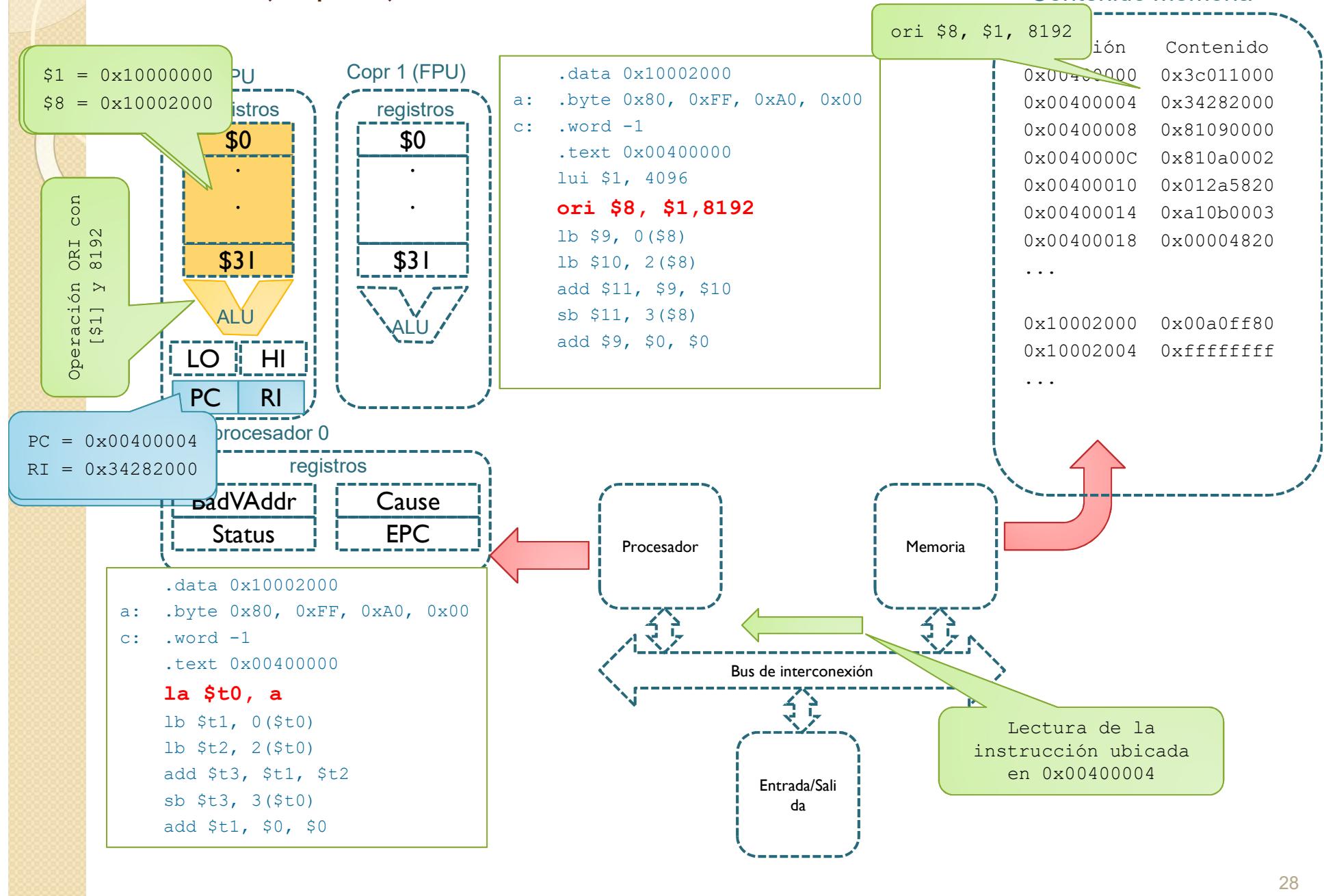
```
.data 0x10002000
a: .byte 0x80, 0xFF, 0xA0, 0x00
c: .word -1
.text 0x00400000
lui $1, 4096
ori $8, $1,8192
lb $9, 0($8)
lb $10, 2($8)
add $11, $9, $10
sb $11, 3($8)
add $9, $0, $0
```



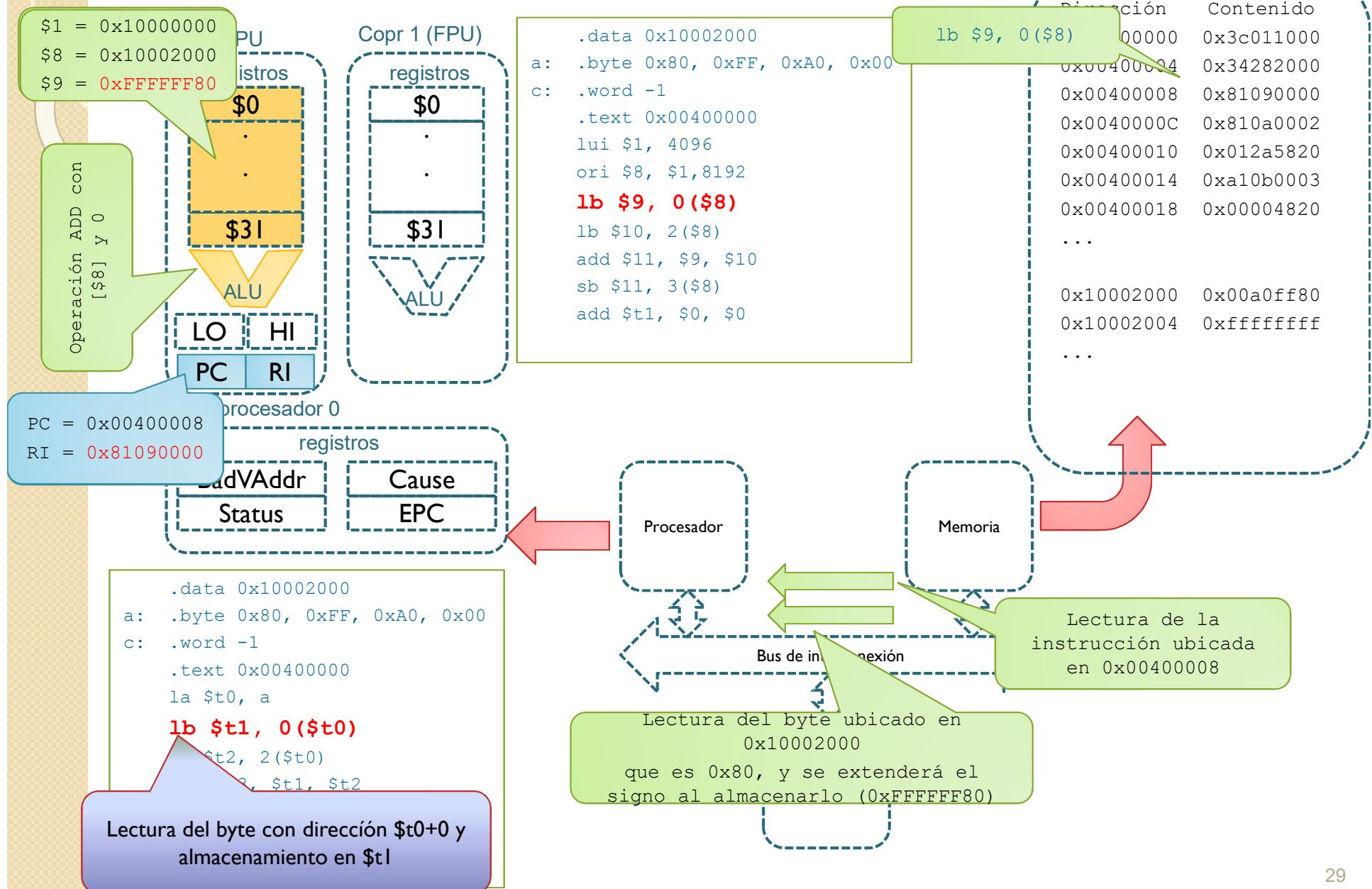
```
.data 0x10002000
a: .byte 0x80, 0xFF, 0xA0, 0x00
c: .word -1
.text 0x00400000
la $t0, a
lb $t1, 0($t0)
lb $t2, 2($t0)
add $t3, $t1, $t2
sb $t3, 3($t0)
add $t1, $0, $0
```



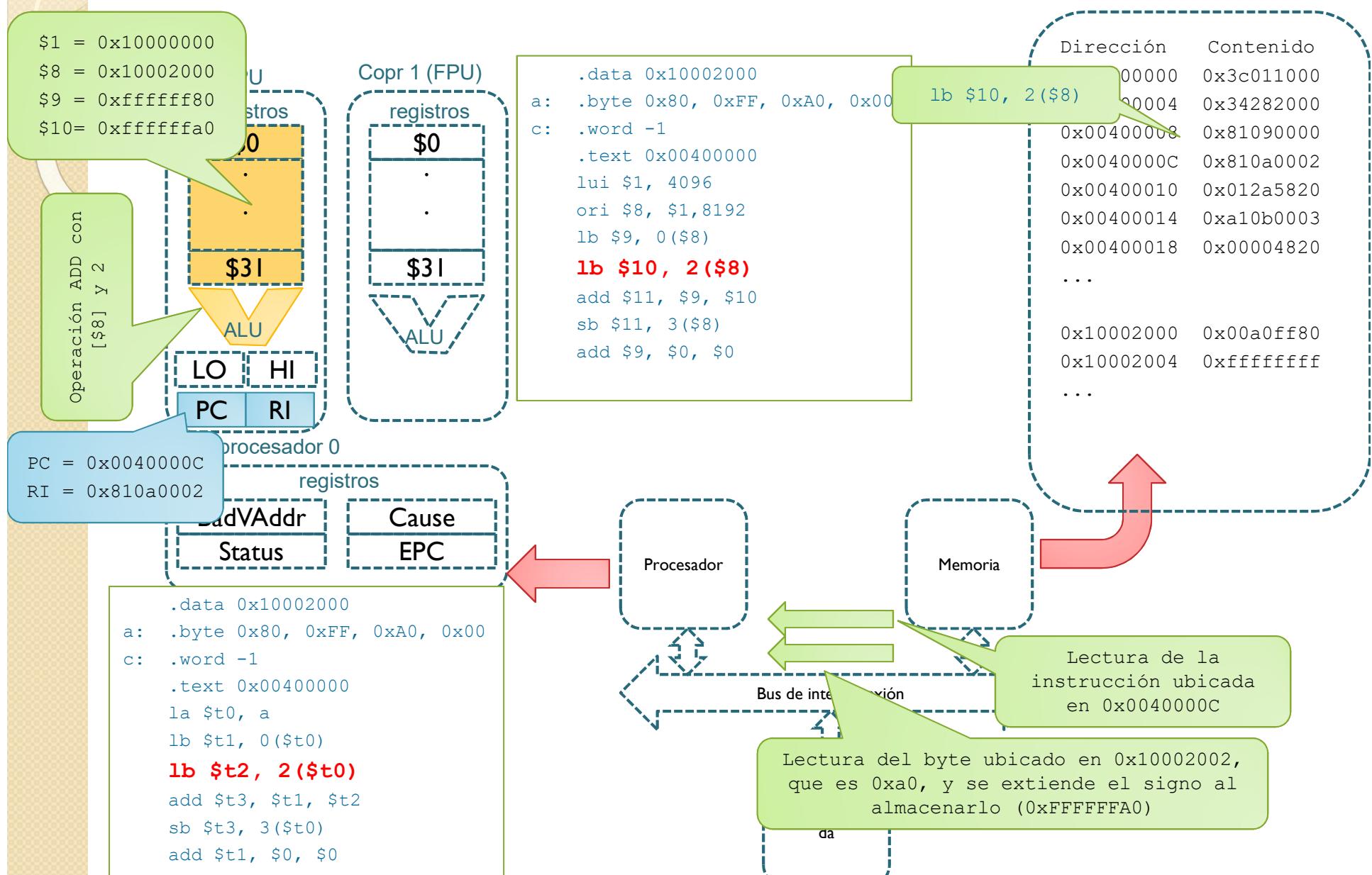
Ejemplo: Ejecución instrucción 2



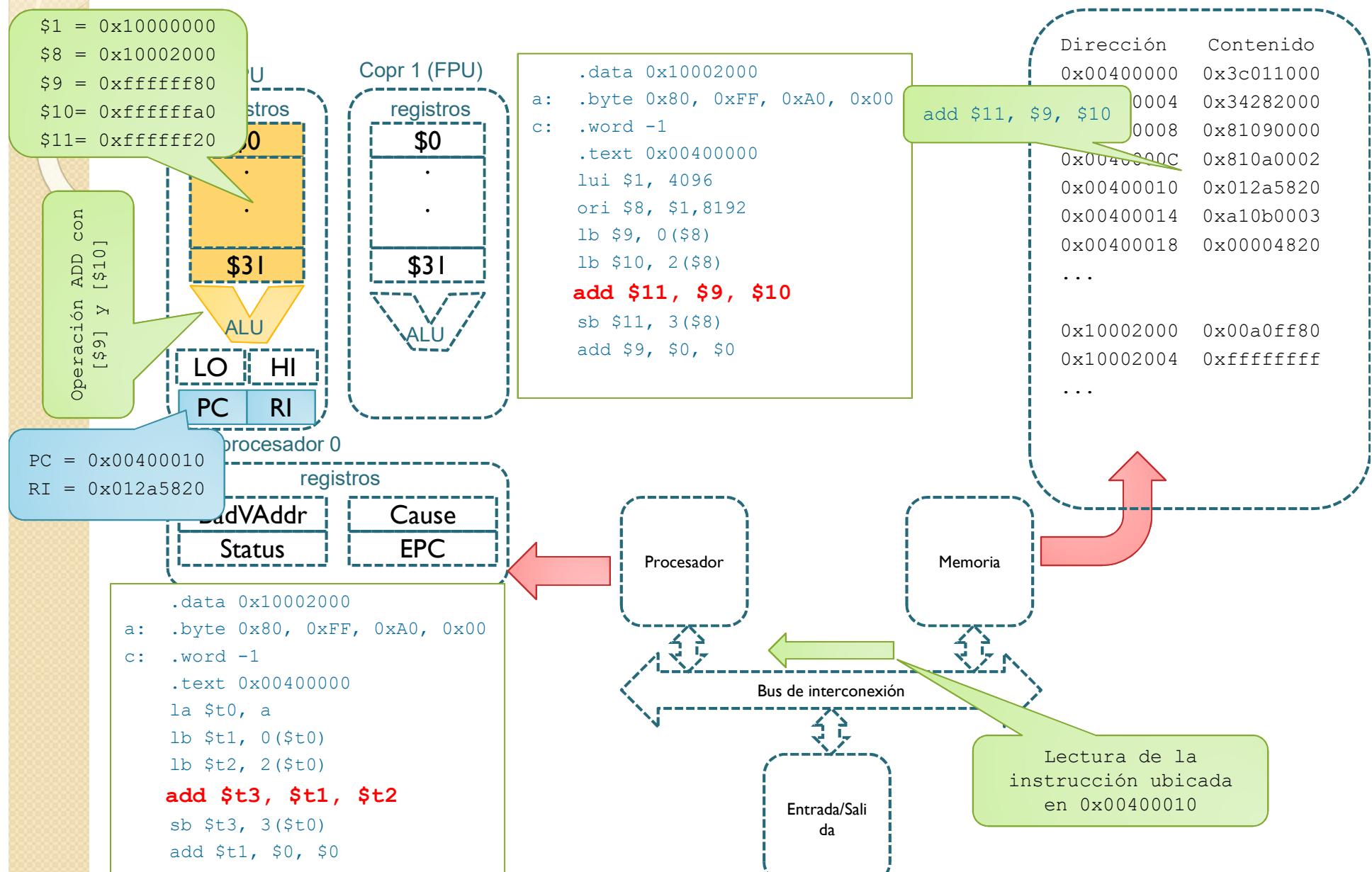
Ejemplo: Ejecución instrucción 3



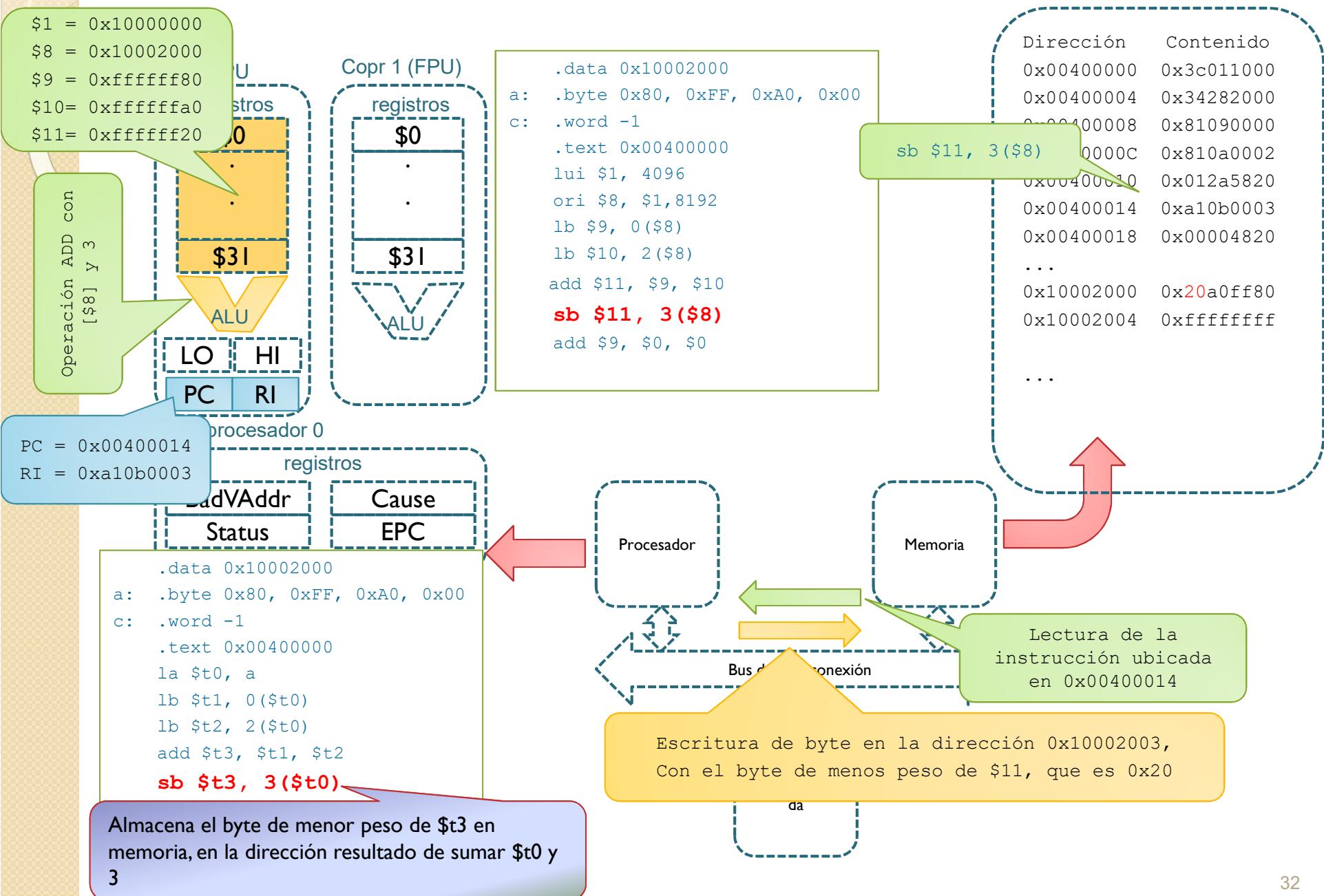
Ejemplo: Ejecución instrucción 4



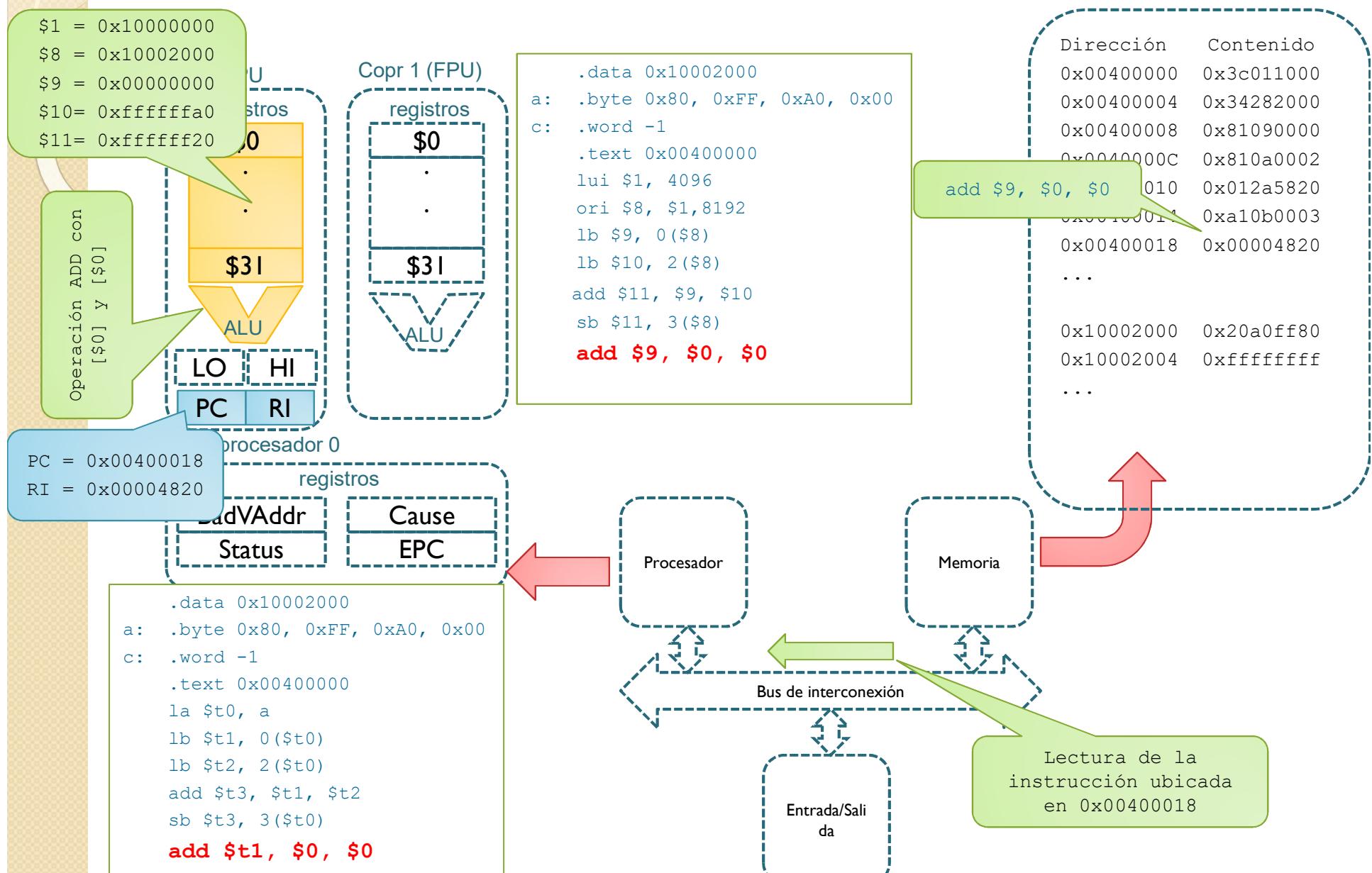
Ejemplo: Ejecución instrucción 5



Ejemplo: Ejecución instrucción 6



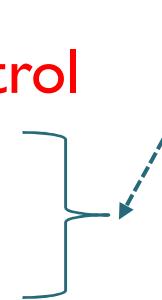
Ejemplo: Ejecución instrucción 7



Contenido y Bibliografía

- I – Arquitectura MIPS32
 - ✓ Introducción y Definición
 - ✓ Características básicas
 - ✓ Ejemplo de ejecución
- 2 – La ruta de datos y la unidad de control
 - ✓ Etapas de búsqueda y decodificación
 - ✓ Diseño de la ruta para aritméticas de tipo R
 - ✓ Diseño de la ruta para aritméticas de tipo I
 - ✓ Diseño de la ruta para instrucciones lw/sw
 - ✓ Diseño de la ruta para instrucciones de salto beq/bne

Si se desea: preparar para clase , visualizando videos 3 y 4



Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4)

Contenido y Bibliografía

- 2 – La ruta de datos y la unidad de control
 - ✓ Etapas de búsqueda y decodificación
 - ✓ Diseño de la ruta para aritméticas de tipo R
 - ✓ Diseño de la ruta para aritméticas de tipo I
 - ✓ Diseño de la ruta para instrucciones lw/sw
 - ✓ Diseño de la ruta para instrucciones de salto beq/bne

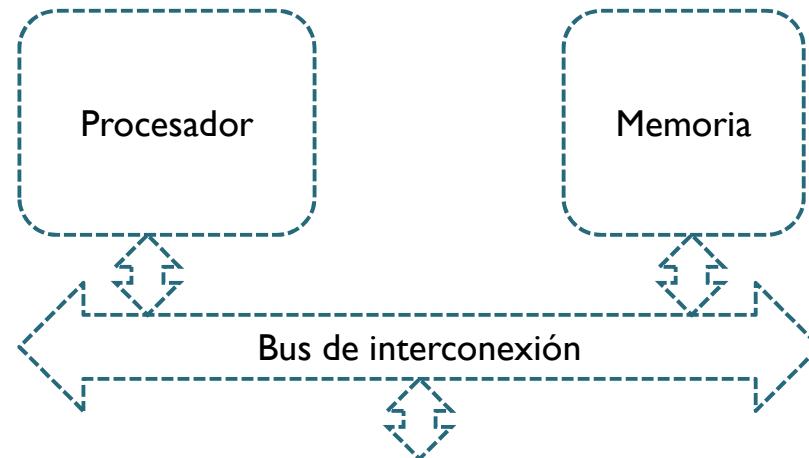
AYUDA EN: Recursos\ Grupos\F\Tema1\videos:

- Vídeo 3. “Ruta de datos: Búsqueda y decodificación”

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011,
Cap 4 (4.1 – 4.4)

La ruta de los datos y la unidad de control

- Diseño del procesador



- ✓ Combinación de elementos lógicos sencillos
 - Circuitos combinacionales: puertas, decodificadores, multiplexores, ALU, ...
 - Circuitos secuenciales: biestables, registros, banco de registros, memoria, ...
- ✓ Las **prestaciones** dependen del número de instrucciones que ejecuta por unidad de tiempo y de la duración del ciclo de reloj
 - **Objetivo inicial: Ejecutar una instrucción por ciclo y reducir el ciclo de reloj**

[Volver](#)

La ruta de los datos y la unidad de control

- Etapas en el diseño
 - ✓ Diseñar la ruta de datos
 - Seleccionar los elementos necesarios para la ejecución de las instrucciones
 - Interconectar los elementos
 - ✓ Diseñar la unidad de control
 - Identificar las señales de control necesarias
 - Diseñar la lógica asociada a cada señal

La ruta de los datos y la unidad de control

- Elección del juego de instrucciones a ejecutar
 - ✓ Subconjunto del juego de instrucciones del MIPS
 - Aritméticas y comparación: add, sub, and, or, slt, addi, slti
 - Carga/almacenamiento: lw, sw
 - Salto: beq, bne **(la instrucción jump será añadida posteriormente)**
- Ciclo único de reloj

AYUDA EN: Recursos\ Grupos\F\Tema1\videos:

Para repasar todas las instrucciones que se van a utilizar y su formatos de codificación se propone ver la siguiente presentación:

- Tema 0. Lenguaje ensamblador.ppsx

Volver



Etapas de búsqueda y decodificación

Todas las instrucciones comienzan con su búsqueda en la memoria. La dirección se encuentra en el PC.

$$RI \leftarrow \text{Mem}[PC]$$

Una vez la instrucción está en RI la unidad de control la decodifica para conocer cuál es y ejecutarla.

- Elementos funcionales necesarios
- Ruta de los datos
- Funcionamiento



Volver
A small teal square with a white arrow pointing to the left.

Elementos funcionales necesarios

Etapas de búsqueda y decodificación

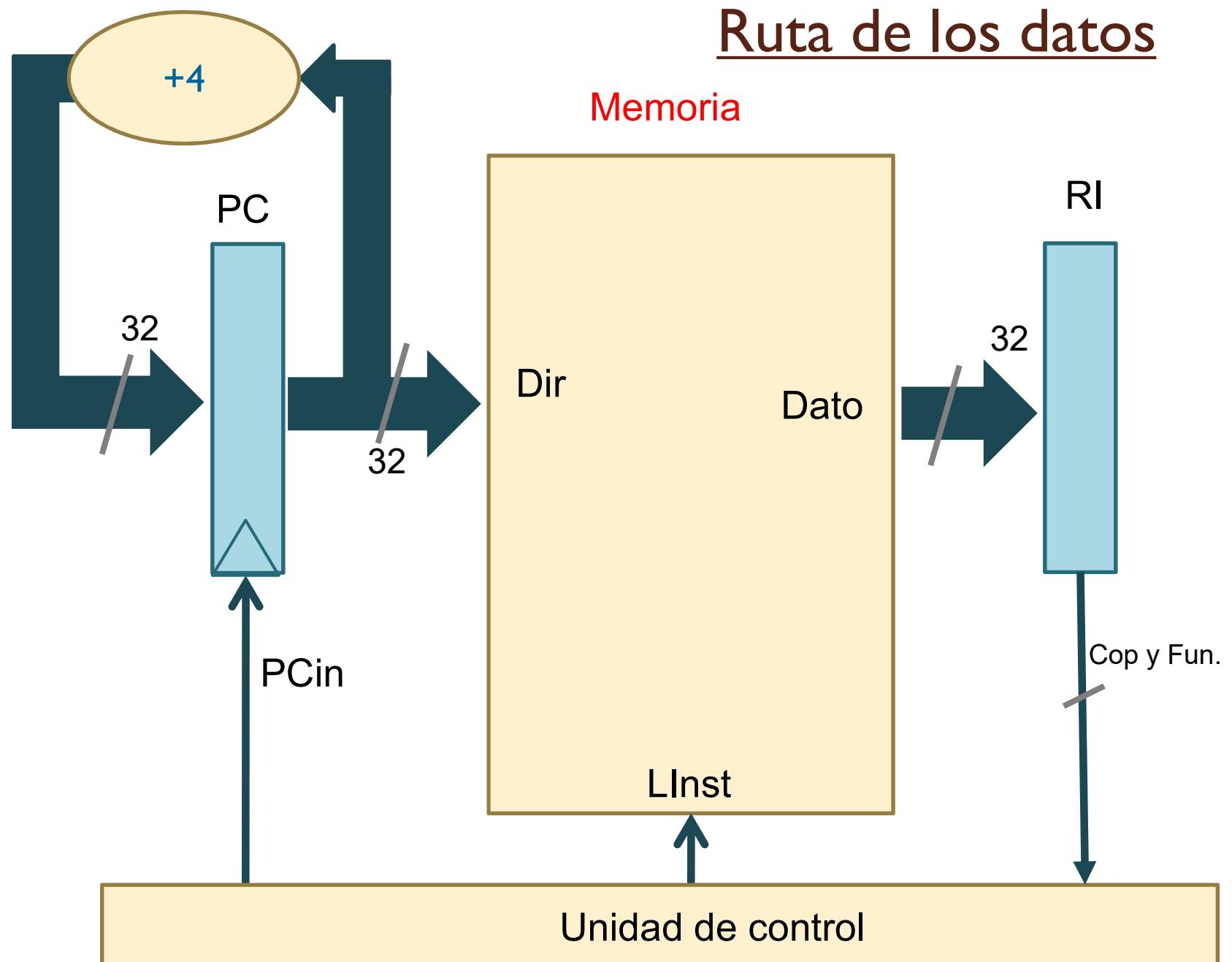
OPERACIONES:

- Leer instrucción empleando PC y almacenar en RI
- Incrementar PC en 4
- Decodificar la instrucción en RI

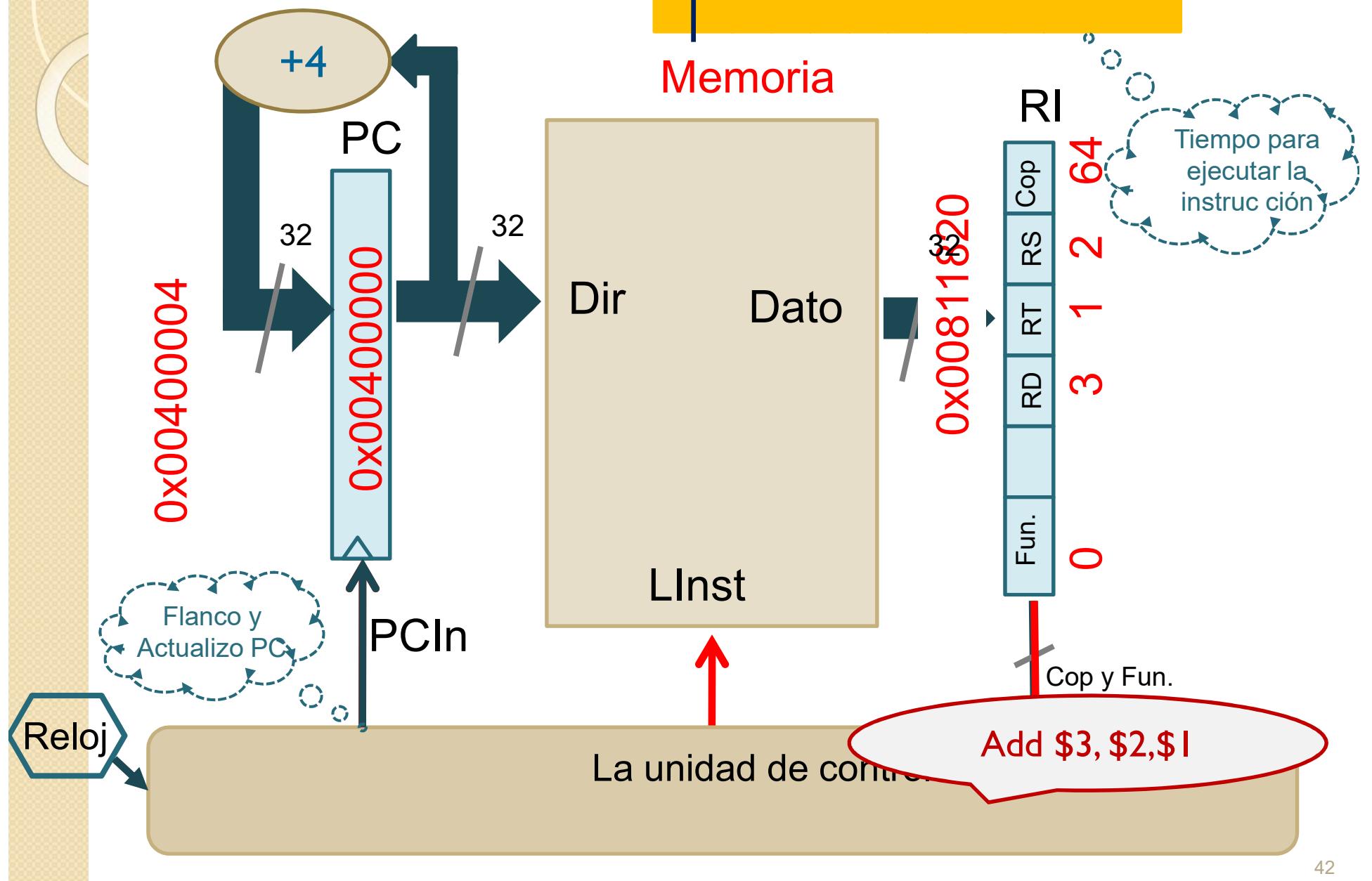
ELEMENTOS:

- Registro PC y RI y memoria de instrucciones.
- Unidad sumadora.
- Campos COP y Función del RI a la unidad de control

Ruta de los datos



Funcionamiento



Contenido y Bibliografía

- **2 – La ruta de datos y la unidad de control**

- ✓ Etapas de búsqueda y decodificación
- ✓ Diseño de la ruta para aritméticas de tipo R
- ✓ Diseño de la ruta para aritméticas de tipo I
- ✓ Diseño de la ruta para instrucciones lw/sw
- ✓ Diseño de la ruta para instrucciones de salto beq/bne

AYUDA EN: Recursos\ Grupos\F\Tema1\videos:

- Vídeo 4: “Ruda Datos tipo R”

Para repasar todas las instrucciones que se van a utilizar y su formatos de codificación se propone ver la siguiente presentación:

- Tema 0. Lenguaje ensamblador.ppsx

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011,
Cap 4 (4.1 – 4.4)

La ruta de los datos aritméticas tipo R

add/sub/and/or/slt \$3, \$1, \$2

COP	RS	RT	RD	FUNC.	
000000	00001	00010	00011	00000	tipo



- ¿Cómo se ejecutan estas instrucciones?

$$RD = RS \text{ operado } RT$$



- Elementos funcionales necesarios



- Ruta de los datos



- Funcionamiento de la ruta ([RutaR.ppsx](#))



- Unidad de control



- Cálculo del tiempo de ciclo

Volver



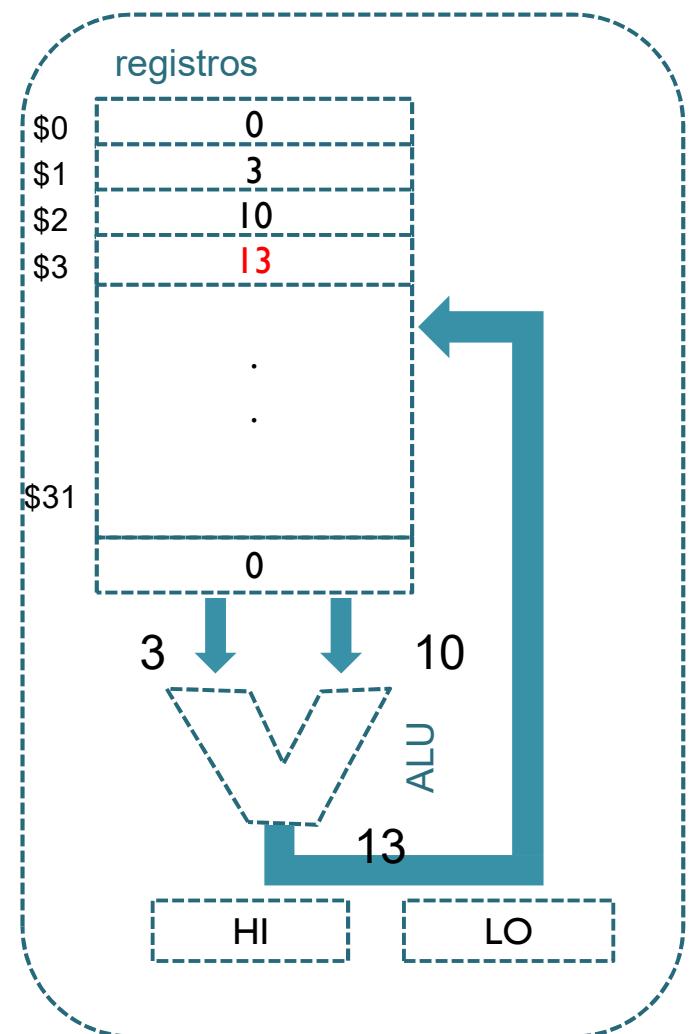
La ruta de los datos aritméticas tipo R

add/sub/and/or/slt \$3, \$1, \$2

COP	RS	RT	RD	FUNC.	
000000	00001	00010	00011	00000	tipo

¿Cómo se ejecutan estas instrucciones?

CPU



add \$3, \$1, \$2

Lectura registros \$1 y \$2

Suma

Escritura resultado en \$3

La ruta de los datos aritméticas tipo R

add/sub/and/or/slt \$3, \$1, \$2

COP	RS	RT	RD	FUNC.	
000000	00001	00010	00011	00000	tipo

Elementos funcionales necesarios

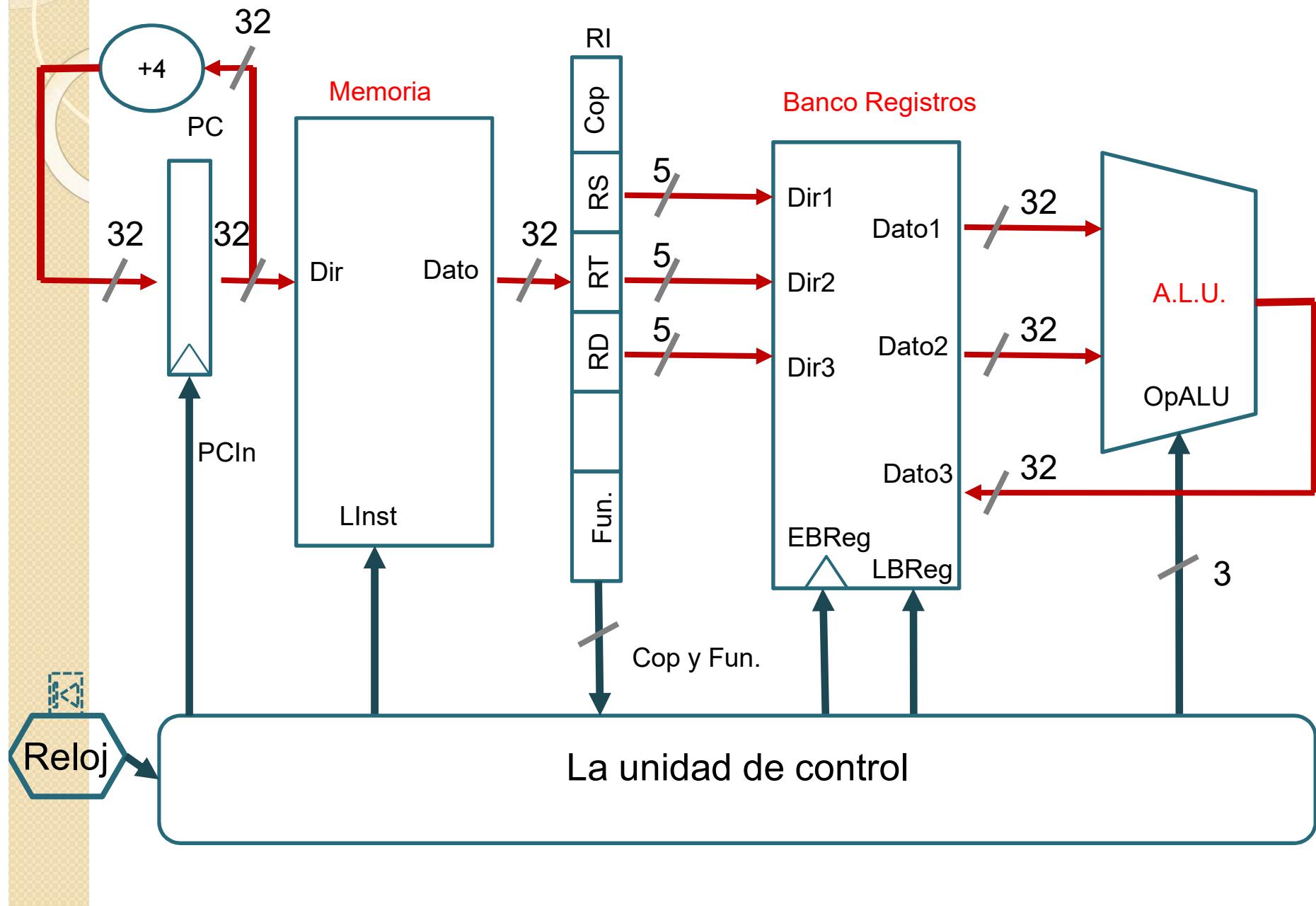
OPERACIONES:

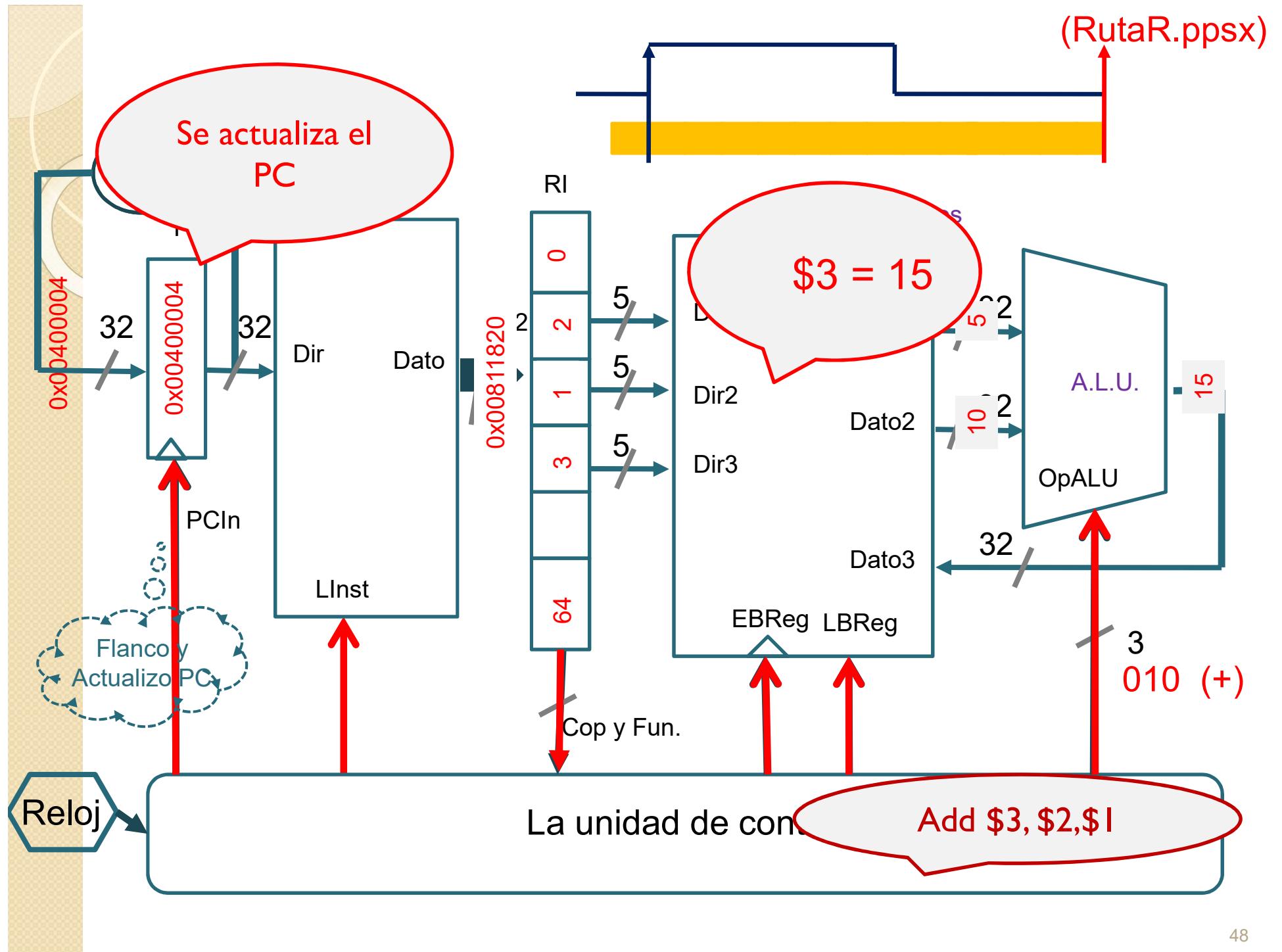
- Leer rs, rt
- Sumar rs y rt
- Guardar en rd dato sumado

ELEMENTOS:

- Banco de registros con dos direcciones de lectura
- Unidad Aritmético y Lógica (ALU en inglés) con las operaciones indicadas
- Banco de registros, con una tercera dirección de escritura

Ruta de los datos aritméticas tipo R





La ruta de los datos aritméticas tipo R

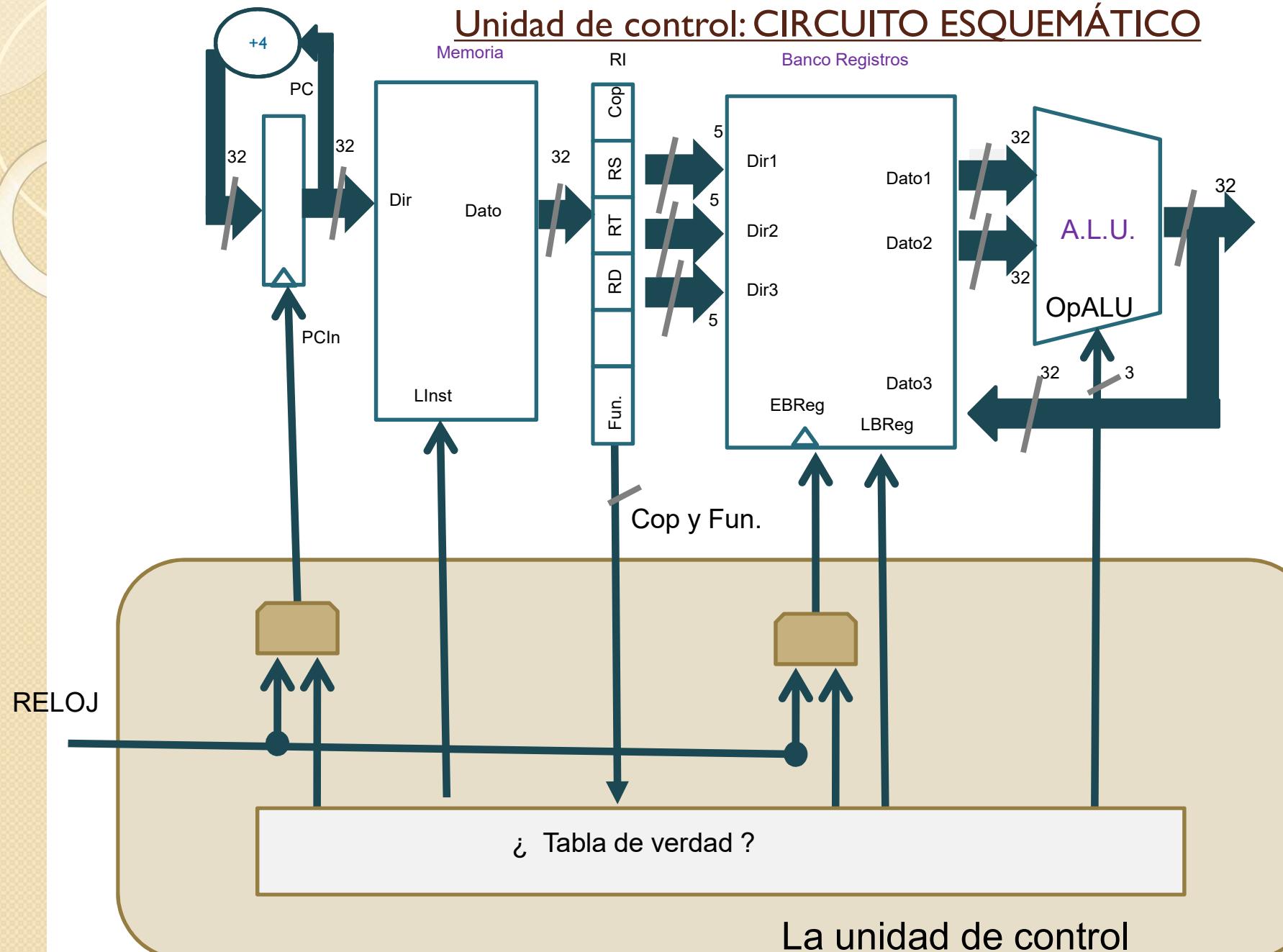
add/sub/and/or/slt \$3, \$1, \$2

COP	RS	RT	RD	FUNC.	
000000	00001	00010	00011	00000	tipo

La unidad de control

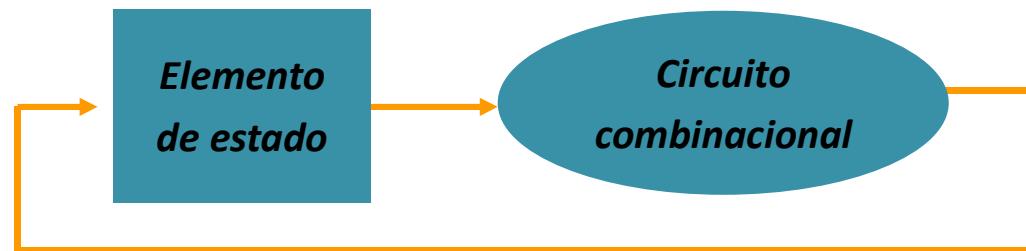
- Circuito esquemático
- ¿Cómo se generan los flancos?
- Tabla de verdad

Unidad de control: CIRCUITO ESQUEMÁTICO

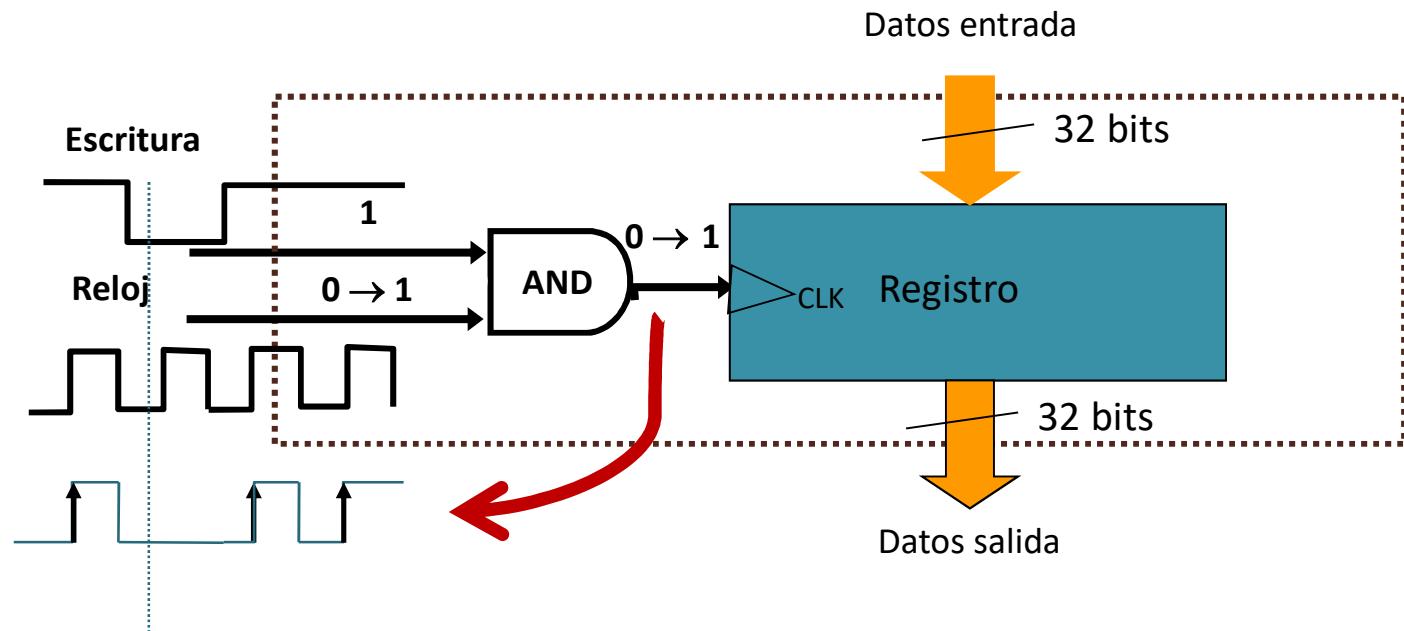


¿Cómo generar flancos en la señal?

- Un elemento puede ser leído y escrito en el mismo ciclo de reloj



- Ejemplo de funcionamiento



Unidad de control: TABLA DE VERDAD

Aritméticas Tipo R: Señales de control

			CP	Mem. Instr.	Banco Registros	ALU	
Instrucción	Cop	Función	PCIn	LInst	LBReg	EBReg	OpALU
add rd, rs, rt	000000	100000	1	1	1	1	010
sub rd, rs, rt	000000	100010	1	1	1	1	110
and rd, rs, rt	000000	100100	1	1	1	1	000
or rd, rs, rt	000000	100101	1	1	1	1	001
slt rd, rs, rt)	000000	101010	1	1	1	1	111



Entradas **Salidas**

OpALU	Operación
000	$a \wedge b$ (and)
001	$a \vee b$ (or)
010	$a + b$ (suma aritmética)
110	$a - b$ (resta)
111	$a < b$ (menor que)

Volver

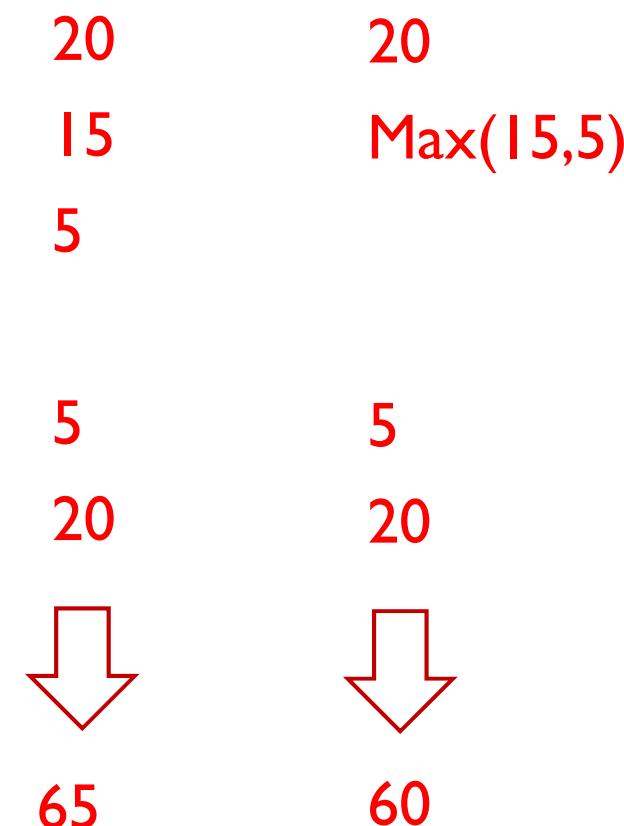

La ruta de los datos aritméticas tipo R

add/sub/and/or/slt \$3, \$1, \$2

COP	RS	RT	RD	FUNC.	
000000	00001	00010	00011	00000	tipo

Cálculo del tiempo de ciclo

- Escritura en registros PC y RI sin coste.
- Lectura en memoria, 20ns.
- Lectura banco de registros, 15ns.
- Unidad de control 5ns en decodificar.
- UAL 5ns todas las operaciones.
- Escritura en banco de registros, 20ns las señales estables a la entrada del banco de registros.



Todas las instrucciones del MIPS leen en el banco de registros, sin saber qué instrucción se está ejecutando, se puede acceder a la vez y ahorrar tiempo

Contenido y Bibliografía

- 2 – La ruta de datos y la unidad de control
 - ✓ Etapas de búsqueda y decodificación
 - ✓ Diseño de la ruta para aritméticas de tipo R
 - ✓ Diseño de la ruta para aritméticas de tipo I
 - ✓ Diseño de la ruta para instrucciones lw/sw
 - ✓ Diseño de la ruta para instrucciones de salto beq/bne

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011,
Cap 4 (4.1 – 4.4)

La ruta de los datos aritméticas tipo I

addi /slti \$3, \$2, 0x1000

COP	RS	RT	Inmediato 16
tipo	00010	00011	0001000000000000

-  ¿Cómo se ejecuta esta instrucción?

RT = RS operando Inmediato extendido a 32b

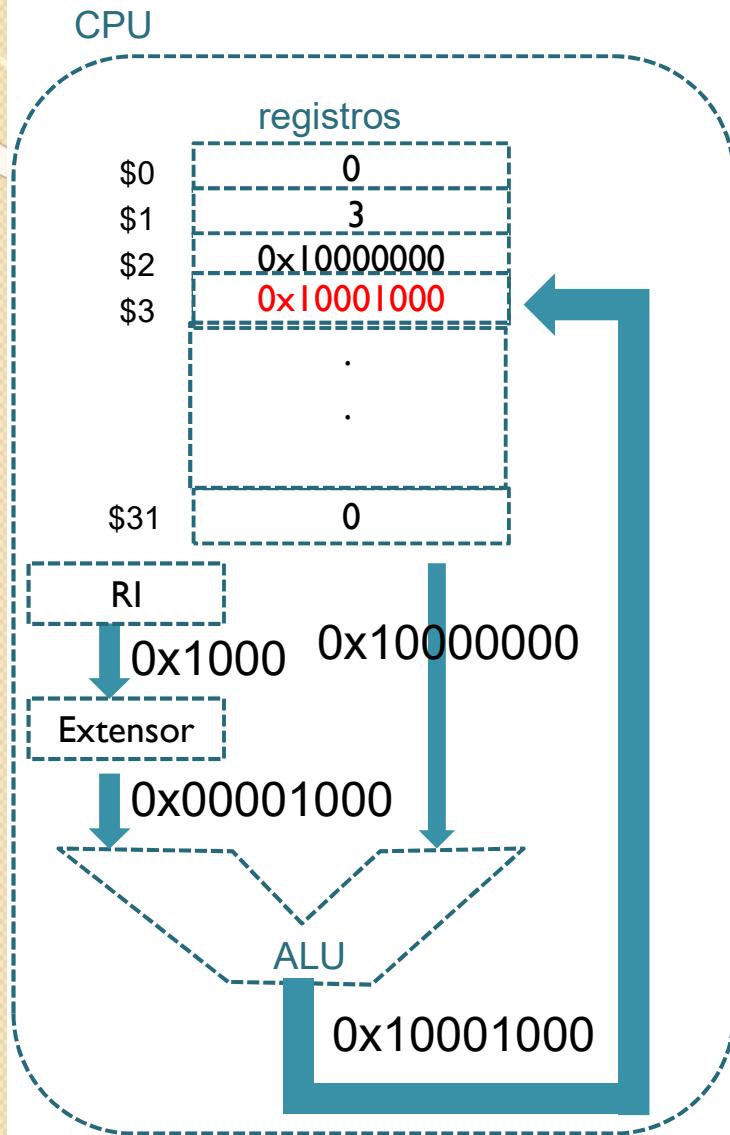
-  Elementos funcionales necesarios
-  Ruta de los datos (Ruta R e I juntas)
-  Funcionamiento de la ruta tipo I (**Rutal.ppsx**)
-  Señales de control (Tabla de verdad)

Volver


La ruta de los datos aritméticas tipo I

addi /slti \$3, \$2, 0x1000			
COP	RS	RT	Inmediato 16
tipo	00010	00011	0001000000000000

¿Cómo se ejecuta esta instrucción?



addi \$3, \$2, 0x1000

Lectura registro \$2 y dato

Suma

Escritura resultado en \$3

La ruta de los datos aritméticas tipo I

addi /slti \$3, \$2, 0x1000

COP	RS	RT	Inmediato 16
tipo	00010	00011	0001000000000000

Elementos funcionales necesarios

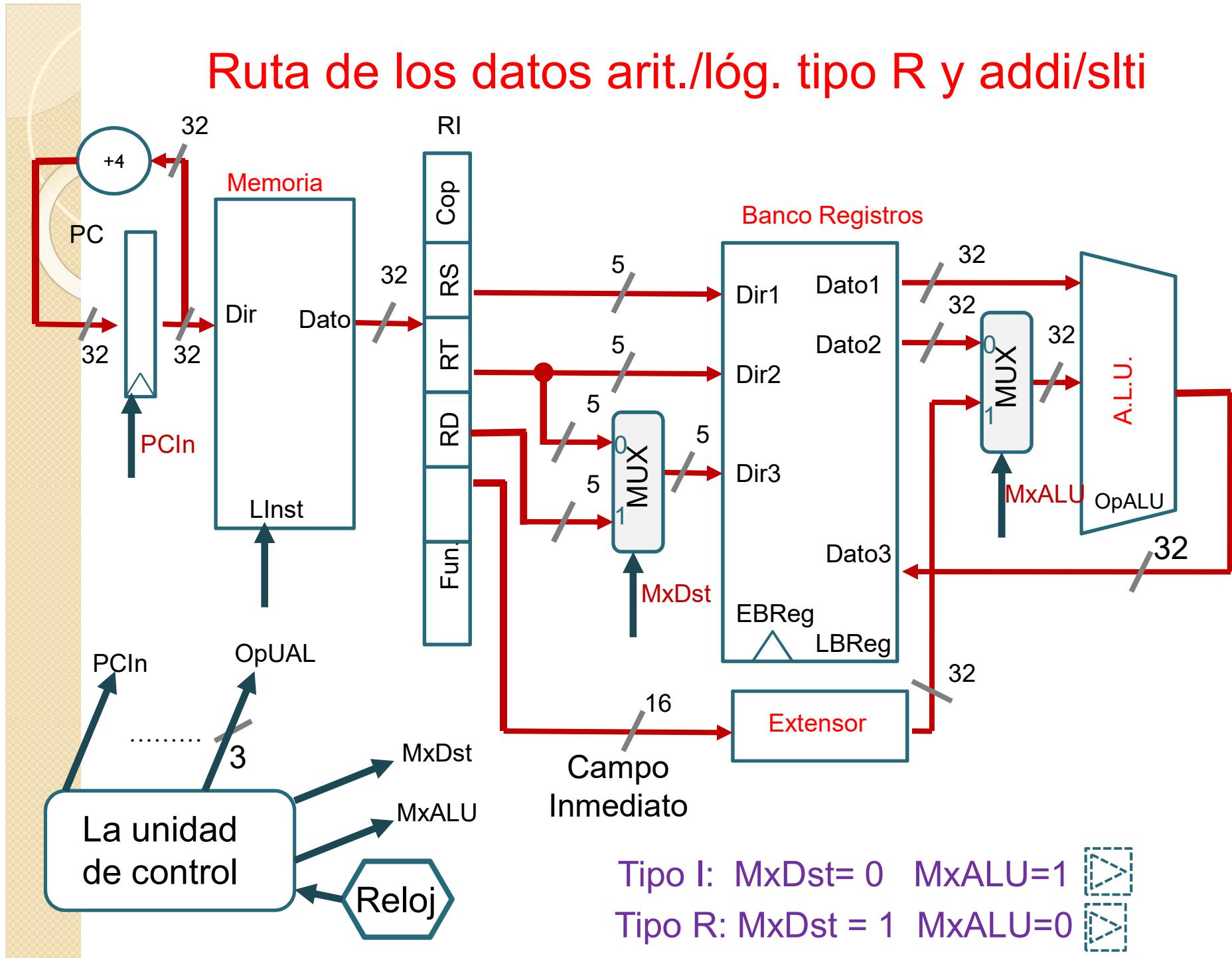
OPERACIONES:

- Leer rs, inmediato 16
- Extender inmediato 16 a 32b
- Realizar la operación
- Guardar en rt dato sumado

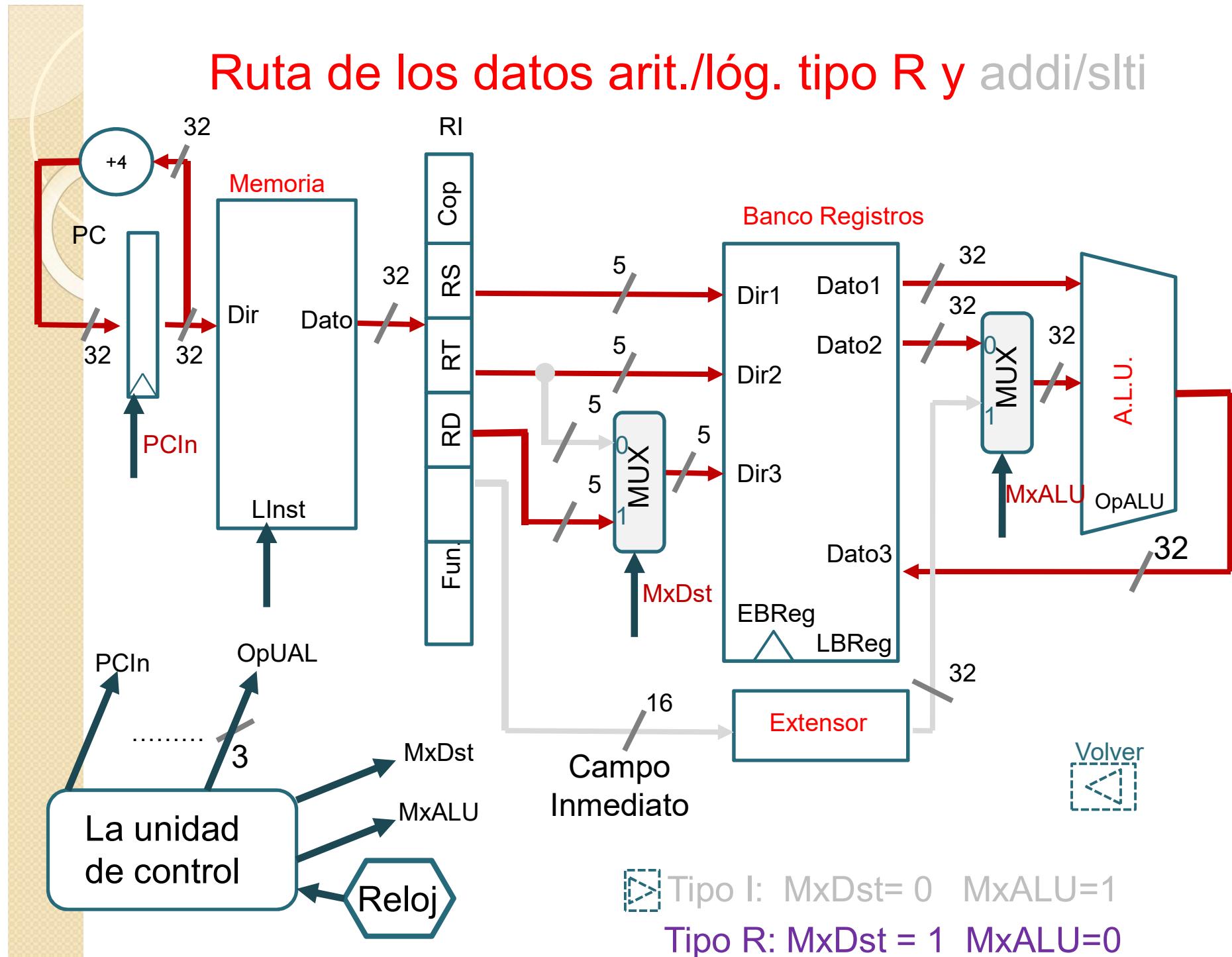
ELEMENTOS:

- Banco de registros (el mismo ruta tipo R)
- Unidad extensora del bit de signo.
- Multiplexor en una entrada de la ALU
- Multiplexores en entradas del Banco de registros.

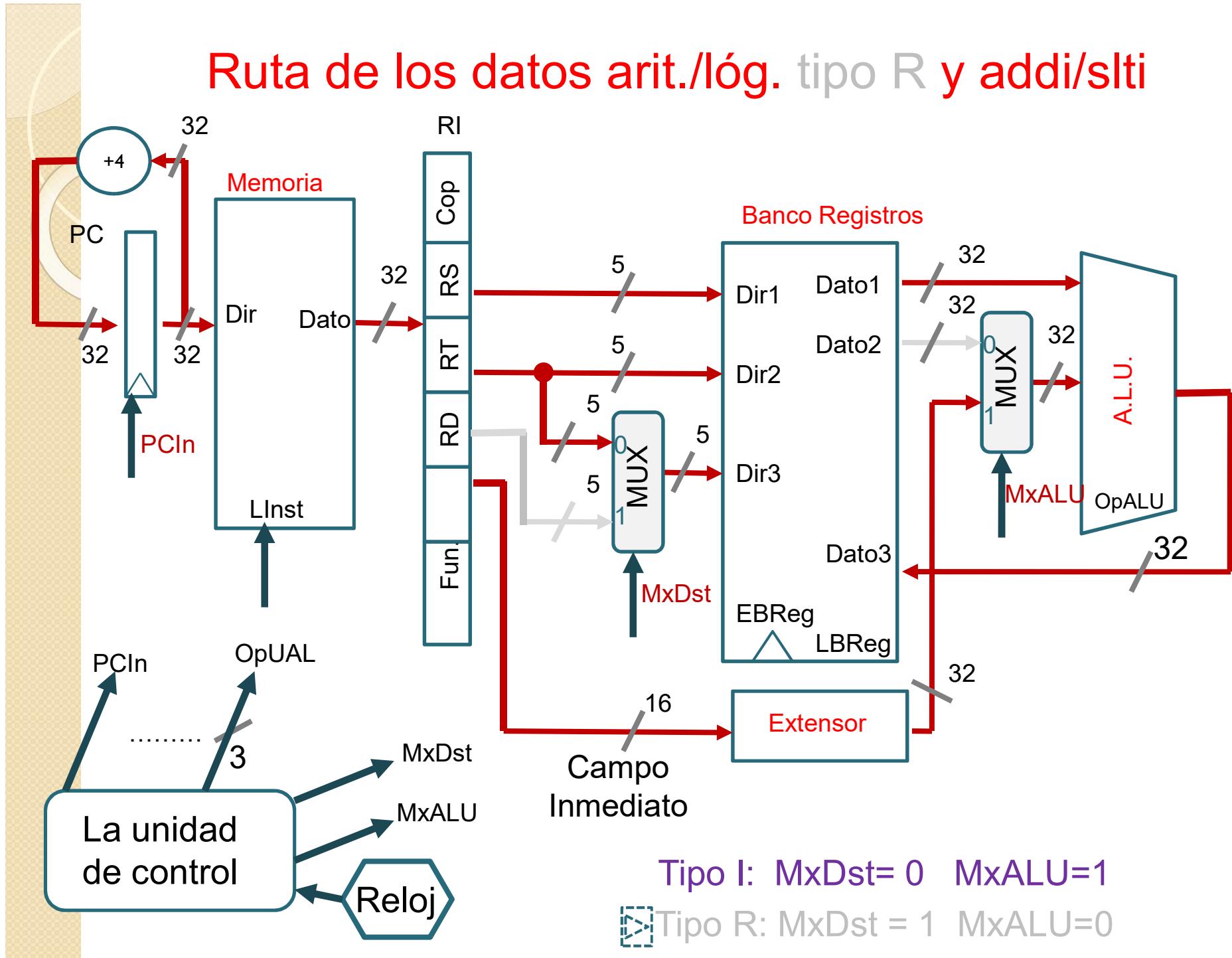
Ruta de los datos arit./lóg. tipo R y addi/slti

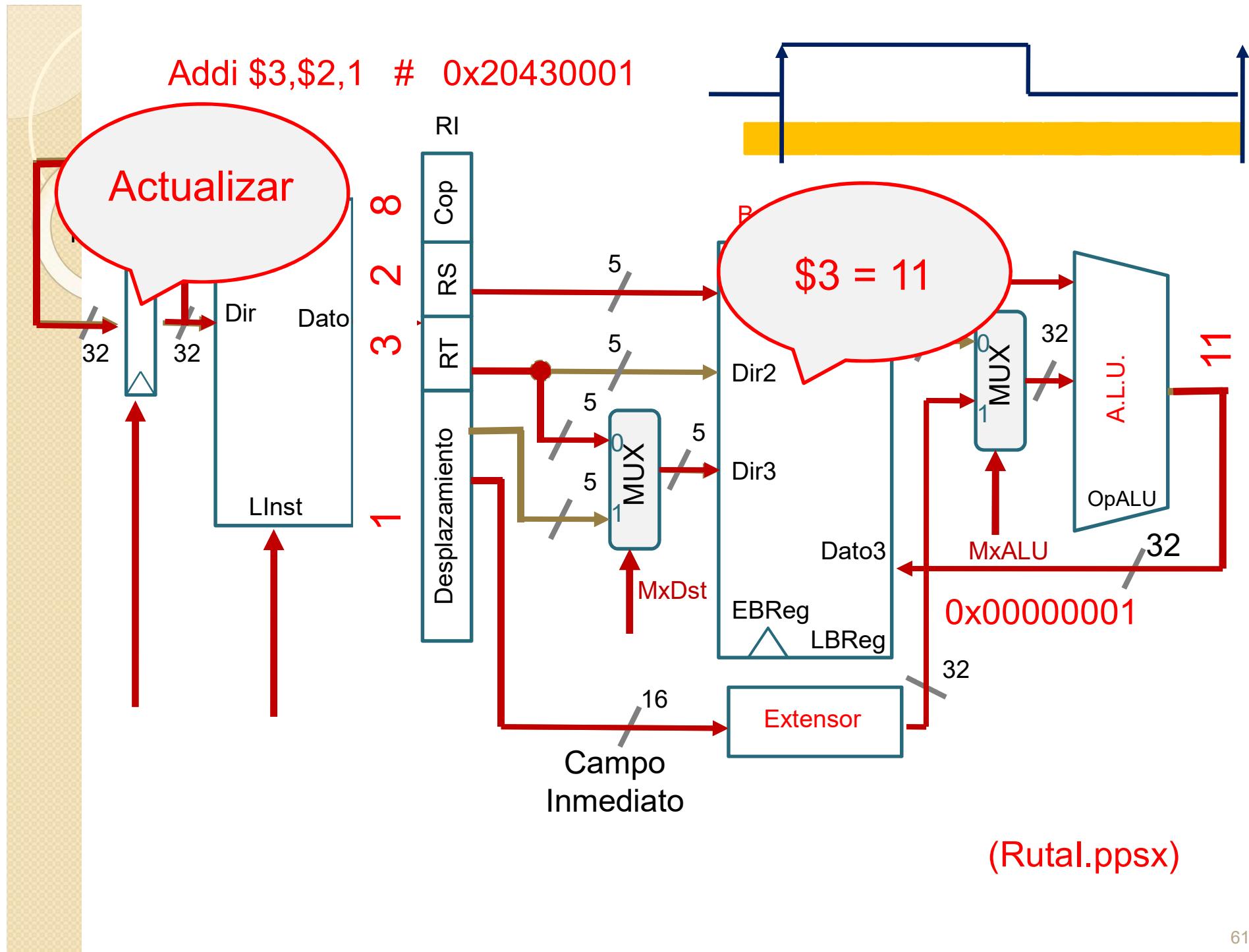


Ruta de los datos arit./lóg. tipo R y addi/slti



Ruta de los datos arit./lóg. tipo R y addi/slti





Aritméticas Tipo I: Señales de control

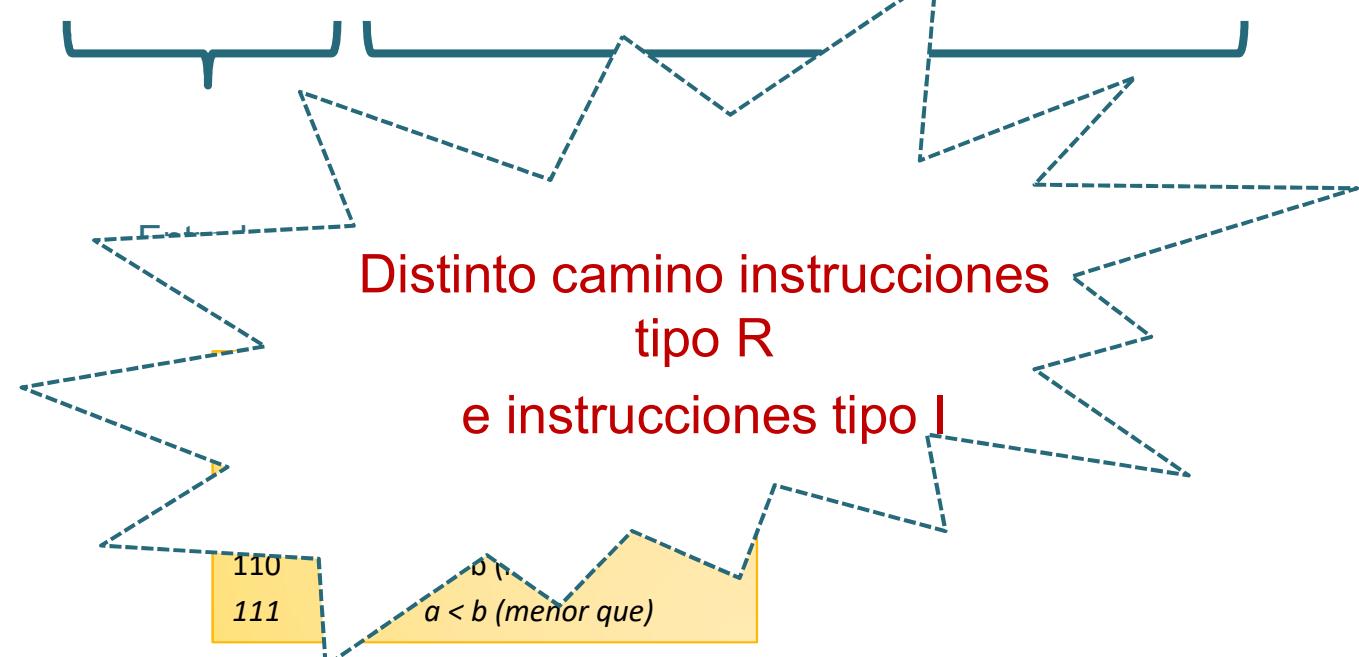
			CP	Mem. Instr.	Banco Registros		ALU	Multiplexores	
Instrucción	Cop	Función	PCIn	LInst	LBReg	EBReg	OpALU	MxALU	MxDst
add rd, rs, rt	000000	100000	1	1	1	1	010	0	1
addi rt, rs, imdto16	001000	-----	1	1	1	1	010	1	0
slti rt, rs, inmdto16	001010	-----	1	1	1	1	111	1	0



OpALU	Operación
000	$a \wedge b$ (and)
001	$a \vee b$ (or)
010	$a + b$ (suma aritmética)
110	$a - b$ (resta)
111	$a < b$ (menor que)

Aritméticas Tipo I: Señales de control

Instrucción	Cop	Función	PCIn	LInst	LBReg	EBReg	OpALU	MxALU	MxDst
add rd, rs, rt	000000	100000	1	1	1	1	010	0	1
addi rt, rs, imdto16	001000	-----	1	1	1	1	010	1	0
slti rt, rs, inmdto16	001010	-----	1	1	1	1	111	1	0



Contenido y Bibliografía

- 1 – Arquitectura MIPS32
 - ✓ Introducción y Definición
 - ✓ Carácterísticas básicas
 - ✓ Ejemplo de ejecución
- 2 – La ruta de datos y la unidad de control
 - ✓ Etapas de búsqueda y decodificación
 - ✓ Diseño de la ruta para aritmético/lógicas de tipo R
 - ✓ Diseño de la ruta para aritmético/lógicas R y aritméticas tipo I
 - ✓ **Diseño de la ruta para instrucciones lw/sw**
 - ✓ Diseño de la ruta para instrucciones de salto beq/bne

Si se desea: preparar para clase visualizando vídeo 5

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4)

Contenido y Bibliografía

• 2 – La ruta de datos y la unidad de control

- ✓ Etapas de búsqueda y decodificación
- ✓ Diseño de la ruta para aritméticas de tipo R
- ✓ Diseño de la ruta para aritméticas de tipo I
- ✓ Diseño de la ruta para instrucciones lw/sw
- ✓ Diseño de la ruta para instrucciones de salto beq/bne

AYUDA EN: Recursos\ Grupos\F\Tema1\videos:

- Vídeo 5: “Añadiendo la instrucción lw a la ruta”

Para repasar todas las instrucciones que se van a utilizar y su formatos de codificación se propone ver la siguiente presentación:

- Tema 0. Lenguaje ensamblador.ppsx

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011,
Cap 4 (4.1 – 4.4)

lw rt, rs, inmdto16 # lw \$3, 0x100(\$2)

COP	RS	RT	desplazamiento
100011	rs	rt	Desplazamiento 16

- ¿Cómo se ejecuta esta instrucción?

$Rt \leftarrow \text{Mem}[Rs + \text{desplazamiento}]$ extendido]

- Elementos funcionales necesarios

- Ruta de los datos (tipo R y tipo I juntas)

- Señales de control (Tabla de verdad)

Volver



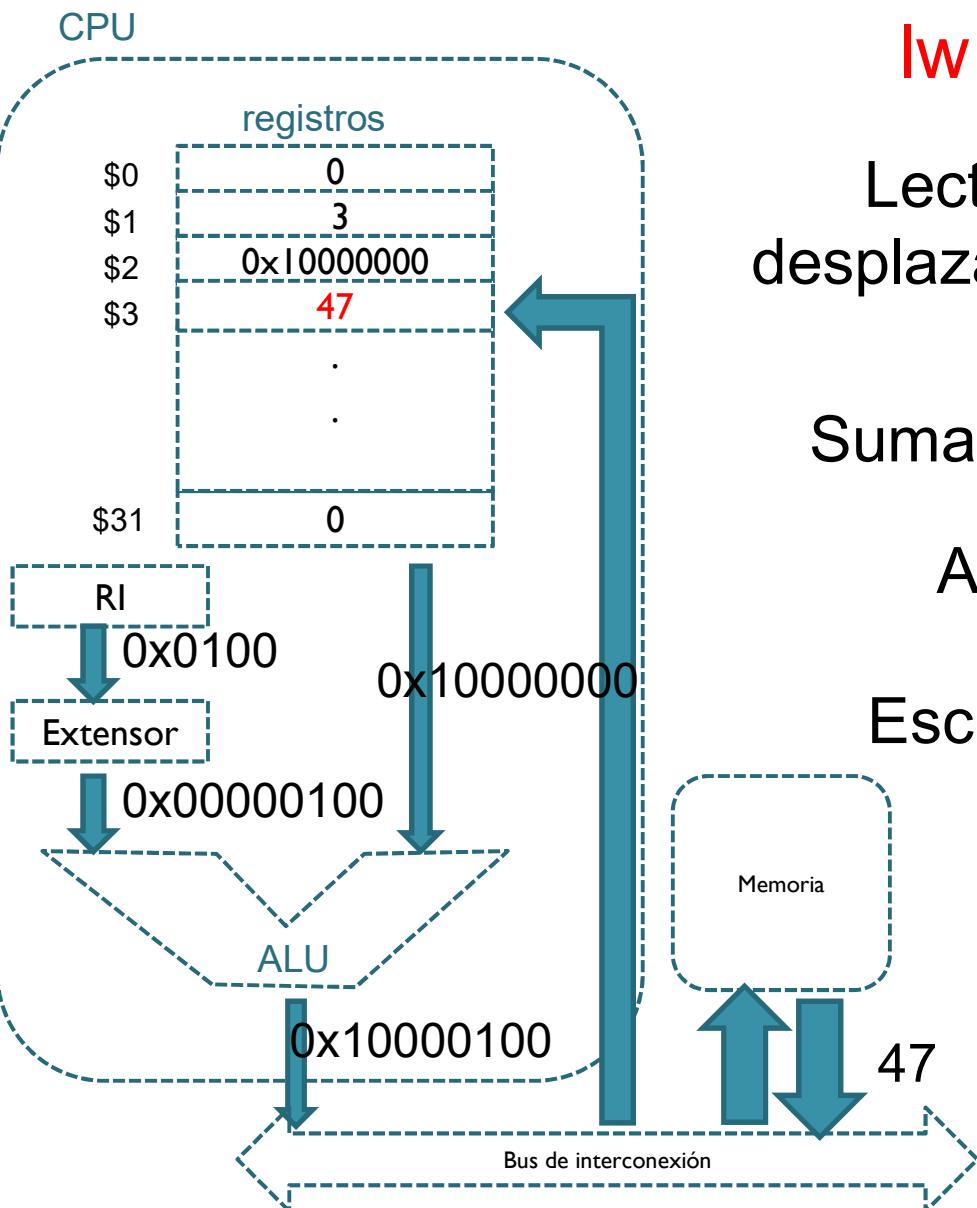
Inst. SW

lw rt, rs, inmdto16

lw \$3, 0x100(\$2)

Rt \leftarrow Mem[Rs + desplazamiento16 extendido]

¿Cómo se ejecuta esta instrucción?



lw \$3, 0x100(\$2)

Lectura registro \$2 (RS) y
desplazamiento 16 extendido a 32
bits

Suma y obtención dirección

Acceso a Memoria

Escritura resultado en \$3 (RT)

lw rt, rs, inmdto16 # lw \$3, 0x100(\$2) Rt → Mem[Rs + desplazamiento16 extendido]

COP	RS	RT	desplazamiento
100011	rs	rt	Desplazamiento 16

Elementos funcionales necesarios

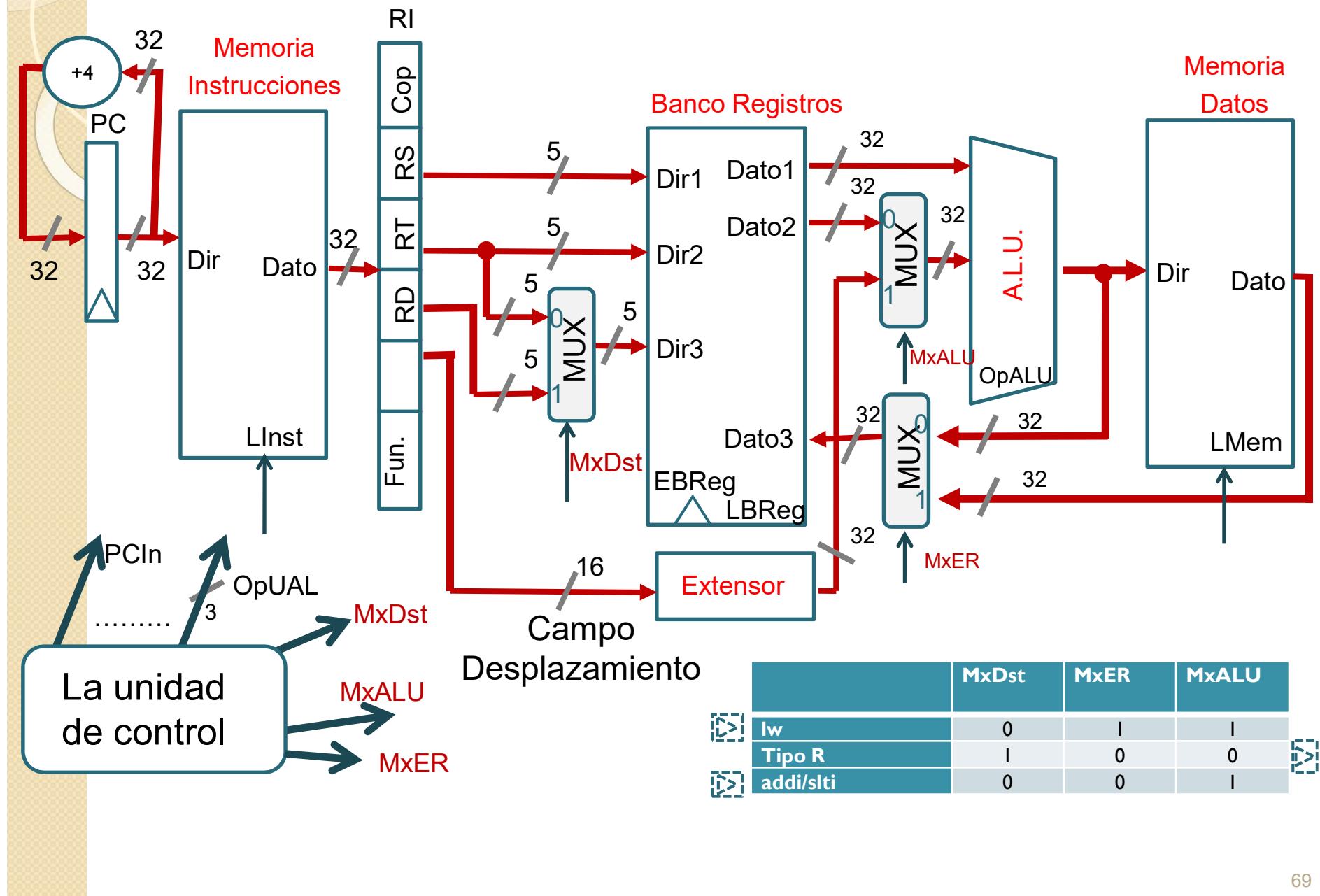
OPERACIONES:

- Leer rs, desplaz16
- Extender Desplazamiento 16 a 32b
- Calcular dirección
- Acceder a memoria
- Guardar en rt dato accedido

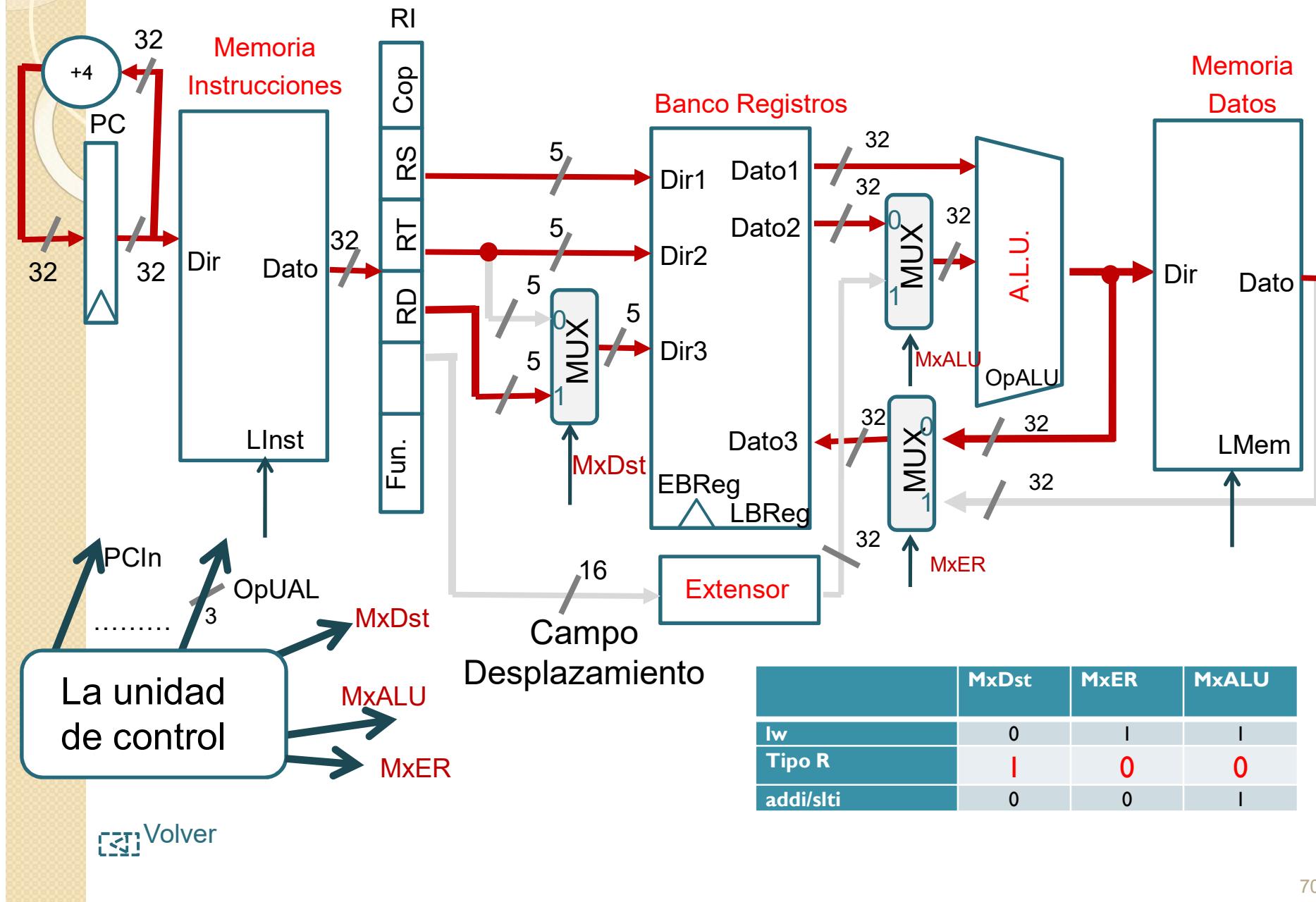
ELEMENTOS:

- Banco de registros
(el mismo ruta tipo R)
- Unidad extensora del bit de signo
(la misma ruta tipo I)
- ALU empleando camino ruta tipo I
- Memoria datos
- Multiplexores en Banco de registros (MxDst como ruta tipo I y UNO NUEVO)

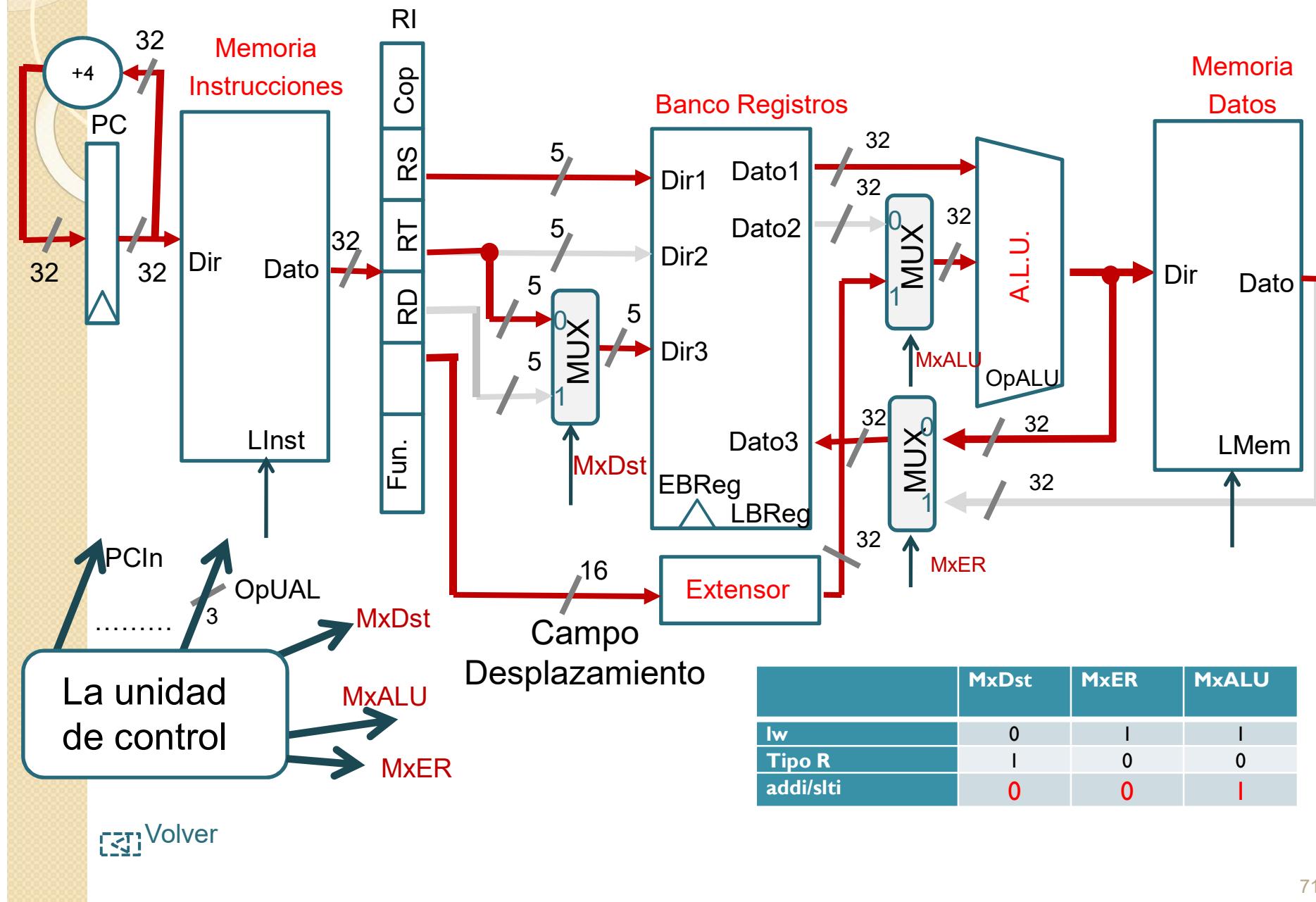
Ruta de los datos arit./lóg. R, addi/slti y lw



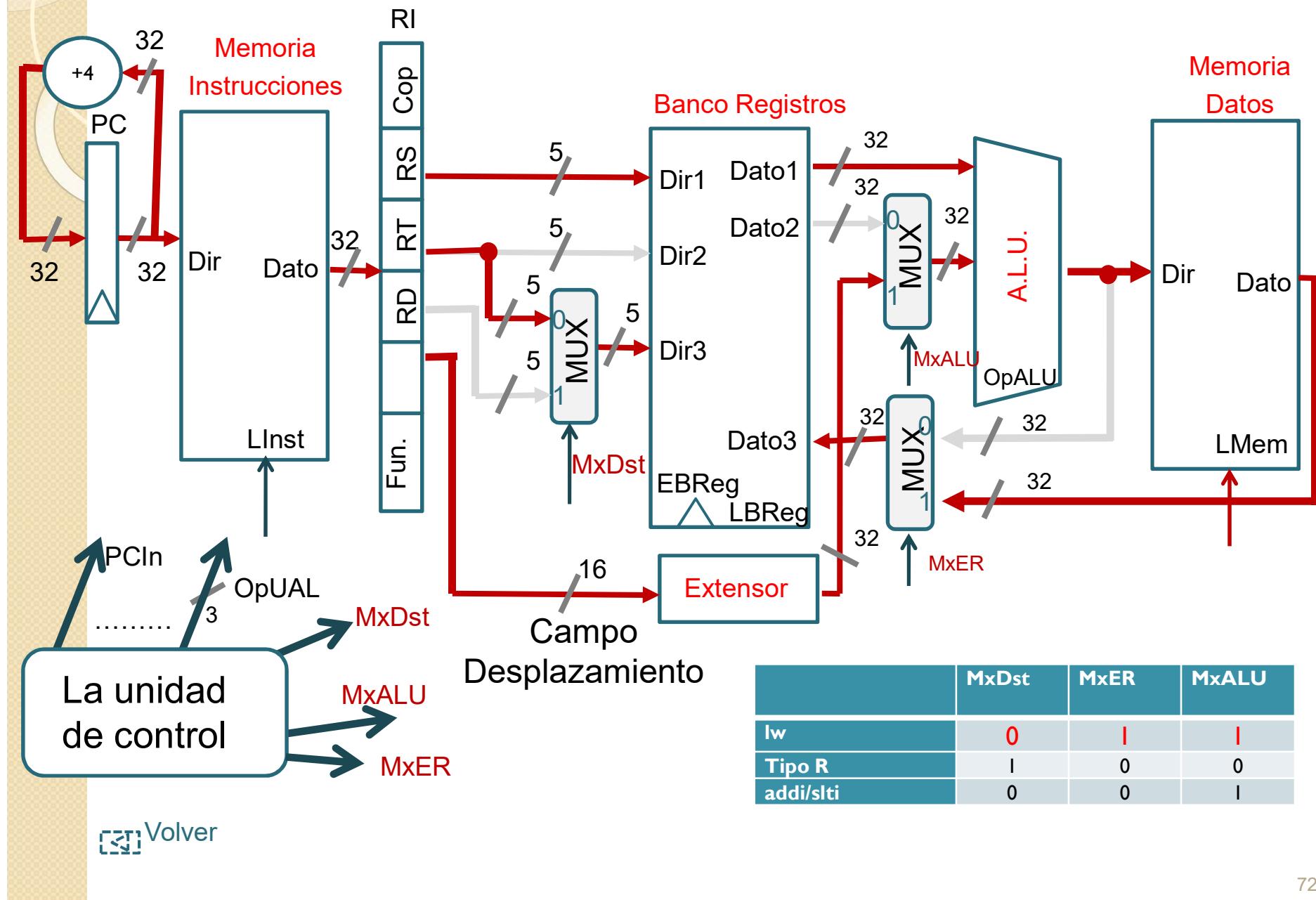
Ruta de los datos arit./lóg. R, addi/slti y lw



Ruta de los datos arit./lóg. R, addi/slti y lw



Ruta de los datos arit./lóg. R, addi/slti y lw



lw rt, rs, inmdto16 # lw \$3, 0x100(\$2)

COP	RS	RT	desplazamiento
100011	rs	rt	Desplazamiento 16

Unidad de control. Tabla de verdad

Instrucción lw: Señales de control

Instrucción	Form	Código Op.	Función	PCIn	LInst	Banco Registros		ALU	Memori a DATOS	Multiplexores		
						LBReg	EBReg			MxALU	MxDst	MxER
add rd, rs, rt	R	000000	100000	1	1	1	1	010	0	0	1	0
addi rt, rs, inmdto16	I	001000		1	1	1	1	000	0	1	0	0
lw rt, desp(rs)	I	100011		1	1	1	1	010	1	1	0	1

The diagram illustrates the flow of control signals. A horizontal bracket at the bottom spans the width of the table, with two vertical tick marks indicating the transition from 'Entradas' (Inputs) to 'Salidas' (Outputs). The 'Entradas' section covers the first six columns of the table, while the 'Salidas' section covers the last seven columns. The ALU and Memory units are positioned below the table, receiving their respective control signals from the output columns.

OpALU	Operación
000	$a \wedge b$ (and)
001	$a \vee b$ (or)
010	$a + b$ (suma aritmética)
110	$a - b$ (resta)
111	$a < b$ (menor que)

sw rt, rs, inmdto16 # sw \$3, 0x100(\$2)

COP	RS	RT	desplazamiento
011011	rs	rt	Desplazamiento 16

- ¿Cómo se ejecuta esta instrucción?

$Rt \rightarrow \text{Mem}[Rs + \text{desplazamiento}]\text{16 extendido}$

- Elementos funcionales necesarios

- Ruta de los datos (Ruta R + Ruta I)

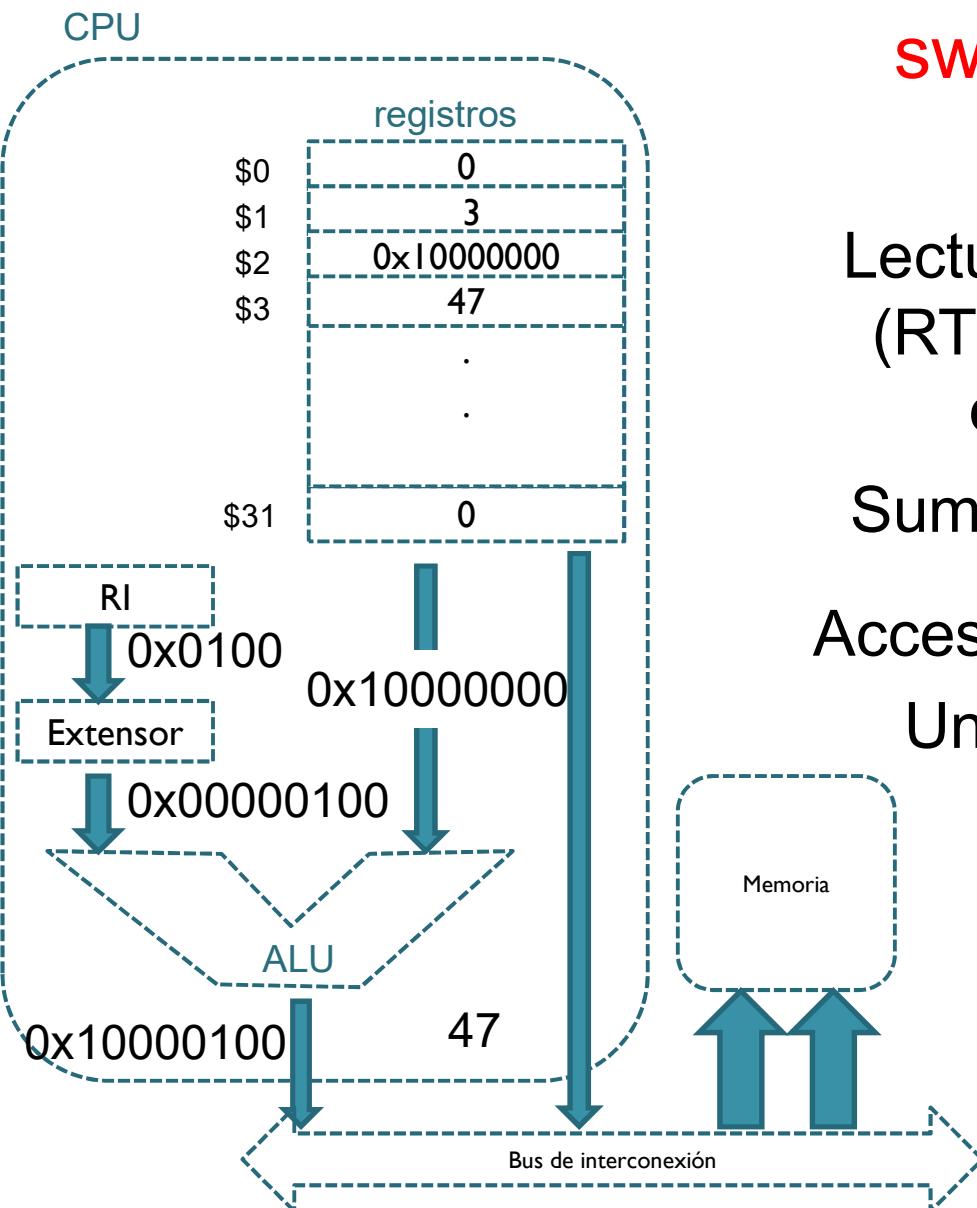
- Señales de control (Tabla de verdad)

Volver


sw rt, rs, inmdto16 # sw \$3, 0x100(\$2) Rt → Mem[Rs + desplazamiento16 extendido]

COP	RS	RT	desplazamiento
011011	rs	rt	Desplazamiento 16

¿Cómo se ejecuta esta instrucción?



sw \$3, 0x100(\$2)

Lectura registro \$2 (RS) , \$3 (RT) y desplazamiento 16 extendido a 32 bits

Suma y obtención dirección

Acceso a Memoria para escribir
Un 47 en dir 0x10000100

sw rt, rs, inmdto16 # sw \$3, 0x100(\$2) Rt → Mem[Rs + desplazamiento16 extendido]

COP	RS	RT	desplazamiento
011011	rs	rt	Desplazamiento 16

Elementos funcionales necesarios

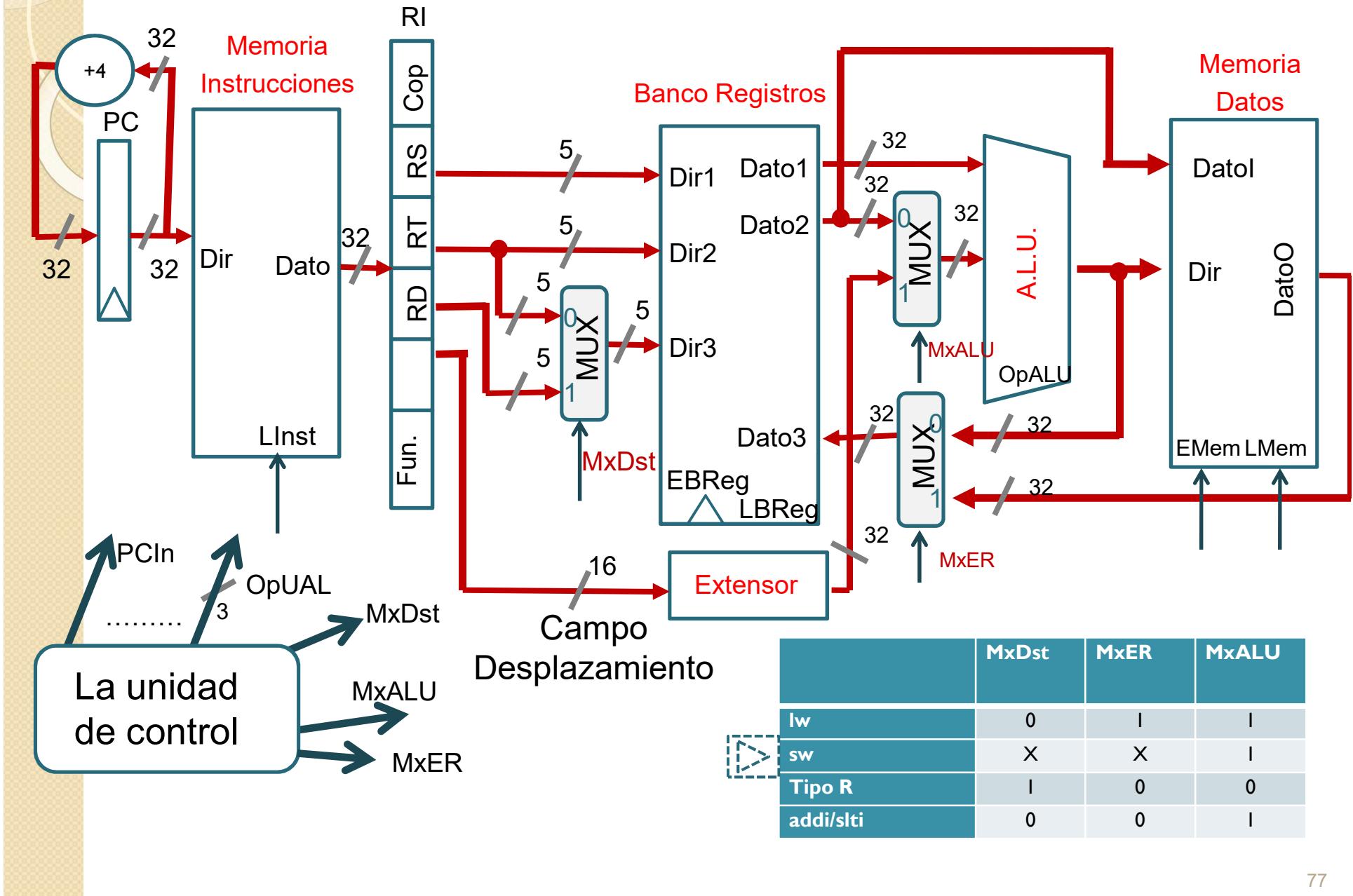
OPERACIONES:

- Leer rs, rt y desplaz16
- Extender Desplazamiento 16 a 32b
- Calcular dirección
- Escribir en memoria el registro rt

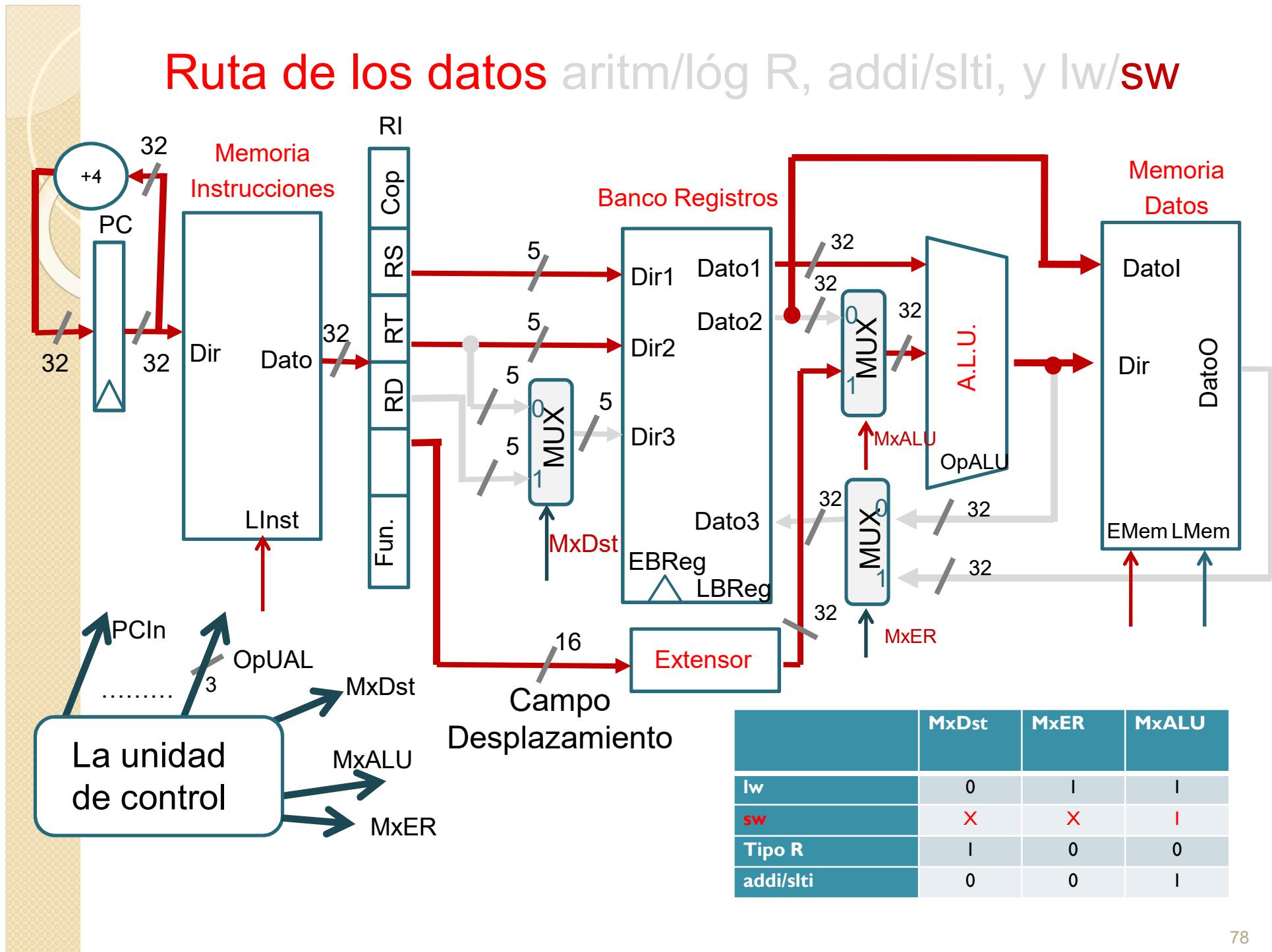
ELEMENTOS:

- Banco de registros
(el mismo ruta tipo R)
- Unidad extensora del bit de signo
(la misma ruta tipo I)
- ALU empleando camino ruta tipo I
- Memoria datos para escribir registro rt

Ruta de los datos aritm/lóg R, addi/slti, y lw/sw



Ruta de los datos aritm/lóg R, addi/slti, y lw/sw



sw rt, rs, inmdto16 # **sw \$3, 0x100(\$2)**

COP	RS	RT	desplazamiento
011011	rs	rt	Desplazamiento 16

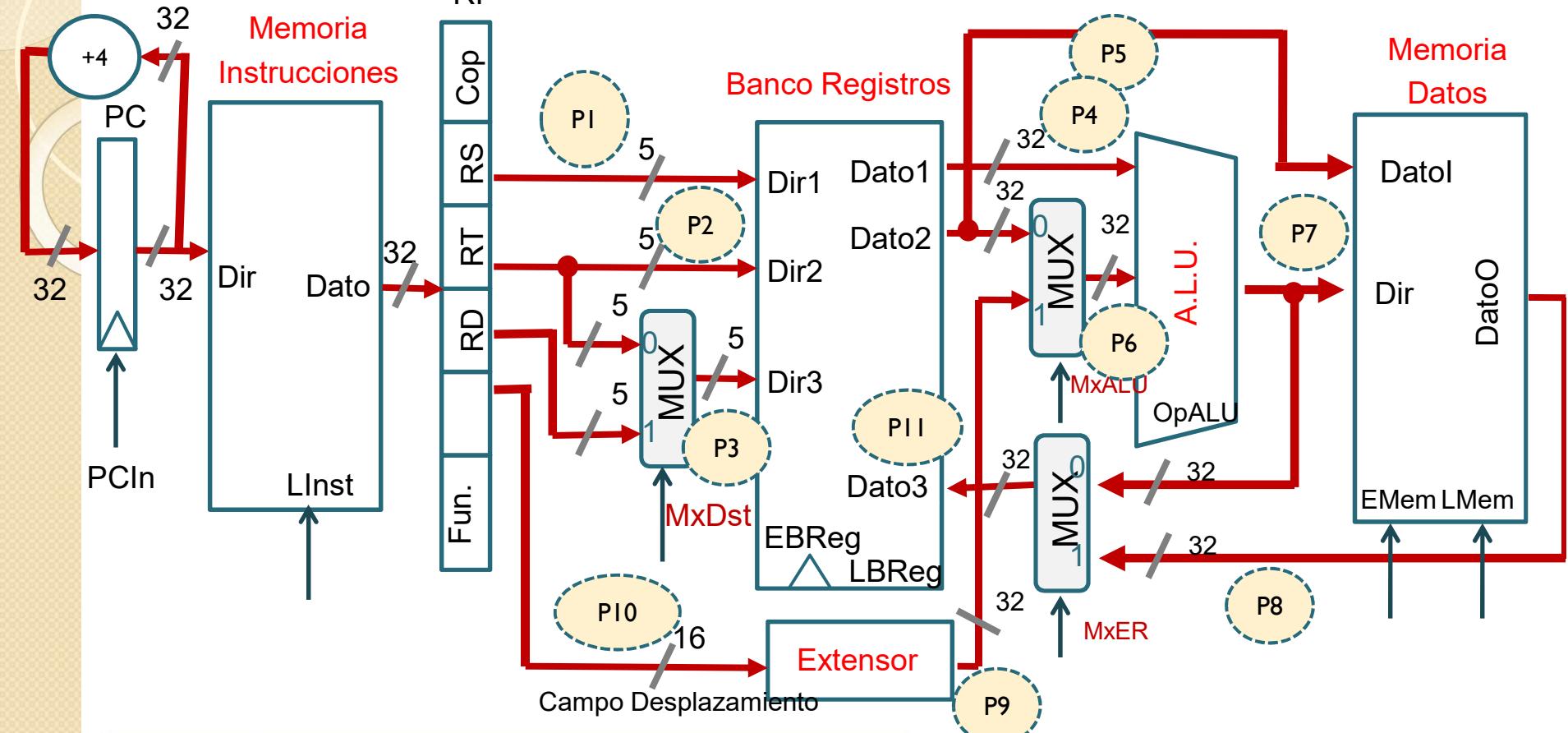
Unidad de control. Tabla de verdad

Instrucción sw: Señales de control

				Reg. CP	Mem. Instr.	Banco Registros		ALU	Memoria DATOS		Multiplexores		
Instrucción	Form	Código Op.	Función	PCIn	LInst	LBReg	EBReg	OpALU	LMem	EMem	MxALU	MxDst	MxER
add rd, rs, rt	R	000000	100000	1	1	1	1	010	0	0	0	1	0
addi rt, rs, inmdto16	I	001000		1	1	1	1	010	0	0	1	0	1
lw rt, desp(rs)	I	100011		1	1	1	1	010	1	0	1	0	1
sw rt, desp(rs)	I	101011		1	1	1	0	010	0	1	1	x	x

OpALU	Operación
000	$a \wedge b$ (and)
001	$a \vee b$ (or)
010	$a + b$ (suma aritmética)
110	$a - b$ (resta)
111	$a < b$ (menor que)

EJERCICIO: RELLENAR LA TABLA (En decimal)



\$3 = 3	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
add \$2,\$3,\$0											
addi \$2,\$3,100											
sw \$2,100(\$0)											
lw 4,100(\$0)											

Contenido y Bibliografía

- 2 – La ruta de datos y la unidad de control
 - ✓ Etapas de búsqueda y decodificación
 - ✓ Diseño de la ruta para aritméticas de tipo R
 - ✓ Diseño de la ruta para aritméticas de tipo I
 - ✓ Diseño de la ruta para instrucciones lw/sw
 - ✓ Diseño de la ruta para instrucciones de salto beq/bne

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011,
Cap 4 (4.1 – 4.4)

beq rt, rs, despl16 # beq \$3, \$2,etiqueta

COP	RS	RT	desplazamiento
000100	00010	00011	Despl 16

-  ¿Cómo se ejecuta esta instrucción?

Si ($rs = rt$) Entonces

$$PC \leftarrow (\text{Despl16 extendido} * 4) + (PC + 4)$$

-  Elementos funcionales necesarios

-  Ruta de los datos

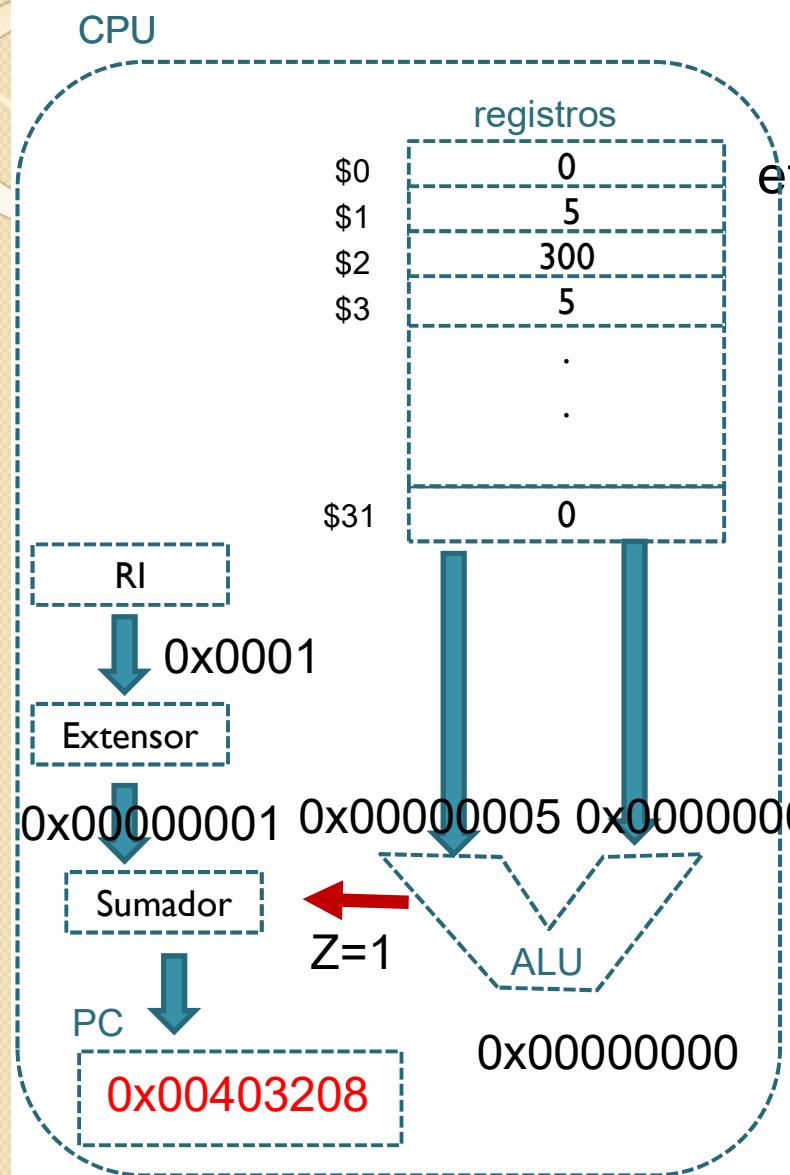
-  Señales de control

beq rt, rs, despl16 # beq \$3, \$2,etiqueta

COP	RS	RT	desplazamiento
000100	00010	00011	Despl 16

Si ($rs = rt$) Entonces $PC \leftarrow (\text{Despl16 extendido} * 4) + (PC + 4)$

¿Cómo se ejecuta esta instrucción?



beq \$3, \$1, etiqueta [0x403200]

add \$2, \$1,\$3 [0x403204]

etiqueta: slti \$4, \$3, 12 [0x403208]

Lectura registros \$1(RS) , \$3 (RT) y Despl 16 y se extiende a 32 bits

En paralelo:

Comparación ¿\$3 (RS) = \$1 (RT) ?

Cálculo dirección:

dato = Despl 16 extendido a 32 bits

PCN = $(PC + 4) + (\text{dato} * 4) = \text{etiqueta}$

Si resultado CIERTO (Z = 1):

$PC = PCN$

Sino (Z=0, siguiente instrucción)

$PC = PC + 4$

beq rt, rs, despl16	#	beq \$3, \$2,etiqueta	Si (rs = rt) Entonces PC \leftarrow (Despl16 extendido * 4) + (PC + 4)
COP 000100	RS 00010	RT 00011	desplazamiento Despl 16

¿Cómo se ejecuta esta instrucción?

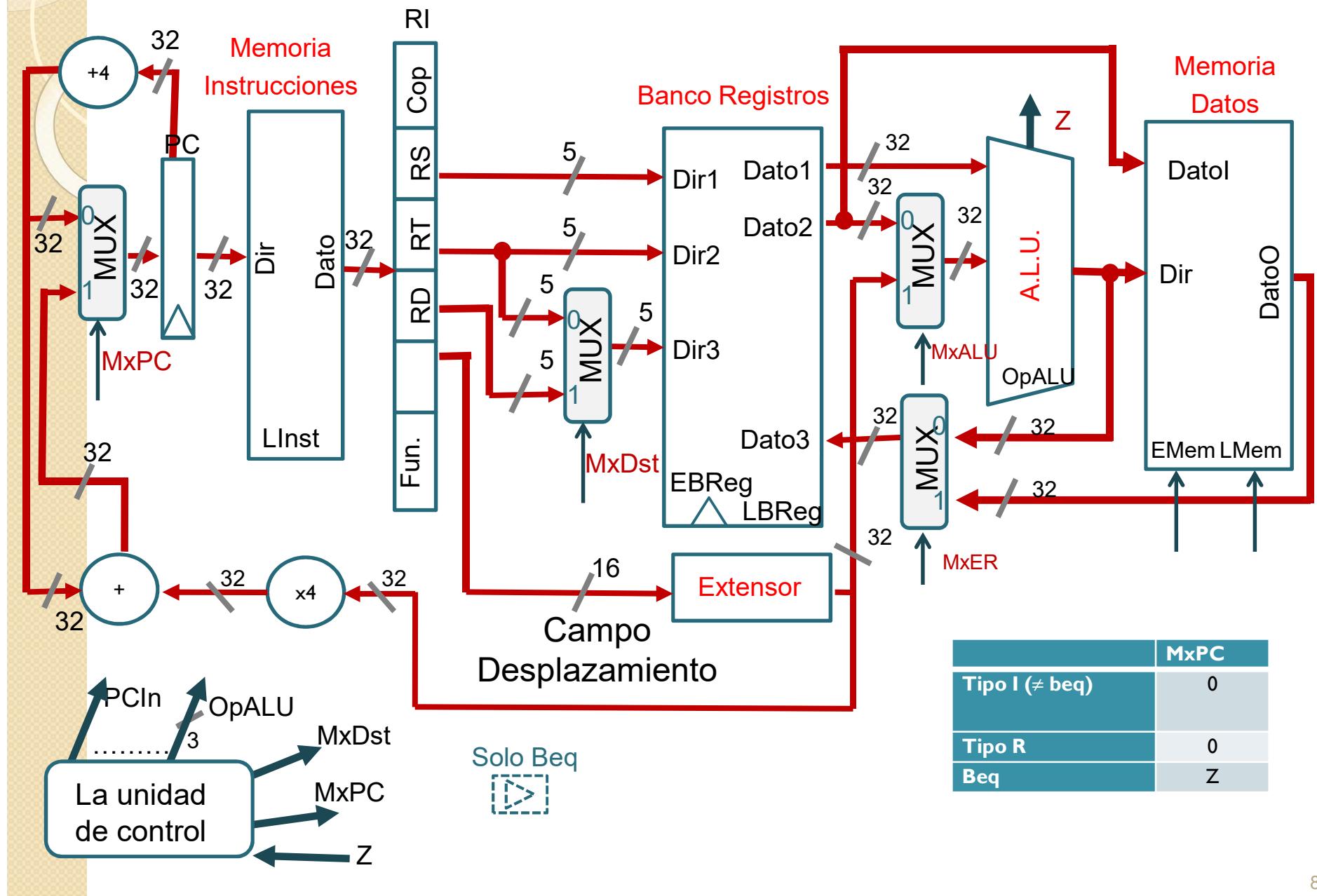
OPERACIONES:

- Leer rs, rt y desplaz16
- Extender Despl 16 a 32b
- Comparar Rs y Rt
- Calcular dirección
- Si Rs=Rt Entonces Modificar PC

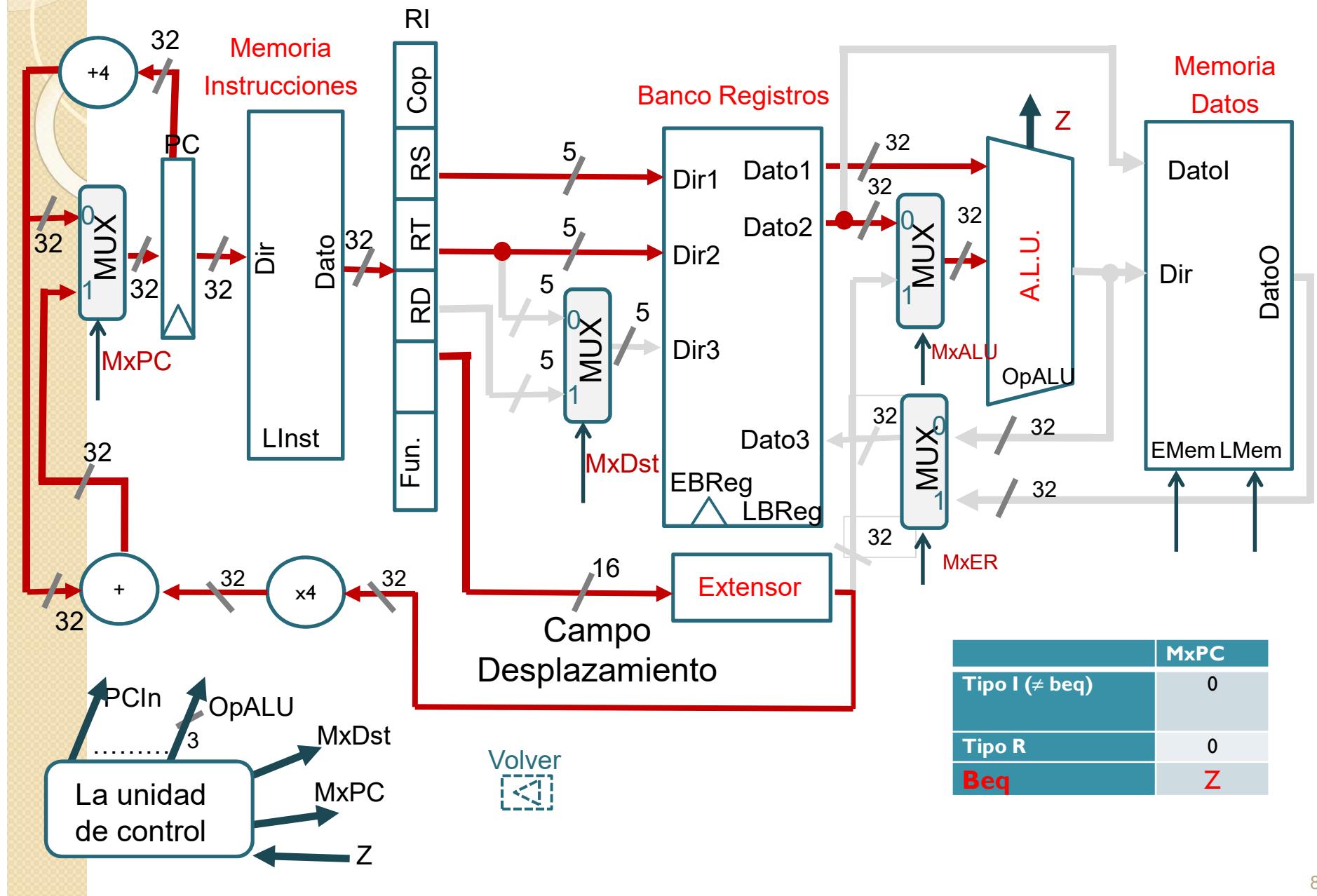
ELEMENTOS:

- Banco de registros (el mismo ruta tipo R)
- Unidad extensora del bit de signo (la misma ruta tipo I)
- ALU con bit Z (operación RESTA)
- Unidad para calcular:
 - Dato * 4 (añadir 2 bits a cero)
 - Sumar a PC+4
- Mux entrada de PC

Ruta de los datos arit./lóg. R, addi/slti, y lw/sw y beq



Ruta de los datos arit./lóg. R, addi/slti, y lw/sw y beq



beq rt, rs, despl16 # beq \$3, \$2,etiqueta

COP	RS	RT	desplazamiento
000100	00010	00011	Despl 16

Unidad de control. Tabla de verdad

Instrucción beq : Señales de control

Instruc.	Tipo	Cop	Función	PCIn	LInst	Banco Registros		ALU	Memoria DATOS		Multiplexores			
						LBReg	EBReg		OpALU	LMem	EMem	MxALU	MxDst	MxER
add	R	000000	100000	1	1	1	1	010	0	0	0	1	0	0
addi	I	001000		1	1	1	1	010	0	0	1	0	0	0
lw	I	100011		1	1	1	1	010	1	0	1	0	1	0
sw	I	101011		1	1	1	0	010	0	1	1	X	X	0
beq	I	000100		1	1	1	0	110	0	0	0	X	X	Z



OpALU	Operación
000	a \wedge b (and)
001	a \vee b (or)
010	a + b (suma aritmética)
110	a - b (resta)
111	a < b (menor que)



Instrucción bne : Señales de control

OpALU	Operación
000	$a \wedge b$ (and)
001	$a \vee b$ (or)
010	$a + b$ (suma aritmética)
110	$a - b$ (resta)
111	$a < b$ (menor que)

Volver



j despl26 # j etiqueta

COP	desplazamiento
000010	Desplazamiento 26

- ¿Cómo se ejecuta esta instrucción?

$$PC_{27..0} \leftarrow (Despl26 * 4)$$

- Elementos funcionales necesarios

- Ruta de los datos

- Señales de control

Volver



Arquitectura MIPS32. Op. Salto tipo J

j etiqueta

000010	0100010101100111000000000000
--------	------------------------------

Como PC tiene 32, se sustituyen los 28 bits de menos peso así:

$$PC = PC_{31..28} \parallel \text{dato} \parallel 00 = \text{etiqueta}$$

Por ejemplo si Etiqueta = 0x04567000=

0000-0100-0101-0110-0111-0000-0000-0000

Elementos necesarios: Añadir a la ruta j despl26

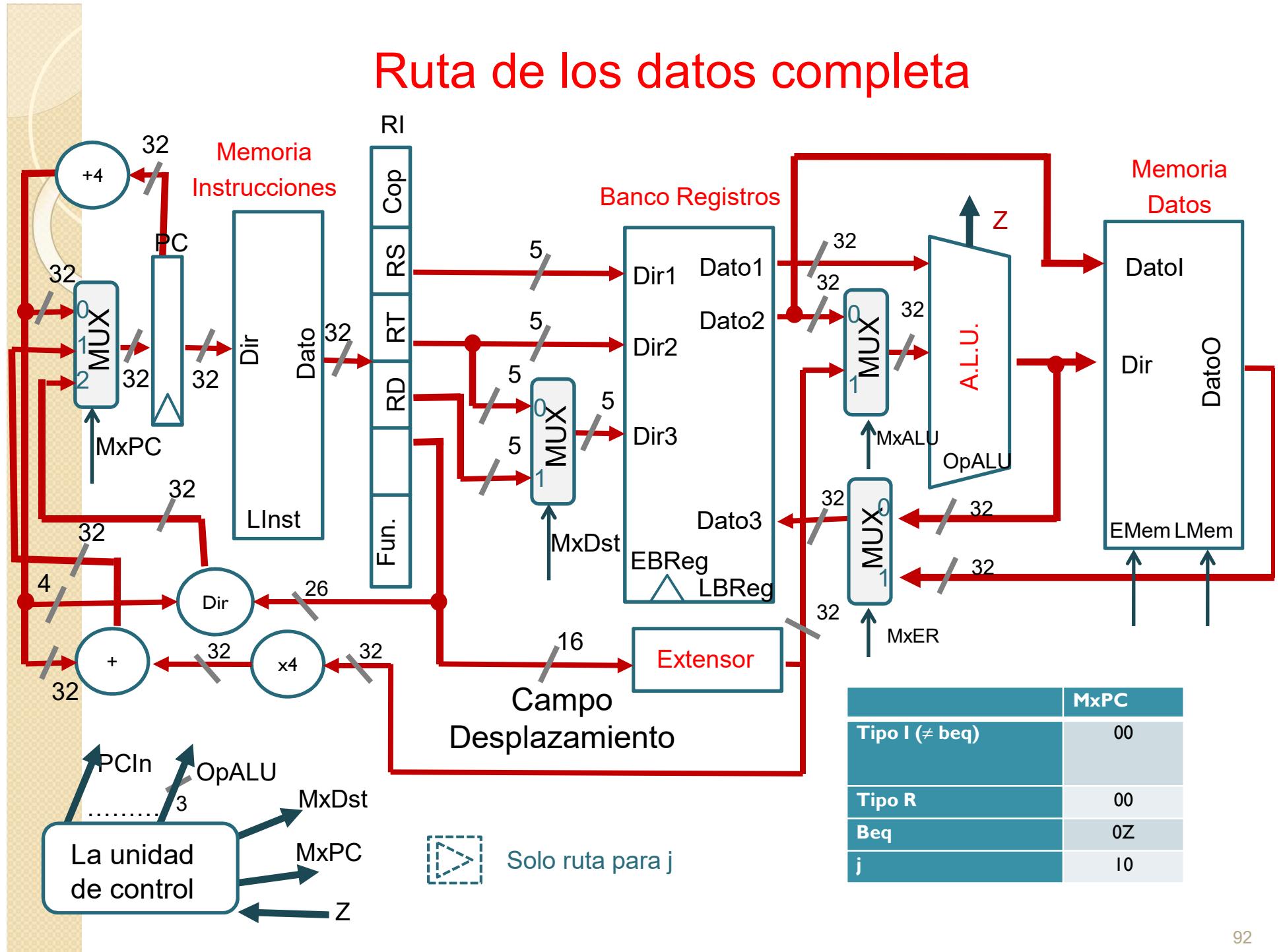
OPERACIONES:

- Calcular dirección de salto
- Escribirla en PC

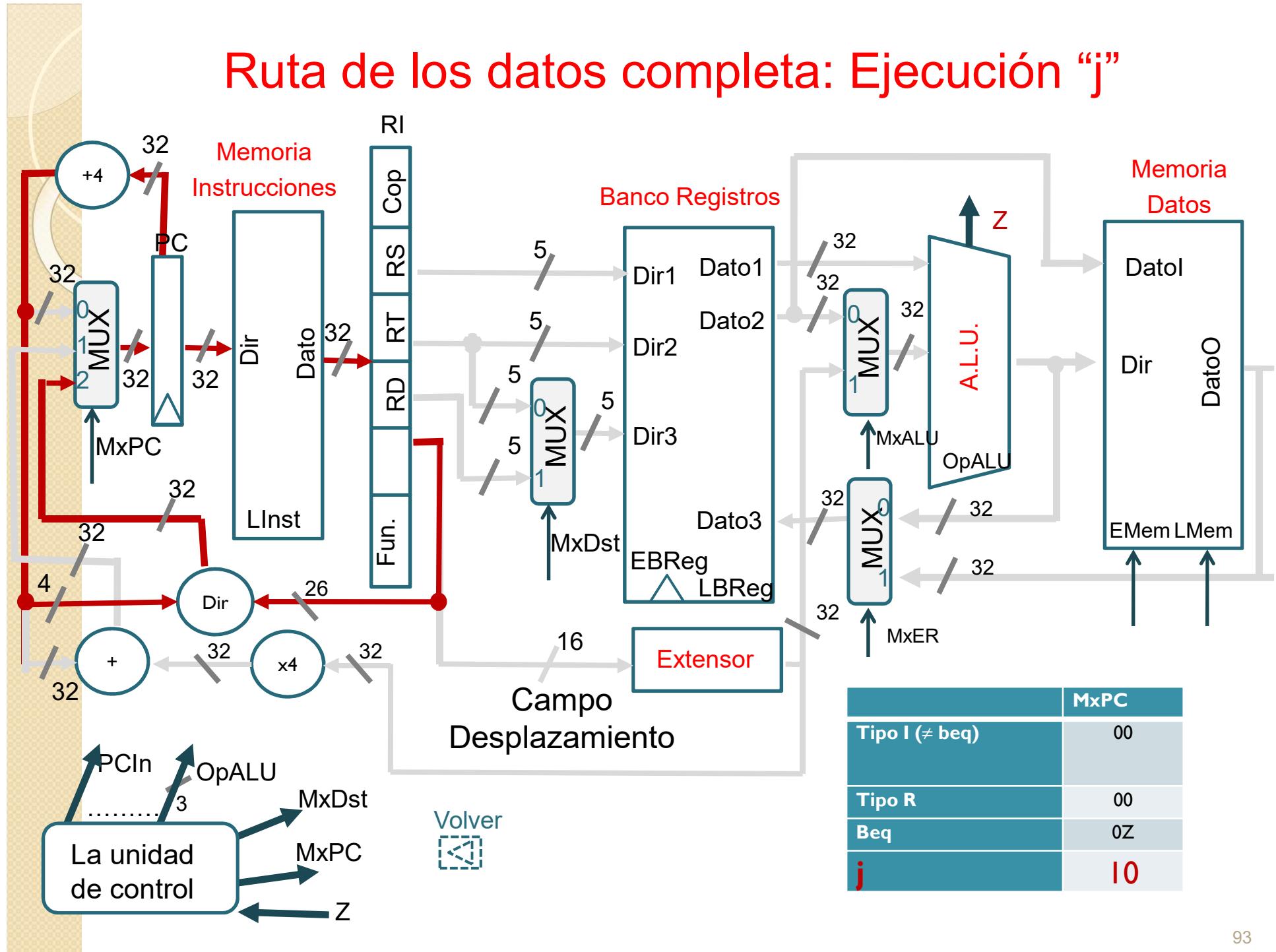
ELEMENTOS:

- Unidad que componga la dirección empleando el RI y algunos bits de PC
- Modificar el Mux de PC

Ruta de los datos completa



Ruta de los datos completa: Ejecución “j”



Instrucción j: Señales de control

Instruc.	Form	Cop	Función	PCIn	LInst	LBReg	EBReg	Banco Registros		ALU	Memoria DATOS		Multiplexores			
								Me m. Instr .	OpALU		LMem	EMem	MxALU	MxDst	MxER	MxPC
add	R	000000	100000	1	1	1	1		010	0	0	0	1	0	00	
addi	I	001000			1	1	1		010	0	0	1	0	0	00	
lw	I	100011			1	1	1		010	1	0	1	0	1	00	
sw	I	101011			1	1	1	0	010	0	1	1	X	X	00	
beq	I	000100			1	1	1	0	110	0	0	0	X	X	0Z	
j	J	000010			1	1	X	0	XXX	0	0	X	X	X	10	

Entradas

Salidas

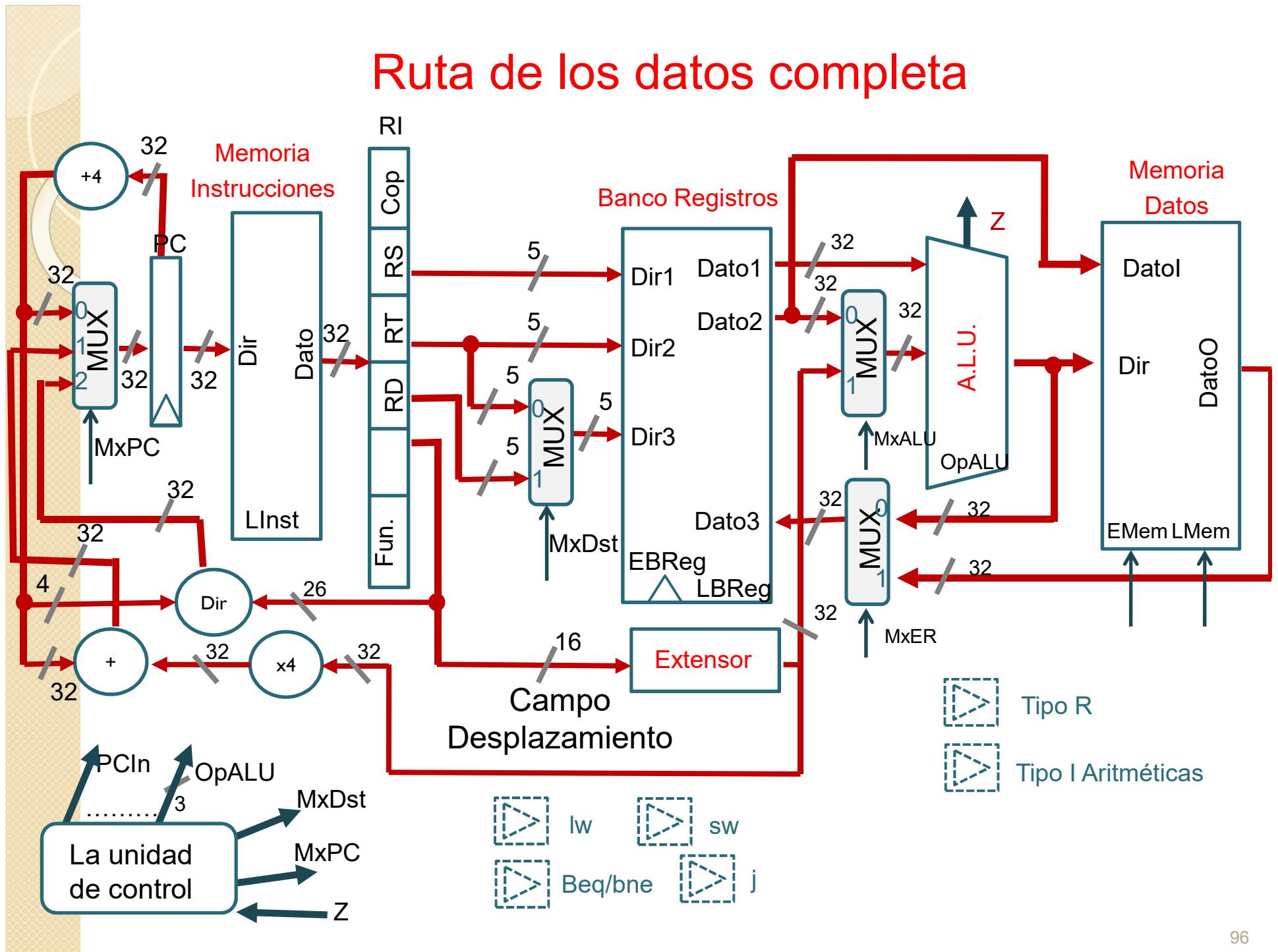
OpALU	Operación
000	$a \wedge b$ (and)
001	$a \vee b$ (or)
010	$a + b$ (suma aritmética)
110	$a - b$ (resta)
111	$a < b$ (menor que)

ANEXO I

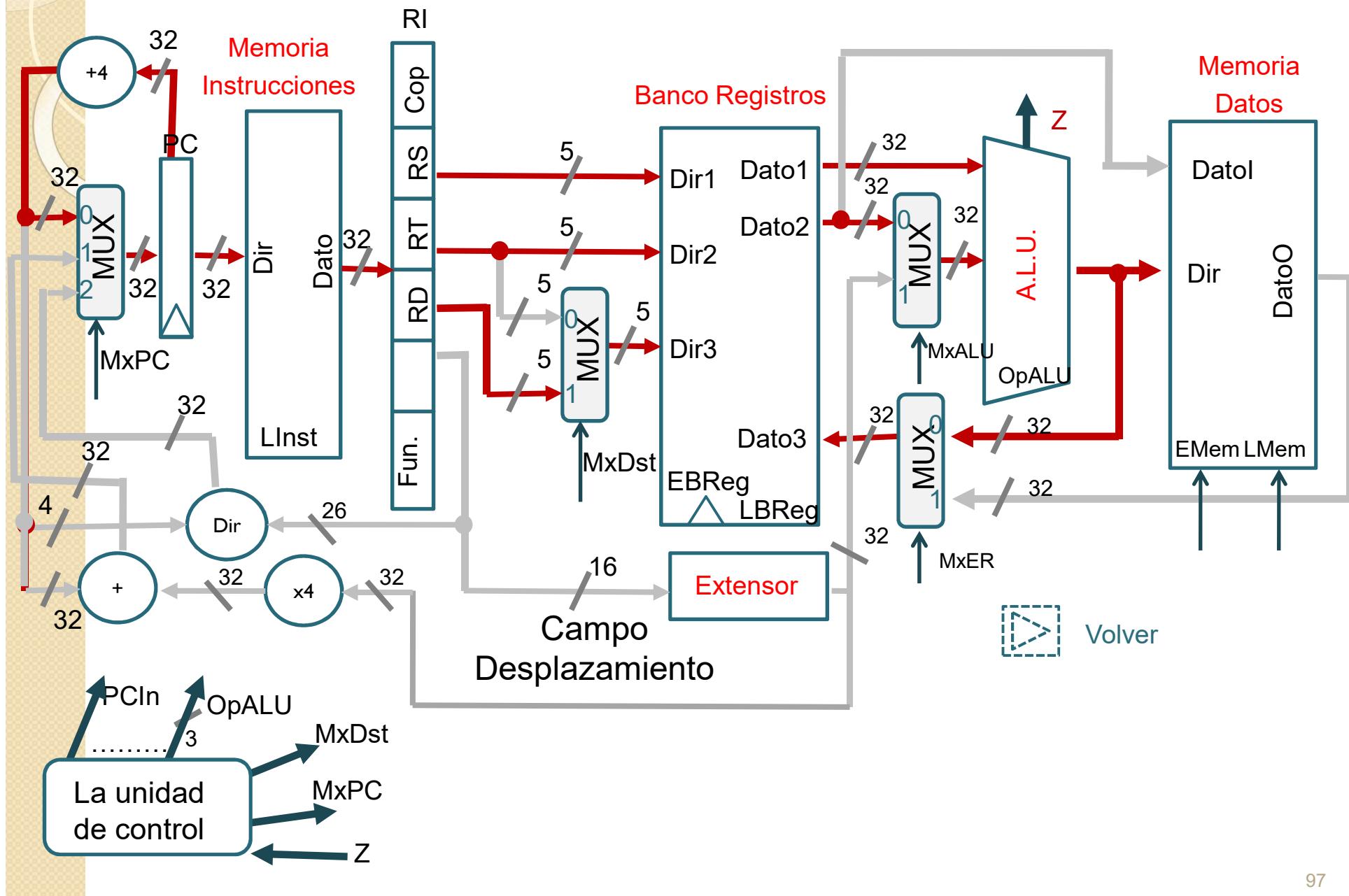
- **La ruta de datos completa**

- ✓ Se añade a continuación la ruta de datos completa y sucesivos ejemplos donde se han marcado en rojo las partes del circuito relevantes en la ejecución de las distintas instrucciones

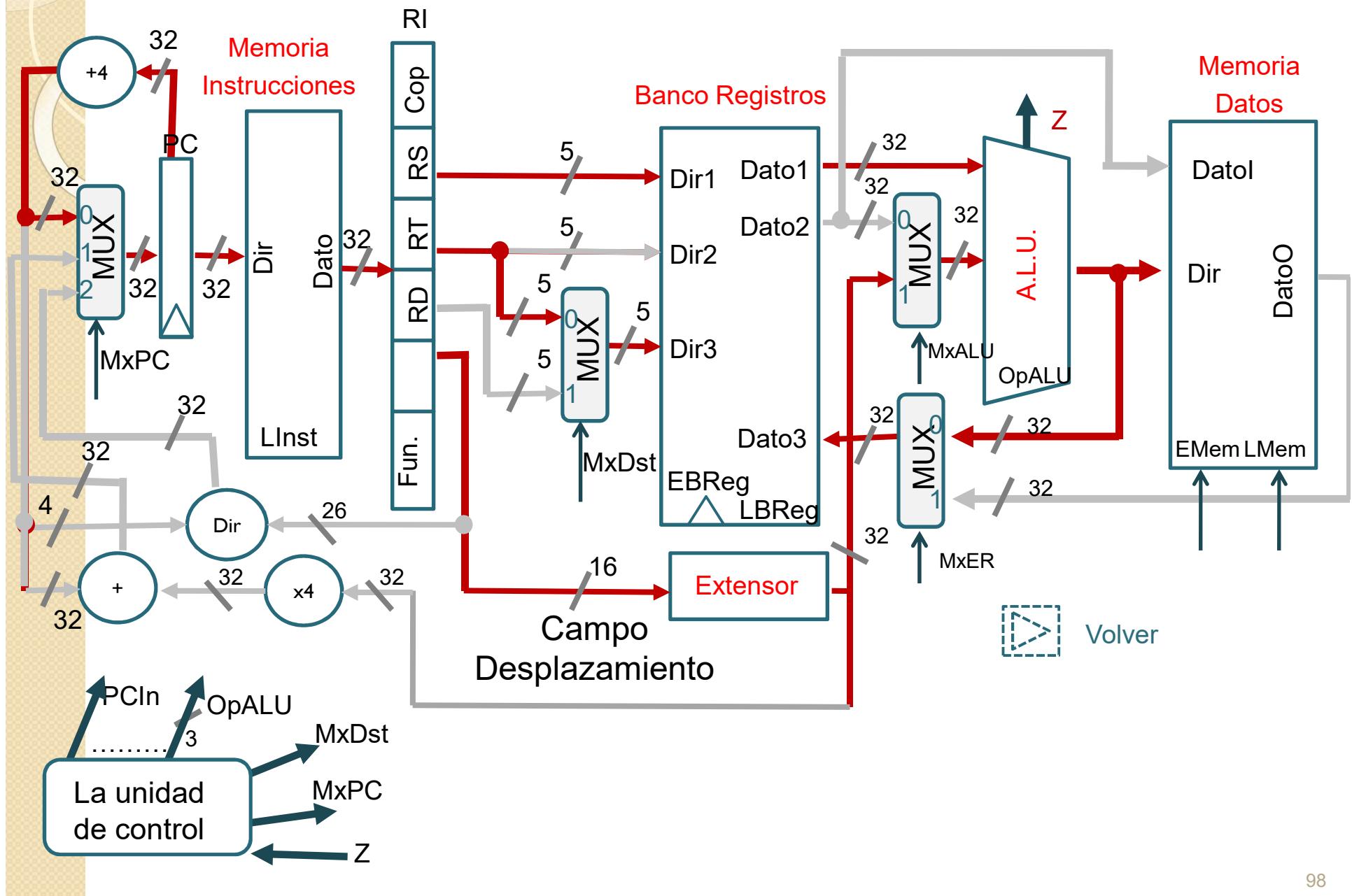
Ruta de los datos completa



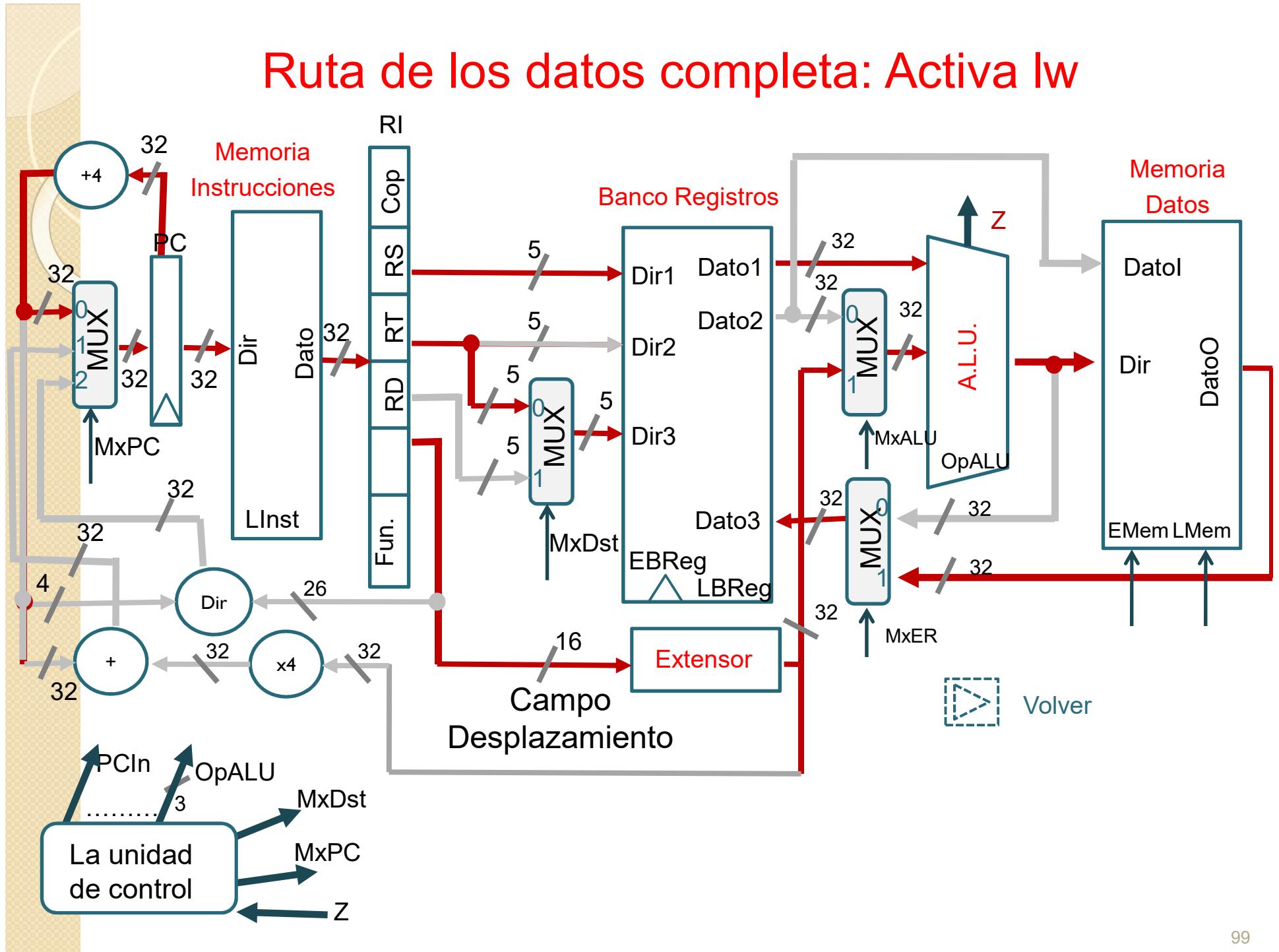
Ruta de los datos completa: Activa tipo R



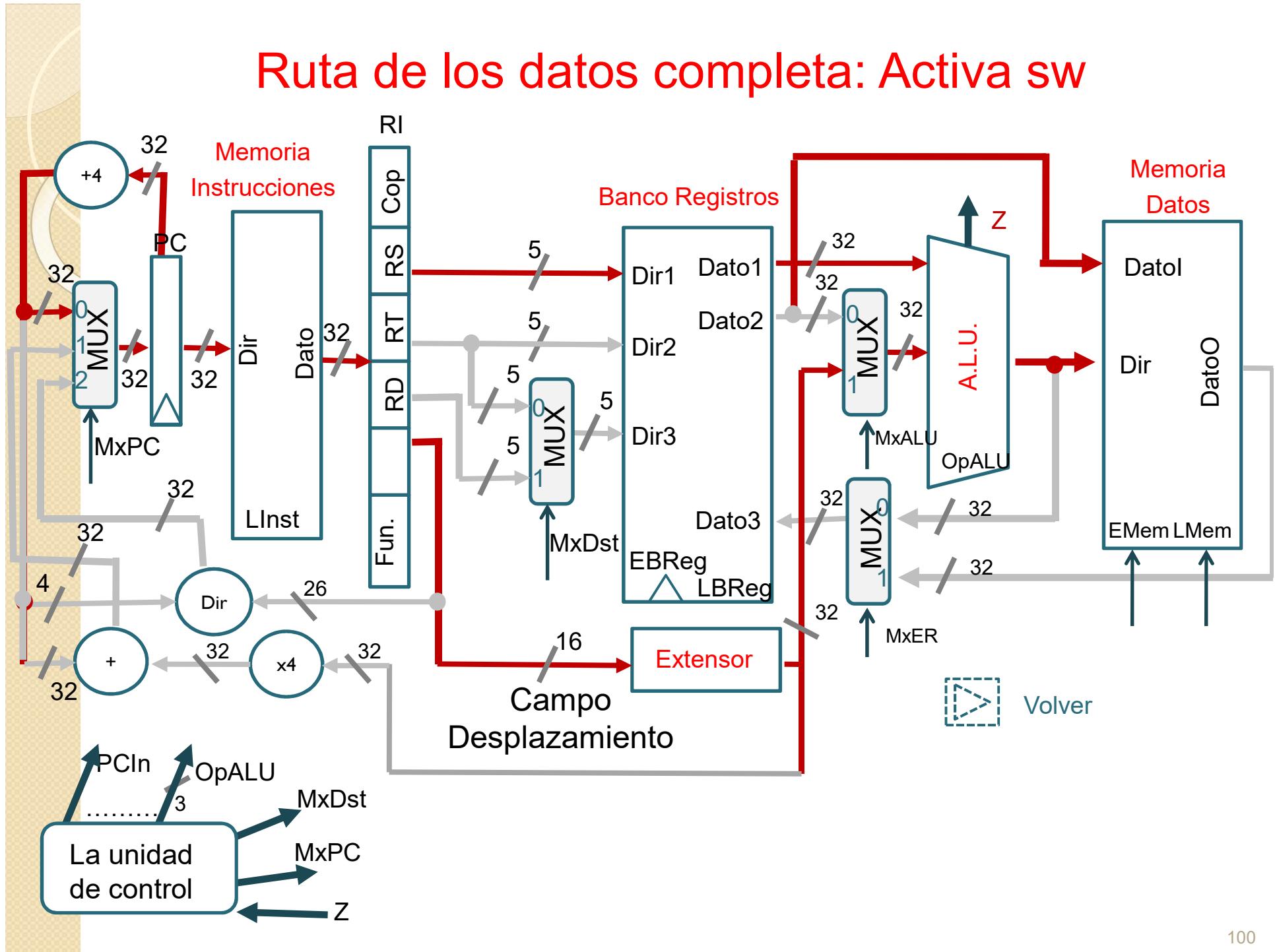
Ruta de los datos completa: Activa aritméticas tipo I



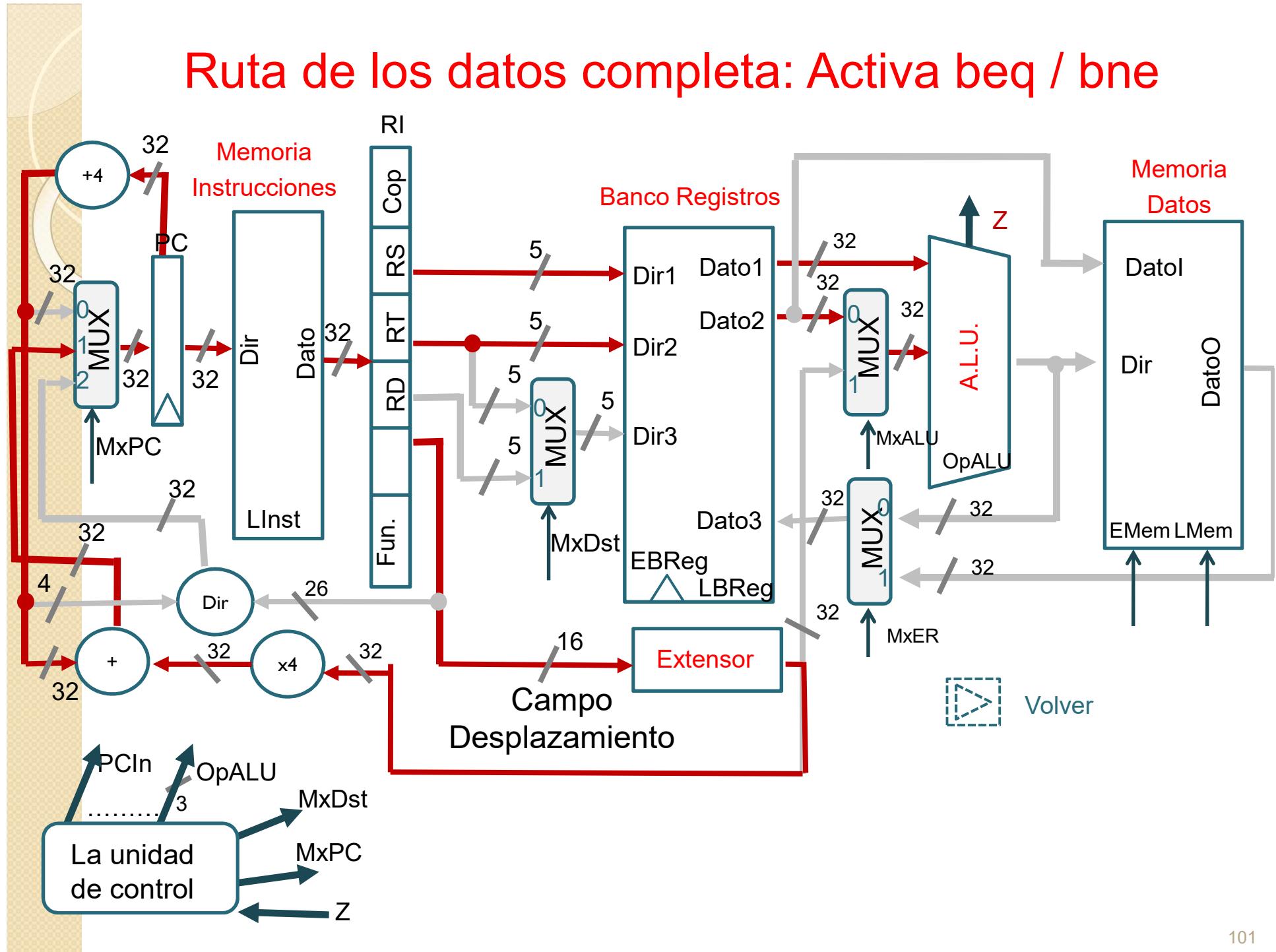
Ruta de los datos completa: Activa lw



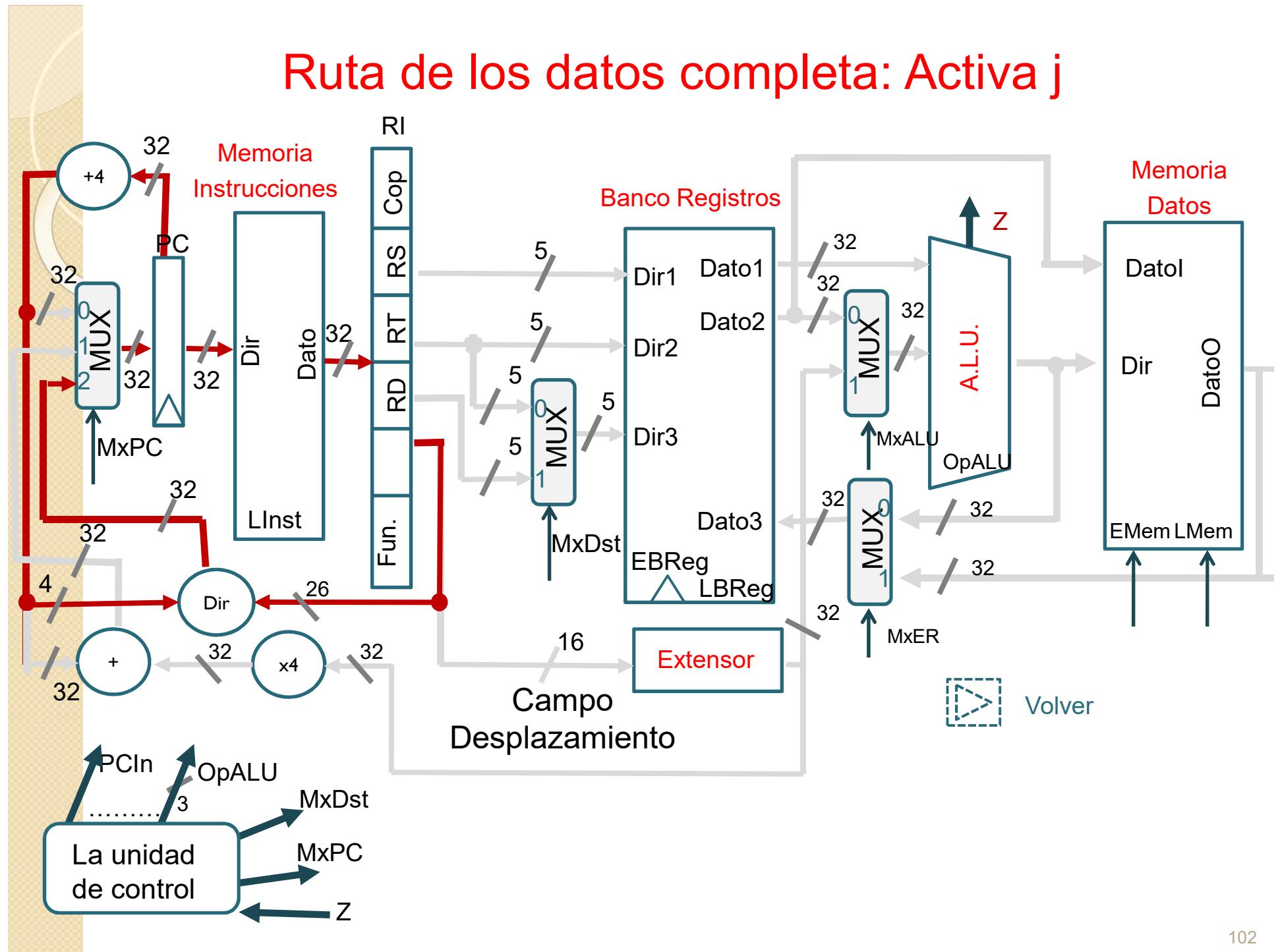
Ruta de los datos completa: Activa sw



Ruta de los datos completa: Activa beq / bne



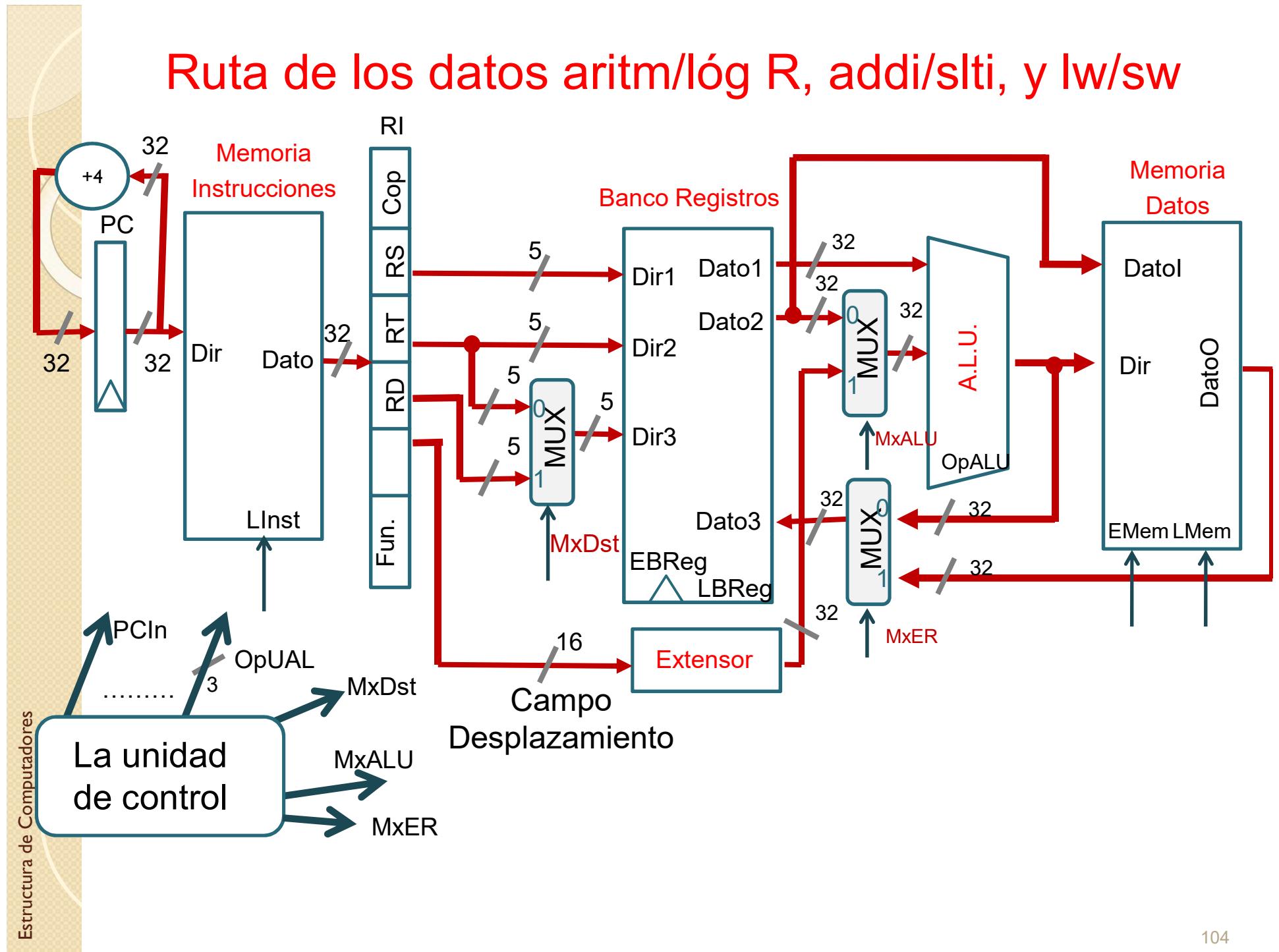
Ruta de los datos completa: Activa j



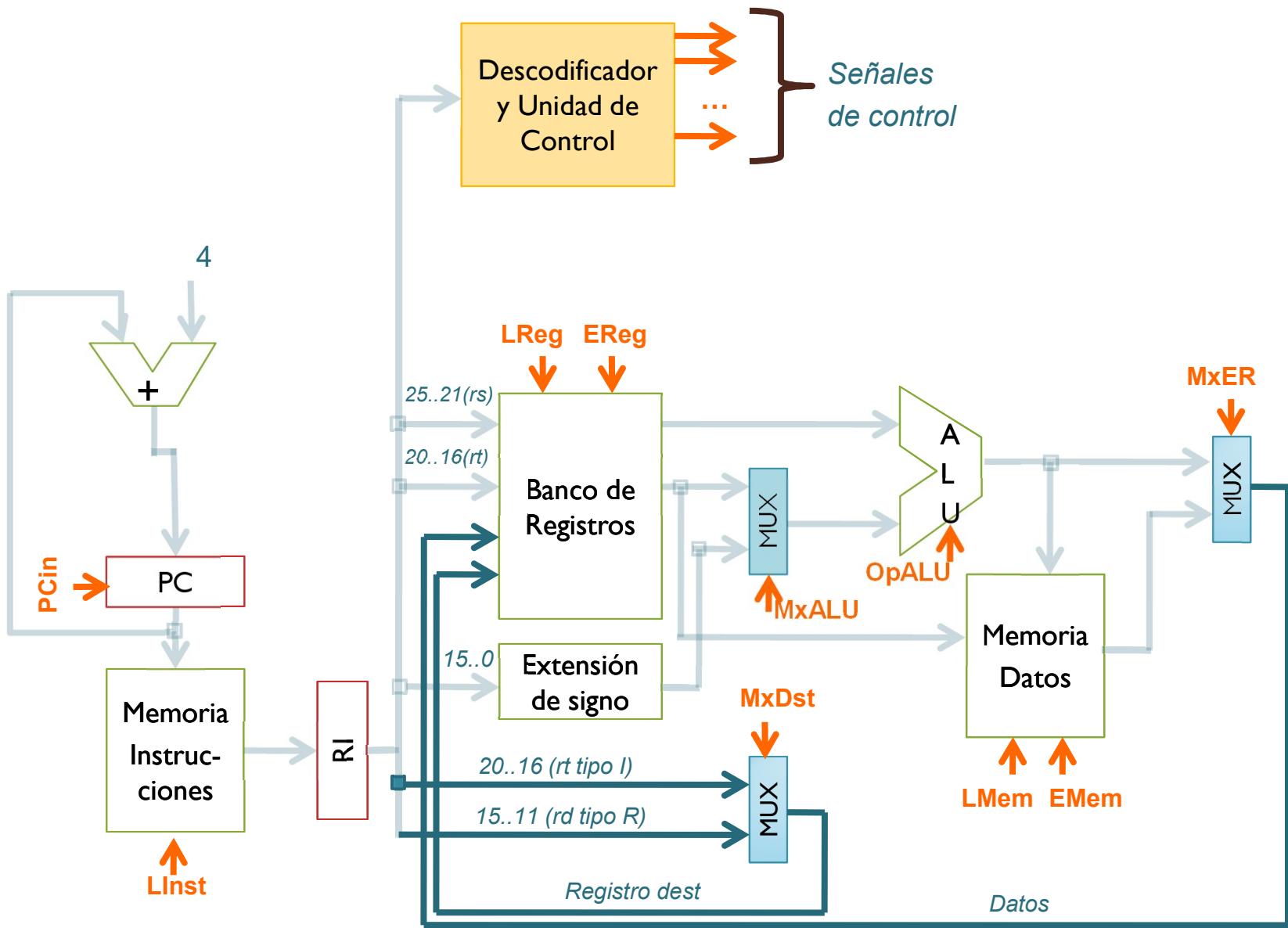
ANEXO II

- Correspondencia entre la ruta de los datos empleada en este tema y la que se empleará en el tema siguiente
 - ✓ Son exactamente iguales, solo varía el dibujo de los elementos funcionales.
 - ✓ Se ha utilizado la ruta sin las instrucciones de salto

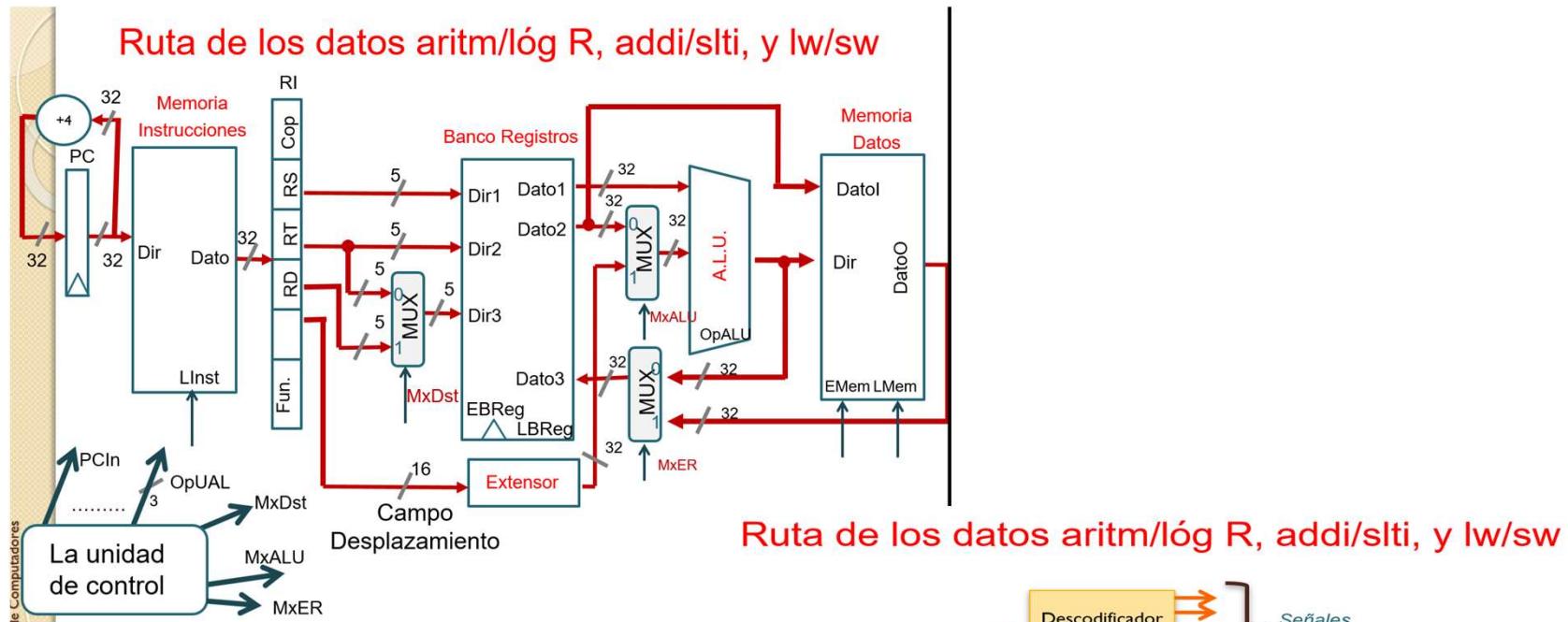
Ruta de los datos aritm/lóg R, addi/slti, y lw/sw



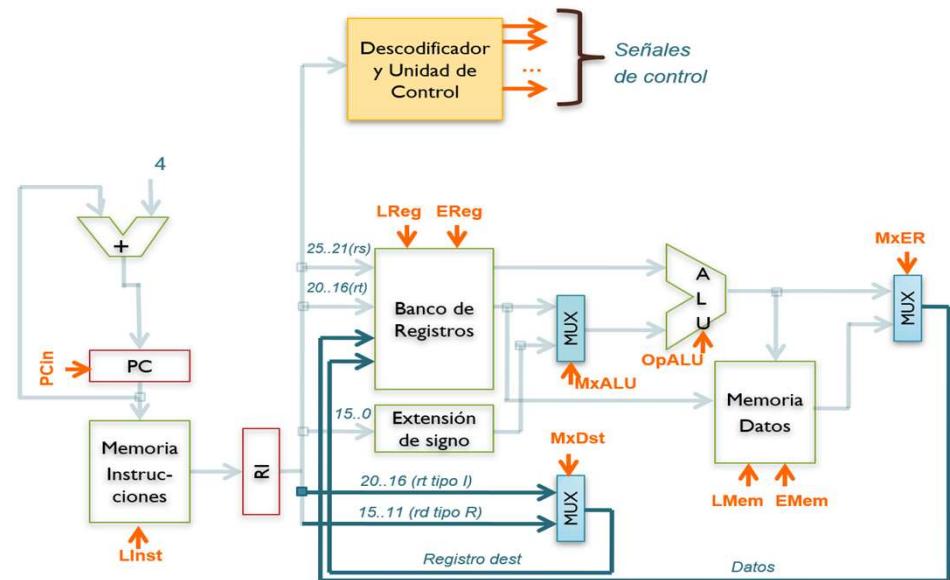
Ruta de los datos aritm/lóg R, addi/slti, y lw/sw



ANEXO II. Comparativa rutas temas 1 y 2



Ruta de los datos aritm/lóg R, addi/slti, y lw/sw



Estructura de Computadores

Grado de Ingeniería Informática
ETSINF

Tema I: El procesador

Fin

