

Estructura de Computadores

Grado de Ingeniería Informática
ETSINF

Tema 2: Segmentación Básica
del Procesador



Objetivos

- Conocer el concepto de segmentación como técnica de mejora de prestaciones
- Conocer la segmentación básica del procesador MIPS
- Detectar conflictos y riesgos sencillos
- Conocer algunas de las técnicas de solución de conflictos que aparecen en la ruta de datos segmentada del MIPS
- Saber comparar prestaciones de una ruta de datos segmentada

Contenidos y Bibliografía

- El proceso de segmentación
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ Identificación de las fases de ejecución de las instrucciones
 - ✓ Diseño de las etapas (excepto beq)
 - ✓ Cálculo del tiempo de ciclo
 - ✓ Añadiendo instrucción beq a la ruta
- Conflictos y riesgos
 - ✓ Conflictos de datos
 - ✓ Conflictos de control (flujo)
- Perspectiva histórica. RISC-CISC y otras alternativas



Contenidos y Bibliografía

- **El proceso de segmentación**
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ Identificación de las fases de ejecución de las instrucciones
 - ✓ Diseño de las etapas (excepto beq)
 - ✓ Cálculo del tiempo de ciclo
 - ✓ Añadiendo instrucción beq a la ruta
- Conflictos y riesgos
 - ✓ Conflictos de datos
 - ✓ Conflictos de control (flujo)
- Perspectiva histórica. RISC-CISC y otras alternativas



Vídeos 1 y 2 repasan este apartado
Archivo con fórmulas:
[Formulas de prestaciones. pdf](#)



Concepto de Segmentación

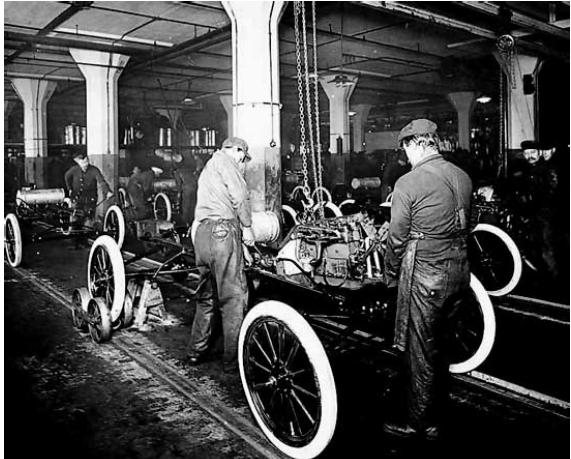
- **Definición de segmentación**
 - ✓ Descomposición de un sistema que ejecuta un determinado proceso en varias etapas, de manera que:
 - Cada etapa se ocupa de una parte del proceso global utilizando recursos propios
 - El proceso global requiere la aplicación ordenada de todas las etapas
 - Todas las etapas trabajan simultáneamente (cada una en un proceso distinto)
- **Objetivo:**Aumentar el paralelismo (temporal) de los procesos y por tanto, aumentar la productividad



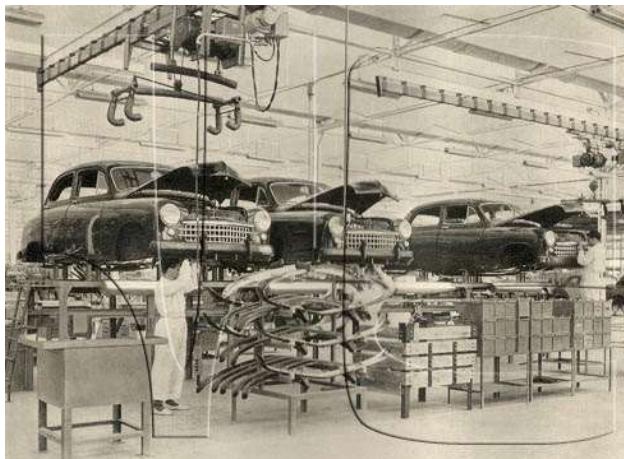
Concepto de Segmentación

- Ejemplos

Ford 1907-1908



Seat Década 50



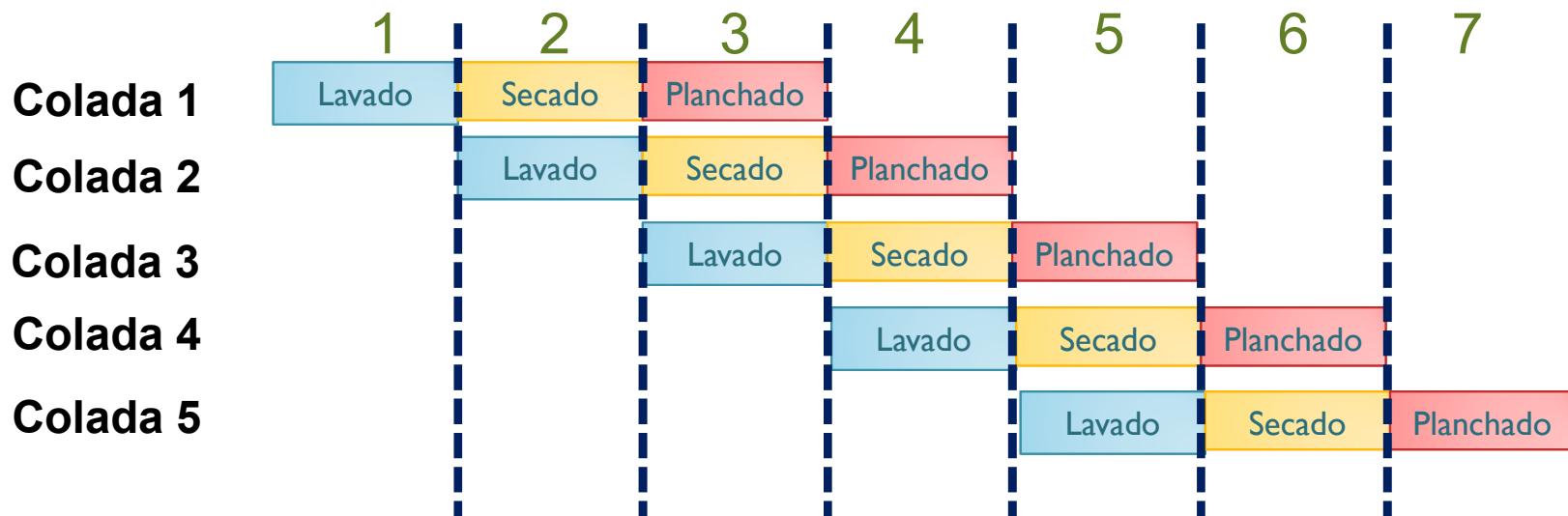
Cadena de montaje de un coche
(montaje motor, ensamblado interior, pintado, ...)

Concepto de Segmentación

- Ejemplos



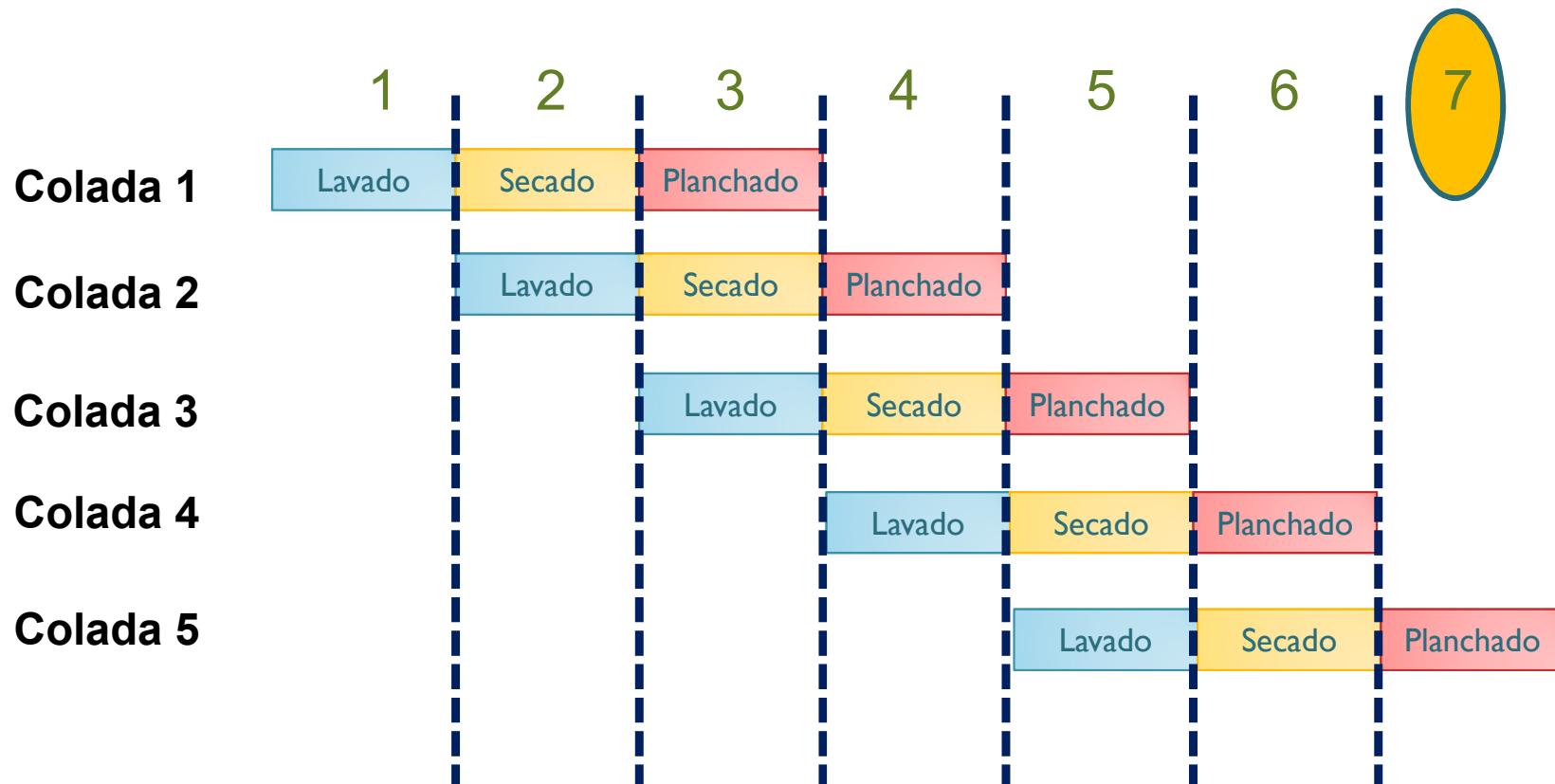
Lavandería
(lavado, secado, planchado)



El símil de la lavandería

3 etapas $K = 3$

5 coladas $n = 5$



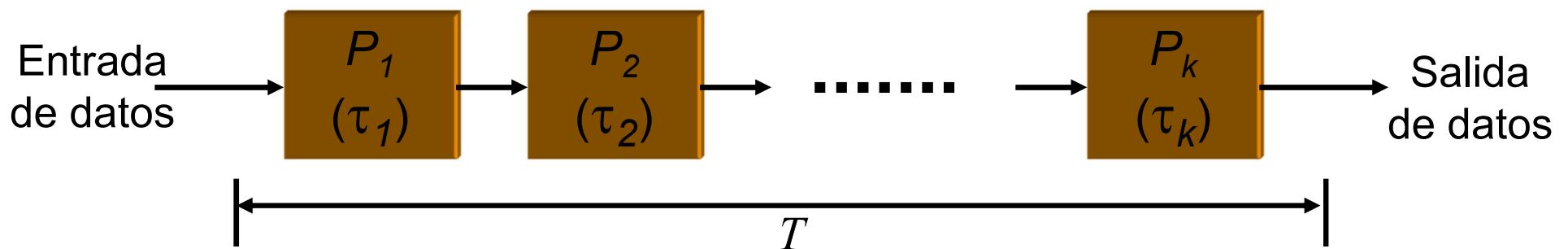
Terminar 5 coladas requerirá el tiempo de duración de un total de 7 etapas:

$$n + K - 1$$

Proceso de Segmentación

- Entrada
 - ✓ Operador lógico P que opera sobre datos de entrada
 - ✓ P se considera constituido por k etapas Pi que operan de manera secuencial sobre los datos
 - ✓ Consideraciones temporales:
 - Retardo del circuito: T
 - Cada etapa Pi tiene necesidades temporales (retardo) distintas: τ_i

$$T = \sum_{i=1}^k \tau_i$$



Proceso de Segmentación

- Salida (circuito segmentado)

- ✓ k etapas (profundidad de segmentación: k)

- Cada etapa va precedida de un registro de etapa (*staging latch*)

- Etapa i:

- Entrada: contenido del registro i

- Salida: se almacena en el registro $i+1$

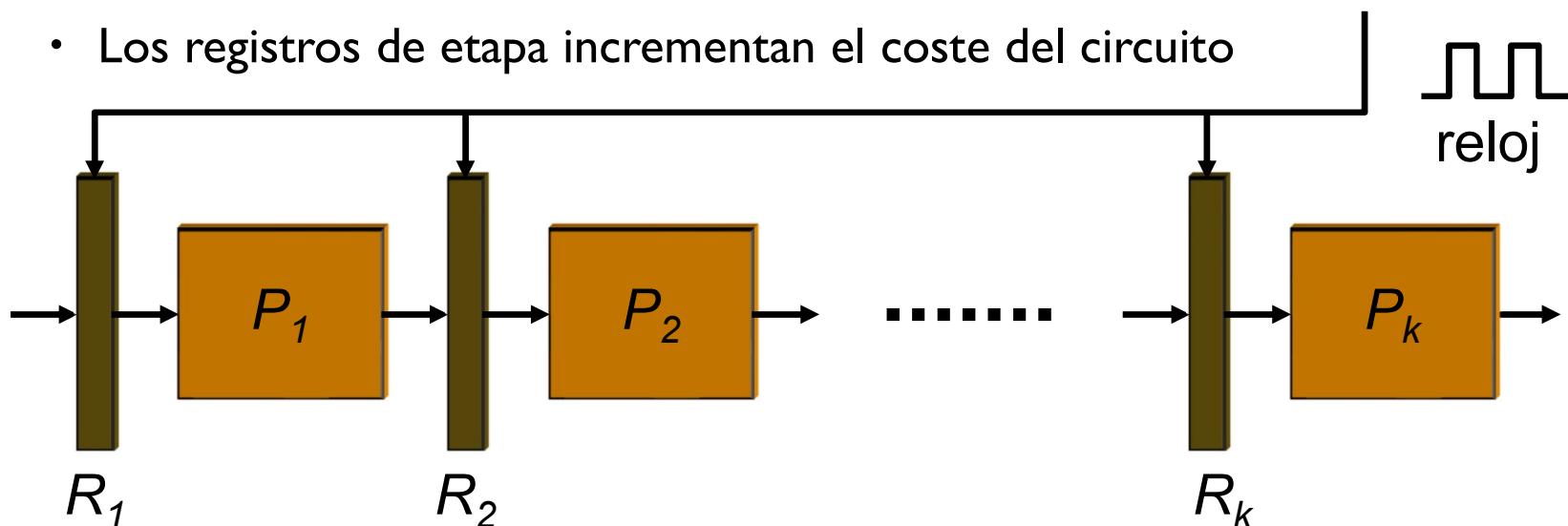
- Registro de etapa i:

- Recoge los datos producidos por la etapa $i-1$

- Suministra datos a la etapa i

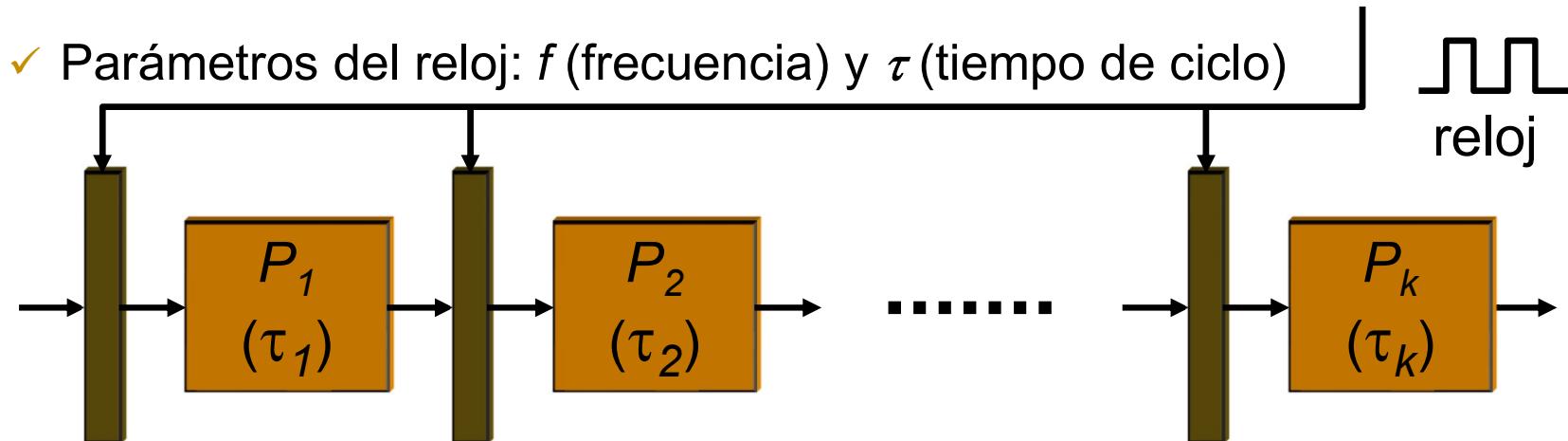
- Todos los registros de etapa se actualizan en el mismo flanco de reloj

- Los registros de etapa incrementan el coste del circuito



Proceso de Segmentación

- Sincronización del circuito



- ✓ Consideraciones temporales:

- El retardo de los registros T_R debe considerarse en el tiempo de ciclo:

$$\tau = \max\{\tau_i\} + T_R \quad f = \frac{1}{\tau}$$

- La segmentación penaliza el tiempo de proceso de una operación individual

$$T \leq k \times \tau$$

Archivo Formulas de prestaciones.pdf

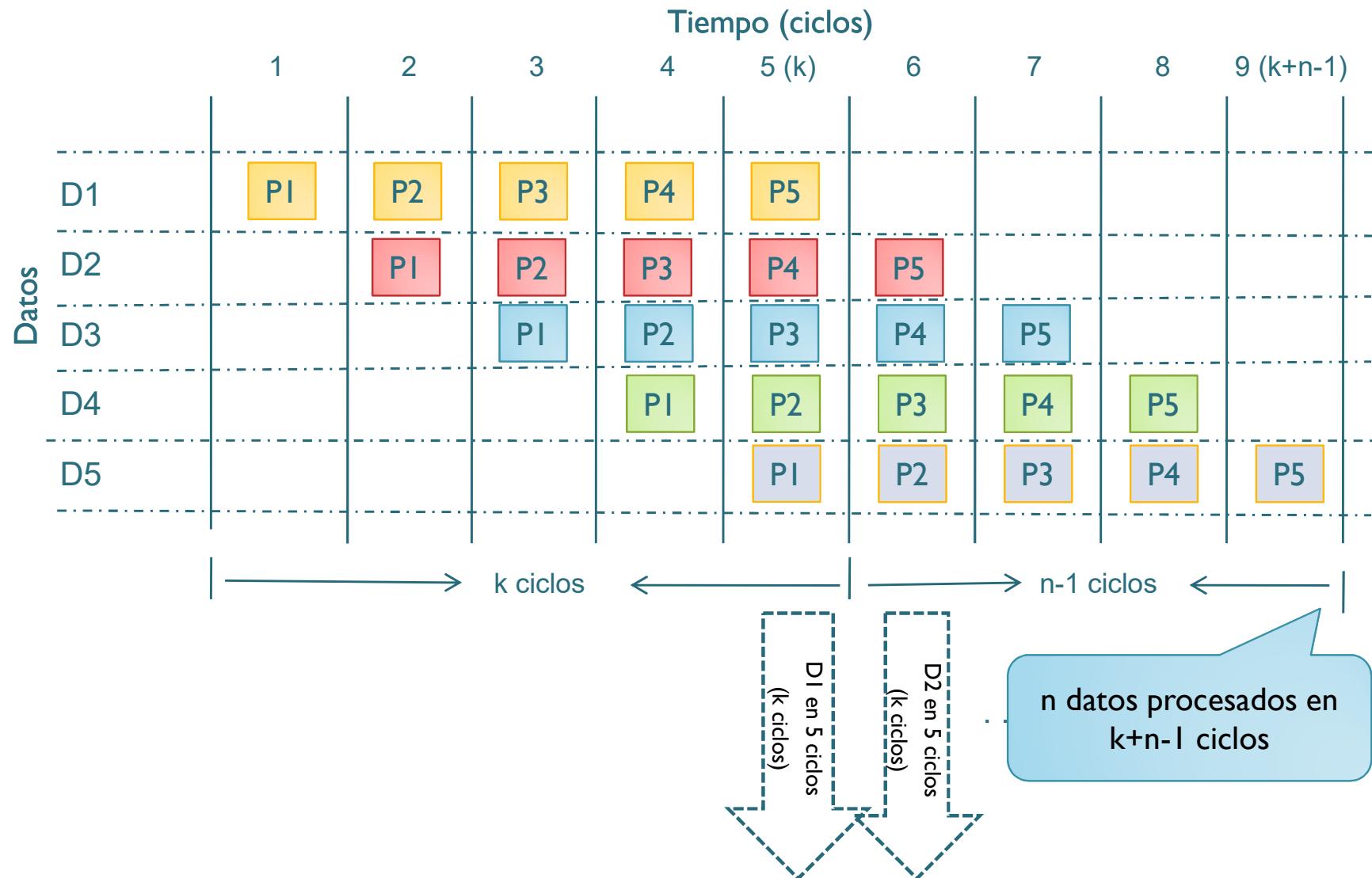
En los recursos de poliformat, dentro de la información del grupo F, en Tema 2, en la carpeta Lecturas, el archivo Formulas de prestaciones.pdf, dónde se resumen las fórmulas que se emplearán en los ejercicios de segmentación, adaptándolas a la segmentación del procesador

Fórmulación de prestaciones

	Procesador No segmentado	Procesador segmentado en k etapas
Tiempo de ciclo (segundos) Frecuencia (Hz= s ⁻¹)	$T = \tau_1 + \tau_2 + \dots + \tau_k$ $f = \frac{1}{T}$ (Hz= s ⁻¹) (k tareas independientes)	$\tau = \max\{\tau_i\} + T_R$ y $f = \frac{1}{\tau}$ (Hz= s ⁻¹) $1 \leq i \leq k$ y T_R tiempo de registros de segmentación
Tiempo de ejecución de un programa con "n" instrucciones	$T_{NS}(n) = n \times T$	$T_S(n) = \tau \times \text{ciclos}$ $T_S(n) = \tau \times \text{ciclos} = \tau \times (k + n - 1)$ (si el procesador <u>no</u> tiene paradas) $T_S(n) = \tau \times \text{ciclos} = \tau \times (p + k + n - 1)$ (si el procesador tiene "p" paradas) $T_S(n) = n \times CPI \times \tau$
CPI Ciclos por instrucción		$CPI = \frac{\text{ciclos} - \text{llenado}}{n} = \frac{\text{ciclos} - (k - 1)}{n}$

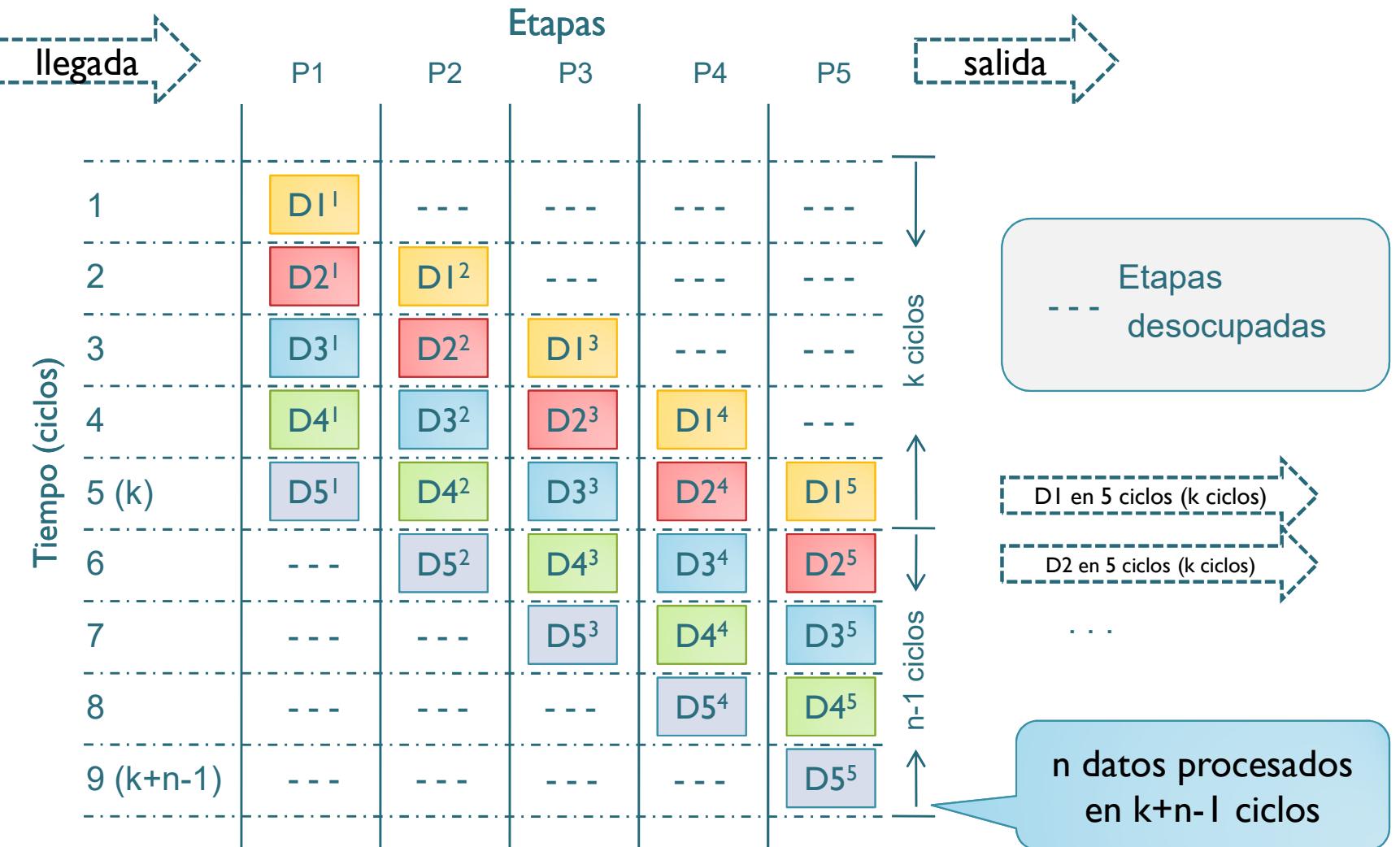
Diagramas Temporales: Diagrama Tiempo/Datos

- Ejemplo de segmentación con 5 etapas ($k=5$) y 5 datos ($n=5$)



Diagramas Temporales: Diagrama Etapas/Tiempo

- Ejemplo de segmentación con 5 etapas ($k=5$) y 5 datos ($n=5$)



DX^y representa Dato X en etapa y ($DX^y \neq DX^z$)

Por simplicidad, obviaremos a partir de ahora las diferencias de los datos entre etapas

Prestaciones de la Segmentación

- Medidas de prestaciones (n datos)

- ✓ **Productividad:** trabajo realizado por unidad de tiempo

$$X(n) = \frac{n}{t_s(n)} = \frac{n}{(n+k-1) \times \tau} \text{ datos/tiempo}$$

- Incrementa:
 - Con el número de datos
 - Con la frecuencia de reloj
- Límite teórico (n suficientemente grande):
 - Una operación por ciclo

$$X(\infty) = \frac{1}{\tau} = f$$

- ✓ **Aceleración:** Ganancia de velocidad respecto al circuito no segmentado:

$$S(n) = \frac{t_{ns}(n)}{t_s(n)} = \frac{n \times T}{(n+k-1) \times \tau}$$

$$S(\infty) = \frac{T}{\tau}$$

Prestaciones de la Segmentación

- Conclusiones

$$S(n) = \frac{t_{ns}(n)}{t_s(n)} = \frac{n \times T}{(n + k - 1) \times \tau}$$

- ✓ Como $T \leq k \cdot \tau$ si $n=1$ entonces $S \leq 1$: la segmentación no beneficia
- ✓ Cuanto mayor sea n (datos a procesar), mayores beneficios (S)
 - Si $n \rightarrow \infty$, entonces $S \rightarrow T/\tau \leq k$
- ✓ **Ganancia teórica máxima**
 - Si $T = k \cdot \tau$ (caso ideal), la ganancia sería $S = k$. Ello se cumpliría sólo si todas las etapas tuvieran igual retardo (τ_i) y el retardo de los registros de segmentación (T_R) fuera igual a cero
 - En principio, y con suficientes datos, a menor τ mayor aceleración

Archivo Formulas de prestaciones.pdf

		CPI = 1 (si no hay paradas) CPI = $(n+p)/n = 1 + p/n$ si hay "p" paradas
Productividad REAL	$\chi(n)$	$\chi_{NS}(n) = \frac{n}{n \times T}$
	$\chi(n) = \frac{\text{instrucciones}}{\text{tiempo}}$	$\chi_S(n) = \frac{n}{\tau \times \text{ciclos}}$ $\chi_S(n) = \frac{n}{\tau \times (k+n-1)}$ (SIN PARADAS) $\chi_S(n) = \frac{n}{\tau \times (p+k+n-1)}$ (CON "p" PARADAS)
Productividad MAXIMA	χ	$\chi_{NS} = \frac{1}{T} = f$
Aceleración REAL	$S(n)$	$S(n) = \frac{T_{NS}(n)}{T_S(n)} = \frac{n \times T}{(k+n-1) \times \tau}$ (SIN PARADAS) y $S(n) = \frac{T_{NS}(n)}{T_S(n)} = \frac{n \times T}{(p+k+n-1) \times \tau}$ (CON "p" PARADAS)
Aceleración MAXIMA	S	$S = \frac{\chi_S}{\chi_{NS}} = \frac{T}{\tau}$
		Aceleración IDEAL $S = k$ Para ruta con k etapas

Prestaciones de la Segmentación

- Consideraciones temporales

- ✓ El período mínimo de reloj es $\tau = \max(\tau_i) + T_R$
- ✓ Para incrementar la productividad ($1/\tau$) se reducirá:
 - el retardo máximo de etapa (aumentando k y manteniendo los retardos equilibrados)
 - el retardo de los registros

Contenidos y Bibliografía

- El proceso de segmentación
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ **Identificación de las fases de ejecución de las instrucciones**
 - ✓ Diseño de las etapas (excepto beq)
 - ✓ Cálculo del tiempo de ciclo
 - ✓ Añadiendo instrucción beq a la ruta
- Conflictos y riesgos
 - ✓ Conflictos de datos
 - ✓ Conflictos de control (flujo)
- Perspectiva histórica. RISC-CISC y otras alternativas



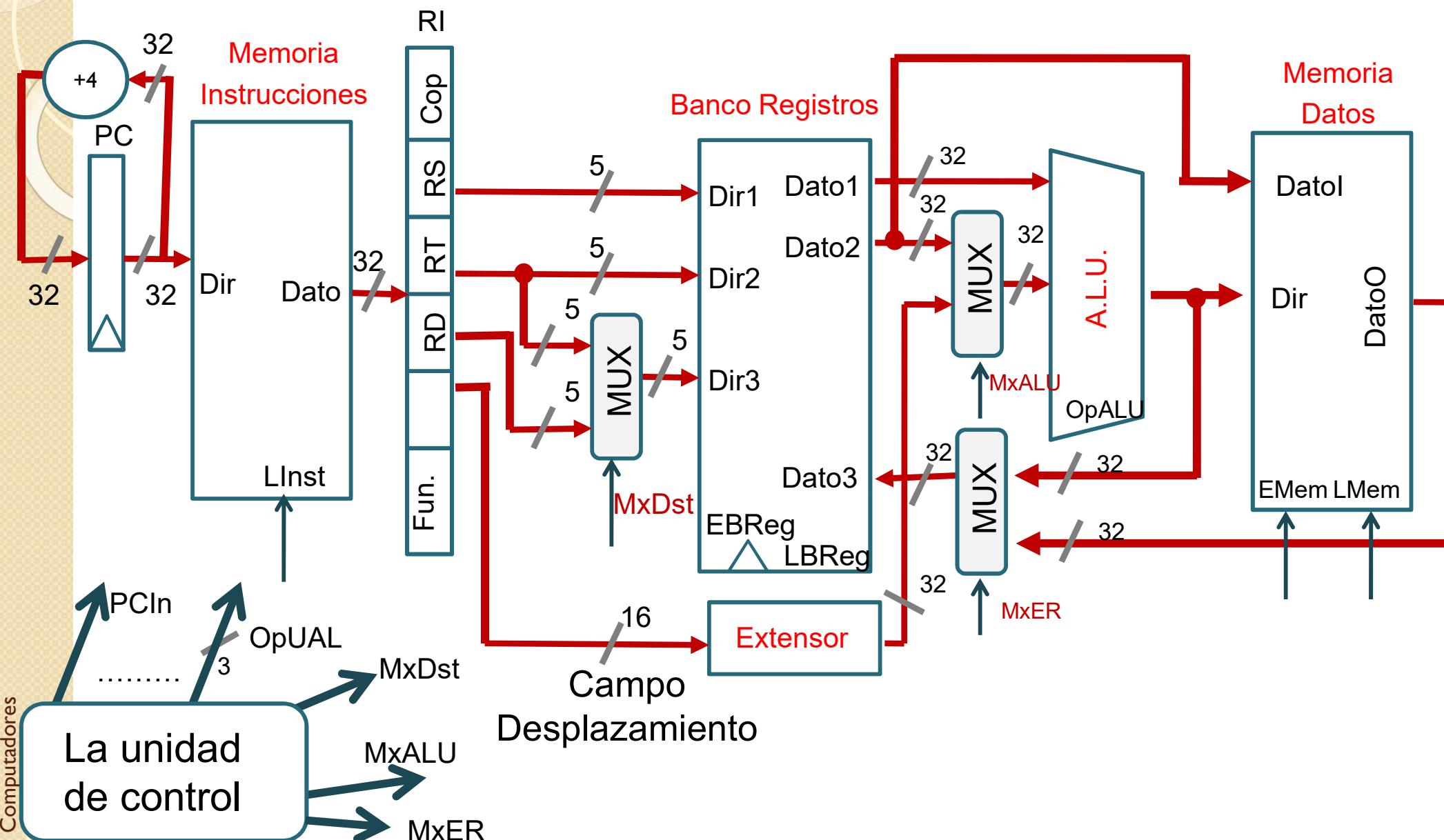
Vídeos 1 y 2 repasan este apartado



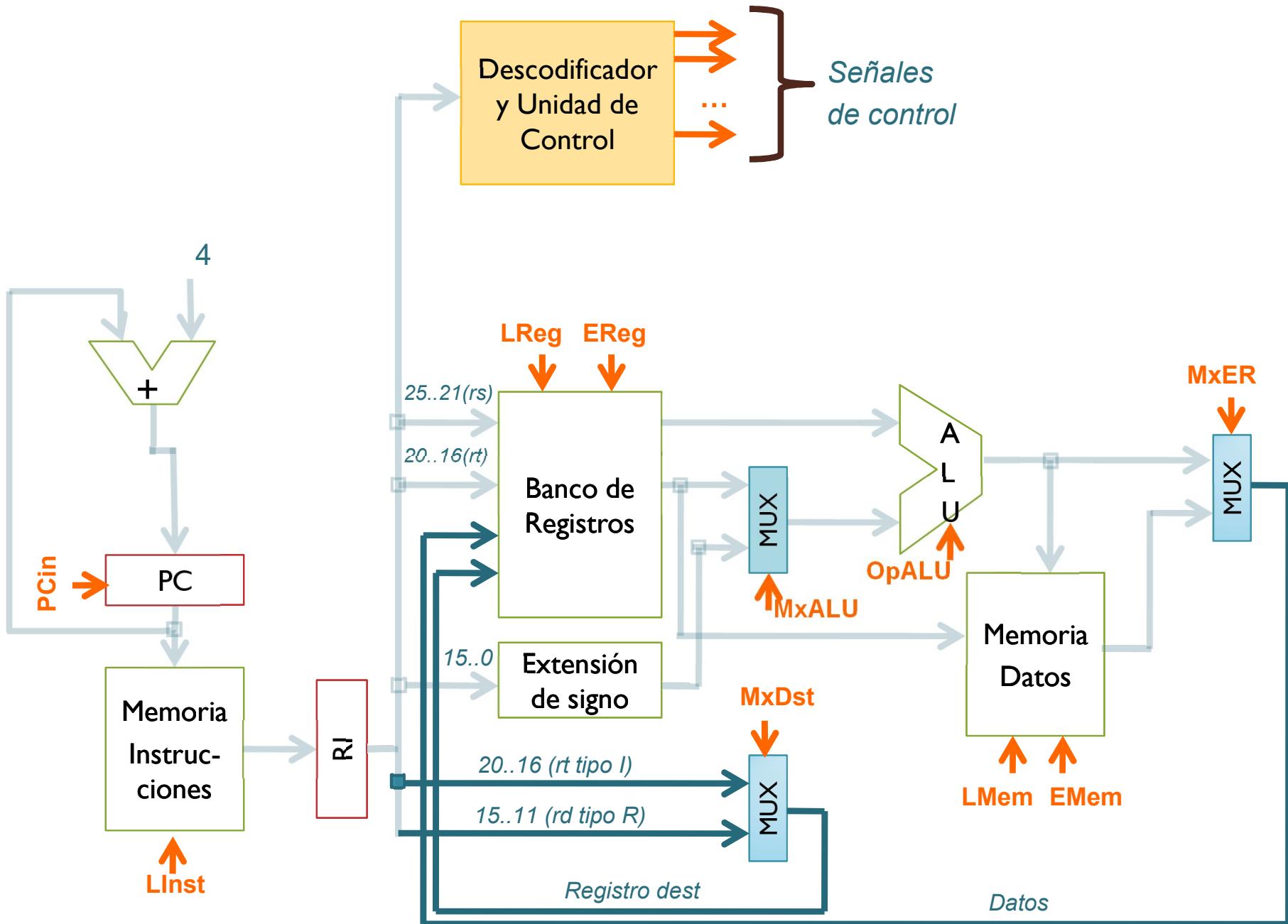
Segmentación de la ruta de datos básica

- Equivalencia entre el dibujo de la ruta de datos vista en tema I, y la que se va a trabajar en el tema 2.
 - ✓ Se empieza con la ruta de los datos sin las instrucciones de salto, para eliminar complejidad
- Identificación de las etapas para la ejecución de las instrucciones aritméticas tipo R e I estudiadas en el Tema I y las de carga lw y sw
- Diseño de las etapas de la ruta de los datos
- Descripción detallada de acciones en cada etapa

TEMA 1: Ruta de los datos aritm/lóg R, addi/slti, y lw/sw



TEMA 2: Ruta de los datos aritm/lóg R, addi/slti, y lw/sw

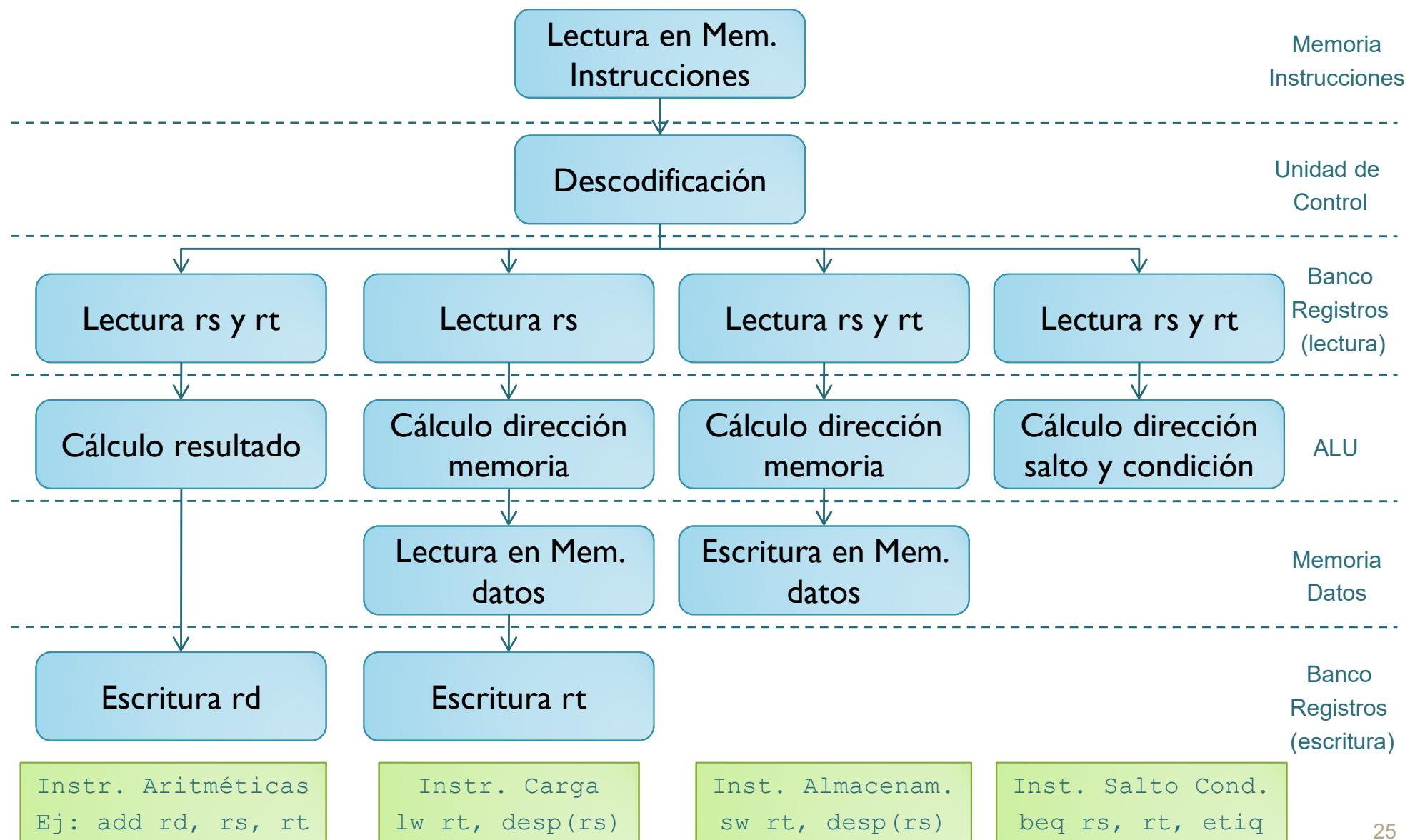


Segmentación de la ruta de datos básica

- Equivalencia entre el dibujo de la ruta de datos vista en tema 1, y la que se va a trabajar en el tema 2.
 - ✓ Se empieza con la ruta de los datos sin las instrucciones de salto, para eliminar complejidad
- Identificación de las etapas para la ejecución de las instrucciones aritméticas tipo R e I estudiadas en el Tema 1 y las de carga lw y sw (están las de salto, pero focalizaremos en las demás)
- Diseño de las etapas de la ruta de los datos
- Descripción detallada de acciones en cada etapa

Segmentación de la Ruta de Datos

- Especificación del ciclo de instrucción
 - ✓ Distribución en etapas de los componentes de la ruta

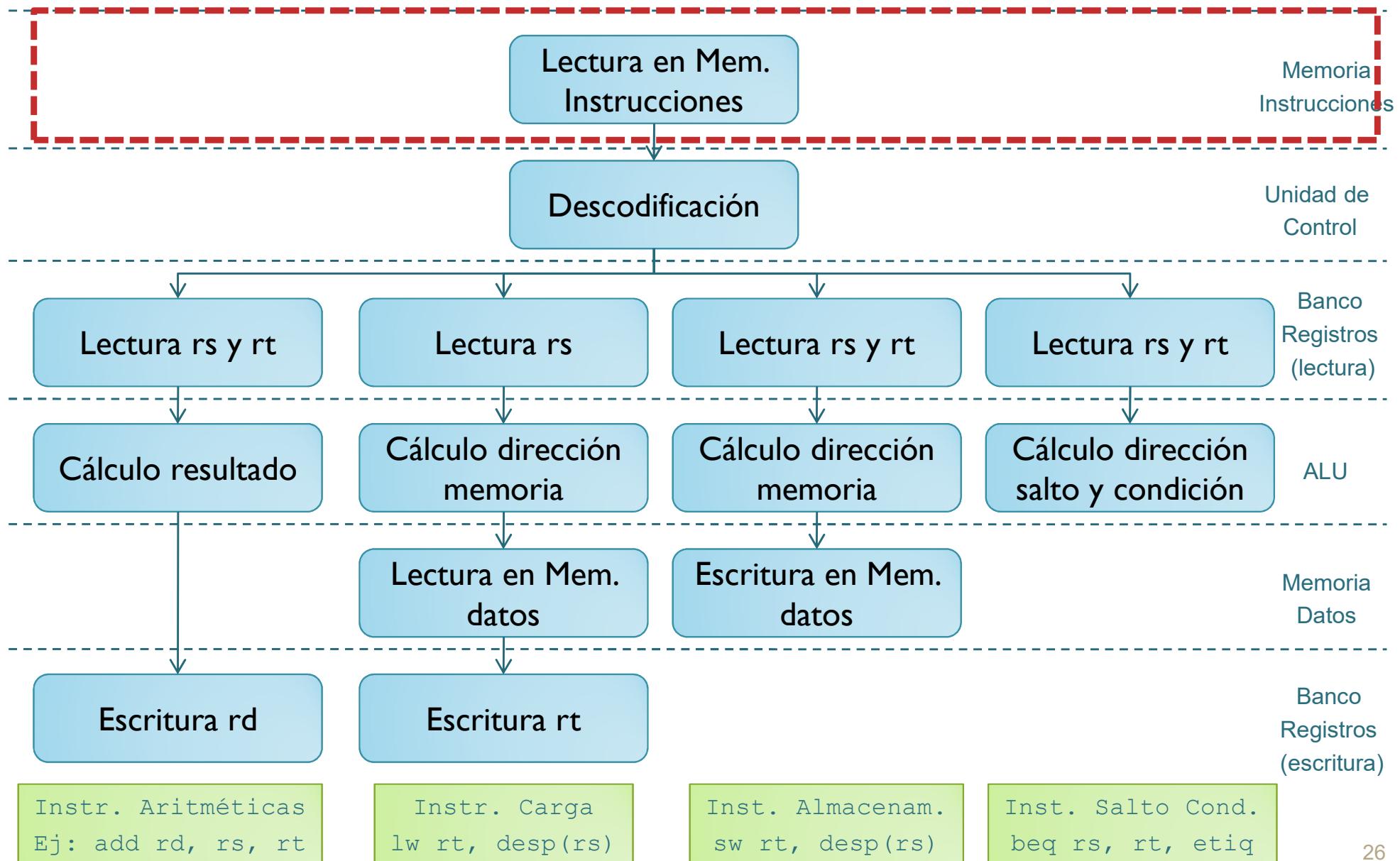


Segmentación de la Ruta de Datos

- Especificación del ciclo de instrucción

- ✓ Distribución en etapas de los componentes de la ruta

LI :Lectura Instrucción

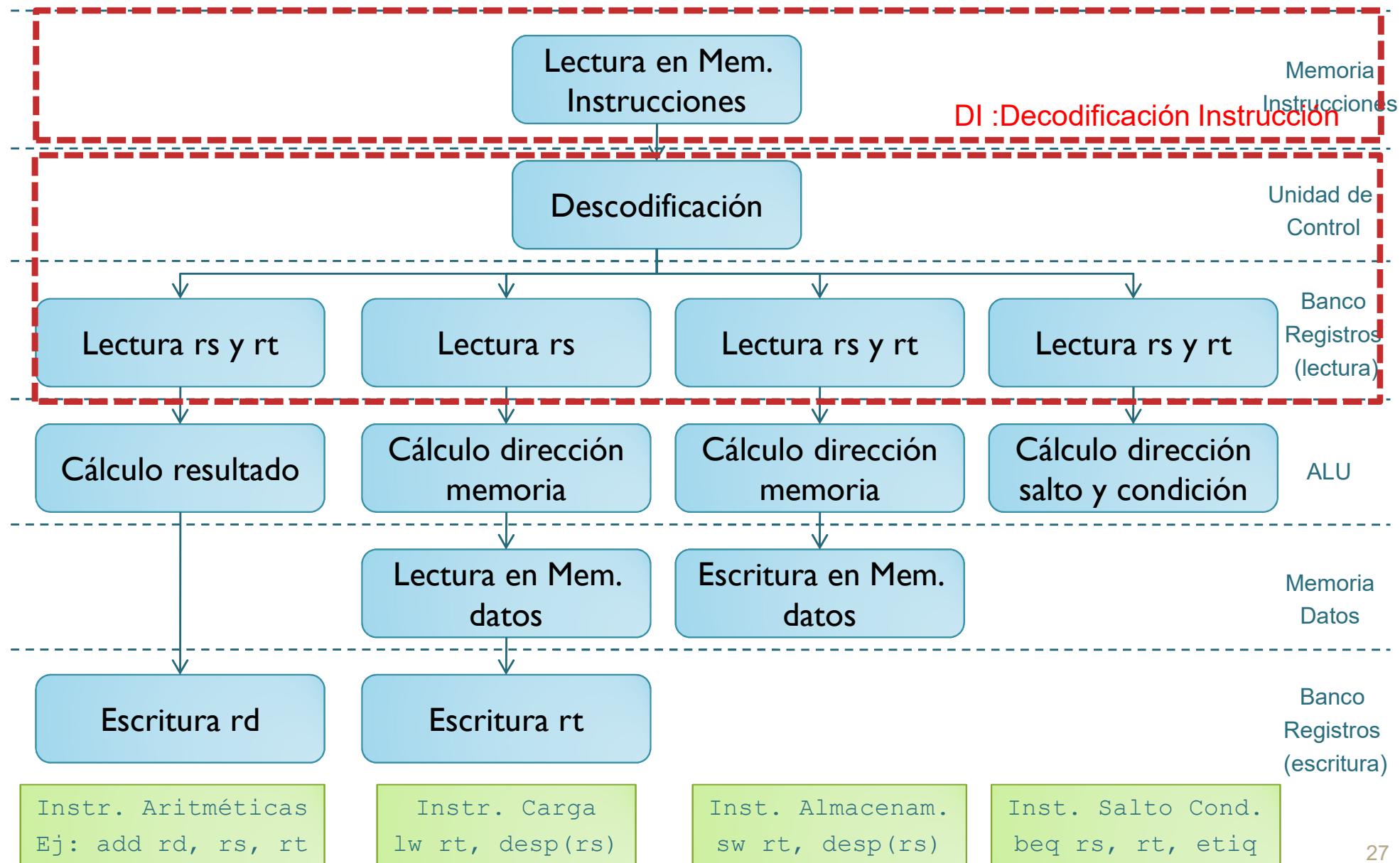


Segmentación de la Ruta de Datos

- Especificación del ciclo de instrucción

- ✓ Distribución en etapas de los componentes de la ruta

LI :Lectura Instrucción

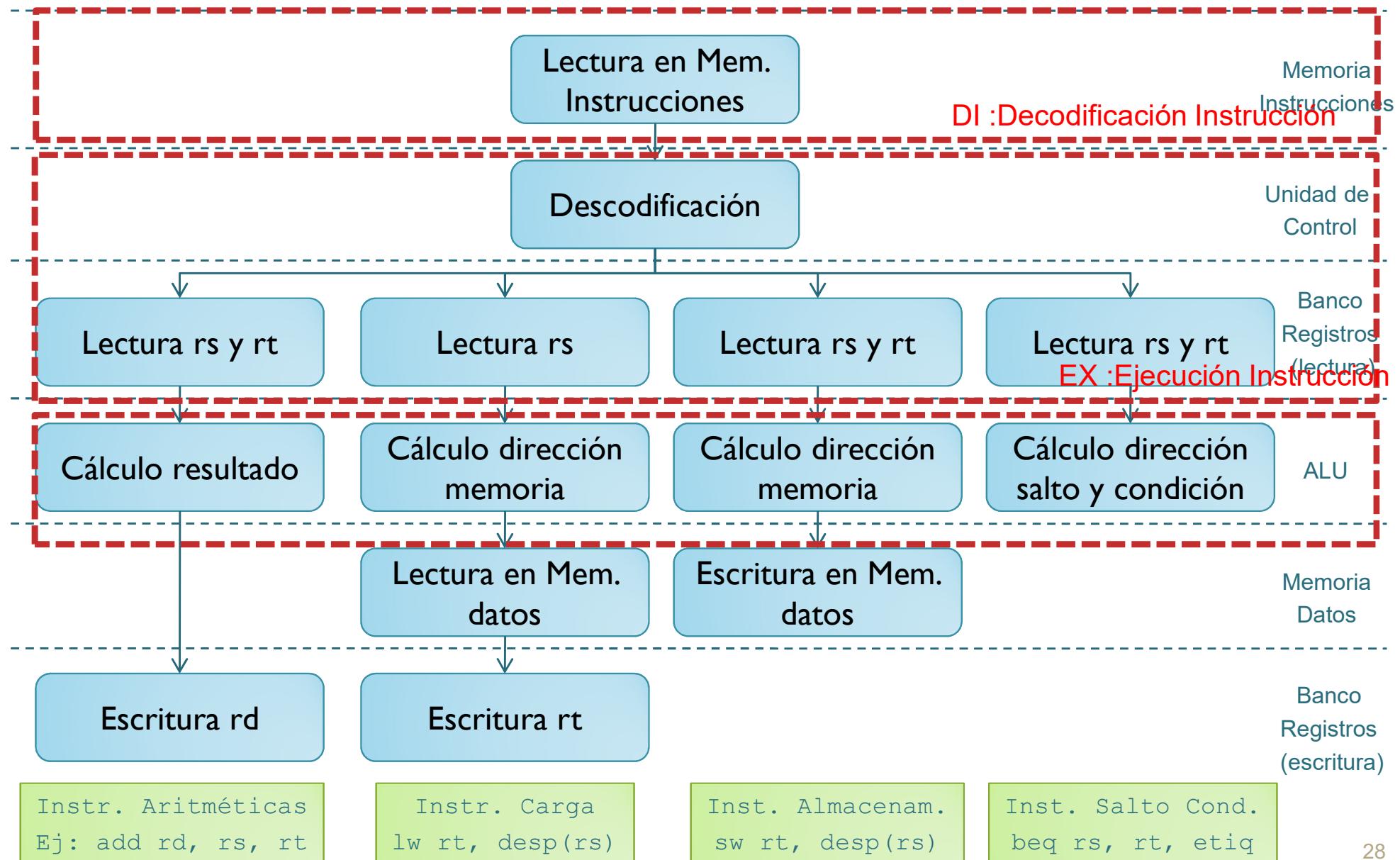


Segmentación de la Ruta de Datos

- Especificación del ciclo de instrucción

- ✓ Distribución en etapas de los componentes de la ruta

LI :Lectura Instrucción

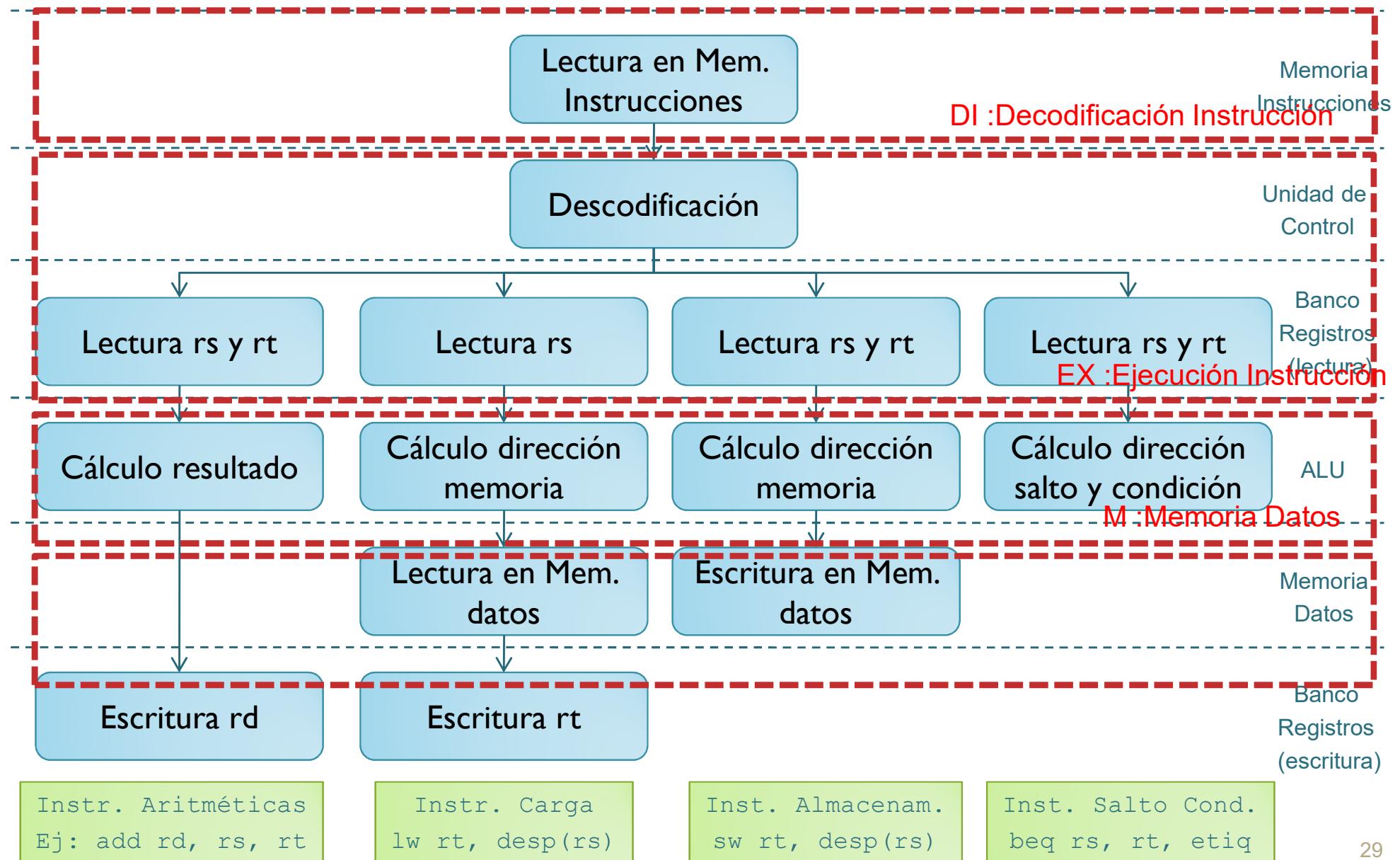


Segmentación de la Ruta de Datos

- ## • Especificación del ciclo de instrucción

- ✓ Distribución en etapas de los componentes de la ruta

LI :Lectura Instrucción



Segmentación de la Ruta de Datos

- Especificación del ciclo de instrucción

- ✓ Distribución en etapas de los componentes de la ruta

LI :Lectura Instrucción

Lectura en Mem.
Instrucciones

Descodificación

Lectura rs y rt

Lectura rs

Lectura rs y rt

DI :Decodificación Instrucción

Memoria
Instrucciones

Unidad de
Control

Banco
Registros
(lectura)

Cálculo resultado

Cálculo dirección
memoria

Cálculo dirección
memoria

Lectura rs y rt
EX :Ejecución Instrucción

ALU

Cálculo dirección
salto y condición

M :Memoria Datos

Lectura en Mem.
datos

Escritura en Mem.
datos

Memoria
Datos

Escritura rd

Escritura rt

ER :Escritura Registros

Instr. Aritméticas
Ej: add rd, rs, rt

Instr. Carga
lw rt, desp(rs)

Inst. Almacenam.
sw rt, desp(rs)

Inst. Salto Cond.
beq rs, rt, etiq

Banco
Registros
(escritura)

Segmentación de la ruta de datos básica

- Equivalencia entre el dibujo de la ruta de datos vista en tema I, y la que se va a trabajar en el tema 2.
 - ✓ Se empieza con la ruta de los datos sin las instrucciones de salto, para eliminar complejidad
- Identificación de las etapas para la ejecución de las instrucciones aritméticas tipo R e I estudiadas en el Tema I y las de carga lw y sw
- Diseño de las etapas de la ruta de los datos
- Descripción detallada de acciones en cada etapa

Segmentación de la Ruta de Datos

- Definición de las etapas

- ✓ Etapas comunes a todas las instrucciones

- **LI:** Etapa de lectura de instrucción (e incremento del PC)
 - **DI:** Etapa de decodificación de instrucción (y lectura de registros)

- ✓ Etapas que dependen del tipo de instrucción

- **EX:** Etapa de ejecución
 - Instrucciones de cálculo: cálculo del resultado
 - Load y Store: cálculo dirección de memoria
 - Instrucciones de salto: Cálculo de la dirección de salto y de la condición de salto
 - **M:** Etapa de memoria
 - Load y Store: acceso a la memoria
 - Instrucciones de cálculo y salto: nada
 - **ER:** Etapa de escritura de registro
 - Instrucciones de cálculo y Load : escritura del registro
 - Store e instrucciones de salto: nada

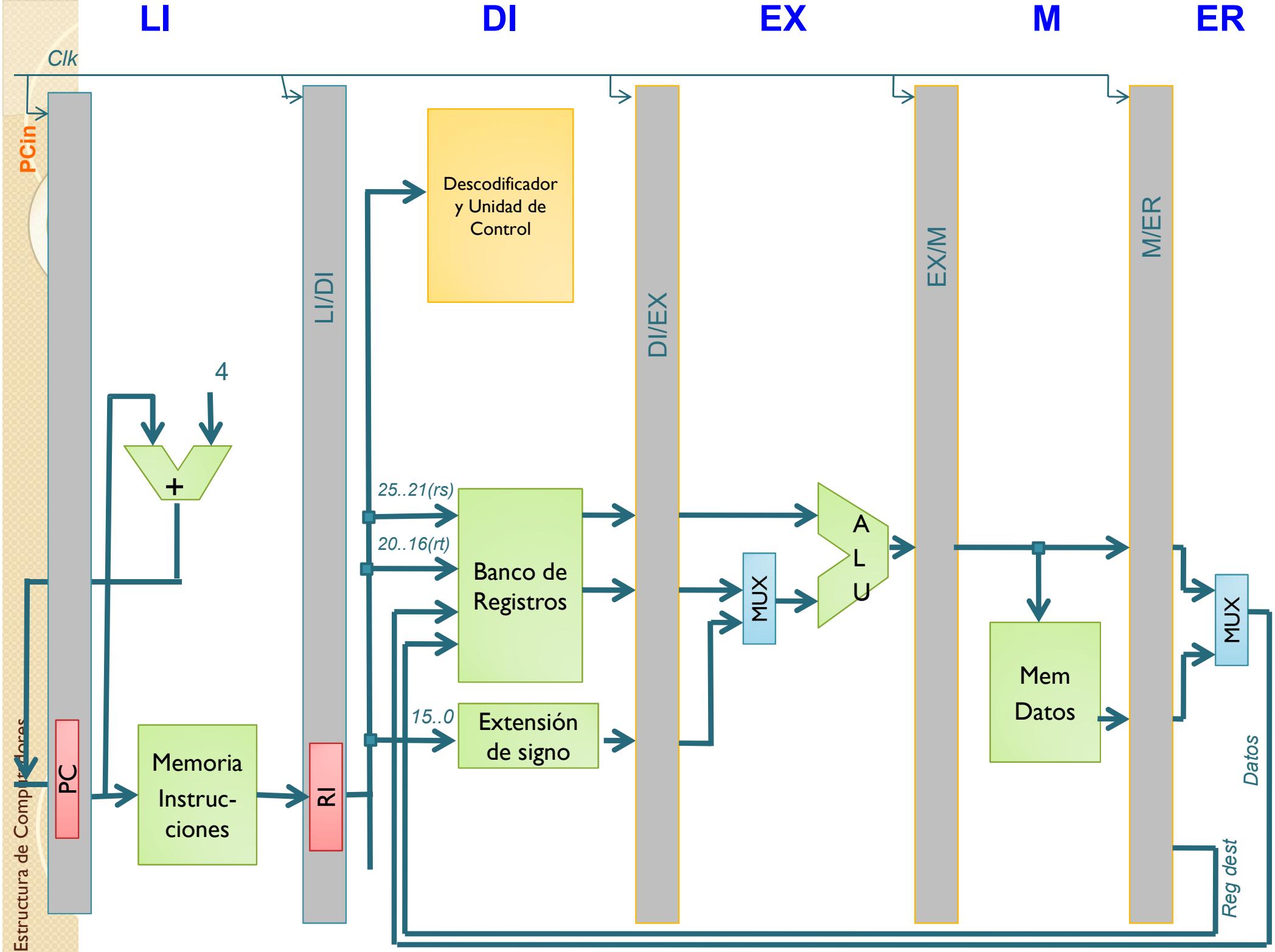
Segmentación de la Ruta de Datos

- Diseño básico del control

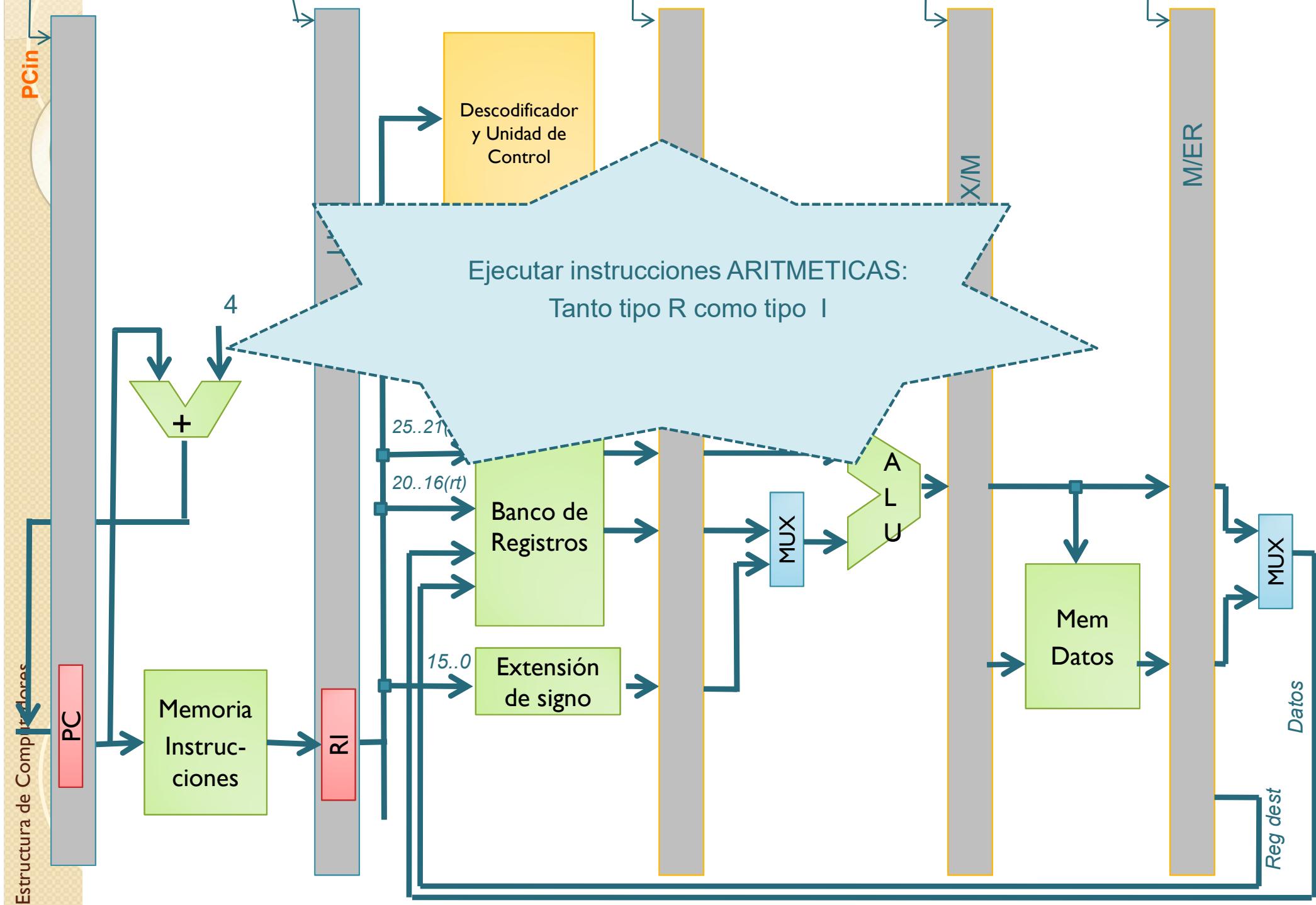
- ✓ Objetivo general: Conseguir que las cinco etapas funcionen ordenadamente y puedan ejecutar las instrucciones escogidas

- ✓ Observaciones

- Cada etapa debe de ser autónoma
 - Las dos primeras etapas, LI y DI, procesan instrucciones no descodificadas.
 - Sus señales de control serán las mismas durante todos los ciclos
 - La etapa DI se encargará de calcular las señales de control de las etapas posteriores y las transferirá a las etapas siguientes
 - Las etapas restantes deben procesar las instrucciones en función del código de operación
 - Las señales de control coinciden con las de la ruta no segmentada. De hecho, las señales de control dependen exclusivamente de la instrucción ejecutada y no de la ruta de datos



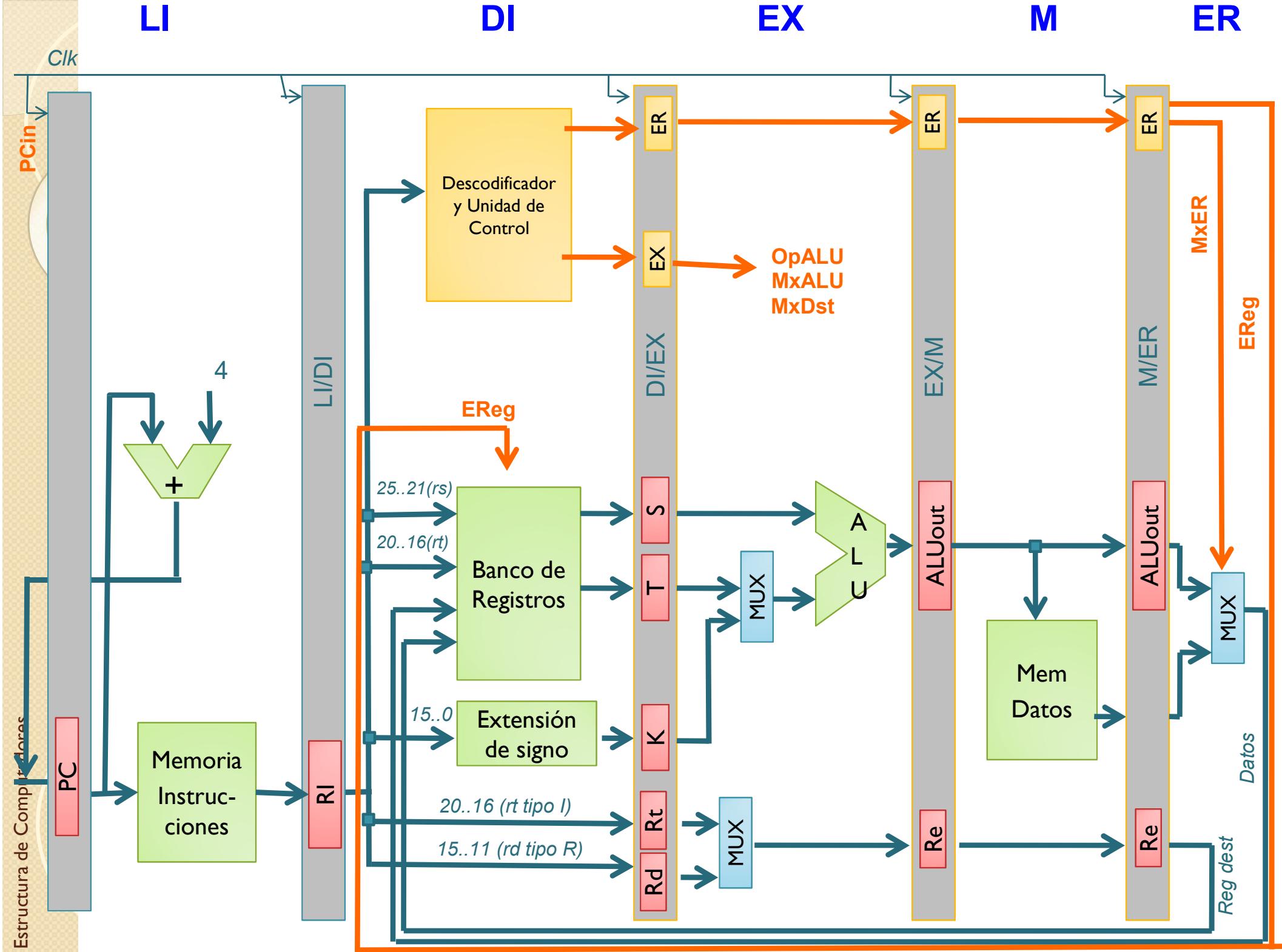
MMD1 LI DI EX M ER

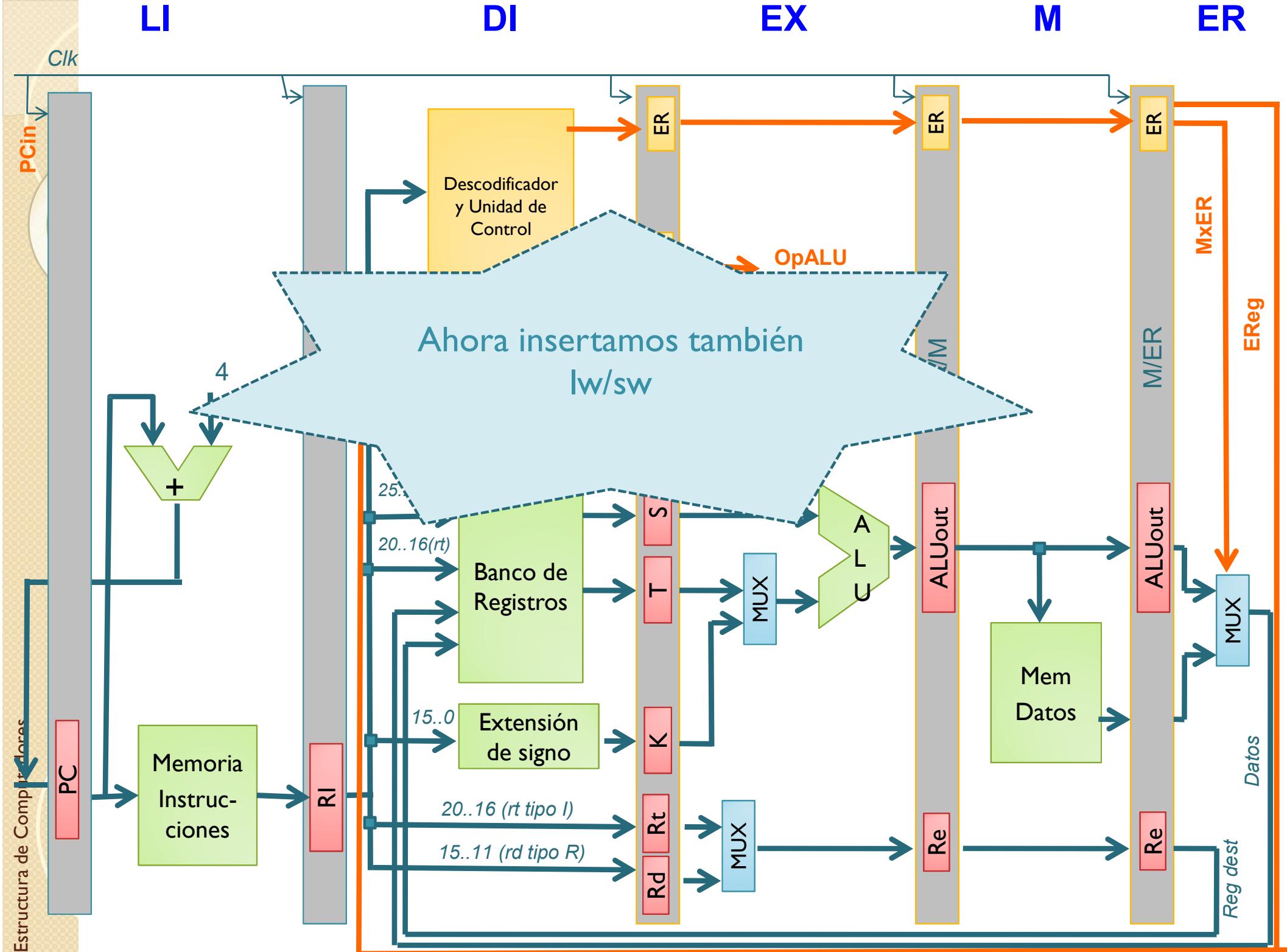


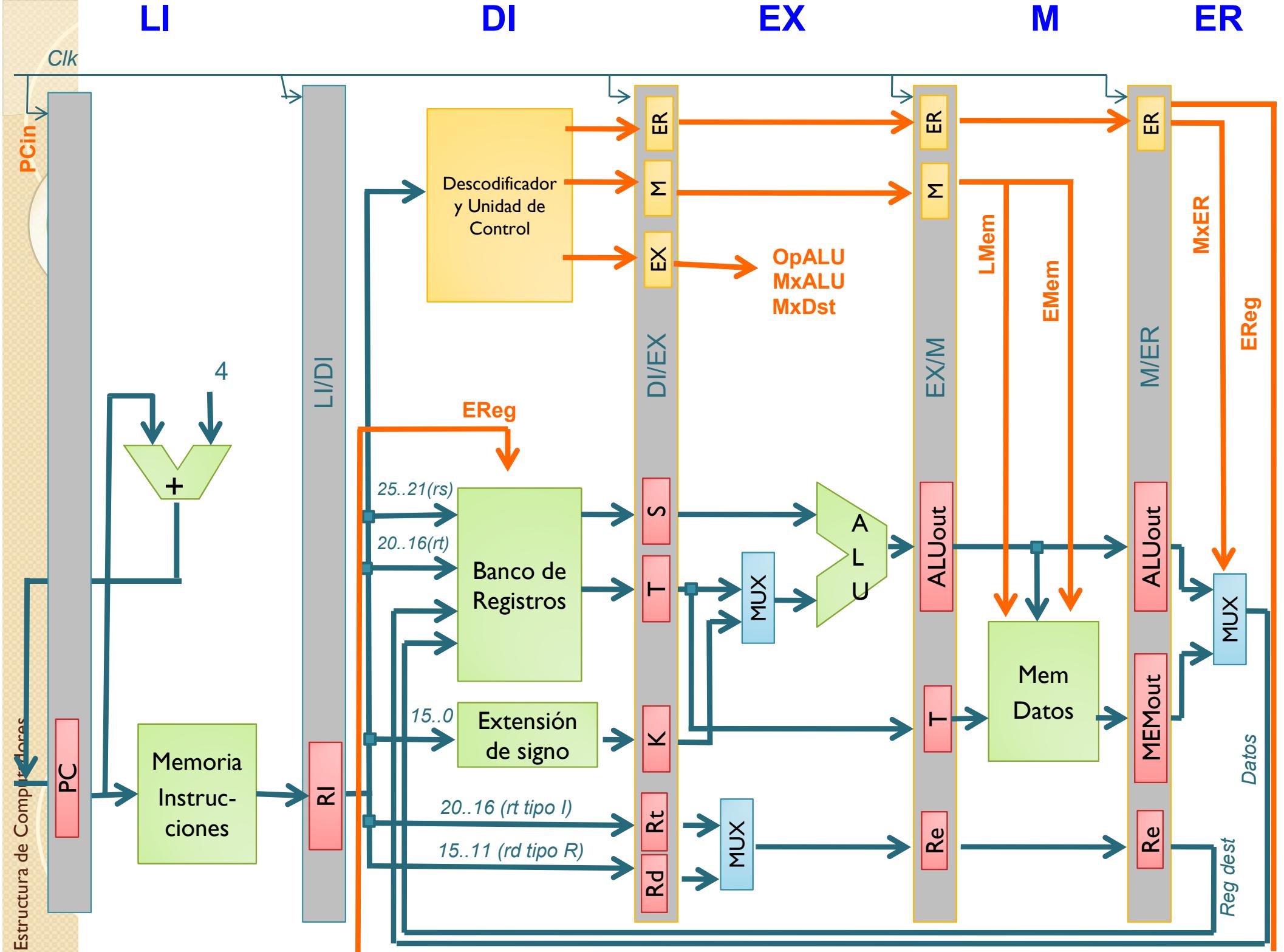
Diapositiva 35

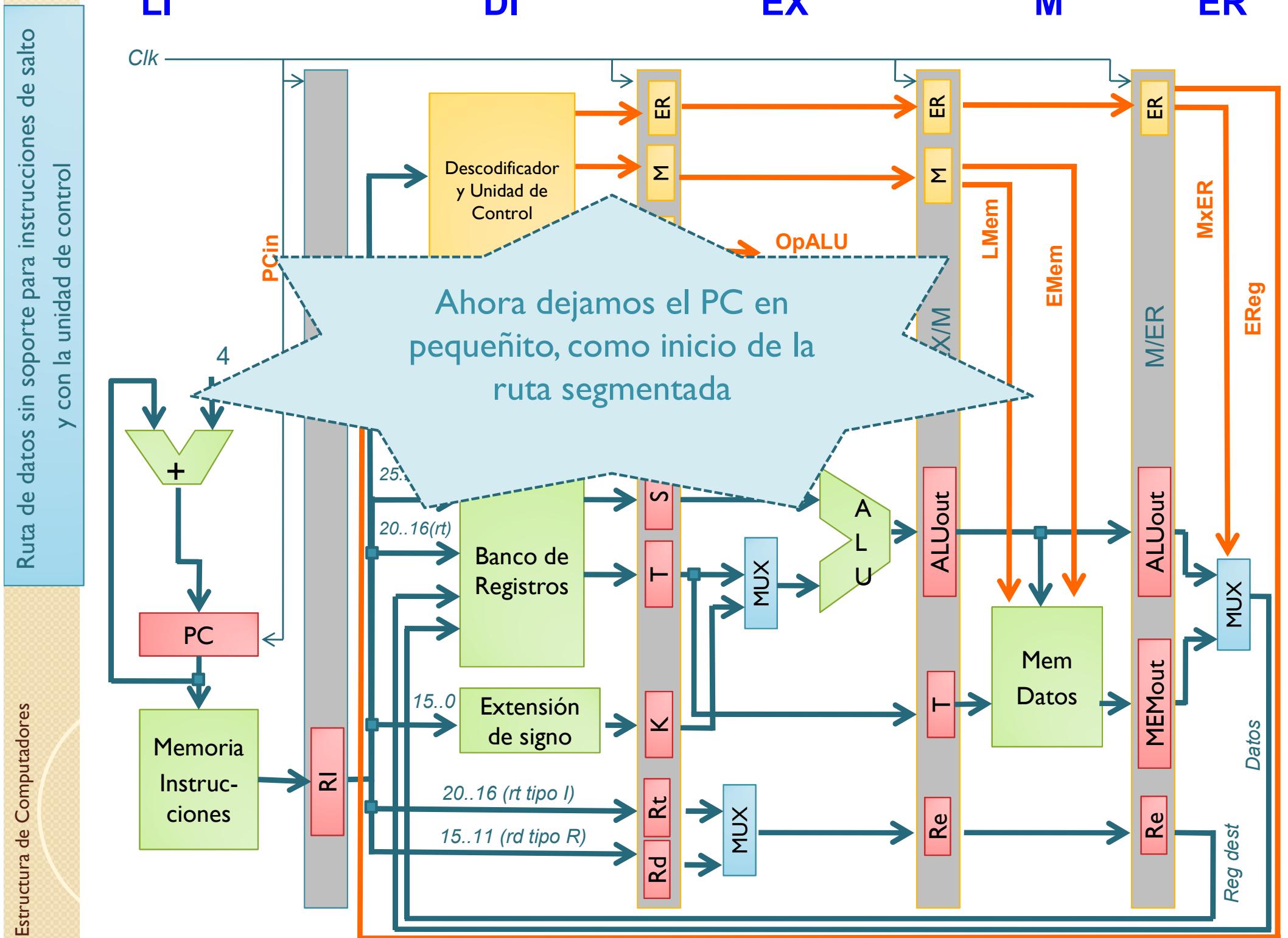
MMD1

Mila Martínez Díaz; 28/09/2020



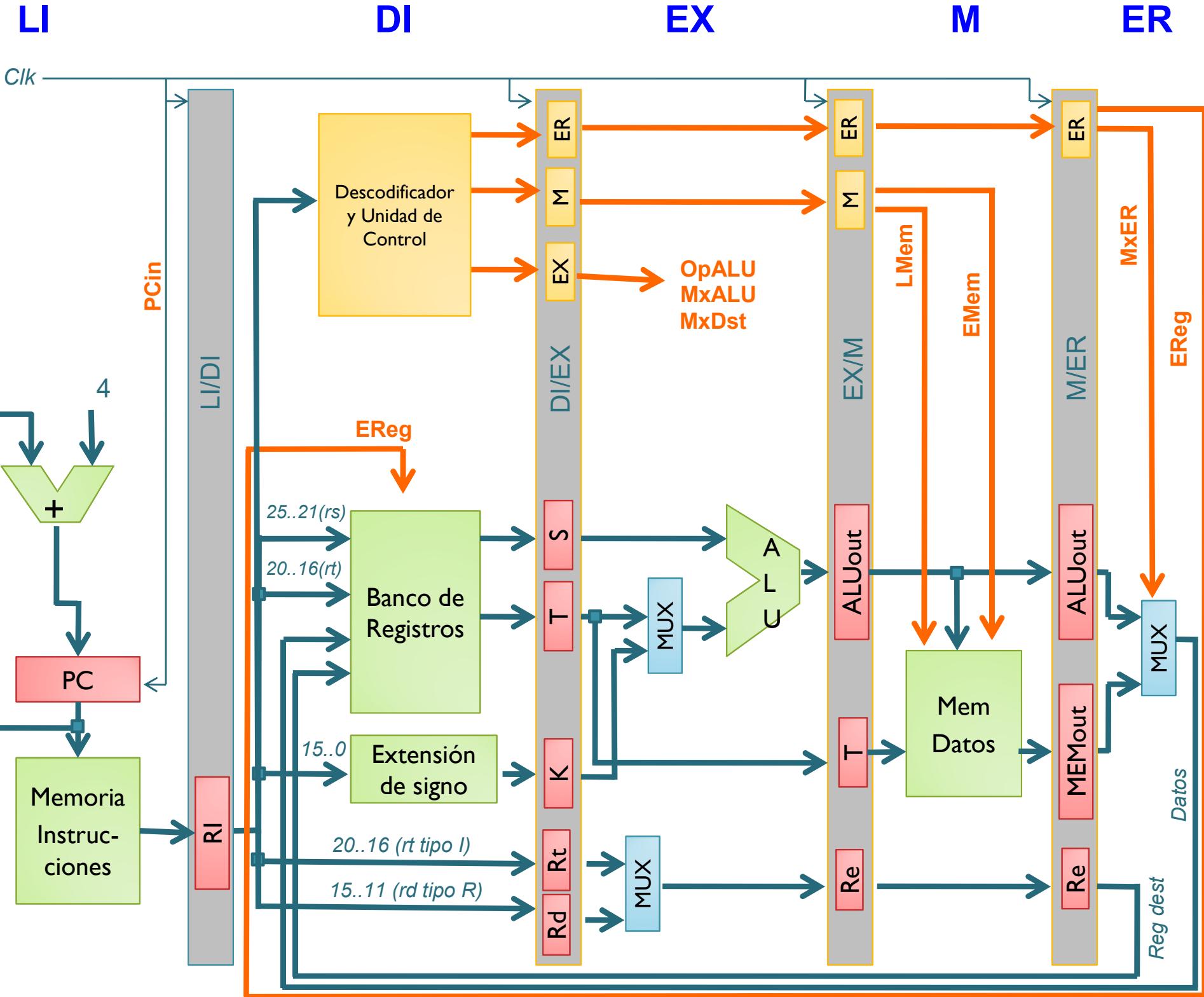






Estructura de Computadores

Ruta de datos sin soporte para instrucciones de salto
y con la unidad de control



Ejercicio I:T2_EjerciciosdePrestaciones.docx

Considere la siguiente secuencia de instrucciones:

etiqueta :	lw \$4, 2000(\$0)	(1)
	addi \$1, \$2, 10	(2)
	sw \$2, 1004(\$3)	(3)
	sub \$3, \$2, \$5	(4)

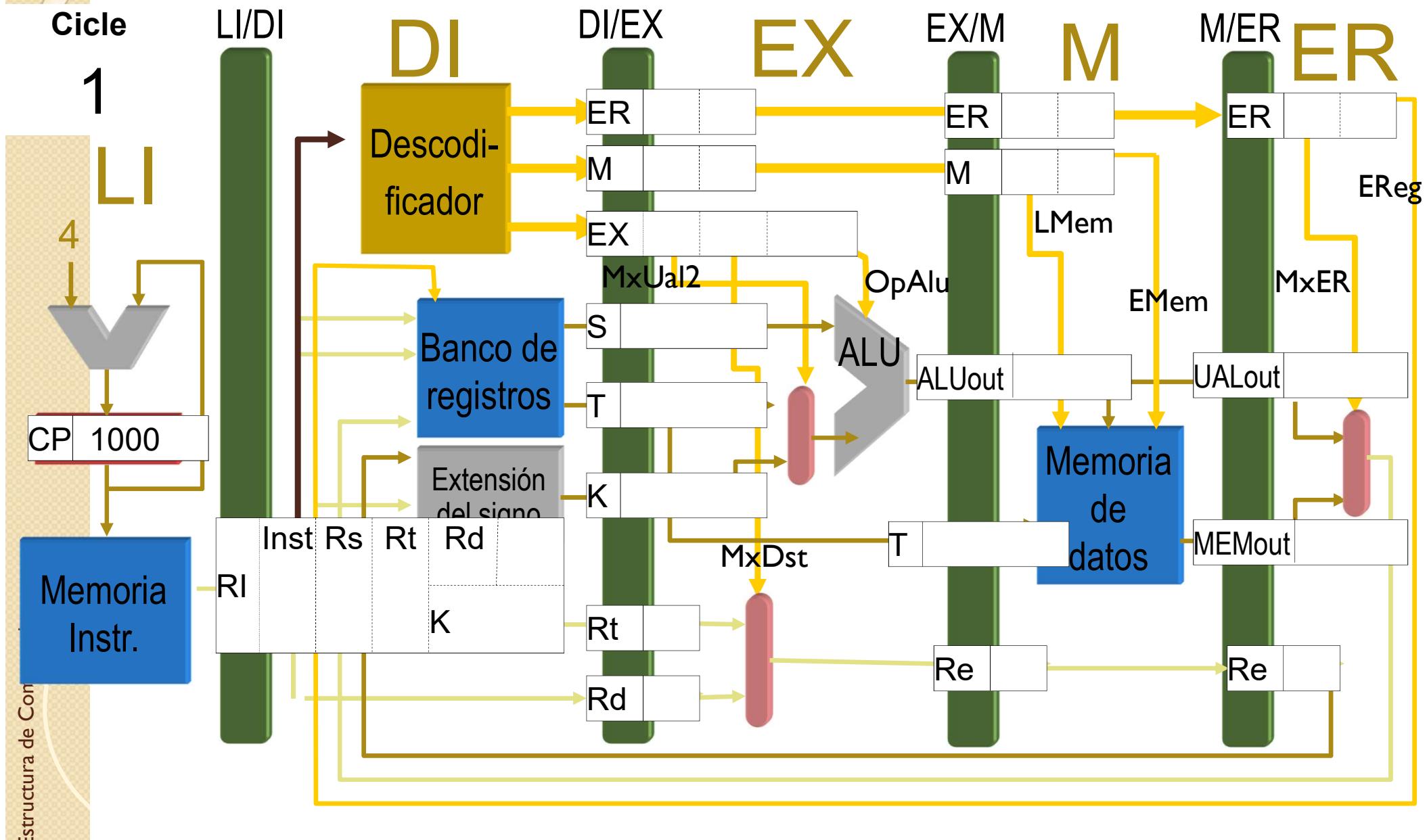
¿Cómo se ejecutaría este conjunto de instrucciones sobre la anterior ruta segmentada?

Asuma que etiqueta = 1000 y que la ruta está vacía



Primer ciclo: Se está accediendo a la instrucción lw, en dirección 1000

lw \$4,2000(\$0)

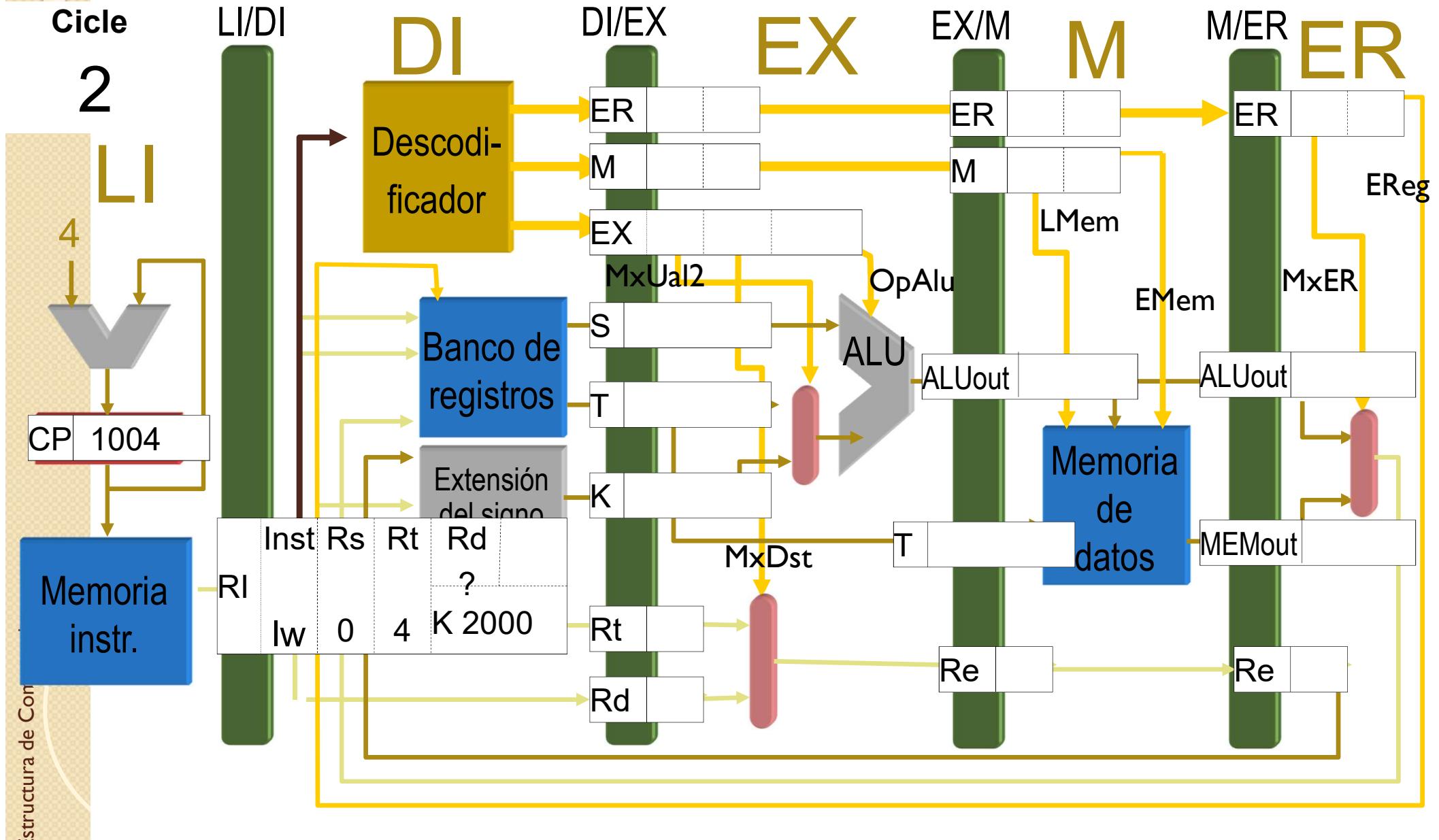


$\$0=0, \$1=4, \$2=20, \$3=1000, \$4=20, \$5=7, (2000)=60, (2004)=70$

addi \$1,\$2,10

lw \$4,2000(\$0)

En el segundo ciclo, lw en etapa DI, y en etapa LI instrucción addi

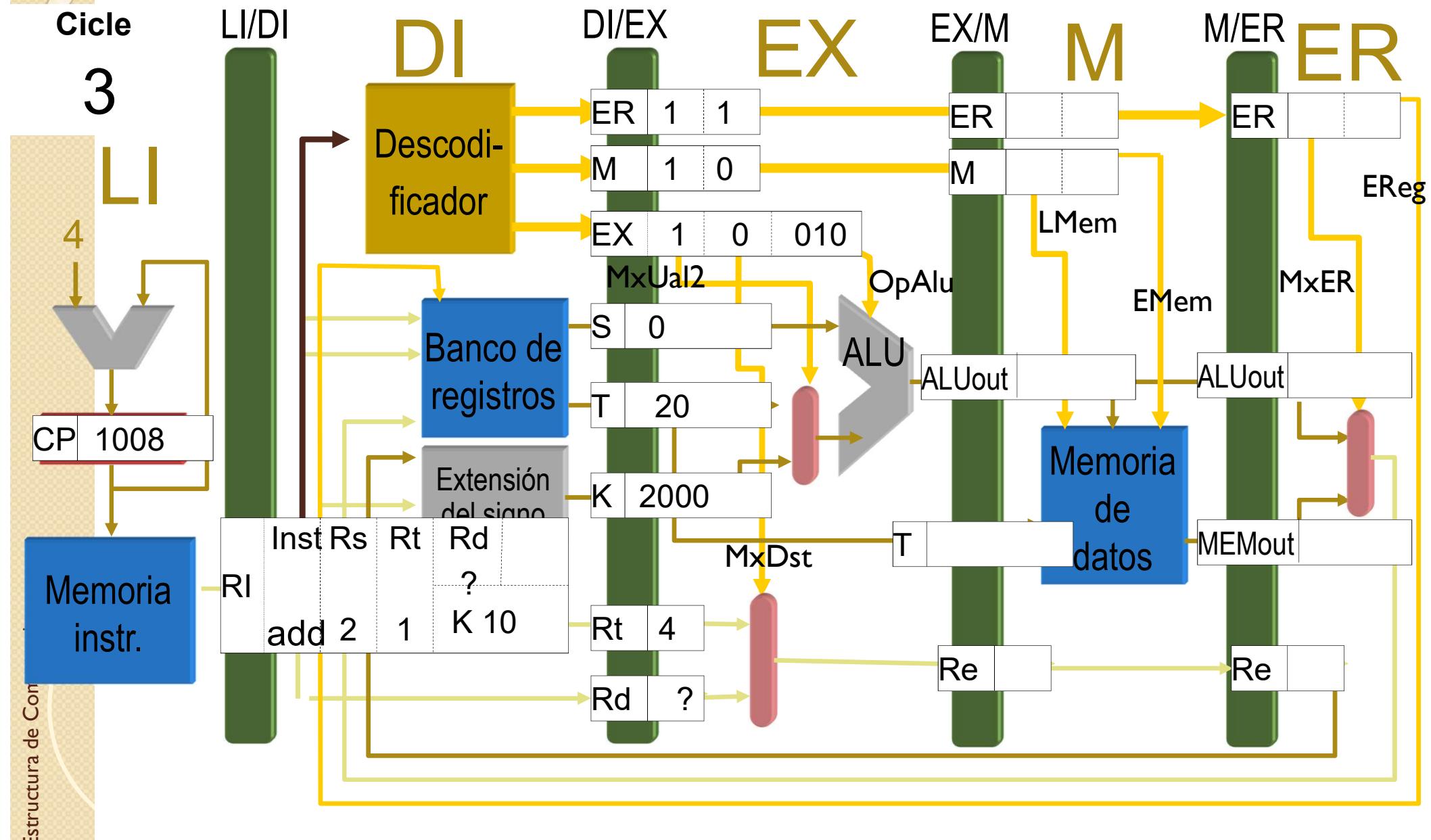


$\$0=0, \$1=4, \$2=20, \$3=1000, \$4=20, \$5=7, (2000)=60, (2004)=70$

sw \$2,1004(\$3)

addi \$1, \$2, 10

lw \$4,2000(\$0)



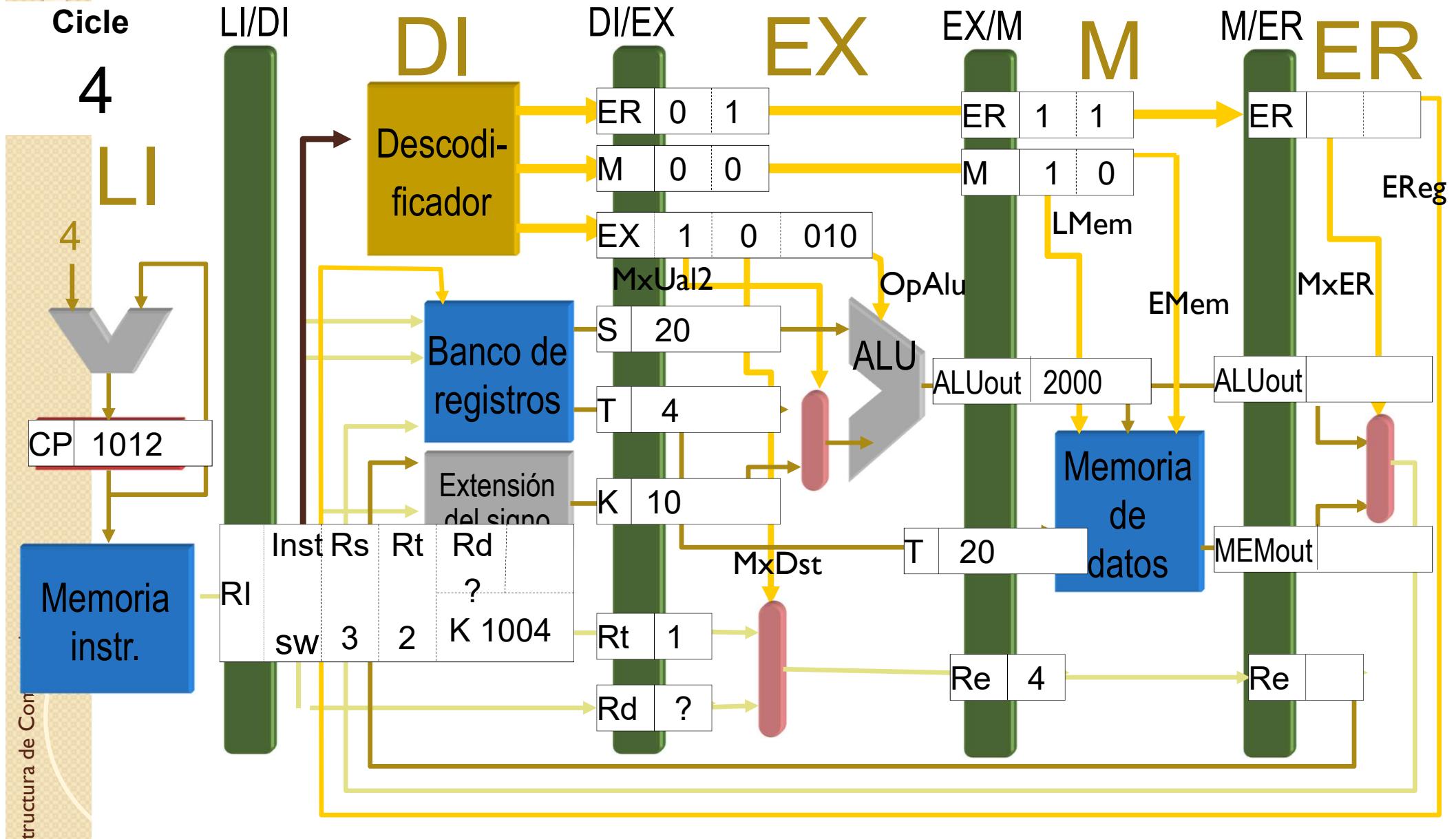
$\$0=0$, $\$1=4$, $\$2=20$, $\$3=1000$, $\$4=20$, $\$5=7$, $(2000)=60$, $(2004)=70$

sub \$3, \$2, \$5

sw \$2, 1004(\$3)

addi \$1, \$2, 10

|w \$4,200(\$0)



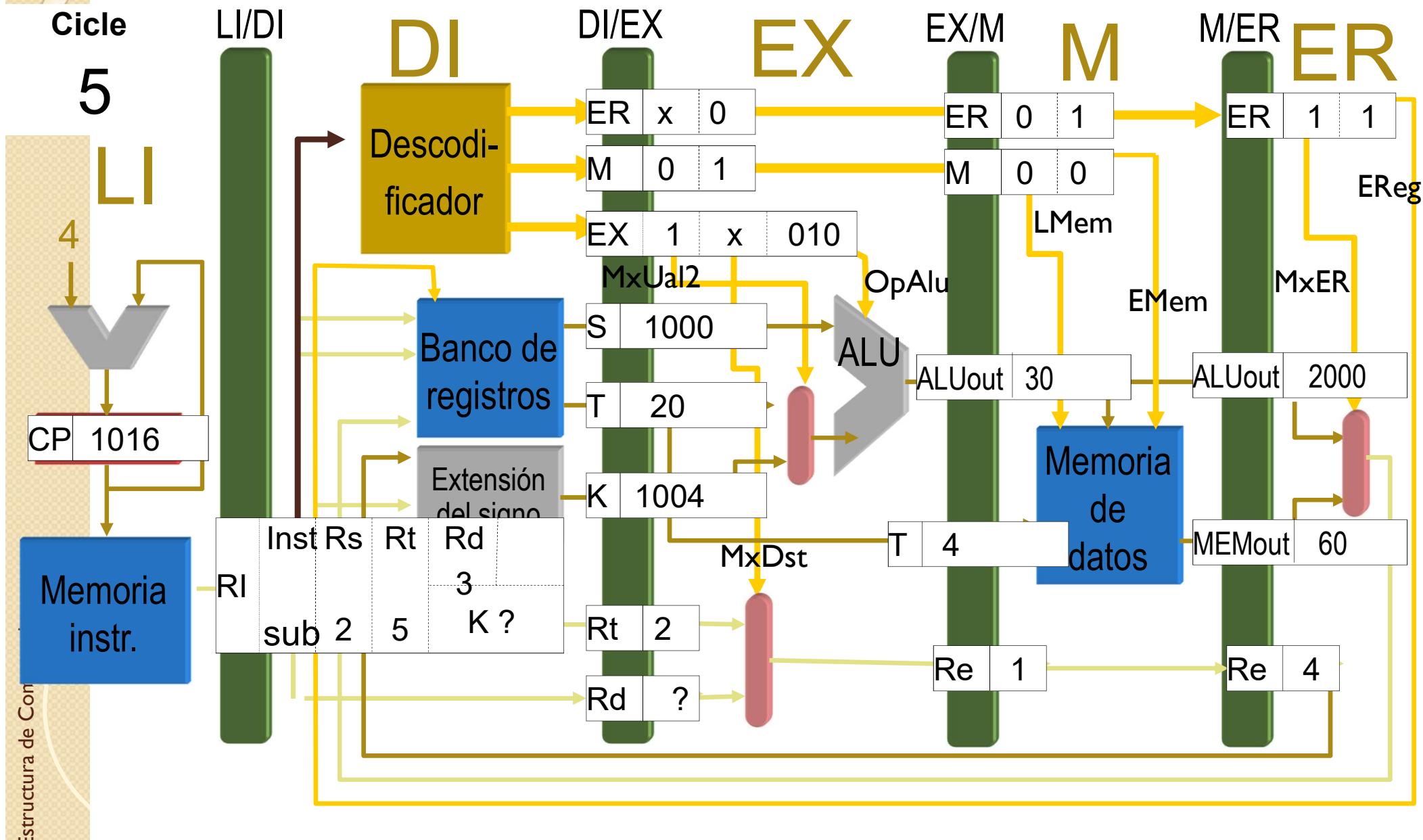
$\$0=0, \$1=4, \$2=20, \$3=1000, \$4=60, \$5=7, (2000)=60, (2004)=70$

sub \$3, \$2, \$5

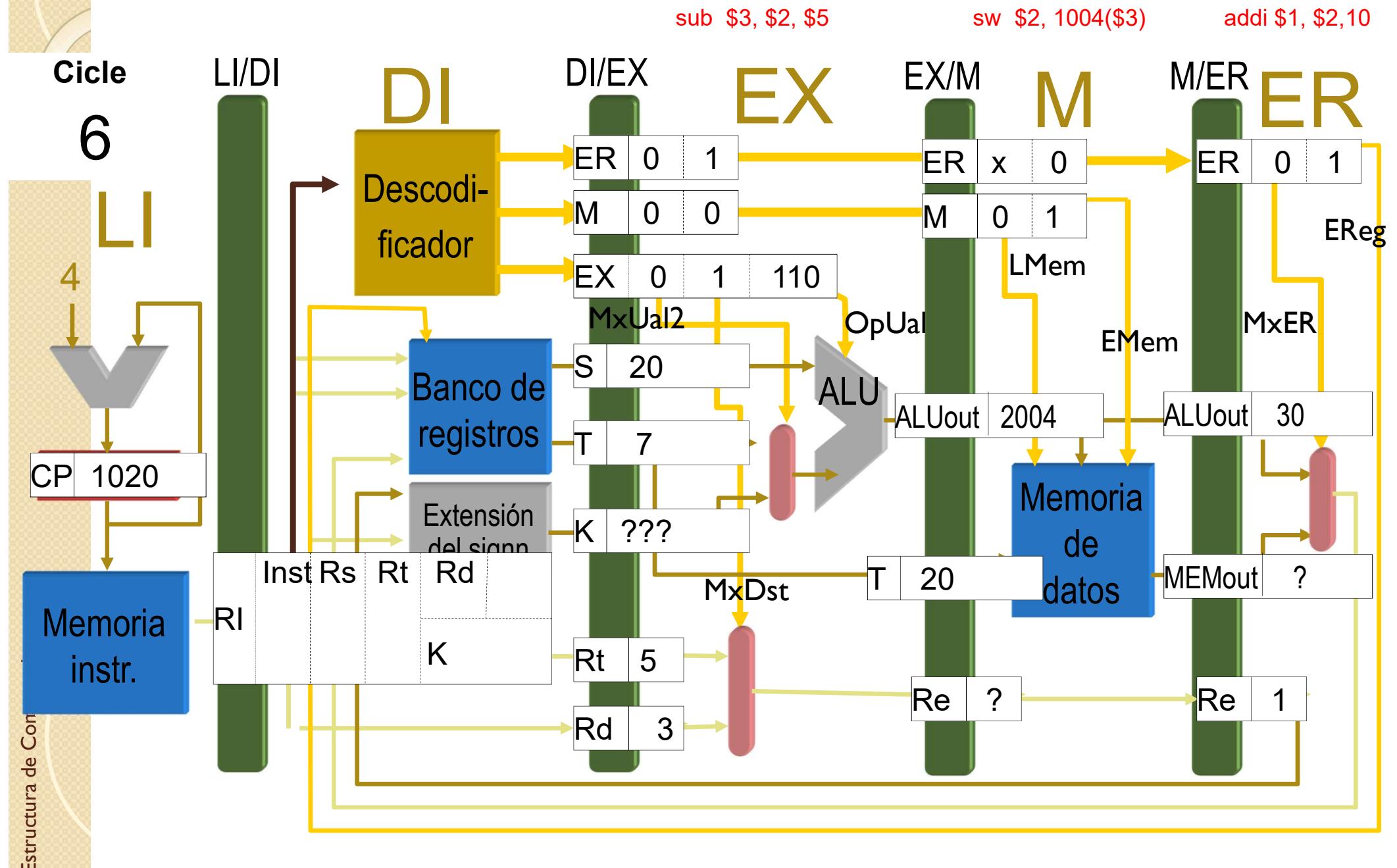
sw \$2, 1004(\$3)

addi \$1, \$2, 10

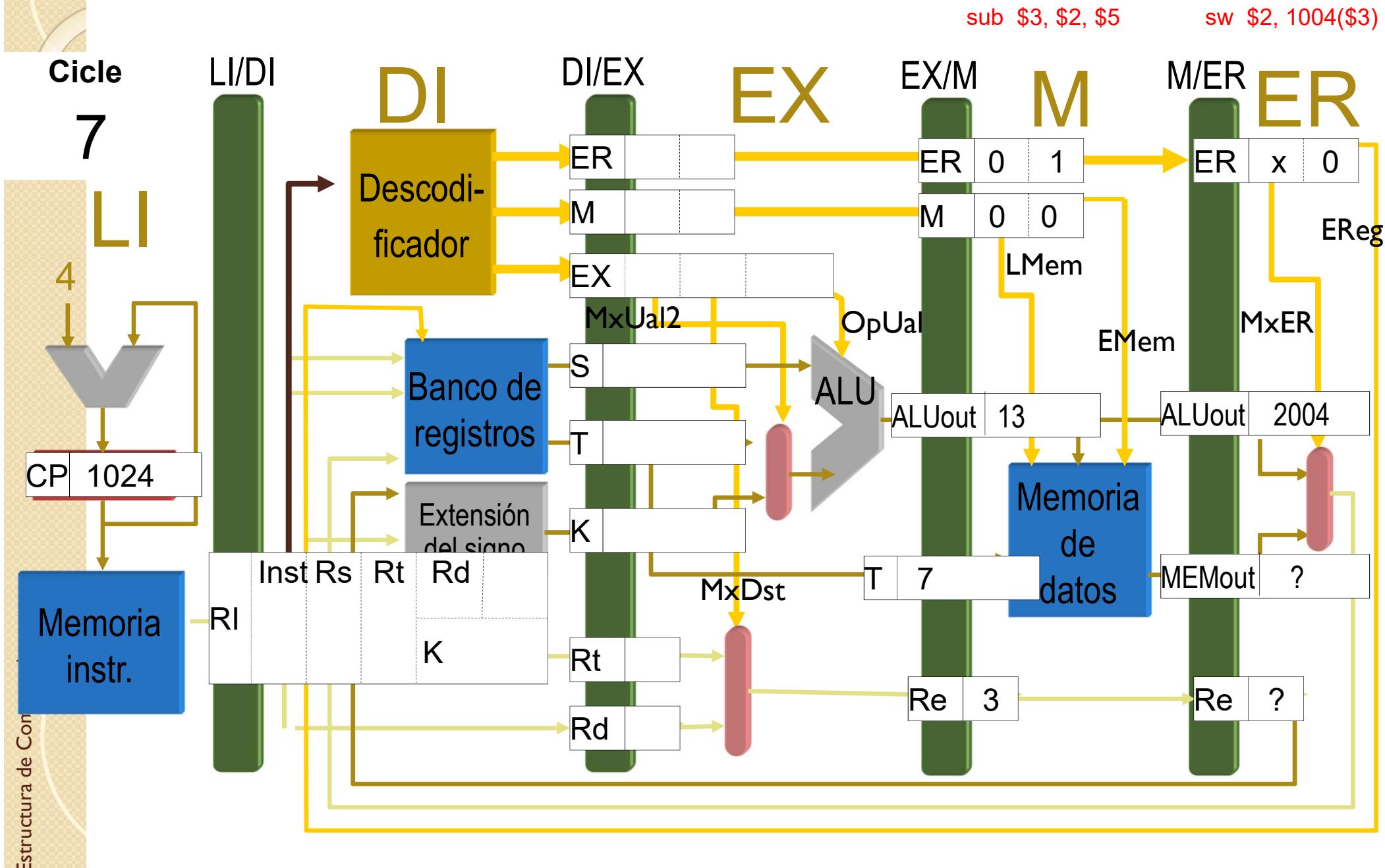
lw \$4, 2000(\$0)



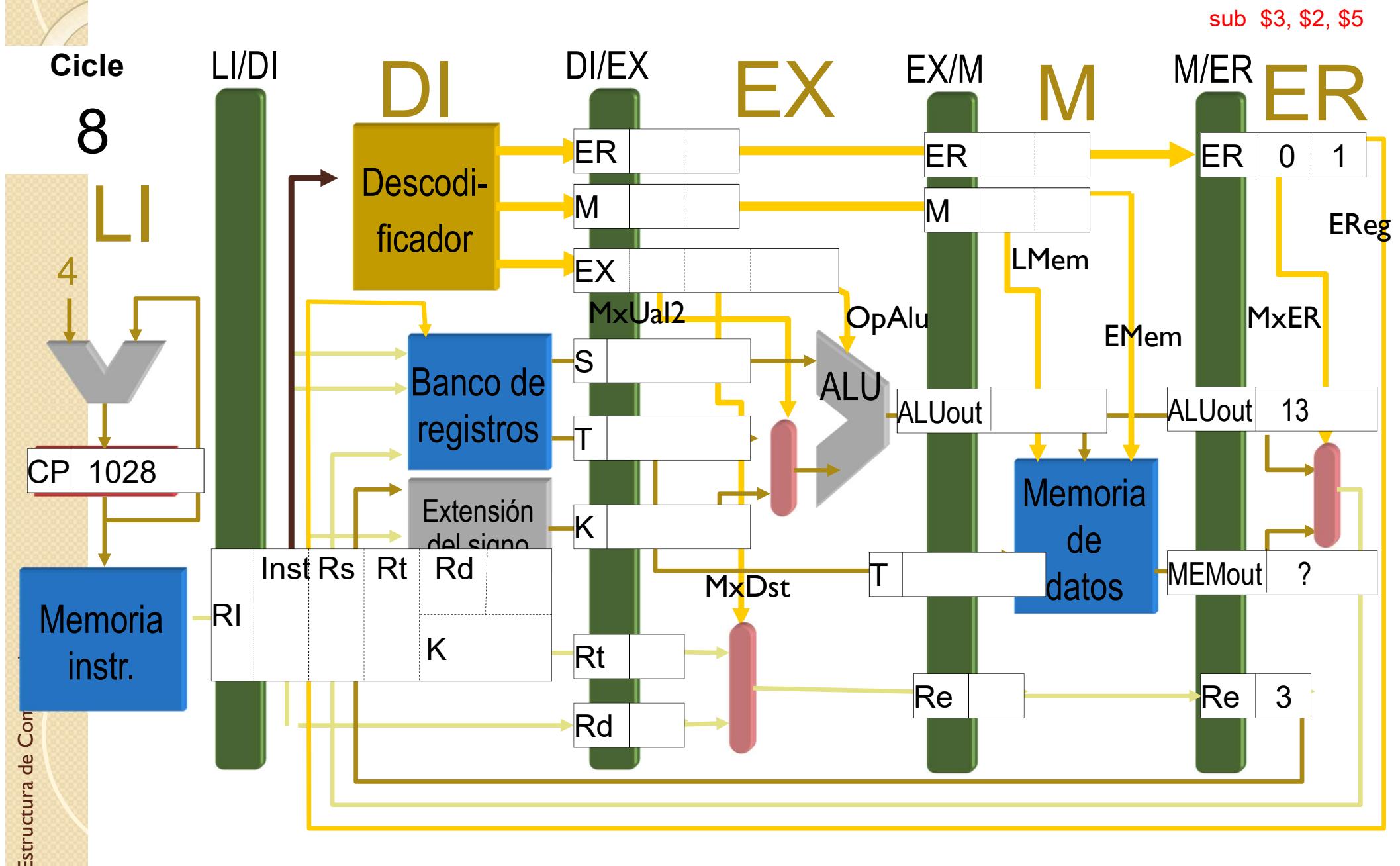
$\$0=0, \$1=30, \$2=20, \$3=1000, \$4=60, \$5=7, (2000)=60, (2004)=20$



$\$0=0$, $\$1=30$, $\$2=20$, $\$3=1000$, $\$4=60$, $\$5=7$, $(2000)=60$, $(2004)=20$



$\$0=0$, $\$1=30$, $\$2=20$, $\$3=13$, $\$4=60$, $\$5=7$, $(2000)=60$, $(2004)=20$



Ejercicio I:T2_EjerciciosdePrestaciones.docx

-
1. Considere la siguiente secuencia de instrucciones,

<u>etiqueta :</u>	<u>lw</u>	\$4, 2000(\$0)	(1)
	<u>addi</u>	\$1, \$2, 10	(2)
	<u>sw</u>	\$2, 1004(\$3)	(3)
	<u>sub</u>	\$3, \$2, \$5	(4)

- a. Realice el diagrama **instrucciones/tiempo**
- ⊕ b. Calcule tiempo de ejecución si el periodo de procesador segmentado es 20ns

	1	2	3	4	5	6	7	8	9	10	
<u>lw</u>	\$3, 2000(\$0)										
<u>addi</u>	\$1, \$2, 10										
<u>sw</u>	\$2, 1004(\$3)										
<u>sub</u>	\$3, \$2, \$5										

Ejercicio I:T2_EjerciciosdePrestaciones.docx

Diagrama
instrucciones/tiempo

Inst	1	2	3	4	5	6	7	8
lw \$4,2000(\$0)	LI	DI	EX	M	ER			
addi \$1,\$2,10		LI	DI	EX	M	ER		
sw \$2,1004(\$3)			LI	DI	EX	M	ER	
sub \$3,\$2,\$5				LI	DI	EX	M	ER

Ciclo	LI	DI	EX	M	ER
1	lw	?	?	?	?
2	addi	lw	?	?	?
3	sw	addi	lw	?	?
4	sub	sw	addi	lw	?
5	?	sub	sw	addi	lw
6	?	?	sub	sw	addi
7	?	?	?	sub	sw
8	?	?	?	?	sub

Diagrama
tiempo/etapas

Ejercicio I:T2_EjerciciosdePrestaciones.docx

Diagrama
instrucciones/tiempo

Inst	1	2	3	4	5	6	7	8
lw \$4,2000(\$0)	LI	DI	EX	M	ER			
addi \$1,\$2,10		LI	DI	EX	M	ER		
sw \$2,1004(\$3)			LI	DI	EX	M	ER	
sub \$3,\$2,\$5				LI	DI	EX	M	ER

Llenado de la ruta segmentada:
Si hay $k=5$ etapas, llenado = $k-1 = 4$

A partir del llenado, termina la ejecución de una instrucción/ciclo

Si se alimenta ininterrumpidamente, esta ruta a partir del ciclo 4 empezaría a terminar cada ciclo una instrucción

Ejercicio I:T2_EjerciciosdePrestaciones.docx

-
1. Considere la siguiente secuencia de instrucciones,

etiqueta : lw \$4, 2000(\$0) (1)
 addi \$1, \$2, 10 (2)
 sw \$2, 1004(\$3) (3)
 sub \$3, \$2, \$5 (4)

- a. Realice el diagrama **instrucciones/tiempo**
- b. Calcule tiempo de ejecución si el periodo de procesador segmentado es 20ns

	1	2	3	4	5	6	7	8	9	10	
<u>lw</u> \$3, 2000(\$0)		LI	DI	EX	M	ER					
<u>addi</u> \$1, \$2, 10			LI	DI	EX	M	ER				
<u>sw</u> \$2, 1004(\$3)				LI	DI	EX	M	ER			
<u>sub</u> \$3, \$2, \$5					LI	DI	EX	M	ER		

¿Tiempo de ejecución?

$$\text{ciclos} = \text{llenado} + \text{instrucciones} = k - 1 + \text{instrucciones} = 4+4 \\ (k = 5 \text{ etapas})$$

Contenidos y Bibliografía

- El proceso de segmentación
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ Identificación de las fases de ejecución de las instrucciones
 - ✓ Diseño de las etapas (excepto beq)
 - ✓ **Cálculo del tiempo de ciclo**
 - ✓ Añadiendo instrucción beq a la ruta
- Conflictos y riesgos
 - ✓ Conflictos de datos
 - ✓ Conflictos de control (flujo)
- Perspectiva histórica. RISC-CISC y otras alternativas



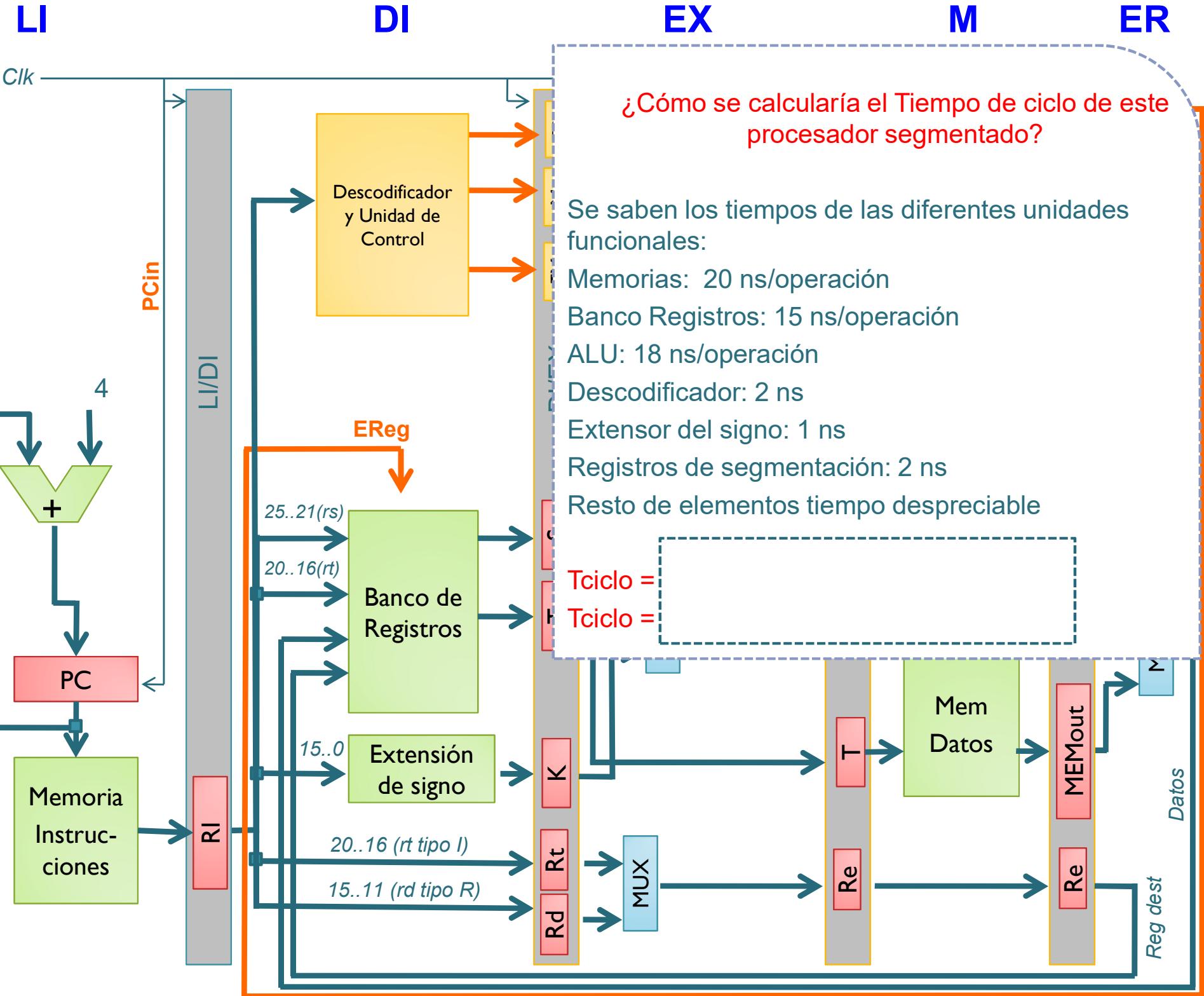
Segmentación de la Ruta de Datos

- La frecuencia de reloj
 - ✓ Tiempo de ciclo de reloj
 - $t_{seg} = \max(\tau_i) + t_R$
 - Habrá que considerar el tiempo de retardo τ_i de cada etapa y fijarse en el más largo
 - Habrá que considerar el retardo t_R de los registros de segmentación
 - ✓ Ejemplo:
 - Si $\tau_{LI} = \tau_M = 30$ ns, $\tau_{DI} = \tau_{ER} = 20$ ns y $\tau_{EX} = 25$ ns; $t_R = 10$ ns
 - Periodo $t_{seg} = 30 + 10 = 40$ ns
 - Frecuencia de reloj $f = 1/40$ ns = 25 MHz

Comparación ruta no segmentada con tiempos de acceso a memoria: 30 ns, a registros: 20 ns, ALU 25ns
Instrucción más lenta: lectura en memoria (load) con 125 ns: $F = 8$ MHz; $S = 3.125$
(30 ns lect. instr. + 20 ns lect. registros + 25 ns ALU cálculo dir. + 30 ns lect. mem + 20 ns escrit. en registro)

Ruta de datos sin soporte para instrucciones de salto
y con la unidad de control

Estructura de Computadores



Prestaciones del Procesador Segmentado

- Generalidades

- ✓ Ecuación del tiempo de ejecución de un programa en un procesador en **régimen estacionario** (no tiene en cuenta el tiempo de llenado de las etapas) :

$$T = I \times CPI \times t_c$$

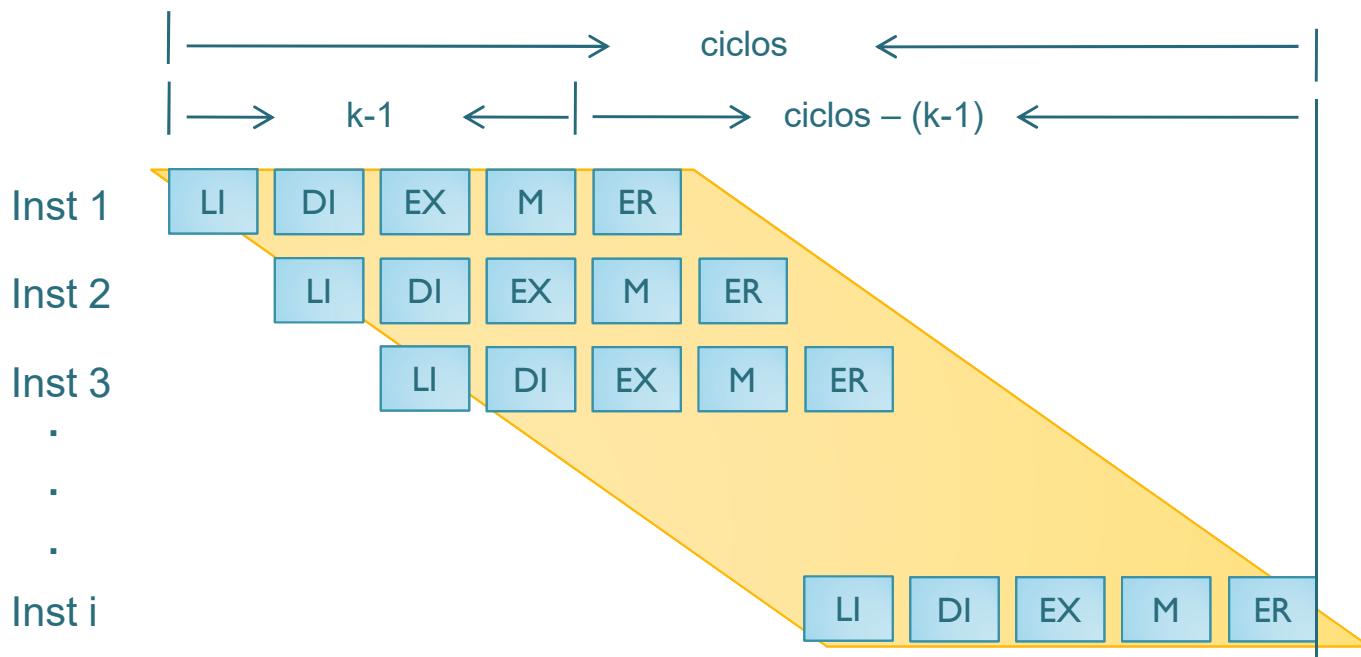
donde:

- T: tiempo de ejecución total del programa
- I: número de instrucciones que se ejecutan
- CPI: número medio de ciclos por instrucción
- t_c : tiempo de ciclo del reloj del procesador
- ✓ En general, convendrá minimizar los tres factores I, CPI y t_c
- ✓ Nos preocuparemos particularmente de los factores I y CPI
- ✓ Vamos a particularizar estos conceptos al caso del procesador segmentado

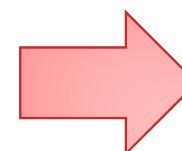
Prestaciones del Procesador Segmentado

• CPI (Ciclos por Instrucción)

- ✓ Índice que se utiliza para cuantificar las prestaciones del procesador
- ✓ Representa el **número medio** de ciclos por instrucción
- ✓ CPI>1 (el pipeline necesita k-1 ciclos para llegar a la última etapa; ciclos > I)



- ✓ Condiciones ideales:
 - cero ciclos de parada
 - CPI ideal = 1 (cota inferior del CPI)



$$CPI = \frac{ciclos - 4}{I}$$

Archivo Formulas de prestaciones.pdf

Fórmulación de prestaciones

	Procesador No segmentado	Procesador segmentado en k etapas
Tiempo de ciclo (segundos) Frecuencia (Hz= s ⁻¹)	$T = \tau_1 + \tau_2 + \dots + \tau_k$ $f = \frac{1}{T}$ (Hz= s ⁻¹) (k tareas independientes)	$\tau = \max\{\tau_i\} + T_R$ y $f = \frac{1}{\tau}$ (Hz= s ⁻¹) $1 \leq i \leq k$ y T_R tiempo de registros de segmentación
Tiempo de ejecución de un programa con "n" instrucciones	$T_{NS}(n) = n \times T$	$T_S(n) = \tau \times \text{ciclos}$ $T_S(n) = \tau \times \text{ciclos} = \tau \times (k + n - 1)$ (si el procesador <u>no</u> tiene paradas) $T_S(n) = \tau \times \text{ciclos} = \tau \times (p + k + n - 1)$ (si el procesador tiene "p" paradas) $T_S(n) = n \times CPI \times \tau$
CPI Ciclos por instrucción		$CPI = \frac{\text{ciclos} - \text{llenado}}{n} = \frac{\text{ciclos} - (k - 1)}{n}$ $CPI = 1$ (si no hay paradas) $CPI = (n+p) / n = 1 + p/n$ si hay "p" paradas



Contenidos y Bibliografía

- El proceso de segmentación
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ Identificación de las fases de ejecución de las instrucciones
 - ✓ Diseño de las etapas (excepto beq)
 - ✓ Cálculo del tiempo de ciclo
 - ✓ **Añadiendo instrucción beq a la ruta**
- Conflictos y riesgos
 - ✓ Conflictos de datos
 - ✓ Conflictos de control (flujo)
- Perspectiva histórica. RISC-CISC y otras alternativas

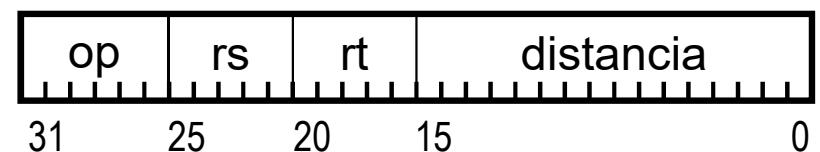


Soporte Instrucciones de Salto

- Instrucciones de salto condicional en el MIPS

- ✓ En el MIPS R2000 tenemos seis instrucciones, todas del formato I
- ✓ Direccionamiento relativo al PC
 - Como distancia de salto, se codifica (en complemento a 2) el número de palabras entre la instrucción siguiente y la instrucción de salto

instrucción	condición de salto
<i>beq rs,rt,eti</i>	$rs = rt$
<i>bne rs,rt,eti</i>	$rs \neq rt$
<i>bgez rs,eti</i>	$rs \geq 0$
<i>bgtz rs,eti</i>	$rs > 0$
<i>blez rs,eti</i>	$rs \leq 0$
<i>bltz rs,eti</i>	$rs < 0$



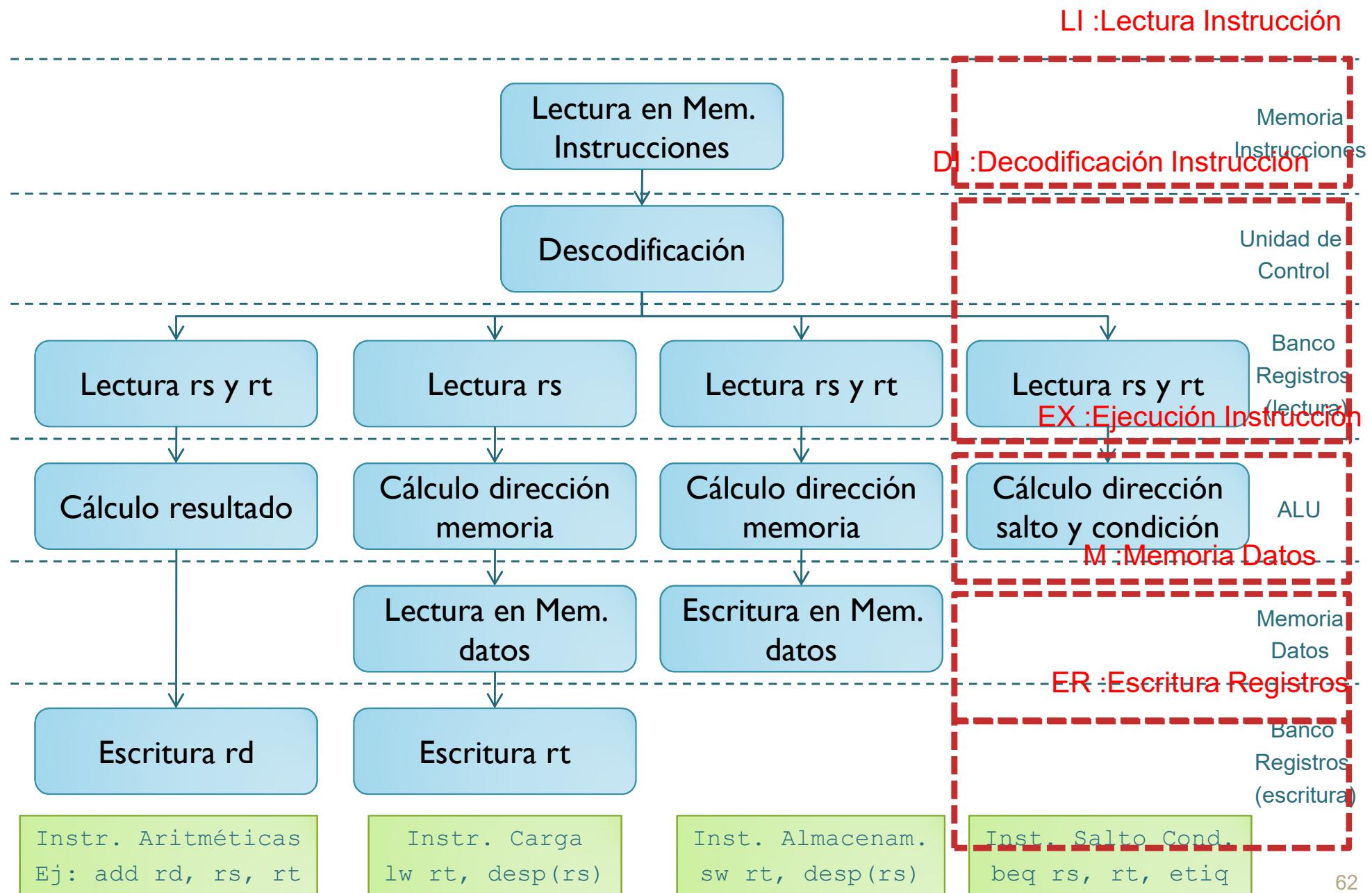
```
.text 0x00400000
      beq $t0, $0, etiq
      sll $t3, $t2, 3
      xor $t4, $0, $0
      ori $t4, $t3, $t1
etiq:   add $t0, $0, $0
      .end
```

Codifica un
+3

$$\text{distancia} = \frac{\text{dirección objetivo} - \text{dirección siguiente}}{4}$$

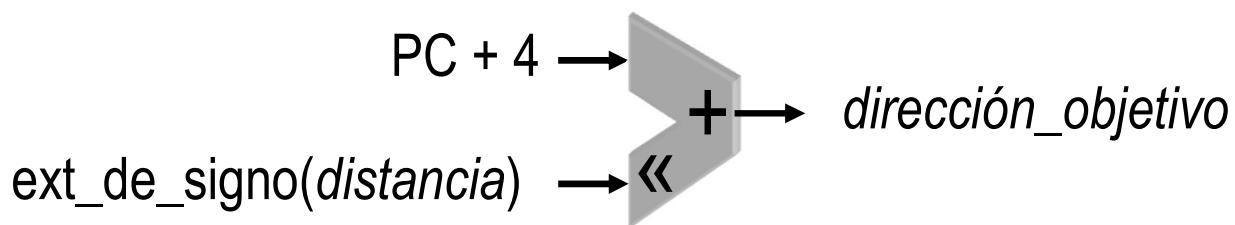
Soporte Instrucciones de Salto

- Especificación del ciclo de instrucción



Cálculo Dirección de Salto

- Hay que calcular la dirección absoluta de salto
 - ✓ dirección_objetivo = (PC+ 4) + ext_de_signo(distancia)*4
 - ✓ El cálculo se puede hacer en EX con un sumador específico adicional
 - ✓ Necesidad de transmitir el valor PC + 4 desde LI hasta EX
- Si la condición se cumple, la etapa M escribe dirección_objetivo en el CP



Cálculo de la Condición de Salto

- Evaluación de la condición

- ✓ La ALU ha de tener la operación identidad: $ALUout = A$
- ✓ Ha de suministrar dos indicadores:
 - Z (resultado igual a zero)
 - S (bit de signo del resultado)

instrucción	condición	op. ALU	COND
beq	$a=b$	resta	Z
bne	$a \neq b$	resta	\overline{Z}
bgez	$a \geq 0$	identidad	S
bgtz	$a > 0$	identidad	$\overline{S} \cdot \overline{Z} = \overline{S+Z}$
blez	$a \leq 0$	identidad	$S+Z$
bltz	$a < 0$	identidad	S

Cálculo de la Condición de Salto

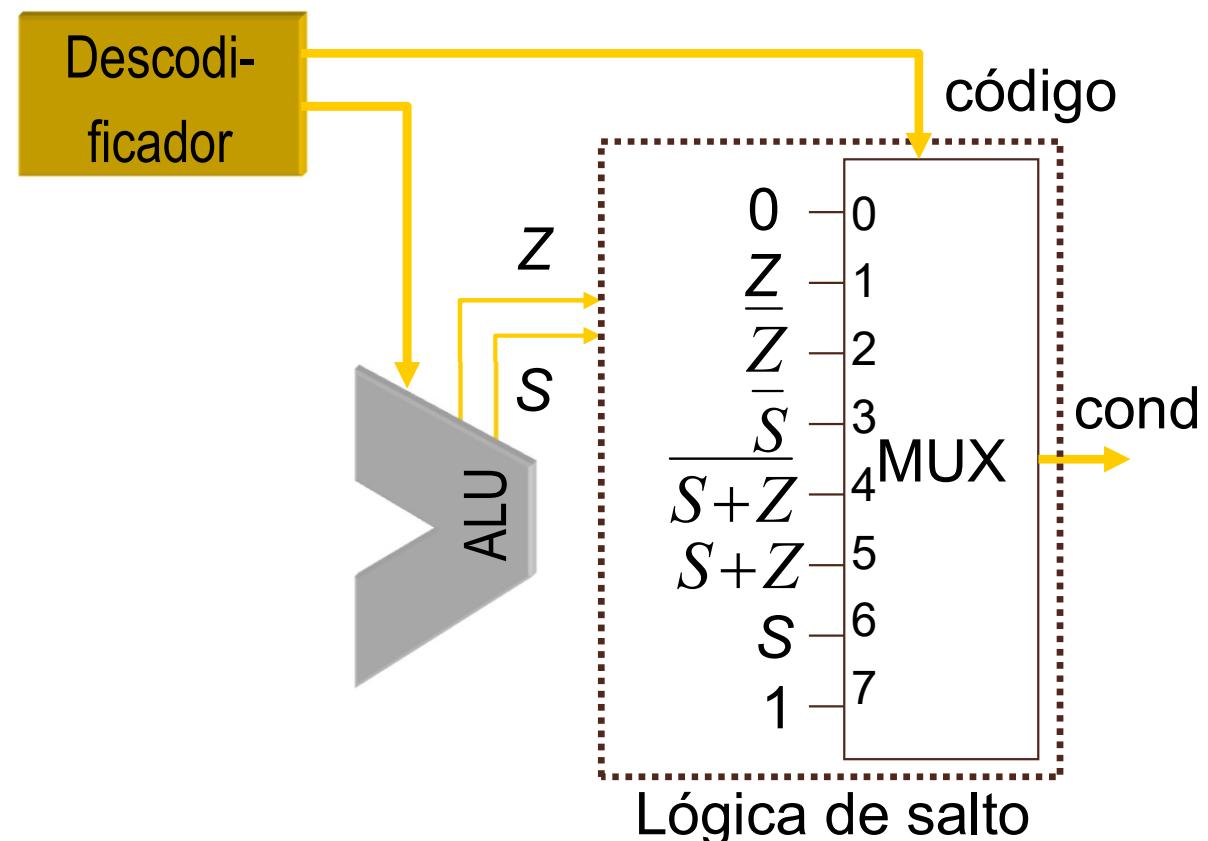
- Control básico de la bifurcación

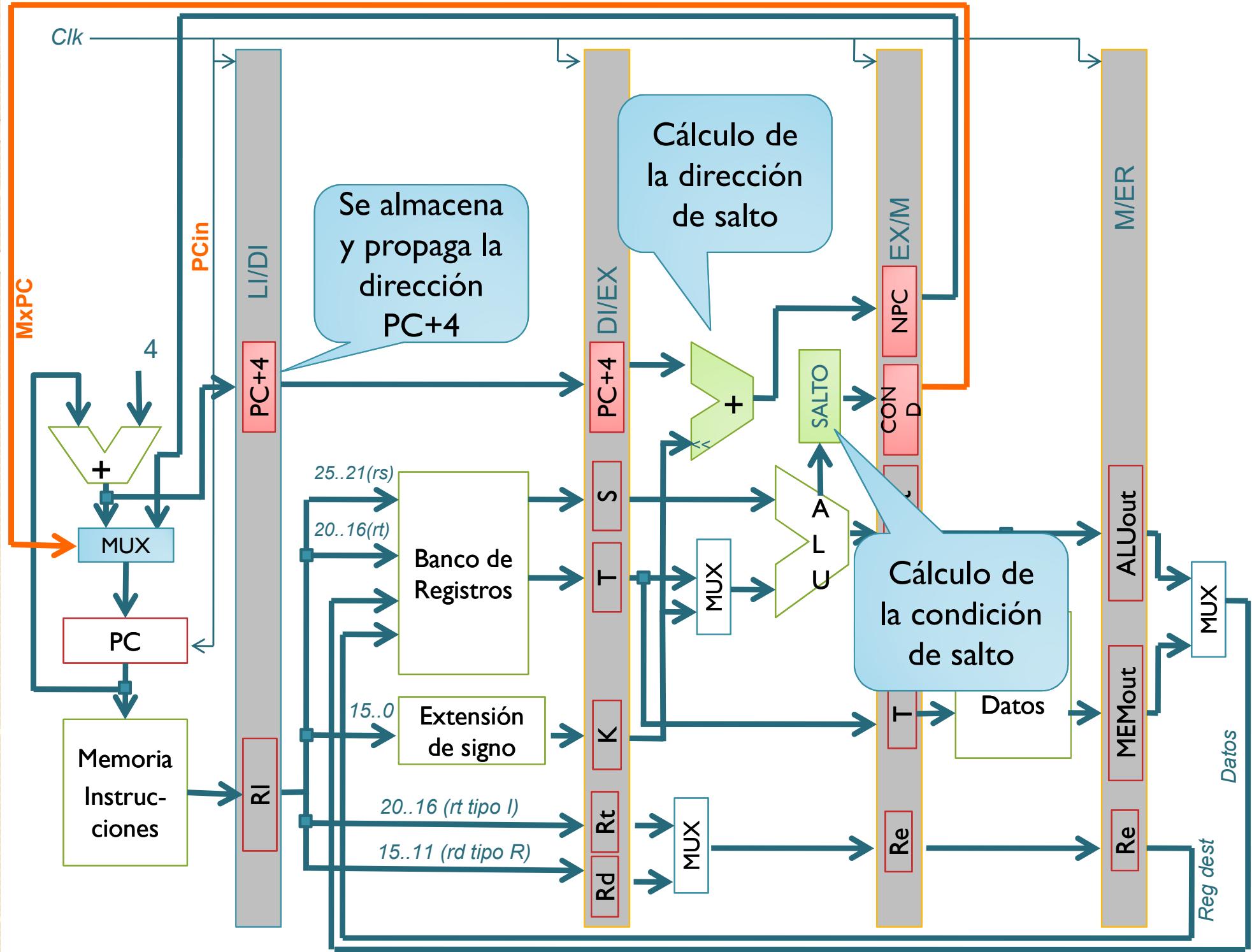
- ✓ Un nuevo registro EX/M.Cond (un bit): indicará si hay bifurcación efectiva

- ✓ Implementación con un MUX

- El descodificador de instrucción, en la etapa DI, calcula la posición del MUX para cada instrucción

instrucción	código
cálculo, l/s, etc.	0
beq	1
bne	2
bgez	3
bgtz	4
blez	5
bltz	6
j	7





Segmentación de la ruta de datos básica

- Equivalencia entre el dibujo de la ruta de datos vista en tema I, y la que se va a trabajar en el tema 2.
 - ✓ Se empieza con la ruta de los datos sin las instrucciones de salto, para eliminar complejidad
- Identificación de las etapas para la ejecución de las instrucciones aritméticas tipo R e I estudiadas en el Tema I y las de carga lw y sw
- Diseño de las etapas de la ruta de los datos
- Descripción detallada de acciones en cada etapa (para el estudiante repasar)

Segmentación de la Ruta de Datos

- Definición de las etapas

- ✓ Etapas comunes a todas las instrucciones

- **LI:** Etapa de lectura de instrucción (e incremento del PC)
 - **DI:** Etapa de decodificación de instrucción (y lectura de registros)

- ✓ Etapas que dependen del tipo de instrucción

- **EX:** Etapa de ejecución
 - Instrucciones de cálculo: cálculo del resultado
 - Load y Store: cálculo dirección de memoria
 - Instrucciones de salto: Cálculo de la dirección de salto y de la condición de salto
 - **M:** Etapa de memoria
 - Load y Store: acceso a la memoria
 - Instrucciones de cálculo y salto: nada
 - **ER:** Etapa de escritura de registro
 - Instrucciones de cálculo y Load : escritura del registro
 - Store e instrucciones de salto: nada

Segmentación de la Ruta de Datos

- Los registros de segmentación
 - ✓ Hacen falta cuatro (porque el PC inicia la etapa LI)
 - Cada uno está estructurado en subregistros
 - ✓ Nomenclatura
 - Nombre de cada registro de etapa: el de las etapas que separa
 - Ej.: registro *LI/DI*
 - Para referirnos a los subregistros: *reg.subreg*
 - Ej.: *LI/DI.RI* = registro de instrucción de la instrucción que está en la etapa DI
 - Para referirnos a un rango de bits de un subregistro: subíndices
 - Ej.: *LI/DI.RI_{31..26}* = código de operación de la instrucción contenida en DI o bien: *LI/DI.RI[Codop]*

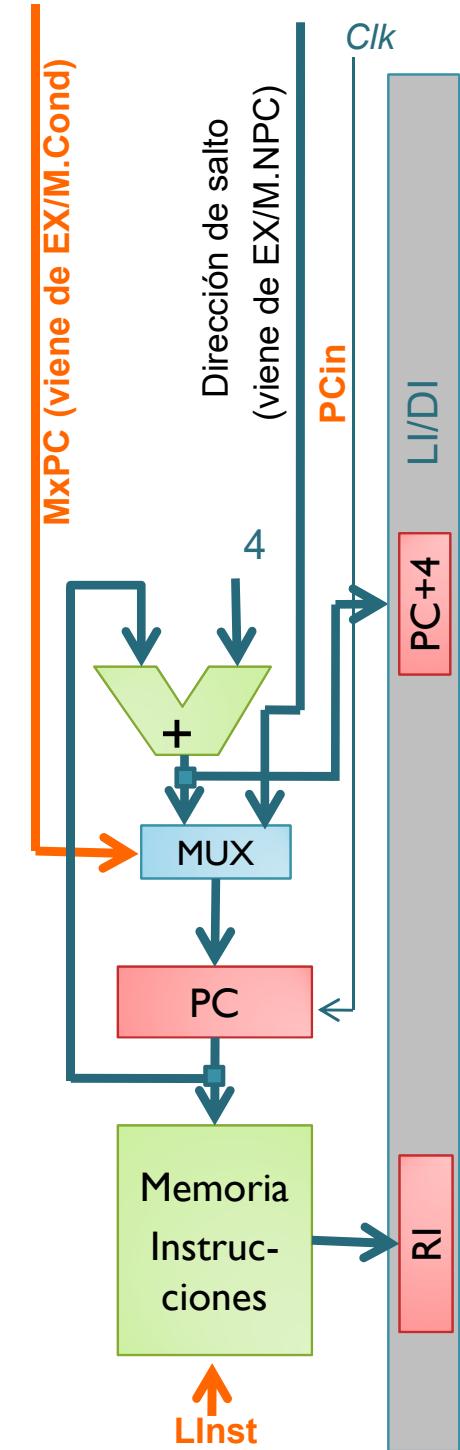
Etapa de Lectura de Instrucción

- Funciones

- ✓ Con todas las instrucciones:
 - Leer la instrucción apuntada por el PC
 - Avanzar el PC para que apunte a la instrucción siguiente
- ✓ Instrucción de salto condicional
 - Seleccionar la siguiente dirección de instrucción, entre $PC+4$ y la dirección de salto (calculada en la etapa EX)

- Señales de control

- ✓ **PCin**: Cargar PC, activada por flanco del reloj
- ✓ **LInst**: Leer en la memoria de instrucciones
- ✓ **PCIn** y **LInst** son constantes (podemos prescindir de ellas)
- ✓ **MxPC**: Calculada en la etapa EX (reg EX/M.Cond)



Etapa de descodificación

- Funciones

- ✓ Con todas las instrucciones:

- Calcular las señales de control para las etapas posteriores
- Leer los registros fuente
- Procesar el campo de desp/inm

- Componentes

- ✓ Extensión de signo (combinacional)

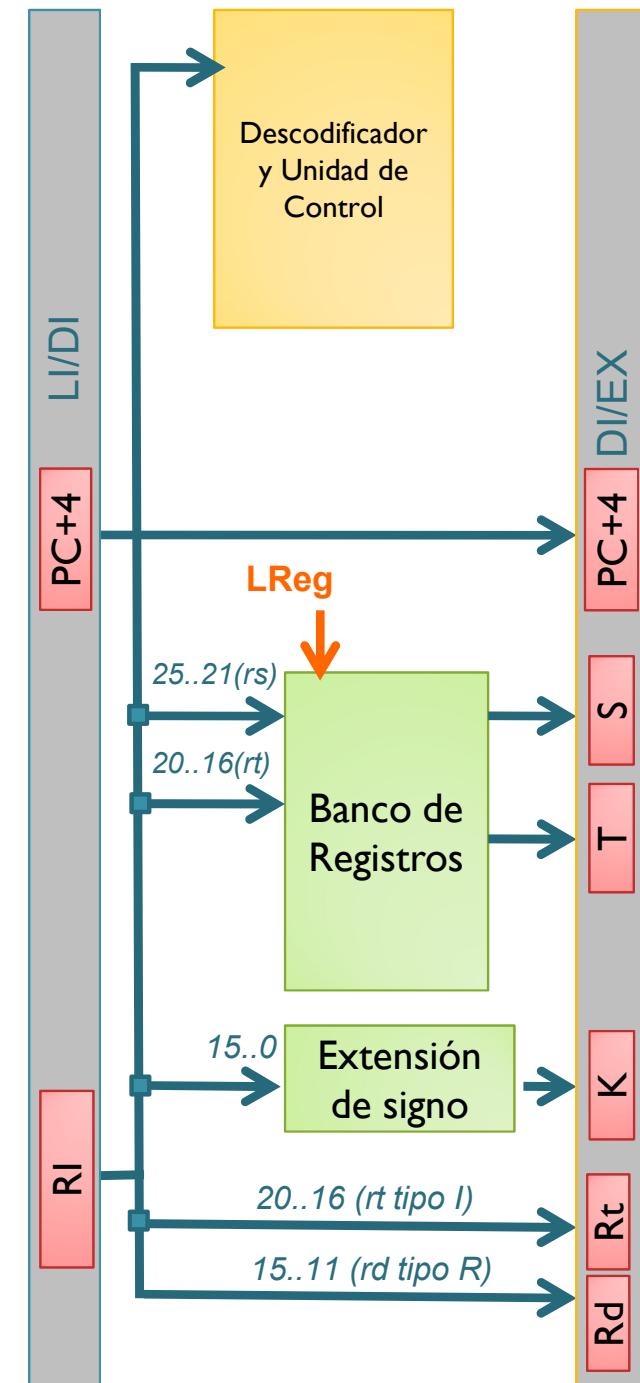
- ✓ Banco de registros (puertos de lectura)

- $S := \text{Reg}[RI_{25..21}]$

- $T := \text{Reg}[RI_{20..16}]$

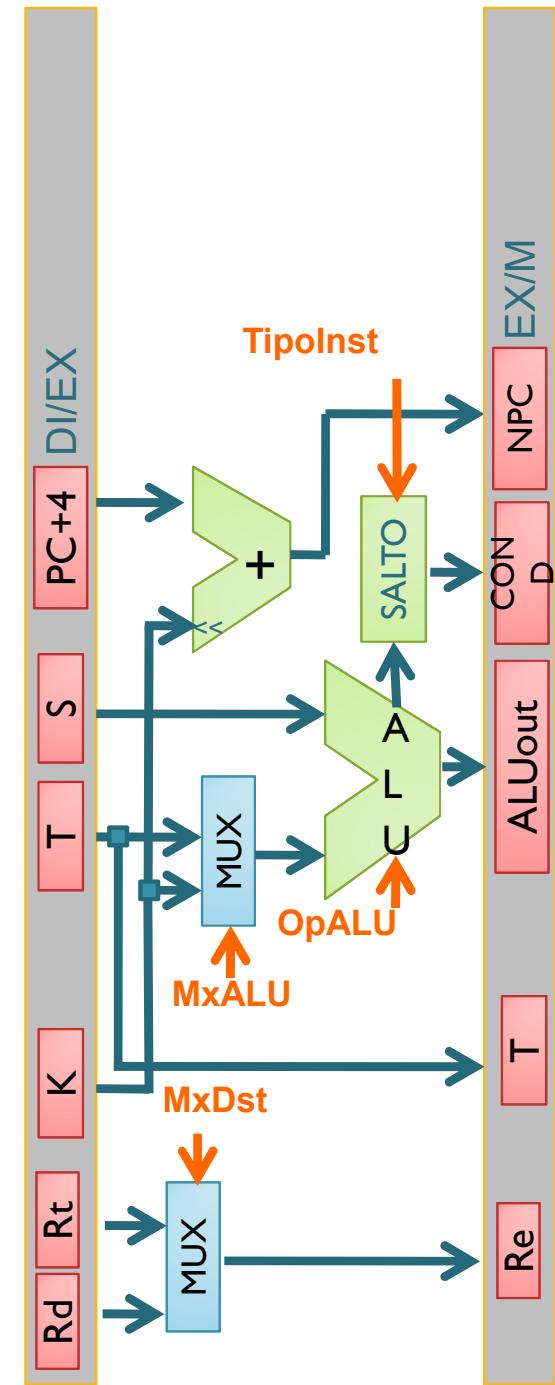
- Señales de control

- ✓ **LReg**: Activa la lectura de los dos registros
(podemos prescindir de esta señal)



Etapa de Ejecución

- Funciones
 - ✓ Instr. de cálculo: obtención del resultado
 - ✓ Instr. load/store: cálculo de la dirección de memoria
 - ✓ Instr. salto: calcula la dirección de salto y evalúa la condición de salto
- Señales de control
 - ✓ **MxALU**: Selecciona segundo operando para la ALU
 - ✓ **MxDst**: Determina el registro destino (escritura)
 - ✓ **OpALU**: Determina la operación que hará la ALU
 - ✓ **TipolInst**: Tipo de instrucción



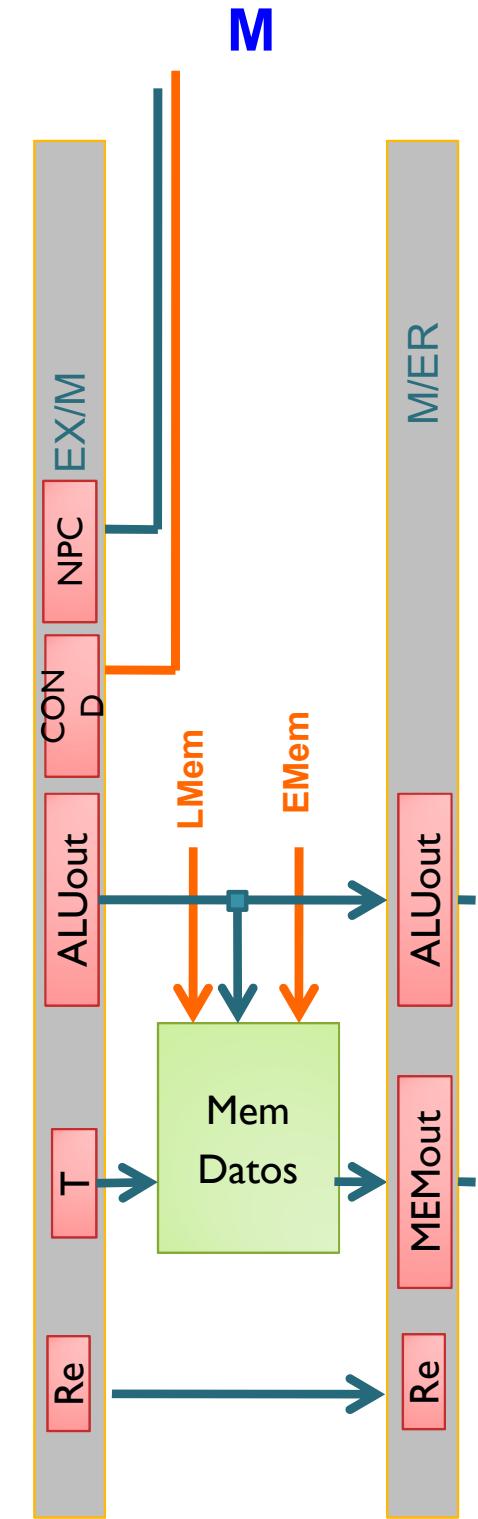
Etapa de Acceso a Memoria

- **Funciones**

- ✓ Instrucciones de cálculo: nada
- ✓ Instrucciones load/store: acceso a memoria
- ✓ Instrucciones de salto: Selecciona siguiente dirección de instrucción a leer

- **Señales de control**

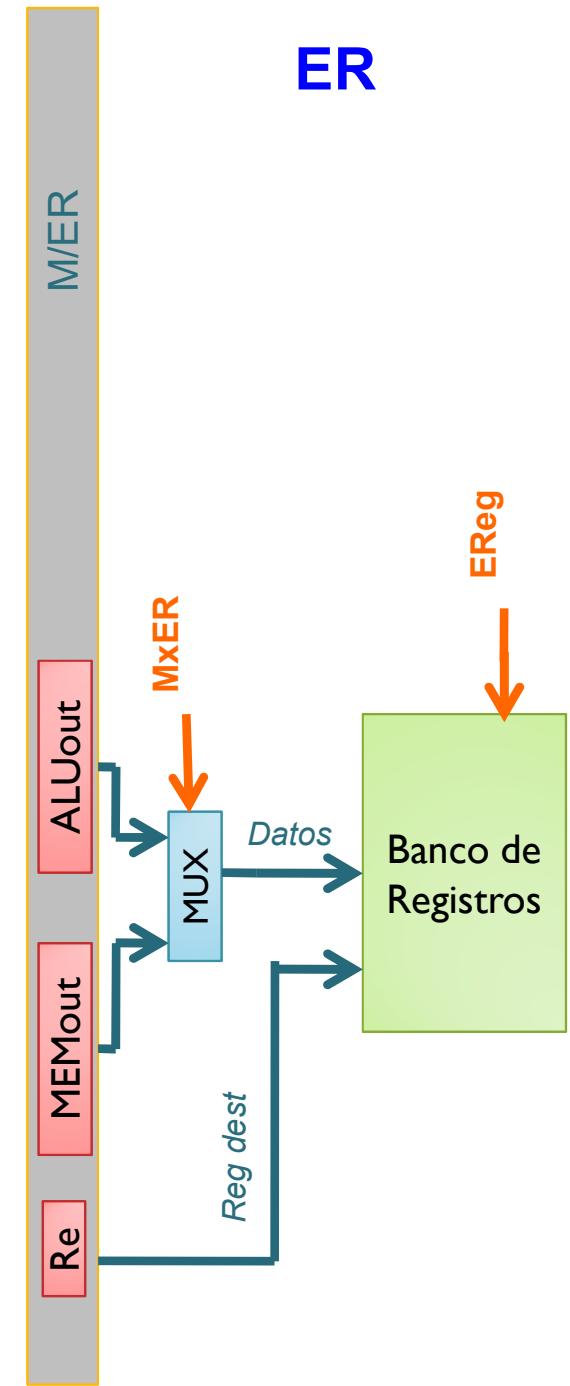
- ✓ **LMem**: activa la lectura de la memoria
- ✓ **EMem**: activa la escritura de la memoria



Etapa de Escritura en Registros

ER

- Funciones
 - ✓ Instrucciones de cálculo: escritura de resultados
 - ✓ Instrucciones de load: escritura del valor leído de la memoria
 - ✓ Instrucciones de store y salto: nada
- Características del banco de registros
 - ✓ Ha de soportar dos accesos de lectura y un acceso de escritura en cada ciclo
- Señales de control
 - ✓ **MxER**: Selecciona el valor que se escribe
 - ✓ **EReg**: Activa la escritura en el banco de registros



Contenidos y Bibliografía

- El proceso de segmentación
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ Identificación de las fases de ejecución de las instrucciones
 - ✓ Diseño de las etapas
- **Conflictos y riesgos**
 - ✓ Conflictos de datos([Lectura de “Riesgos por dependencias de datos\(notas\).pdf para repasar](#))
 - ✓ Conflictos de control (flujo)
- Perspectiva histórica. RISC-CISC y otras alternativas

Introducción a los Conflictos

- **Conflictos o riesgos**

- ✓ Son situaciones producidas por la segmentación del procesador, en las que la ejecución de una o más instrucciones no debe avanzar
 - Si el procesador no estuviera segmentado, desaparecerían

- ✓ **Tipos**

- Estructurales (**no los vemos; nuestra ruta no tiene conflictos estructurales**)
- De datos
- De control

- ✓ **Soluciones**

- Ciclos de parada
- Modificación del software (inserción instrucciones NOP)

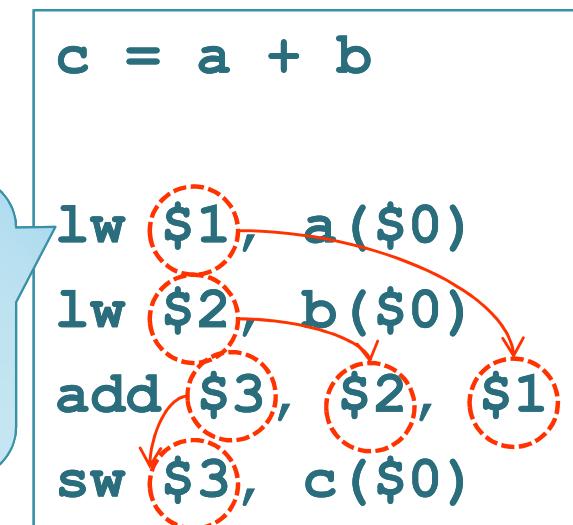
Los conflictos estructurales se producen en el uso de los recursos por parte de instrucciones cuya ejecución se solapa. Son causa de recursos insuficientemente. Ejemplo: Disponer de una única memoria de datos e instrucciones.

Los Conflictos de Datos

• Dependencias de datos

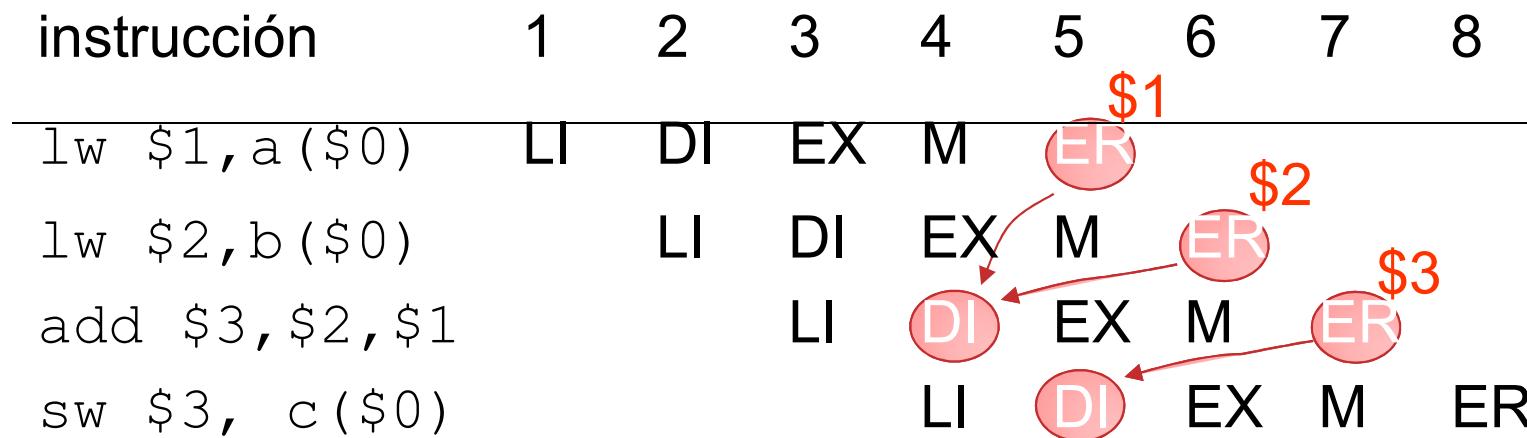
- ✓ Las instrucciones están fuertemente relacionadas entre ellas, resultado de la compilación de sentencias escritas en un lenguaje de programación de alto nivel
- ✓ ¿Qué son las dependencias de datos?
 - Las instrucciones se pueden ver como productoras y consumidoras de datos
 - Dependencia de datos:
cuando una instrucción consume un dato producido por otra
 - Ejemplo:
 - Compilación de $c := a + b$

Existe dependencia entre la instrucción 1 y la 3 por el registro \$1



Los Conflictos de Datos

- Conflictos de datos dentro de un procesador segmentado
 - ✓ Aparecen cuando una instrucción ha de operar con un valor que todavía no ha suministrado una instrucción anterior:



- El valor de \$1 se escribe en el ciclo 5; pero ha de ser leído por *add* en el ciclo 4
- El valor de \$2 se escribe en el ciclo 6; pero ha de ser leído por *add* en el ciclo 4
- El valor de \$3 se escribe en el ciclo 7; pero ha de ser leído por *sw* en el ciclo 5

Los Conflictos de Datos

- Técnicas básicas de resolución de conflictos
 - ✓ Técnicas de urgencia
 - Solución por software: inserción de instrucciones NOP en el código
 - Solución por hardware: generación de ciclos de parada

Los Conflictos de Datos

- Solución de urgencia por software: inserción de instrucciones *nop*
 - ✓ Al generar el código máquina, el compilador puede insertar instrucciones ***nop***
 - ✓ Ejemplo: compilación de *c:=a+b*

instrucción	1	2	3	4	5	6	7	8	9	10	11	12
lw \$1, a (\$0)	LI	DI	EX	M	ER	\$1						
lw \$2, b (\$0)	LI	DI	EX	M	ER	\$2						
<i>nop</i>		LI	DI	EX	M	ER						
<i>nop</i>			LI	DI	EX	M	ER					
add \$3, \$2, \$1			LI	DI	EX	M	ER	\$3	ER			
<i>nop</i>				LI	DI	EX	M	ER				
<i>nop</i>					LI	DI	EX	M	ER			
sw \$3, c (\$0)					LI	DI	EX	M	ER			

$$CPI = \frac{12 - 4}{8} = 1$$

Los Conflictos de Datos

- Solución de urgencia
 - ✓ Al generar el resultado se actualiza la señal de escritura en memoria
 - ✓ Ejemplo: comando `lw`

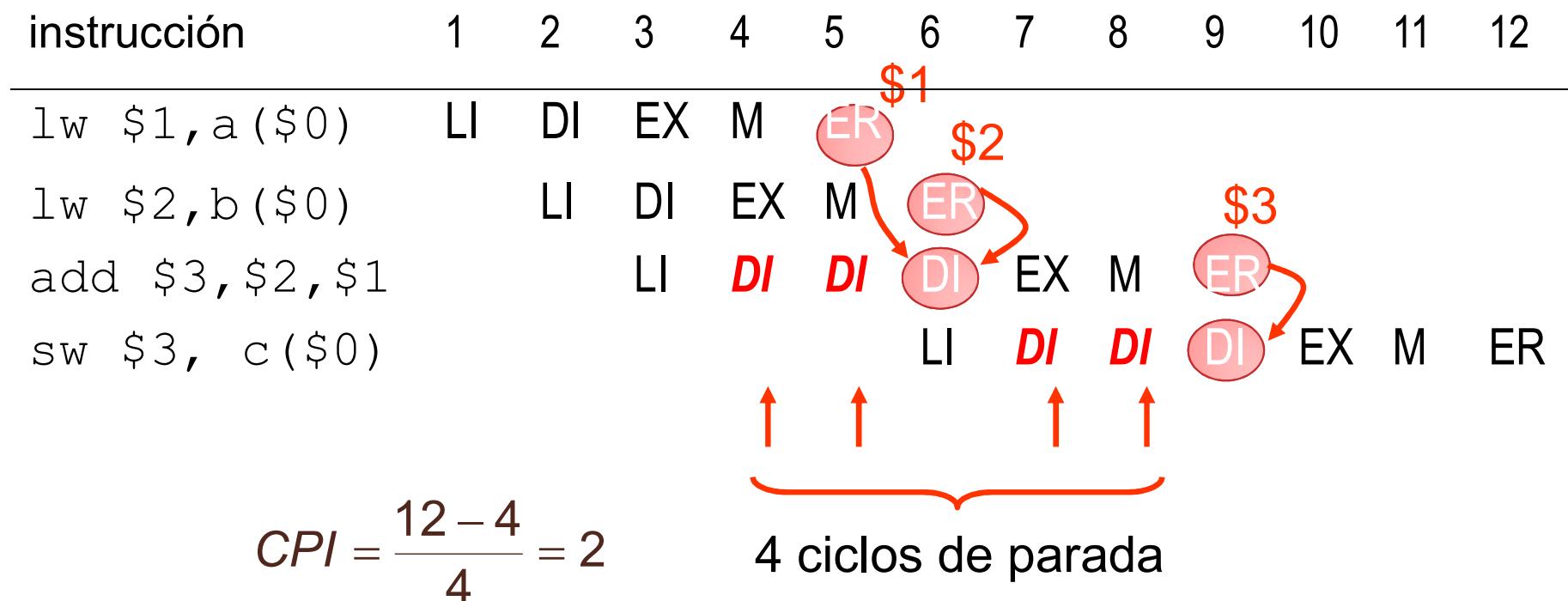
Se escribe en etapa ER y se lee en etapa DI, porque se modifica la señal de escritura en registros, ahora debe ser por nivel. Se sale de la etapa DI con el valor actualizado en etapa ER.

instrucción	1	2	3	4	5	6	7	8	9	10	11	12
<code>lw \$1, a(\$0)</code>	LI	DI	EX	M	ER \$1							
<code>lw \$2, b(\$0)</code>	LI	DI	EX	M	ER \$2	ER						
<code>nop</code>		LI	DI	EX	M	ER						
<code>nop</code>			LI	DI	EX	M	ER					
<code>add \$3, \$2, \$1</code>			LI	DI	EX	M	ER \$3					
<code>nop</code>				LI	DI	EX	M	ER				
<code>nop</code>					LI	DI	EX	M	ER			
<code>sw \$3, c(\$0)</code>						DI	EX	M	ER			

$$CPI = \frac{12 - 4}{8} = 1$$

Los Conflictos de Datos

- Solución de urgencia por hardware: ciclos de parada
 - ✓ El control del procesador puede prever las dependencias de datos y generar ciclos de parada para resolverlas



- ✓ Podemos calcular el CPI a partir del número de ciclos de parada (P)

$$CPI = \frac{I + P}{I} = 1 + \frac{P}{I} = 1 + \frac{4}{4} = 2$$

Soluciones y prestaciones

- Inserción de NOP

$$CPI = \frac{12 - 4}{8} = 1$$

El tiempo de ejecución en régimen estacionario:

$$T_{\text{programa}} = 8 \text{ instrucciones} * 1 \text{ CPI} * T_{\text{ciclo}}$$

- Detención (ciclos de espera)

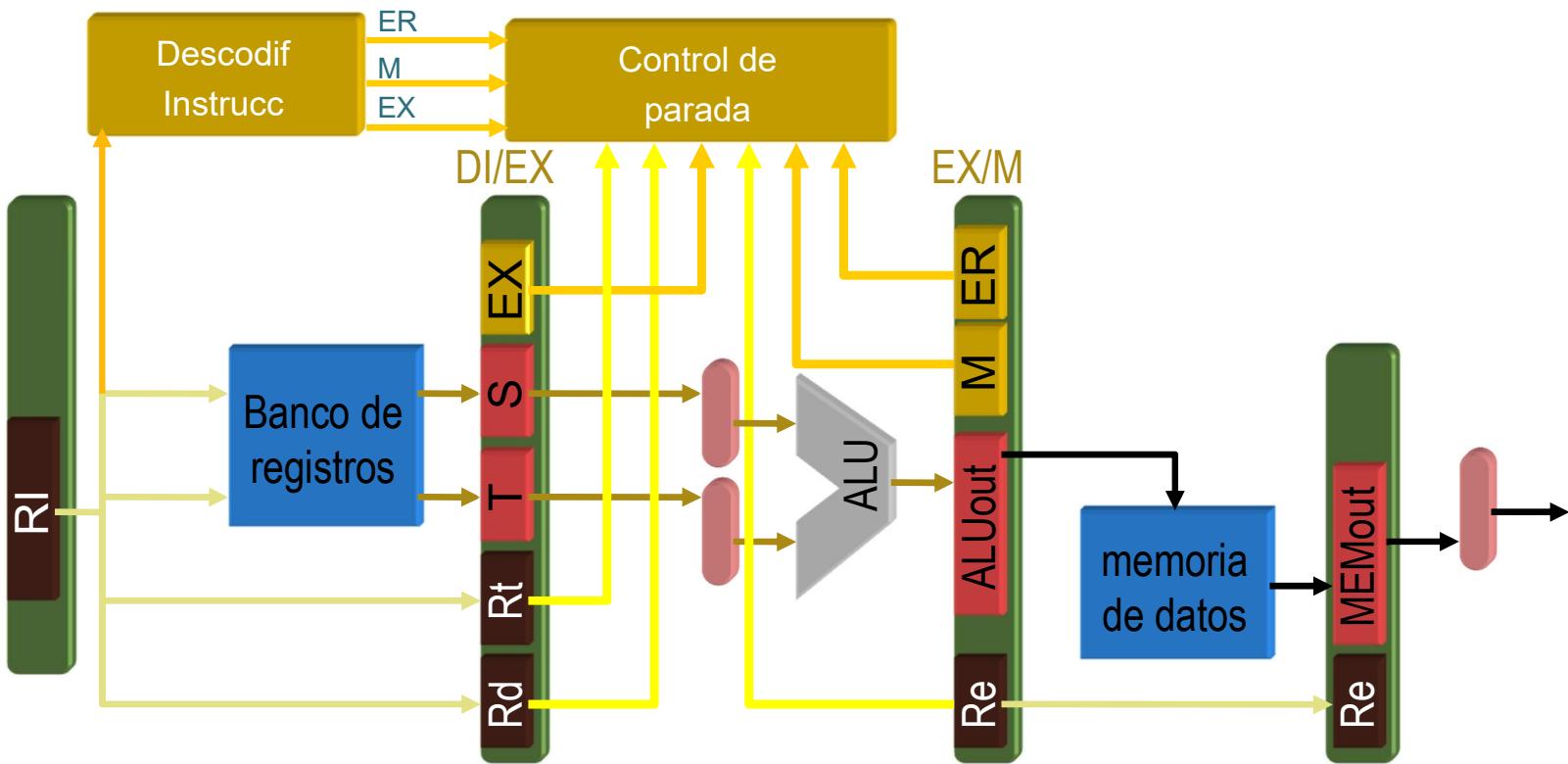
$$CPI = \frac{12 - 4}{4} = 2$$

El tiempo de ejecución en régimen estacionario:

$$T_{\text{programa}} = 4 \text{ instrucciones} * 2 \text{ CPI} * T_{\text{ciclo}}$$

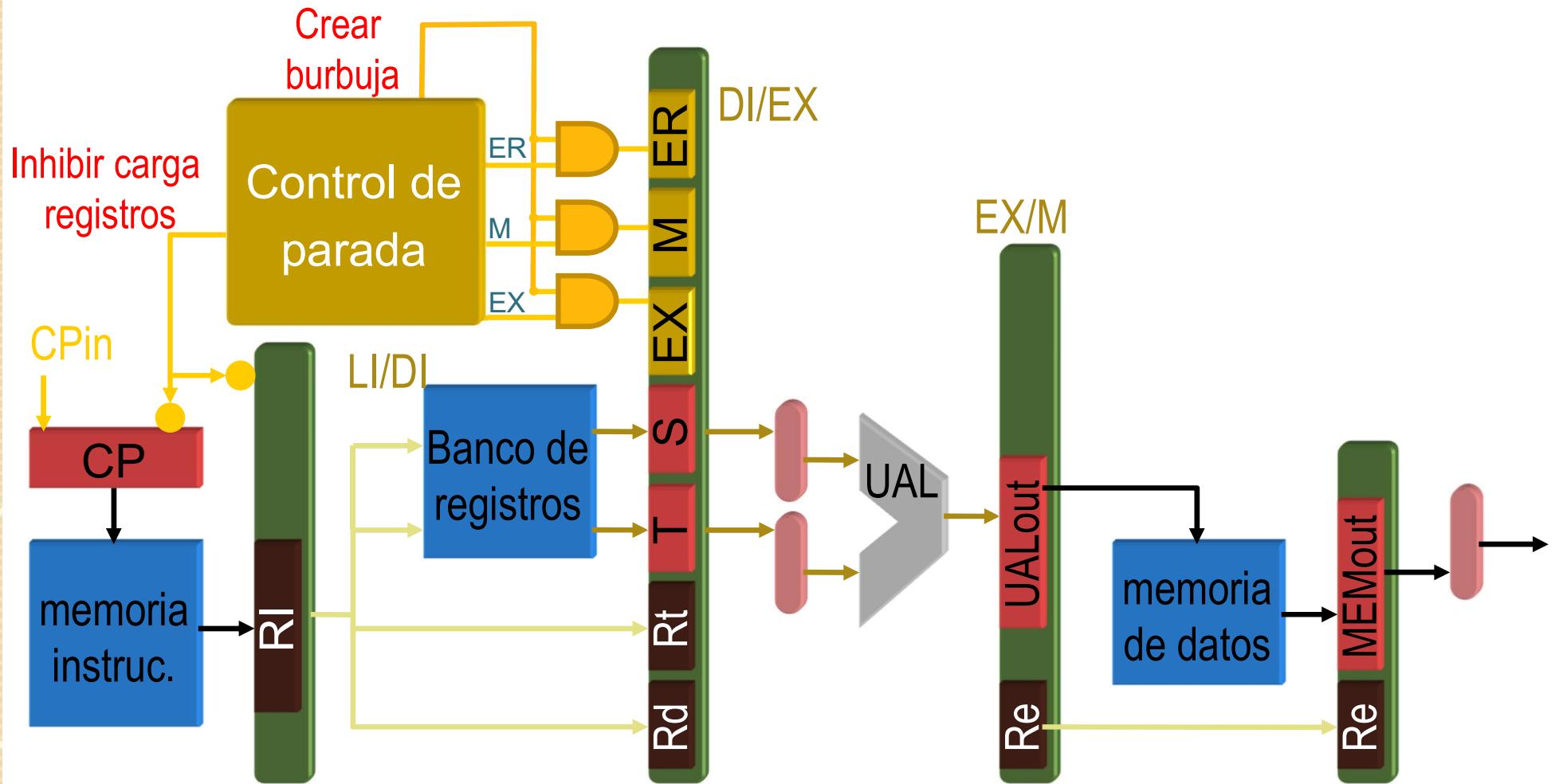
Lógica de detección de la dependencia

- Detección (algunos ejemplos de funciones de detección):
 - ✓ EX[LMem]=0 & EX[EMem]=0 & (la instrucción actual no es 'load' ni 'store')
 - ✓ (DI/EX.Rd=RI.Rs + (DI/EX[MxALU2]=1 & DI/EX.Rt=RI.Rs))
 - ✓ (DI/EX.Rd=RI.Rt + (DI/EX[MxALU2]=1 & DI/EX.Rt=RI.Rt))
 - ✓ (EX/M.Re = RI.Rs + (EX[MxALU2]=0 & EX/M.Re = RI.Rt))



Lógica de inserción del ciclo de parada

- Las etapas LI y DI han de repetir instrucción
- Hay que detener el avance en EX (creación de una burbuja) ($LMem=0$; $EMem=0$; $EReg=0$)



Ejercicio 1:T2_EjerciciosGenerales.docx

Preg. 1 Considere la siguiente secuencia de instrucciones,

```
etiqueta : addi    $t0, $t0, 10      (1)
           lui     $t1, 0x1080      (2)
           or     $t1, $t1, $t0      (3)
           lw      $t2, 0($t1)       (4)
           sw      $t2, 100($t1)     (5)
```

- a. Rellene la siguiente tabla para indicar los riesgos por dependencia de datos que se encuentran en este fragmento de código.

(En las casillas “Se escribe en” y “Se lee en” indique el número de instrucción, que se encuentra a la derecha de las instrucciones entre paréntesis, en la cual se escribe o lee el registro.)

	Registro	Se escribe en	Se lee en
Riesgo 1			
Riesgo 2			
Riesgo 3			
Riesgo 4			
Riesgo 5			

- b. Realice el diagrama **instrucciones/tiempo**. Utilice la técnica de solución de riesgos de datos por ciclos de parada. Calcule también el CPI.

Resumen : Conflictos de datos

- Instrucción Productora y a continuación Consumidora

add \$4, \$3,\$3

sub \$2, \$4, \$1

	1	2	3	4	5	6	7
add \$4,\$3,\$3	LI	DI	EX	M	ER		
sub \$2,\$4,\$1		LI	DI	EX	M	ER	

\$4 : instrucción add lo escribe en ciclo 5 etapa ER

instrucción sub lo en ciclo 3, etapa DI

Resumen : Conflictos de datos

- Instrucción Productora y a continuación Consumidora

add \$4, \$3,\$3
sub \$2, \$4, \$1

	I	2	3	4	5	6	7
add \$4,\$3,\$3	LI	DI	EX	M	ER		
sub \$2,\$4,\$1		LI	DI	EX	M	ER	

Solución : Etapa DI de sub, en mismo ciclo que etapa ER de add

Mediante paradas en procesador

	I	2	3	4	5	6	7	8
add \$4,\$3,\$3	LI	DI	EX	M	ER			
sub \$2,\$4,\$1		LI	DI	DI	DI	EX	M	ER

Mediante inserción de nop por compilador

	I	2	3	4	5	6	7	8
add \$4,\$3,\$3	LI	DI	EX	M	ER			
nop		LI	DI	EX	M	ER		
nop			LI	DI	EX	M	ER	
sub \$2,\$4,\$1				LI	DI	EX	M	ER

Se debe modificar el banco de REGISTROS de la ruta segmentada.

Señal de escritura por NIVEL.

Resumen : Conflictos de datos

- Instrucción Productora y a continuación Consumidora

add \$4, \$3,\$3

sub \$2, \$4, \$1

	1	2	3	4	5	6	7
add \$4,\$3,\$3	LI	DI	EX	M	ER		
sub \$2,\$4,\$1		LI	DI	EX	M	ER	

Solución : Etapa DI de sub, en mismo ciclo que etapa ER de add

Mediante paradas en procesador

	1	2	3	4	5	6	7	8
add \$4,\$3,\$3	LI	DI	EX	M	ER			
sub \$2,\$4,\$1		LI	DI	DI	DI	EX	M	ER

Paradas: etapas LI y DI se congelan,
en el resto se borran los registros de segmentación

Mediante inserción de nop por compilador

	1	2	3	4	5	6	7	8
add \$4,\$3,\$3	LI	DI	EX	M	ER			
nop			LI	DI	EX	M	ER	
nop				LI	DI	EX	M	ER
sub \$2,\$4,\$1				LI	DI	EX	M	ER

Resumen : Conflictos de datos

- Instrucción Productora y a continuación Consumidora

add \$4, \$3,\$3

sub \$2, \$4, \$1

	1	2	3	4	5	6	7
add \$4,\$3,\$3	LI	DI	EX	M	ER		
sub \$2,\$4,\$1		LI	DI	EX	M	ER	

Solución : Etapa DI de sub, en mismo ciclo que etapa ER de add

Mediante paradas en procesador

	1	2	3	4	5	6	7	8
add \$4,\$3,\$3	LI	DI	EX	M	ER			
sub \$2,\$4,\$1		LI	DI	DI	DI	EX	M	ER

$$\text{CPI} = \text{ciclos-4}/\text{instrucciones} =$$
$$\text{CPI} = 1 + \text{paradas}/\text{instrucciones}$$

Paradas: etapas LI y DI se congelan,
en el resto se borran los registros de segmentación

Mediante inserción de nop por compilador

	1	2	3	4	5	6	7	8
add \$4,\$3,\$3	LI	DI	EX	M	ER			
nop			LI	DI	EX	M	ER	
nop				LI	DI	EX	M	ER
sub \$2,\$4,\$1				LI	DI	EX	M	ER

$$\text{CPI} = 1$$

Resumen : Conflictos de datos

- Instrucción Productora y a continuación Consumidora

add \$4, \$3,\$3

sub \$2, \$4, \$1

	1	2	3	4	5	6	7
add \$4,\$3,\$3	LI	DI	EX	M	ER		
sub \$2,\$4,\$1		LI	DI	EX	M	ER	

- Instrucción Productora y Consumidora próxima

add \$4, \$3,\$3

add \$9, \$10,\$11

sub \$2, \$4, \$1

	1	2	3	4	5	6	7
add \$4,\$3,\$3	LI	DI	EX	M	ER		
add \$9,\$10,\$11		LI	DI	EX	M	ER	
sub \$2,\$4,\$1			LI	DI	EX	M	ER

Resumen : Conflictos de datos

- Instrucción Productora y a continuación Consumidora

add \$4, \$3,\$3

sub \$2, \$4, \$1

	I	2	3	4	5	6	7
add \$4,\$3,\$3	LI	DI	EX	M	ER		
sub \$2,\$4,\$1		LI	DI	EX	M	ER	

- Instrucción Productora y Consumidora próxima

add \$4, \$3,\$3

add \$9, \$10,\$11

sub \$2, \$4, \$1

	I	2	3	4	5	6	7
add \$4,\$3,\$3	LI	DI	EX	M	ER		
add \$9,\$10,\$11		LI	DI	EX	M	ER	
sub \$2,\$4,\$1			LI	DI	EX	M	ER

Mediante paradas en procesador

	I	2	3	4	5	6	7	8	
add \$4,\$3,\$3	LI	DI	EX	M	ER				
add \$9,\$10,\$11		LI	DI	EX	M	ER			
sub \$2,\$4,\$1			LI	DI	DI	DI	EX	M	ER

Mediante inserción de nop, sería SOLO una

Resumen : Conflictos de datos

- **Valoración de las dos soluciones vistas**

- ✓ **Semejanzas**

- El número de ciclos en tiempo de ejecución es el mismo
 - El compilador puede ayudar a bajar el tiempo de ejecución por ejemplo reordenando el código para evitar los conflictos.

- ✓ **Diferencias**

- La inserción de instrucciones inútiles incrementa el factor I
 - Los ciclos de parada incrementan el factor CPI
 - La ruta de datos segmentada es **compatible binaria** con la no segmentada (es decir, la ejecución del mismo programa en ambos procesadores produciría los **mismos resultados**)
 - La complejidad de la lógica de inserción de ciclos de parada podría alargar el retardo de las etapas y habría que bajar la frecuencia del reloj



Contenidos y Bibliografía

- El proceso de segmentación
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ Identificación de las fases de ejecución de las instrucciones
 - ✓ Diseño de las etapas
- Conflictos y riesgos
 - ✓ Conflictos de datos ([Lectura de “Riesgos por dependencias de datos\(notas\).pdf para repasar](#))
 - ✓ **Conflictos de control (flujo)** [Vídeos 3. Los riesgos de control](#)
- Perspectiva histórica. RISC-CISC y otras alternativas

Conflictos de Control

- Las instrucciones de salto

- ✓ Son instrucciones que rompen la secuencia lineal de ejecución de los programas

- ✓ Tipos

- Salto incondicional (jump)
 - Salto condicional o bifurcación (branch)
 - Llamada y retorno de subprograma (call/jump&link y retorno)

- ✓ Modos de direccionamiento

- Absoluto
 - Relativo a PC
 - Indirecto

- ✓ Frecuencia de aparición de las instrucciones de salto: 10 al 20% del código

- depende del tipo de procesador, de las técnicas de compilación y del programa en concreto

Conflictos de Control

- Instrucciones de salto condicional en el MIPS
Ejemplos

```
if (x>y)
/*then*/ z=x
else z=y;
```

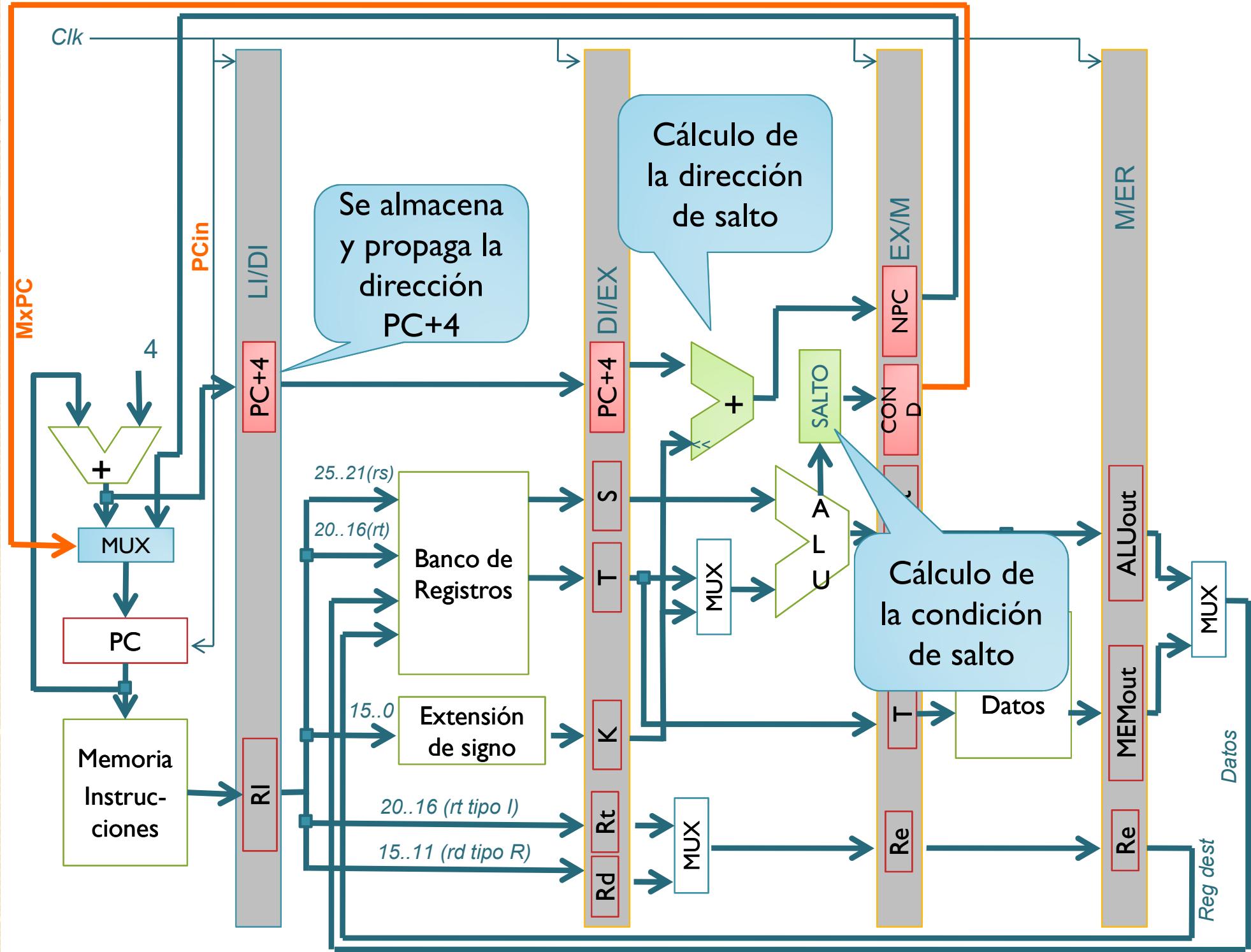


```
if:    lw $t0, x($0)
       lw $t1, y($0)
       sub $t2, $t1, $t0
       bgez $t2, else
then:   sw $t0, z($0)
        j endif
else:   sw $t1, z($0)
endif:
```

```
z=0;
do      z=z+y;
        x=x-1;
while  (x!=0)
```



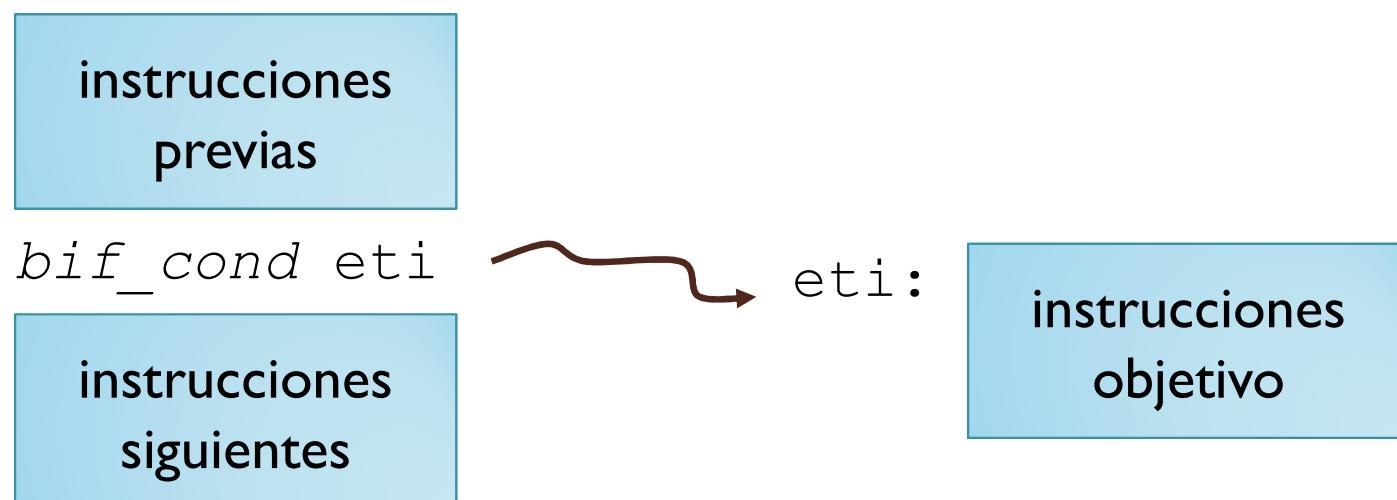
```
add $t2, $zero, $zero
lw $t0, x($0)
lw $t1, y($0)
do:    add $t2, $t2, $t1
       addi $t0, $t0, -1
while: bne $t0, $zero, do
enddw: sw $t2, z($0)
```



Conflictos de Control

• Vocabulario

- ✓ Instrucción objetivo (*target*): la instrucción destinataria del salto
- ✓ Las bifurcaciones saltan si se cumple una condición
 - si bifurcan, diremos que el salto es efectivo (*taken*)
 - en caso contrario, diremos que el salto es no efectivo (*not taken*)
- ✓ Una instrucción de salto condicional relaciona tres grupos de instrucciones:



Conflictos de Control

- Análisis del conflicto

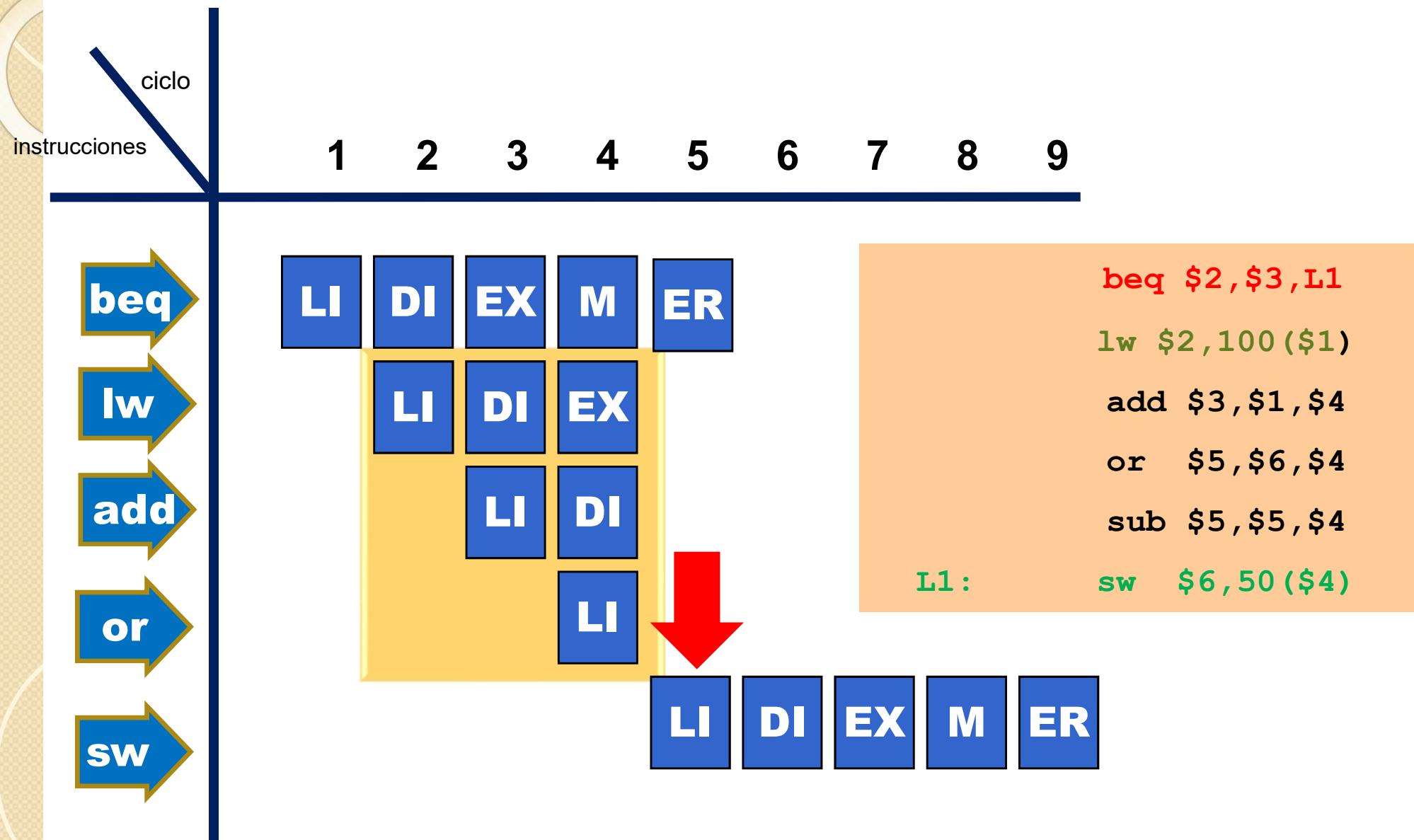
- ✓ Cuando el salto es efectivo en la etapa M, el procesador tiene tres instrucciones siguientes en proceso. En la etapa ER se escribe el nuevo PC

Instrucción	1	2	3	4	5	6	7	8	9
previa	LI	DI	EX	M	ER				
bifurcación		LI	DI	EX	M	ER			
siguiente 1		LI	DI	EX	M	ER			
siguiente 2		LI	DI	EX	M	ER			
siguiente 3		LI	DI	EX	M	ER			
objetivo		LI	DI	EX	M	ER			

Ver ruta

ciclo	LI	DI	EX	M	ER
4	siguiente2	siguiente1	bifurcación	previa	?
5	siguiente3	siguiente2	siguiente1	bifurcación	previa
6	objetivo	siguiente3	siguiente2	siguiente1	bifurcación

Análisis del conflicto



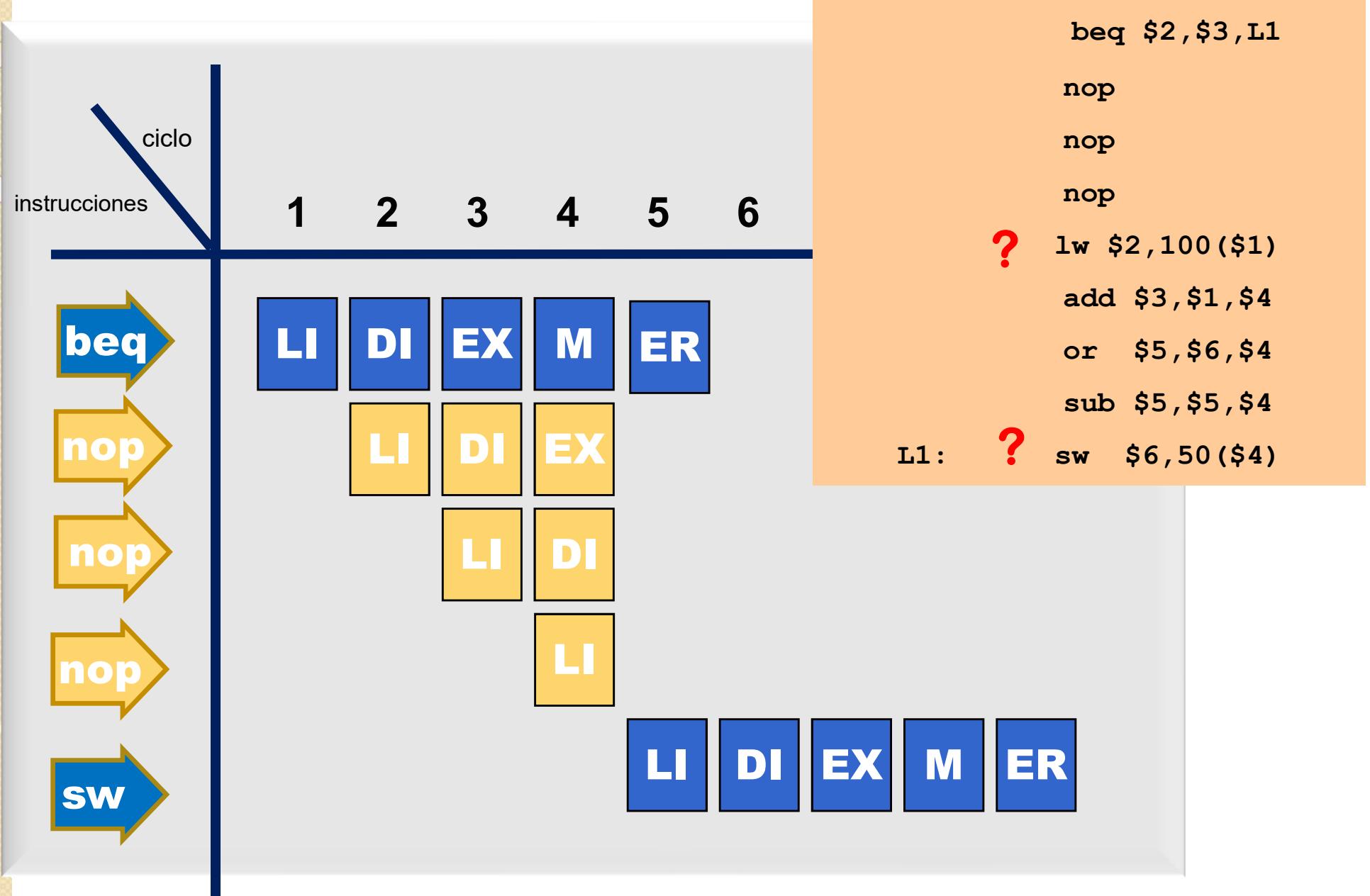
Latencia de salto

- Número de ciclos desperdiciados en caso de que el salto se produzca
- Número de instrucciones “erróneas” en el procesador si el salto se produce
- **Latencia de salto** = número de etapa en la que el salto es efectivo – 1
- En el caso anterior la **latencia de salto** = 3
- Cuanto más temprana sea la etapa en que se actualiza el CP, menor latencia y por tanto menor penalización

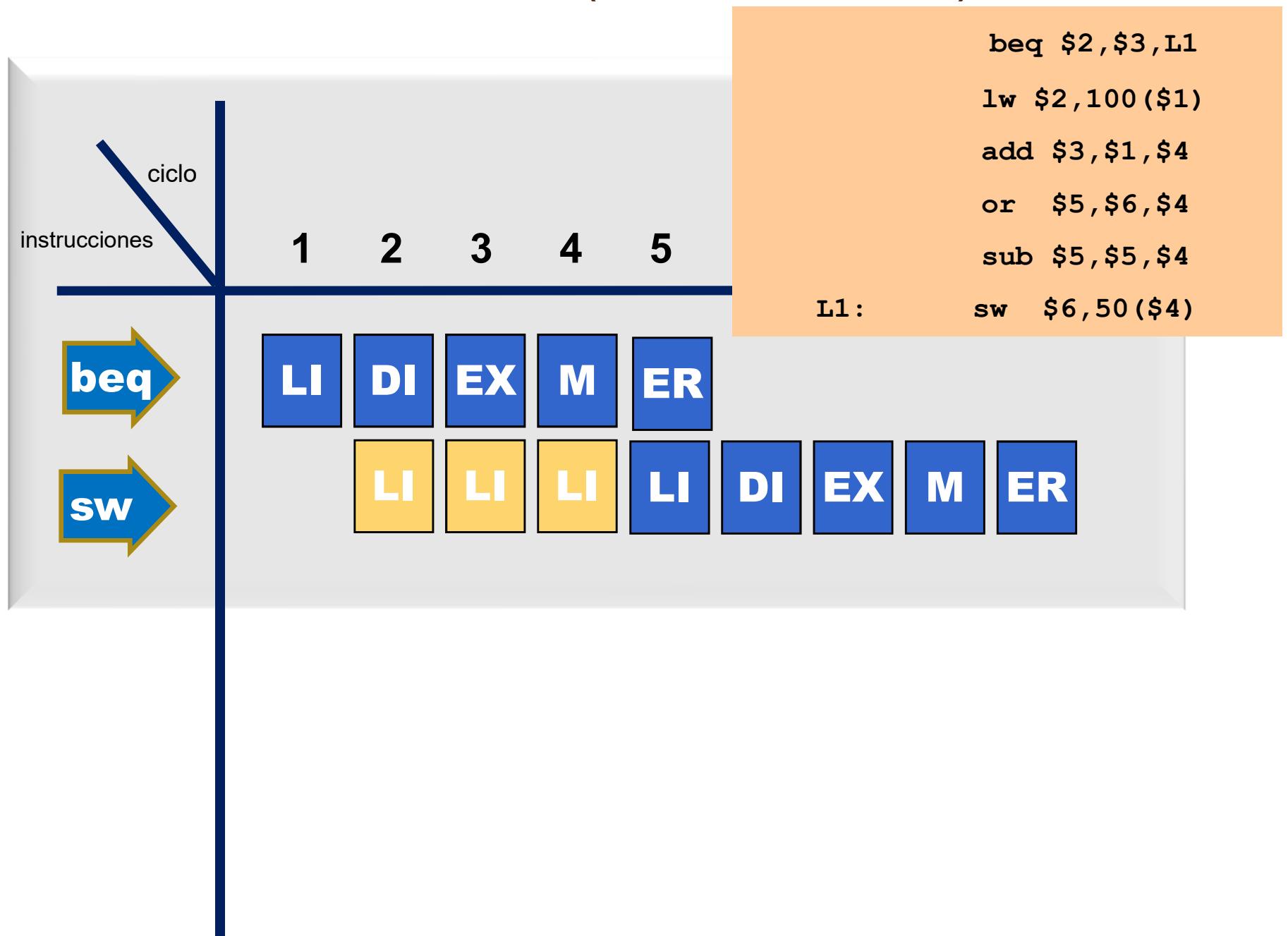
Tratamiento de los Conflictos de Control

- Primeras soluciones al conflicto
 - ✓ Soluciones conservativas o de urgencia:
 - Hardware
 - El decodificador inserta ciclos de parada al detectar un salto
 - Software
 - El compilador inserta NOP

Solución conservativa (SOFTWARE)



Solución conservativa (HARDWARE)

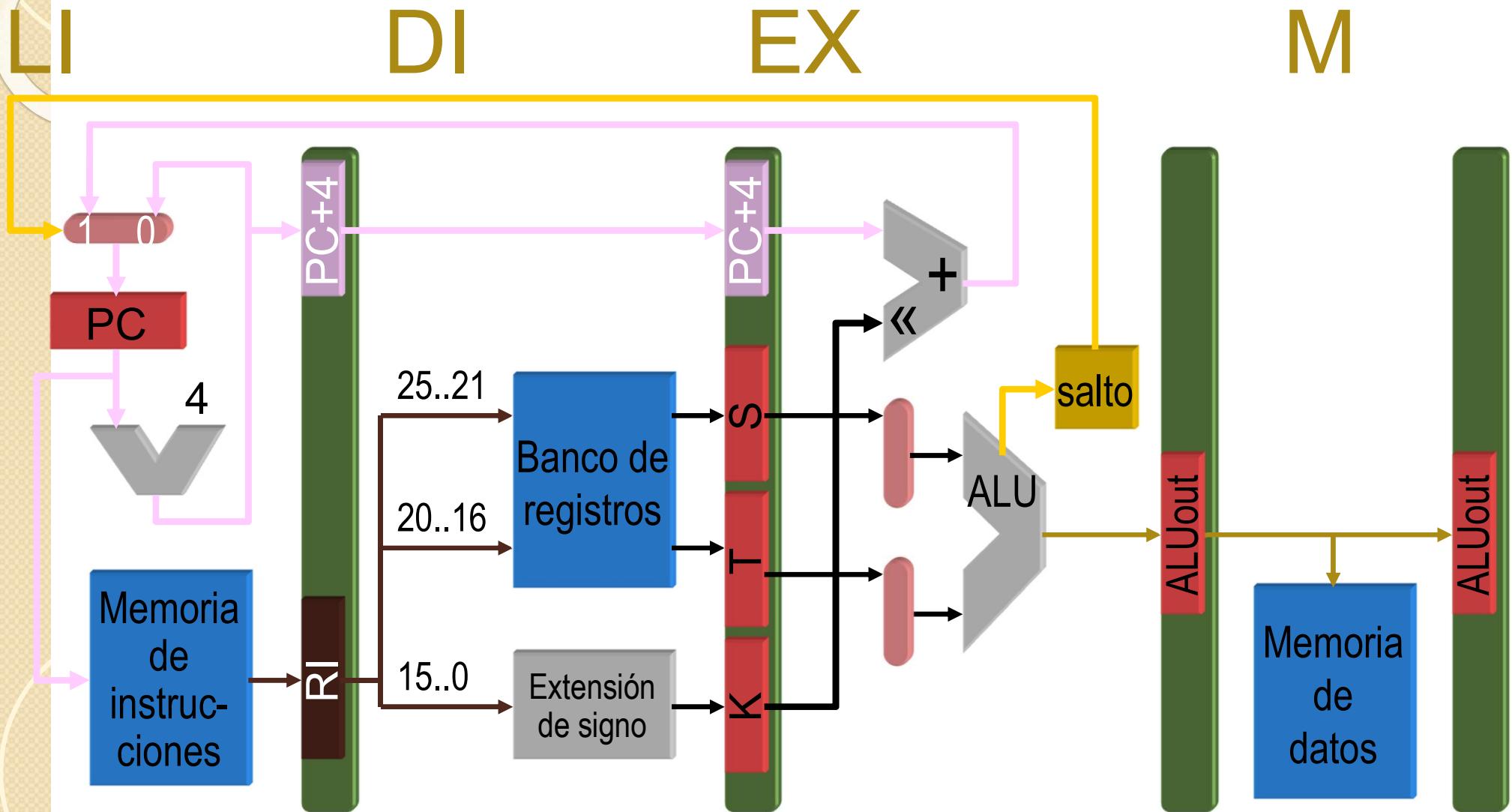


Latencia de salto

- Número de ciclos desperdiciados en caso de que el salto se produzca
- Número de instrucciones “erróneas” en el procesador si el salto se produce
- **Latencia de salto** = número de etapa en la que el salto es efectivo – 1
- En el caso anterior la latencia de salto = 3
- Cuanto más temprana sea la etapa en que se actualiza el CP, menor latencia y por tanto menor penalización
- Para reducir el conflicto, **se puede modificar el diseño de la ruta de datos y avanzar el momento de la escritura del PC.** Ver siguiente transparencia

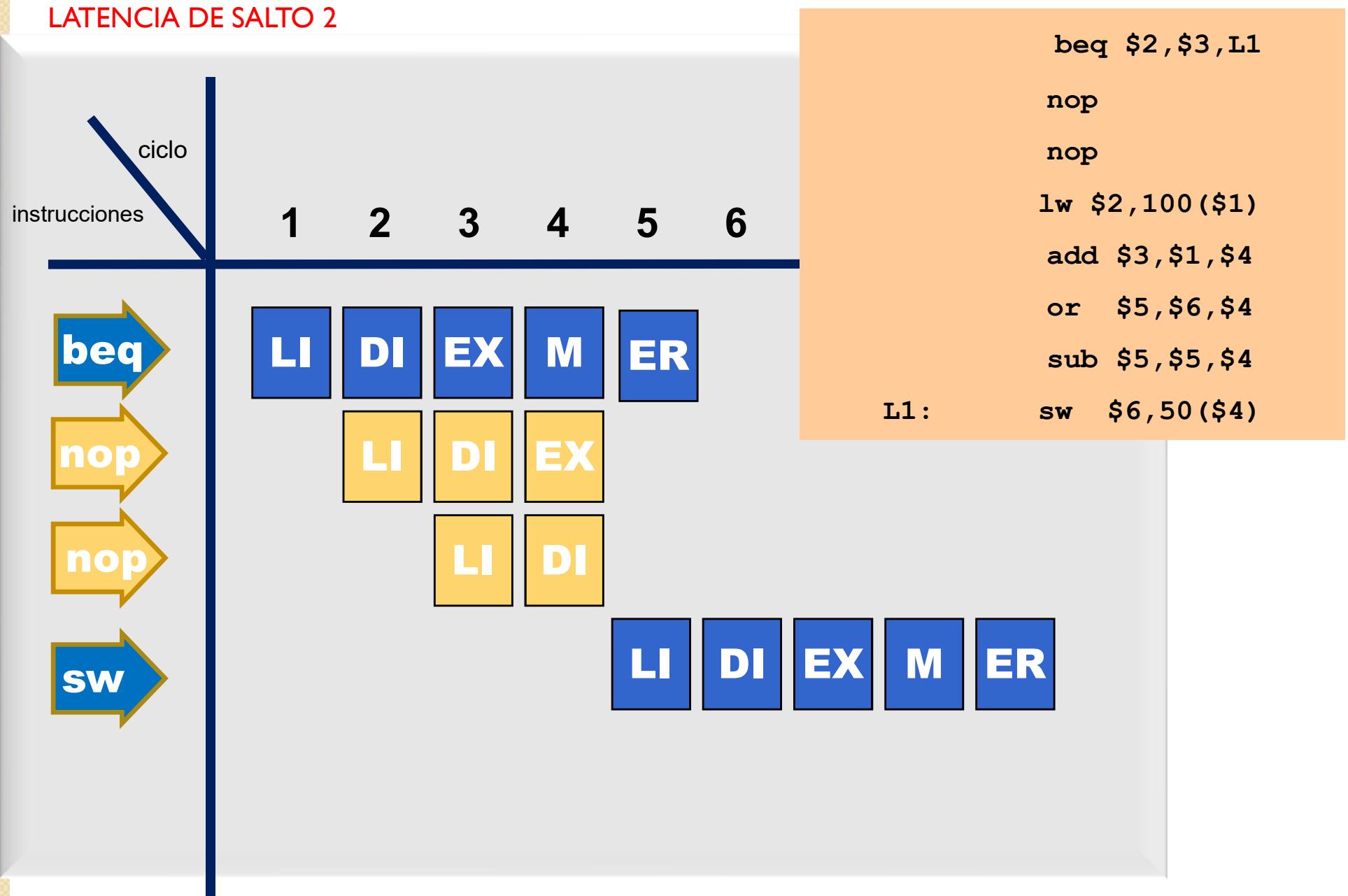
Latencia del Salto

- Ruta de datos con latencia de salto = 2



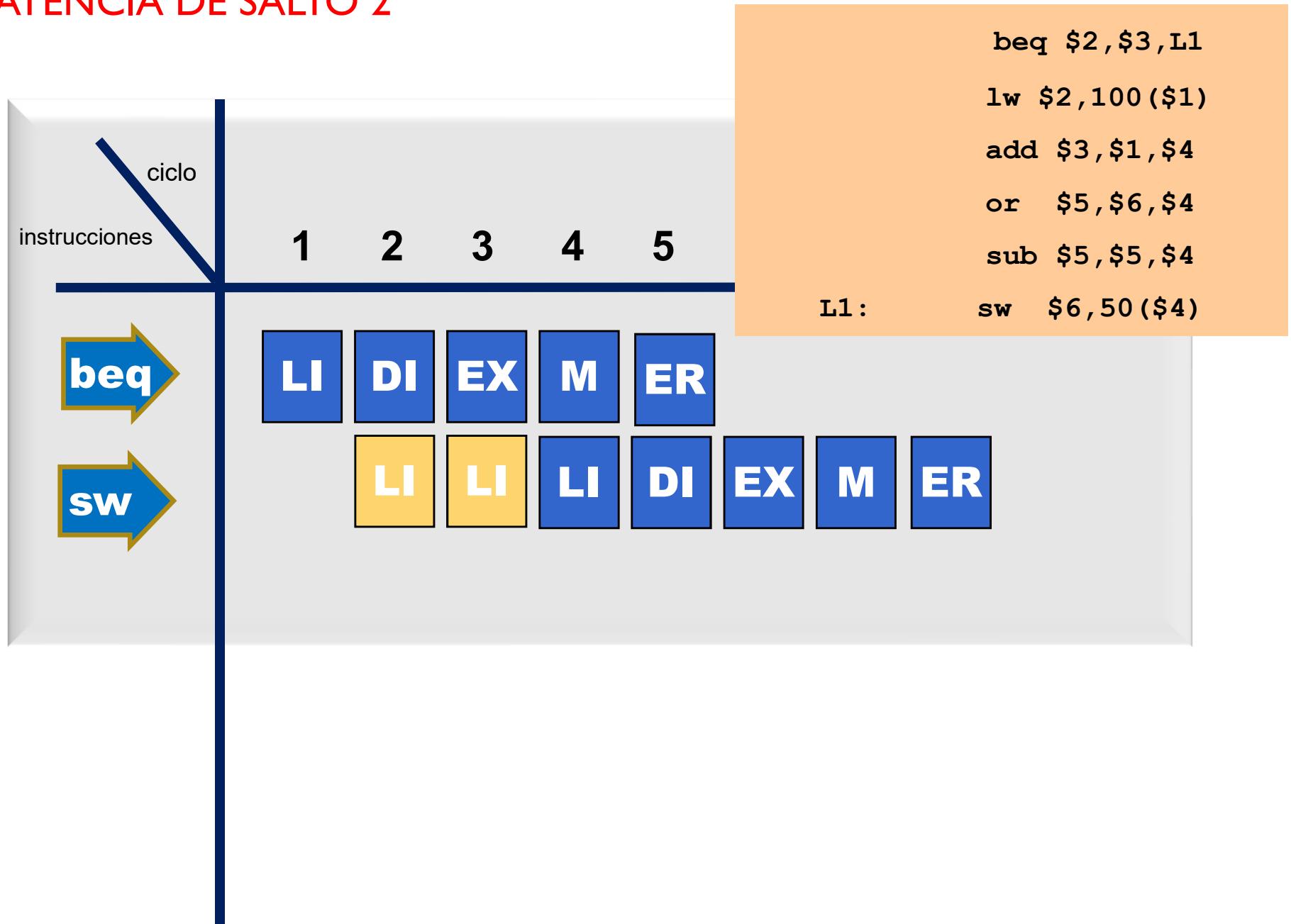
Solución conservativa (SOFTWARE)

LATENCIA DE SALTO 2



Solución conservativa (HARDWARE)

LATENCIA DE SALTO 2



Resumen : Riesgos de control

- Instrucción de salto (condicional o incondicional)

```
beq $4, $3,etiqueta  
sub $2, $9, $5  
addi $10, $10, -1  
lui $12, 0x1000  
etiqueta: ori $5, $0, 100
```

LATENCIA SALTO: 1,2 ó 3

Resumen : Riesgos de control

- Instrucción de salto (condicional o incondicional)

beq \$4, \$3,etiqueta
sub \$2, \$9, \$5
addi \$10, \$10, -1
lui \$12, 0x1000
etiqueta: ori \$5, \$0, 100

LATENCIA SALTO: 1,2 ó 3

	I	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Beq	LI	DI	EX	M	ER										
sub		LI	DI	EX	M	ER									
addi			LI	DI	EX	M	ER	M							
lui				LI	DI	EX	M	ER							
ori					LI	DI	EX	M	ER						

Ejemplo: Latencia 3



En ese instante entra la nueva
instrucción

Resumen : Riesgos de control

- Instrucción de salto (condicional o incondicional)

beq \$4, \$3,etiqueta
sub \$2, \$9, \$5
addi \$10, \$10, -1
lui \$12, 0x1000
etiqueta: ori \$5, \$0, 100

LATENCIA SALTO: 1,2 ó 3

	I	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Beq	LI	DI	EX	M	ER										
sub		LI	DI	EX	M	ER									
addi			LI	DI	EX	M	ER	M							
lui				LI	DI	EX	M	ER							
ori					LI	DI	EX	M	ER						

Ejemplo: Latencia 3



En ese instante entra la nueva instrucción



Si no hago algo, entran 3 instrucciones en la ruta de los datos, QUE NO DEBEN ENTRAR

Resumen : Riesgos de control

- Instrucción de salto (condicional o incondicional)

beq \$4, \$3,etiqueta
sub \$2, \$9, \$5
addi \$10, \$10, -1
lui \$12, 0x1000
etiqueta: ori \$5, \$0, 100

LATENCIA SALTO: 1,2 ó 3

Solución: Paradas o NOP

	I	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Beq	LI	DI	EX	M	ER										
sub		LI	DI	EX	M	ER									
addi			LI	DI	EX	M	ER	M							
lui				LI	DI	EX	M	ER							
ori					LI	DI	EX	M	ER						

Ejemplo: Latencia 3



En ese instante entra la nueva instrucción



Si no hago algo, entran 3 instrucciones en la ruta de los datos, QUE NO DEBEN ENTRAR

Resumen : Riesgos de control

- Instrucción de salto (condicional o incondicional)

beq \$4, \$3,etiqueta
sub \$2, \$9, \$5
addi \$10, \$10, -1
lui \$12, 0x1000
etiqueta: ori \$5, \$0, 100

LATENCIA SALTO: 1,2 ó 3

Solución: Paradas

	I	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Beq	LI	DI	EX	M	ER										
??		LI	LI	LI	LI	DI	EX	M	ER						

Ejemplo: Latencia 3



En ese instante entra la nueva
instrucción : sub o ori

Hay tantas paradas como
latencia de salto

Resumen : Riesgos de control

- Instrucción de salto (condicional o incondicional)

beq \$4, \$3,etiqueta
sub \$2, \$9, \$5
addi \$10, \$10, -1
lui \$12, 0x1000
etiqueta: ori \$5, \$0, 100

LATENCIA SALTO: 1,2 ó 3

Solución: Paradas

	I	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Beq	LI	DI	EX	M	ER										
??		LI	LI	LI	DI	EX	M	ER							



En ese instante entra la nueva instrucción : sub o ori

Hay tantas paradas como latencia de salto

Resumen : Riesgos de control

- Instrucción de salto (condicional o incondicional)

```
addi $4,$4,-1  
beq $4, $3,etiqueta  
sub $2, $9, $5  
addi $10, $10, -1  
lui $12, 0x1000  
etiqueta: ori $5, $0, 100
```

LATENCIA SALTO: 1,2 ó 3

¿Y si además tiene la
instrucción de salto Riesgo de
Datos?

Solución: Paradas

Resumen : Riesgos de control

- Instrucción de salto (condicional o incondicional)

addi \$4,\$4,-1
beq \$4, \$3,etiqueta
sub \$2, \$9, \$5
addi \$10, \$10, -1
lui \$12, 0x1000
etiqueta: ori \$5, \$0, 100

LATENCIA SALTO: 1,2 ó 3

¿Y si además tiene la
instrucción de salto Riesgo de
Datos?

Solución: Paradas

	I	2	3	4	5	6	7	8	9	10	11	12	13	14	15
add	LI	DI	EX	M	ER(\$t4)										
Beq		LI	DI	DI	DI	ER	M	ER							
i?				LI	LI	LI	LI	DI	EX	M	ER				

Ejemplo: Latencia 2



En ese instante entra la nueva
instrucción : sub o ori

Ejercicio I:T2_EjerciciosGenerales.docx

Preg. 4 Supóngase el siguiente código ejecutándose en un procesador que posee una latencia de salto de 2 ciclos y los riesgos se resuelven mediante la inserción de ciclos de parada.

```
(1)    __start:    addi $t0, $zero, 10
(2)                  lui  $t1, 0x1000
(3)                  ori  $t1, $t1, 0xA
(4)    bucle:      lw   $t2, 0($t1)
(5)                  lw   $t3, 0xA0($t1)
(6)                  add  $t4, $t3, $t2
(7)                  sw   $t3, 0($t1)
(8)                  addi $t0, $t0, -1
(9)                  bne $t0, $zero, bucle
(10)                 sw   $t4, 0xB0($t1)
```

- a) Identificar los riesgos por dependencia de datos y rellene la siguiente tabla. Nótese que el bucle se ejecuta 10 veces.

	Registro	Se escribe en	Se lee en	Ciclos de parada a insertar
Riesgo 1				
Riesgo 2				
Riesgo 3				
Riesgo 4				
Riesgo 5				
Riesgo 6				
Riesgo 7				



Tratamiento de los Conflictos de Control

- Primeras soluciones al conflicto

- ✓ Soluciones conservativas o de urgencia:

- Hardware
 - El decodificador inserta ciclos de parada al detectar un salto
 - Software
 - El compilador inserta NOP

- ✓ Soluciones avanzadas: predicción

- Predicción fija
 - *predict-not taken*
 - *predict taken*

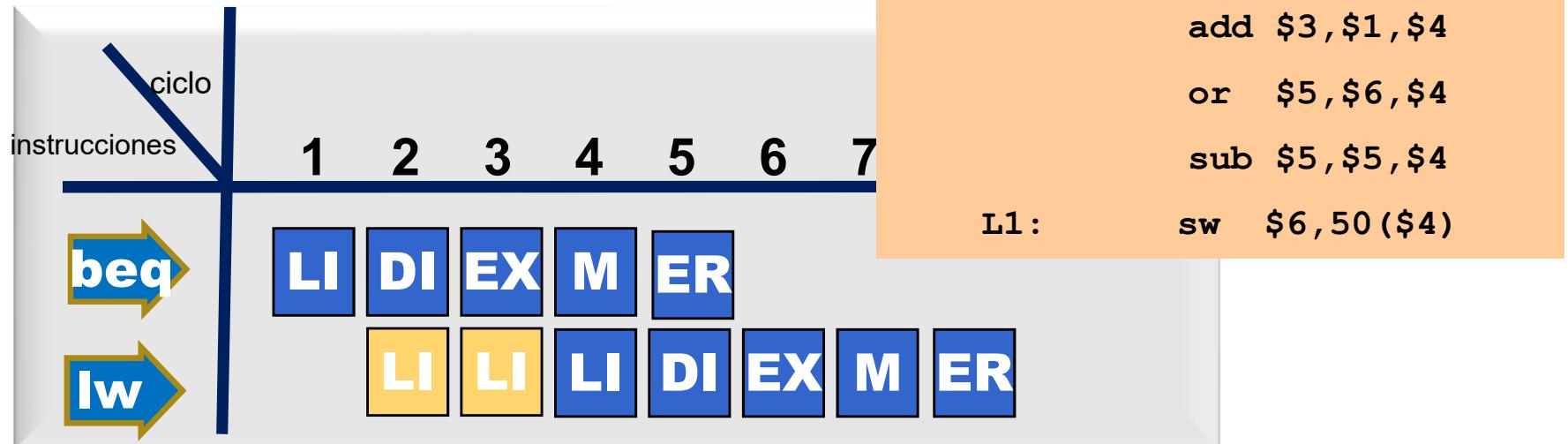
Intel i486

Sun SuperSparc

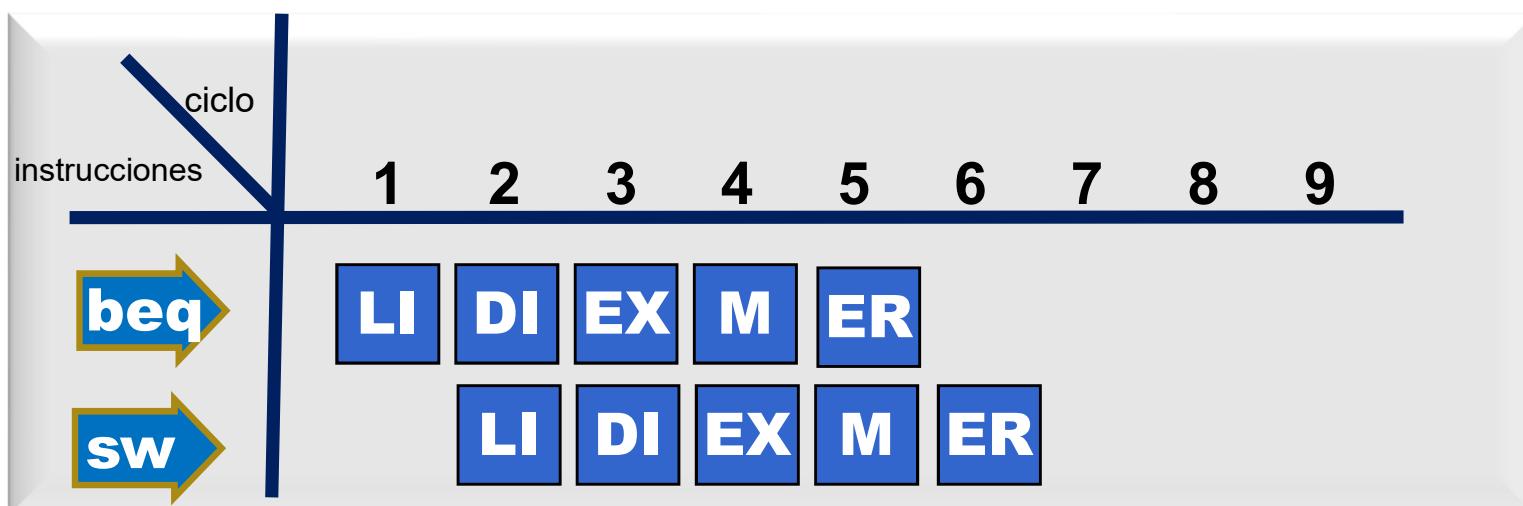
Solución PREDICTIVA: predict taken

LATENCIA DE SALTO 2

CASO DE FALLO: O sea NO SE SALTA



CASO DE ACIERTO: O sea SI SE SALTA



Ejercicio 3:T2_EjerciciosGenerales.docx

Preg. 3 Los siguientes fragmentos de código contienen riesgos cuando se ejecutan en un procesador segmentado como el visto en clase. Completa el cronograma tiempo-etapas e indique claramente los riesgos rodeando los registros que producen el riesgo. Asuma que la solución empleada para solucionar los conflictos es por ciclos de parada. Latencia de salto 2.

Considera que **SI** que se produce el salto y el procesador utiliza la técnica predict taken.

	1	2	3	4	5	6	7	8	9	10	11
lw \$6,50(\$7)											
beq \$7,\$8,L											
and \$1,\$2,\$3											
nop											
nop											
L:or \$7,\$6,\$8											

Procesador Predictivo: Hace una predicción fija, si acierta no hay problemas y solo si falla introduce las paradas

Predict-taken: Predice SIEMPRE que SALTA

Predict-No taken: Predice SIEMPRE que NO SALTA

Ejercicio 3:T2_EjerciciosGenerales.docx

Preg. 3 Los siguientes fragmentos de código contienen riesgos cuando se ejecutan en un procesador segmentado como el visto en clase. Completa el cronograma tiempo-etapas e indique claramente los riesgos rodeando los registros que producen el riesgo. Asuma que la solución empleada para solucionar los conflictos es por ciclos de parada. Latencia de salto 2.

Considera que **SI** que se produce el salto y el procesador utiliza la técnica predict taken.

	1	2	3	4	5	6	7	8	9	10	11
lw \$6,50(\$7)											
beq \$7,\$8,L											
and \$1,\$2,\$3											
nop											
nop											
L:or \$7,\$6,\$8											

Procesador Predictivo: Hace una predicción fija, si acierta no hay problemas y solo si falla introduce las paradas

Predict-taken: Predice SIEMPRE que SALTA

Predict-No taken: Predice SIEMPRE que NO SALTA

Según enunciado Predicción ACERTADA

Ejercicio 3:T2_EjerciciosGenerales.docx

Preg. 3 Los siguientes fragmentos de código contienen riesgos cuando se ejecutan en un procesador segmentado como el visto en clase. Completa el cronograma tiempo-etapas e indique claramente los riesgos rodeando los registros que producen el riesgo. Asuma que la solución empleada para solucionar los conflictos es por ciclos de parada. Latencia de salto 2.

Considera que **SI** que se produce el salto y el procesador utiliza la técnica predict taken.

	1	2	3	4	5	6	7	8	9	10	11
lw \$6,\$0(\$7)	U	DI	EX	M	ER						
beq \$7,\$8,L		U	DI	EX	M	ER					
and \$1,\$2,\$3											
nop											
nop											
L:or \$7,\$6,\$8			U	*	DI	EX	M	ER			



Esa parada es por la dependencia de datos

Procesador Predictivo: Hace una predicción fija, si acierta no hay problemas y solo si falla introduce las paradas

Predict-taken: Predice SIEMPRE que SALTA

Predict-No taken: Predice SIEMPRE que NO SALTA

Según enunciado Predicción ACERTADA

Ejercicio 3:T2_EjerciciosGenerales.docx

Preg. 3 Los siguientes fragmentos de código contienen riesgos cuando se ejecutan en un procesador segmentado como el visto en clase. Completa el cronograma tiempo-etapas e indique claramente los riesgos rodeando los registros que producen el riesgo. Asuma que la solución empleada para solucionar los conflictos es por ciclos de parada. Latencia de salto 2.

¿Cómo habría sido la solución si el enunciado dijera que NO SALTA?

Considera que ~~\$1~~ que se produce el salto y el procesador utiliza la técnica predict taken.

	1	2	3	4	5	6	7	8	9	10	11
lw \$6,50(\$7)											
beq \$7,\$8,L											
and \$1,\$2,\$3											
nop											
nop											
L:or \$7,\$6,\$8											

Procesador Predictivo: Hace una predicción fija, si acierta no hay problemas y solo si falla introduce las paradas

Predict-taken: Predice SIEMPRE que SALTA

Predict-No taken: Predice SIEMPRE que NO SALTA

Ejercicio 3:T2_EjerciciosGenerales.docx

Preg. 3 Los siguientes fragmentos de código contienen riesgos cuando se ejecutan en un procesador segmentado como el visto en clase. Completa el cronograma tiempo-etapas e indique claramente los riesgos rodeando los registros que producen el riesgo. Asuma que la solución empleada para solucionar los conflictos es por ciclos de parada. Latencia de salto 2.

¿Cómo habría sido la solución si el enunciado dijera que NO SALTA?

Considera que ~~\$1~~ que se produce el salto y el procesador utiliza la técnica predict ~~taken~~.

	1	2	3	4	5	6	7	8	9	10	11
lw \$6,50(\$7)	U	DI	EX	M	ER						
beq \$7, \$8, L		U	DI	EX	M	ER					
and \$1, \$2, \$3			•	•	U	DI	EX	M	ER		
nop											
nop											
<u>L:or \$7,\$6,\$8</u>											

FALLO de predicción y entonces PARADAS

Procesador Predictivo: Hace una predicción fija, si acierta no hay problemas y solo si falla introduce las paradas

Predict-taken: Predice SIEMPRE que SALTA

Predict-No taken: Predice SIEMPRE que NO SALTA

Ejercicio 4:T2_EjerciciosGenerales.docx

d) Considere que se ha ejecutado todo el programa completo. Rellene la siguiente tabla:

```

(1) __start:    addi $t0, $zero, 10
(2)              lui  $t1, 0x1000
(3)              ori  $t1, $t1, 0xA
(4) bucle:      lw   $t2, 0($t1)
(5)              lw   $t3, 0xA0($t1)
(6)              add  $t4, $t3, $t2
(7)              sw   $t3, 0($t1)
(8)              addi $t0, $t0, -1
(9)              bne $t0, $zero, bucle
(10)             sw   $t4, 0xB0($t1)

```

Instrucciones ejecutadas (I)	$I = 3 + (6 * 10) + 1 = 64 \text{ instrucciones}$
Ciclos de parada (P)	$P = 4_{\text{DATOS}} + (4_{\text{DATOS}} + 10 \text{ veces}) + (2_{\text{BNE}} * 10 \text{ veces}) = 64$
Ciclos totales de ejecución (T)	$T = \text{llenado} + I + P = 4 + 64 + 64 = 132$
CPI	$CPI = 1 + P / I = 1 + 64 / 64 = 2$

e) Si la solución a los conflictos de control fueran solucionadas por la técnica predict taken. ¿Cuántos ciclos se tardaría y cuál sería el CPI resultante?

Instrucciones ejecutadas (I)
Ciclos de parada (P)
Ciclos totales de ejecución (T)
CPI

b) Rellene el diagrama de instrucciones/tiempo adjunto solo hasta la primera ejecución del bucle.



Latencia de SALTO: 2

Prestaciones



Las soluciones conservativas (inserción de instrucciones NOP o ciclos de espera) no permiten mejorar el rendimiento del procesador

Para no tener ciclos improductivos el compilador puede encontrar instrucciones útiles en lugar de NOP. Esto supone re-organizar el código.

Alternativa: Predicción de salto



- Predicción estática
 - Salto efectivo
 - Salto No efectivo
- Predicción dinámica
 - (tiene en cuenta el histórico)

Conclusiones

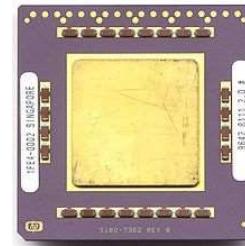


- Las instrucciones de salto (control de flujo) suponen la ruptura de la secuencia de ejecución de un programa y por tanto una pérdida de rendimiento
- Las soluciones conservativas se limitan a resolver el problema para garantizar la correcta secuenciación del programa
- Para mejorar el rendimiento intentaremos que se puedan aprovechar las instrucciones que ya han entrado en el procesador.
- Las técnicas de predicción de salto van encaminadas a ello.

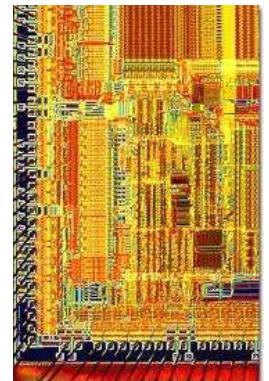
Contenidos y Bibliografía

- El proceso de segmentación
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ Identificación de las fases de ejecución de las instrucciones
 - ✓ Diseño de las etapas
- Conflictos y riesgos
 - ✓ Conflictos de datos
 - ✓ Conflictos de control (flujo) ([Visualizar vídeo 3](#))
- Perspectiva histórica. RISC-CISC y otras alternativas
[\(Leer Segmentacion Procesadores Actuales.pdf\)](#)

Ejemplos de Segmentación Básica



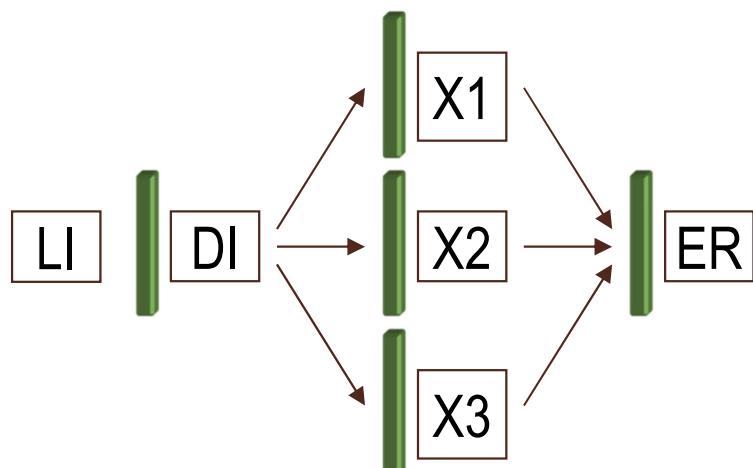
- La segmentación básica en los procesadores comerciales
 - ✓ El modelo de segmentación en cinco etapas es la base del diseño de los procesadores modernos (excepto la familia Intel x86)
 - ✓ Ejemplos:
 - MIPS R2000, R3000, ...
 - Sun SPARC V7
 - Hewlett Packard – Precision Architecture (HP-PA)
 - ✓ Familia Intel x86
 - 80486: también estaba segmentado en 5 etapas, pero con diferente organización



Otros Modelos de Segmentación

- Segmentación con etapas multiciclo
 - ✓ Permite la ejecución de operaciones de larga duración
 - aritmética de coma flotante, multimedia, etc.
 - ✓ Dispone de una colección de diversos operadores, cada uno con latencia L y tasa de repetición R específicos
 - acceso a memoria (típicamente $L \geq I$, $R=I$)
 - aritmética entera ($L=I$, $R=I$)
 - aritmética de CF ($L>I$, $R>=I$)

Ejemplo:



unidad	L	R
X1 (mem)	2	I
X2 (enteros)	I	I
X3 (CF)	3	3

Otros Modelos de Segmentación

- Segmentación con etapas multiciclo (cont.)
 - ✓ Hace falta una lógica de emisión (*issue*) que distribuya el trabajo entre los diversos operadores
 - ✓ El orden en que acaba la ejecución de las instrucciones puede ser distinto al orden en que han sido decodificadas

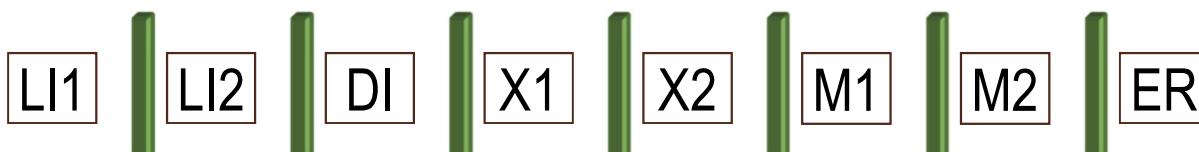
instrucción	1	2	3	4	5	6	7
lw		LI	DI	XI	XI	ER	
add.s (CF)			LI	DI	X3	X3	X3 ER
add (enters)				LI	DI	X2	ER

- ✓ La ruta de datos puede padecer conflictos estructurales
 - Ejemplo: ¿qué pasa si van seguidas dos instrucciones de CF?

Otros Modelos de Segmentación

- Supersegmentación

- ✓ La tarea a realizar en alguna/s etapa/s del modelo básico se reparte (segmenta) entre varias etapas
 - Aumento del número de etapas
 - Objetivo: incrementar la frecuencia de reloj (etapas más cortas)
 - CPI ideal = 1
- ✓ Ejemplo: MIPS R4000: 8 etapas LI1,LI2,DI,EX,M1,M2,M3,ER
 - LI1 y LI2 hacen la lectura de instrucción
 - M1, M2 y M3 realizan el acceso a la memoria de datos



- ✓ Otros ejemplos

- SPARC-V8 (9 etapas)
- Alpha 21064 (7 etapas)

Otros Modelos de Segmentación

- Procesadores superescalares

- ✓ Cada etapa puede procesar n instrucciones

- Intención: reducir CPI ideal a $1/n$ manteniendo la frecuencia de reloj
 - Se utiliza el índice IPC (Instrucciones Por Ciclo) = n

- ✓ Ejemplo con n=2 (CPI ideal = 0.5; IPC ideal = 2)

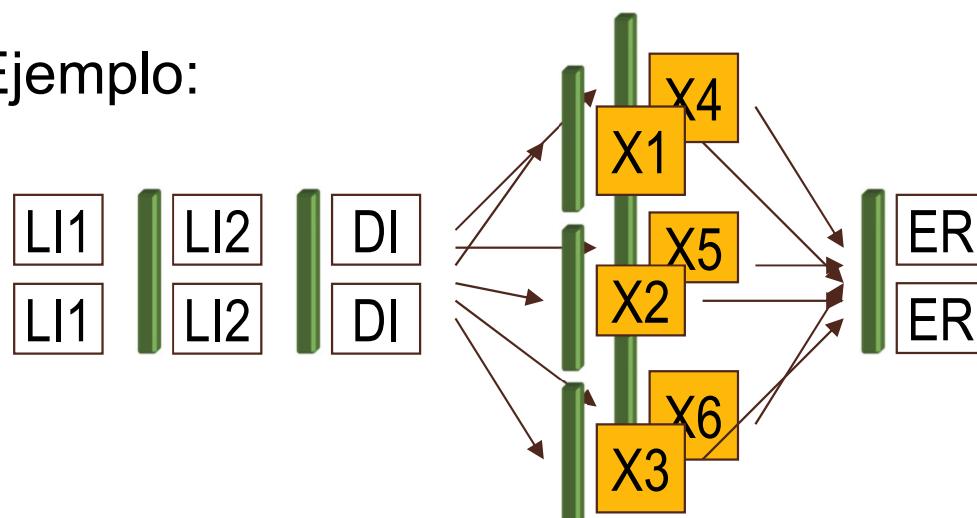
Instr.	1	2	3	4	5	6	7	LI	DI	EX	M	ER
i1		LI	DI	EX	M	ER		LI	DI	EX	M	ER
i2		LI	DI	EX	M	ER		LI	DI	EX	M	ER
i3			LI	DI	EX	M	ER					
i4			LI	DI	EX	M	ER					
i5				LI	DI	EX	M	ER				
i6				LI	DI	EX	M	ER				

Otros Modelos de Segmentación

- Combinaciones

- ✓ Los procesadores actuales combinan diversos modelos de segmentación
- ✓ Parámetros significativos:
 - Número de etapas
 - Grado de escalabilidad = número de instrucciones que se pueden leer o descodificar por ciclo
 - Surtido de operadores y características (L, R) de cada uno

Ejemplo:



Supersegmentado, superescalar de grado = 2

Unidad	L	R
X1 (I/s memoria)	3	1
X2 (s/r enteros)	1	1
X3 (p/d enteros)	4	3
X4 (s/r CF)	5	3
X5 (p/d CF)	15	15
X6 (multimedia)	3	1

Ejercicio 4:T2_EjerciciosGenerales.docx

Preg. 4 Supóngase el siguiente código ejecutándose en un procesador que posee una latencia de salto de 2 ciclos y los riesgos se resuelven mediante la inserción de ciclos de parada.

```
(1)    __start:    addi $t0, $zero, 10
(2)                  lui  $t1, 0x1000
(3)                  ori  $t1, $t1, 0xA
(4)    bucle:      lw   $t2, 0($t1)
(5)                  lw   $t3, 0xA0($t1)
(6)                  add  $t4, $t3, $t2
(7)                  sw   $t3, 0($t1)
(8)                  addi $t0, $t0, -1
(9)                  bne $t0, $zero, bucle
(10)                 sw   $t4, 0xB0($t1)
```

- g) ¿Qué se podría esperar en términos de productividad máxima del empleo de un procesador superescalar de 2 vías trabajando a una frecuencia de reloj igual a de la del procesador anterior?

RISC vs CISC: Dos Estrategias de Diseño

CISC: Complex Instruction Set Computer	RISC: Reduced Instruction Set Computer
Juego de instrucciones grande y complejo	Juego de instrucciones reducido y sencillo
Formato de instrucciones variable (heterogéneo) con diferentes longitudes	Formato de instrucciones fijo (homogéneo) con la misma longitud
Múltiples modos de direccionamiento complejos	Pocos modos de direccionamiento y sencillos
Número de registros reducido	Número de registros grande
Múltiples instrucciones pueden acceder a memoria	Arquitectura de carga-almacenamiento
Unidad de control compleja (microprogramada)	Unidad de control sencilla (cableada)
Objetivo: instrucciones complejas	Objetivo ejecutar cada instrucción en un solo ciclo
Menor complejidad en el compilador	Mayor complejidad en el compilador
Mayor dificultad para la segmentación	Mayor facilidad en la segmentación
Arquitectura típica de los años 70 y 80	Arquitectura popular en los años 90
Intel x86, celeron, Pentium,AMD (Duron,Athlon)	Power PC, DEC Alpha, MIPS, ARM, SPARC , HP
En la actualidad no hay una clara diferencia entre CISC-RISC. Los procesadores no obedecen a una clasificación tan polarizadas. Los procesadores Intel se consideran CISC-in RISC-out	

RISC vs CISC: Dos Estrategias de Diseño

- Estudios llevados a cabo en los años 80 demostraron:
 - ✓ Los procesadores (CISC) dedicaban el 80% de su tiempo a ejecutar solo un 20% de las instrucciones disponibles en su juego de instrucciones
 - ✓ Algunas instrucciones no se utilizaban nunca y otras muy pocas veces
- Al reducir el juego de instrucciones y homogeneizar el formato
 - ✓ Se ejecutan las instrucciones más rápidamente
 - ✓ Unidades de control más simples, rápidas, y ocupan menos espacio
 - ✓ Utilizar el espacio del chip para incluir más registros y memoria cache
 - ✓ Hacer la segmentación más eficiente
- CISC reduce I a costa de incrementar el CPI
- RISC reduce el CPI aumentando I

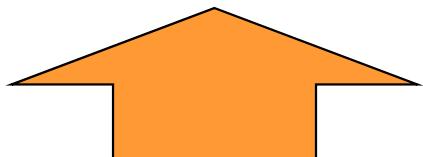
$$\frac{\text{Tiempo}}{\text{programa}} = T_{\text{ciclo}} * \frac{\text{ciclos}}{\text{instrucción}} * \frac{\text{instrucciones}}{\text{programa}}$$

CISC vs RISC: Dos Estrategias de Diseño

- Multiplicación de dos números que se encuentran ubicados en memoria guardando el producto en la dirección del operando I:
 - ✓ dir1: dirección de memoria del operando 1
 - ✓ dir2: dirección de memoria del operando 2

CISC

MULT dir1, dir2



Poco trabajo para el compilador

Baja ocupación de la memoria

Mucho trabajo ejecutado por cada instrucción

Elevado tiempo de ejecución

RISC

li \$t0, dir1

li \$t1, dir2

lw \$t3, 0(\$t0)

lw \$t4, 0(\$t1)

mul \$t3, \$t4

mflo \$t5

sw \$t5, 0(\$t0)



Más trabajo para el compilador

Mayor utilización de la memoria

Más instrucciones pero más simples

Menor tiempo de ejecución por instrucción

Contenidos y Bibliografía

- El proceso de segmentación
 - ✓ Concepto
 - ✓ Diagramas temporales
 - ✓ Prestaciones
- Segmentación de la ruta de datos básica
 - ✓ Identificación de las fases de ejecución de las instrucciones
 - ✓ Diseño de las etapas
- Conflictos y riesgos
 - ✓ Conflictos de datos
 - ✓ Conflictos de control (flujo)
- Perspectiva histórica. RISC-CISC y otras alternativas
(Leer “Segmentación de procesadores actuales.pdf”)

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4^a edición, Ed. Reverté, 2011, Cap 4 (4.5 – 4.8)