



# Arquitectura de Computadores



## Tema 1-2. Procesadores Segmentados



# Sumario

- **Parte 1**
  - Conexiones externas
  - Elementos de la microarquitectura del procesador
  - Ejecución multiciclo de instrucciones: ALU, MEM, Salto
  - Etapas de ejecución de las instrucciones
  - Ejecución segmentada
  - Registros de segmentación
  - Ejecución de instrucciones en un procesador segmentado: 1 resultado/ciclo



# Sumario

## • Parte 2

- Modelo segmentado del procesador DLX32p
  - » Cálculo de speed-up
  - » Dependencias estructurales
- Dependencias Estructurales
  - » Números de puertos de la cache L1
- Dependencias de datos
  - » Dependencias de datos verdaderas (RAW)
  - » Antidependencias de datos (WAR)
  - » Dependencias de salida (WAW)
  - » Anticipación en DLX32p
- Dependencias de control en DLX32p
- Procesador segmentado con unidades funcionales multicilo DLX32mf
  - » Latencia
  - » Intervalo de repetición
  - » Dependencias estructurales
  - » Dependencia datos
- Evaluación de prestaciones de MIPS R4000 con SPEC CPU 89



# Arquitectura de Computadores

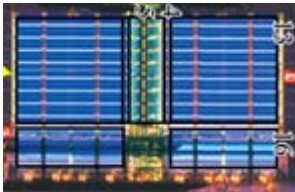


## Tema 1-2. Procesadores Segmentados, Parte 1



# Conexiones Externas del Procesador

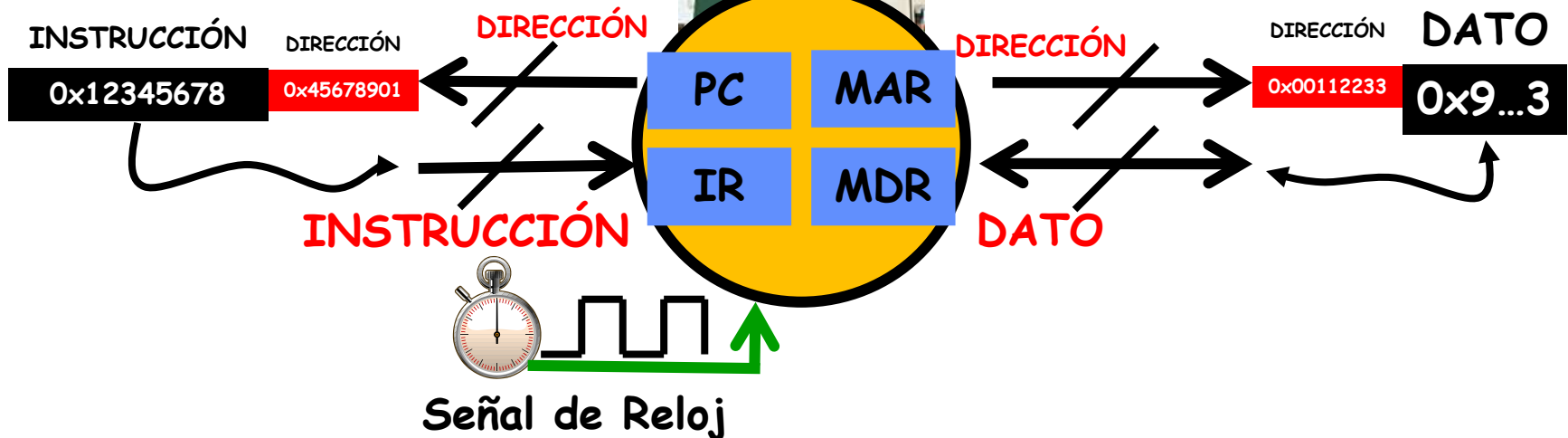
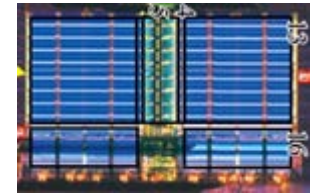
Memoria de Programa

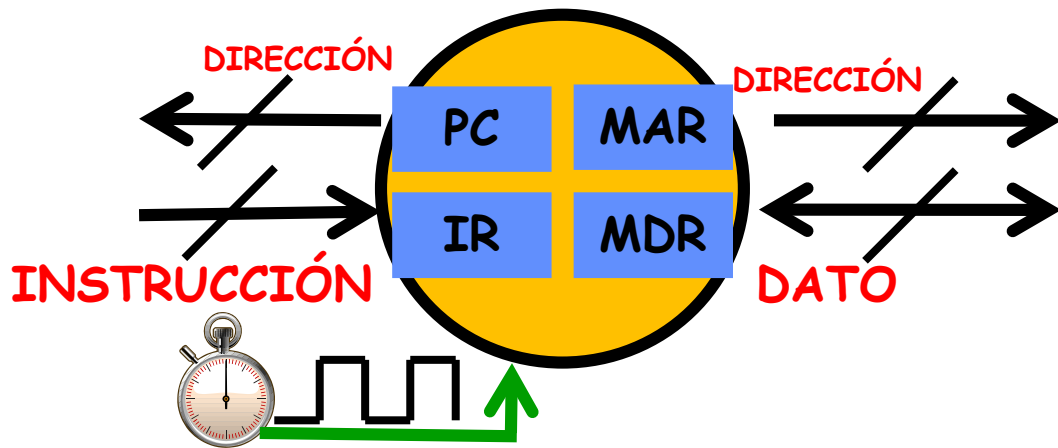


Procesador  
RISC

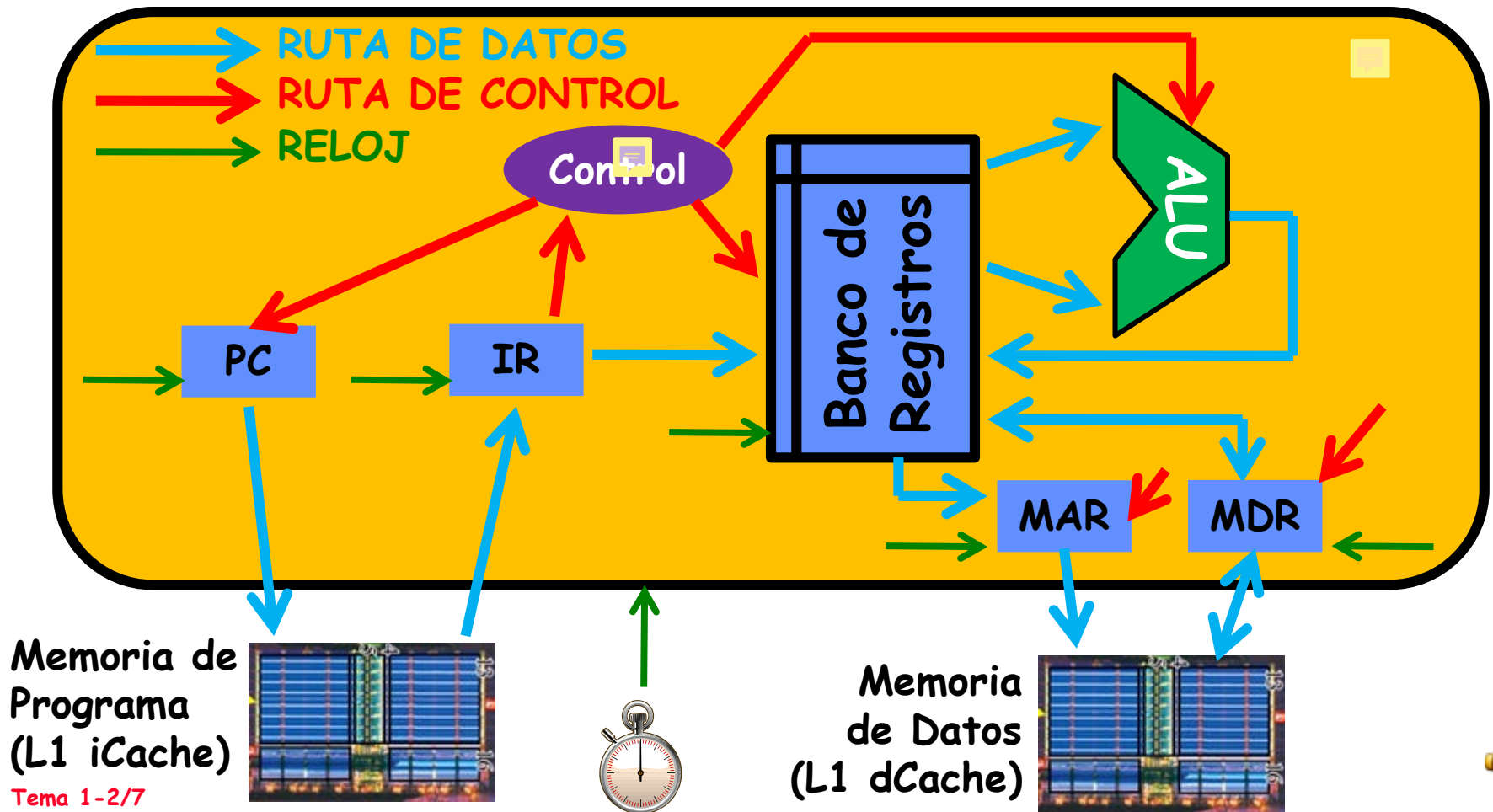


Memoria de Datos





# Estructura interna del procesador



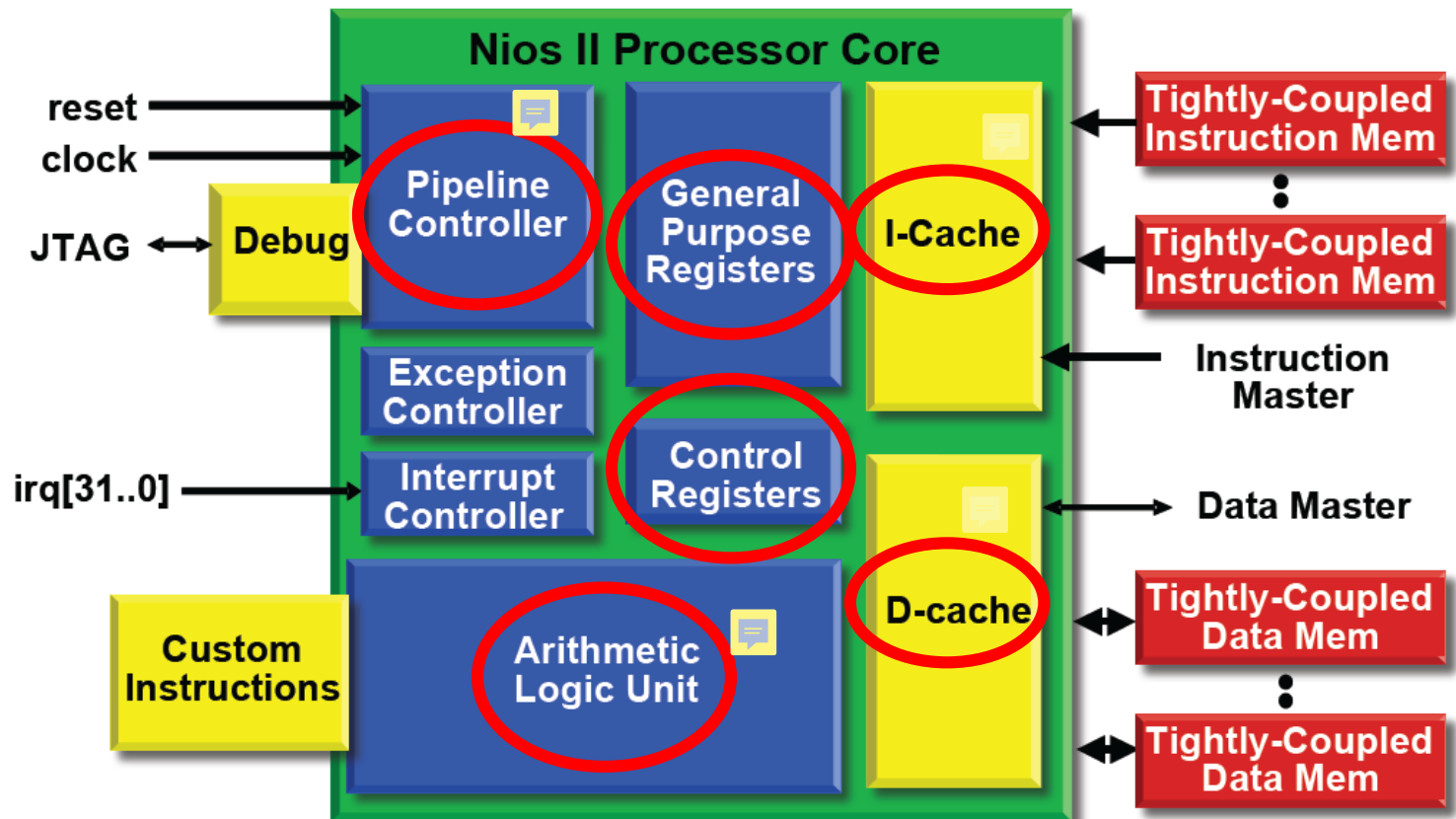
Memoria de Programa (L1 iCache)

Memoria de Datos (L1 dCache)

# Estructura interna de NIOS II



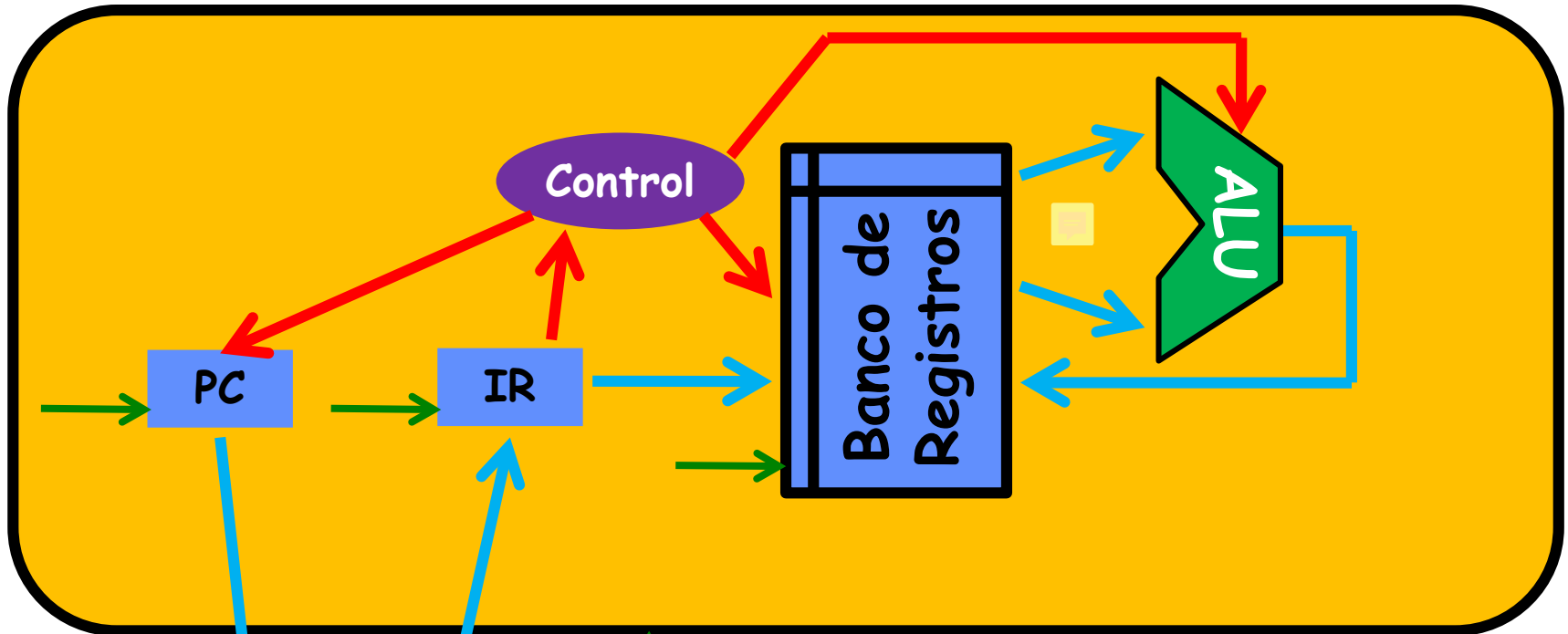
## Nios II Processor Block Diagram



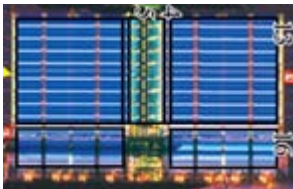
# Ejecución de una instrucción aritmético-lógica



→ RUTA DE DATOS  
→ RUTA DE CONTROL  
→ RELOJ



Memoria de  
Programa  
(L1 iCache)

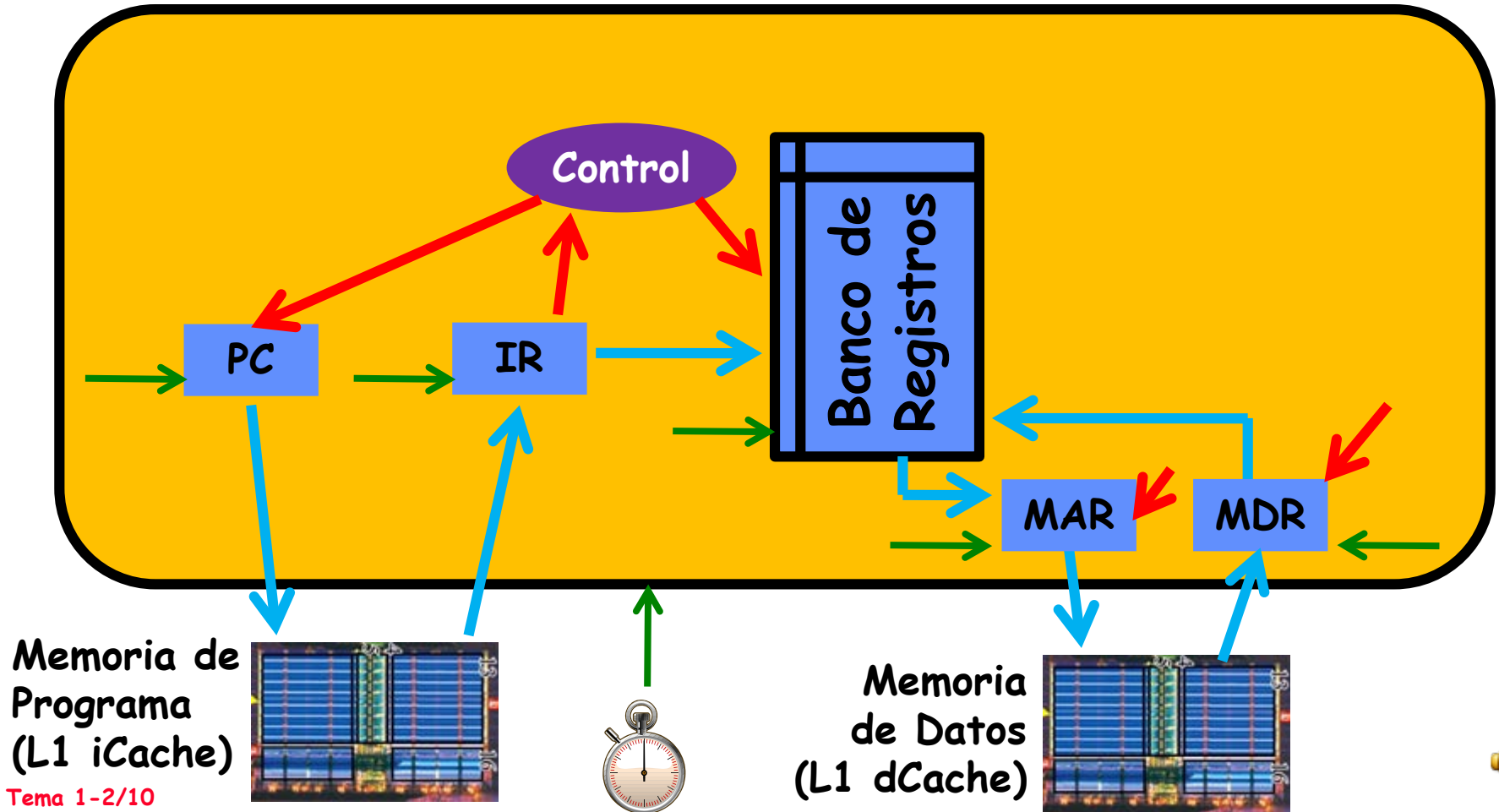




# Ejecución de una instrucción de carga desde memoria



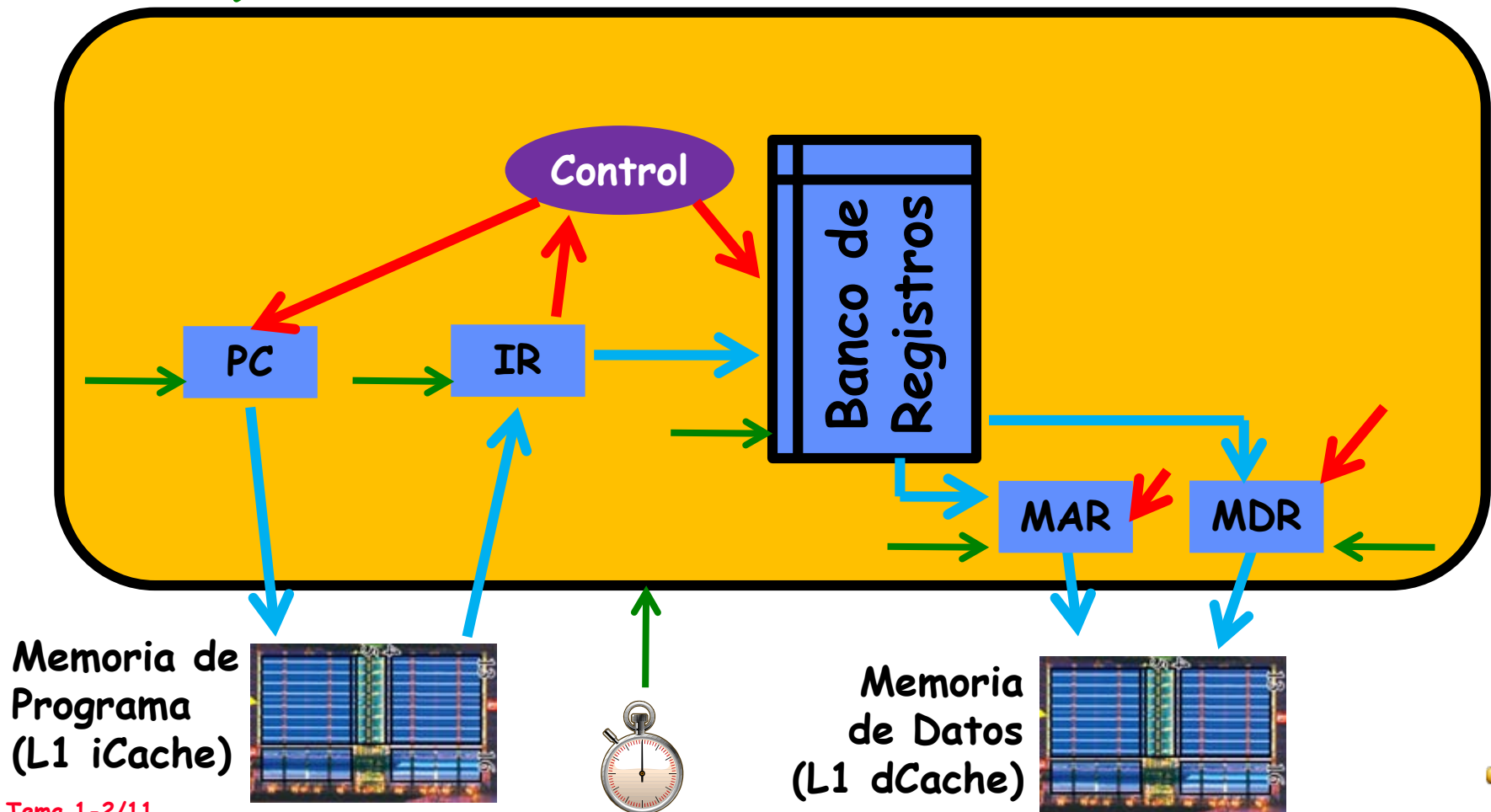
→ RUTA DE DATOS  
→ RUTA DE CONTROL  
→ RELOJ



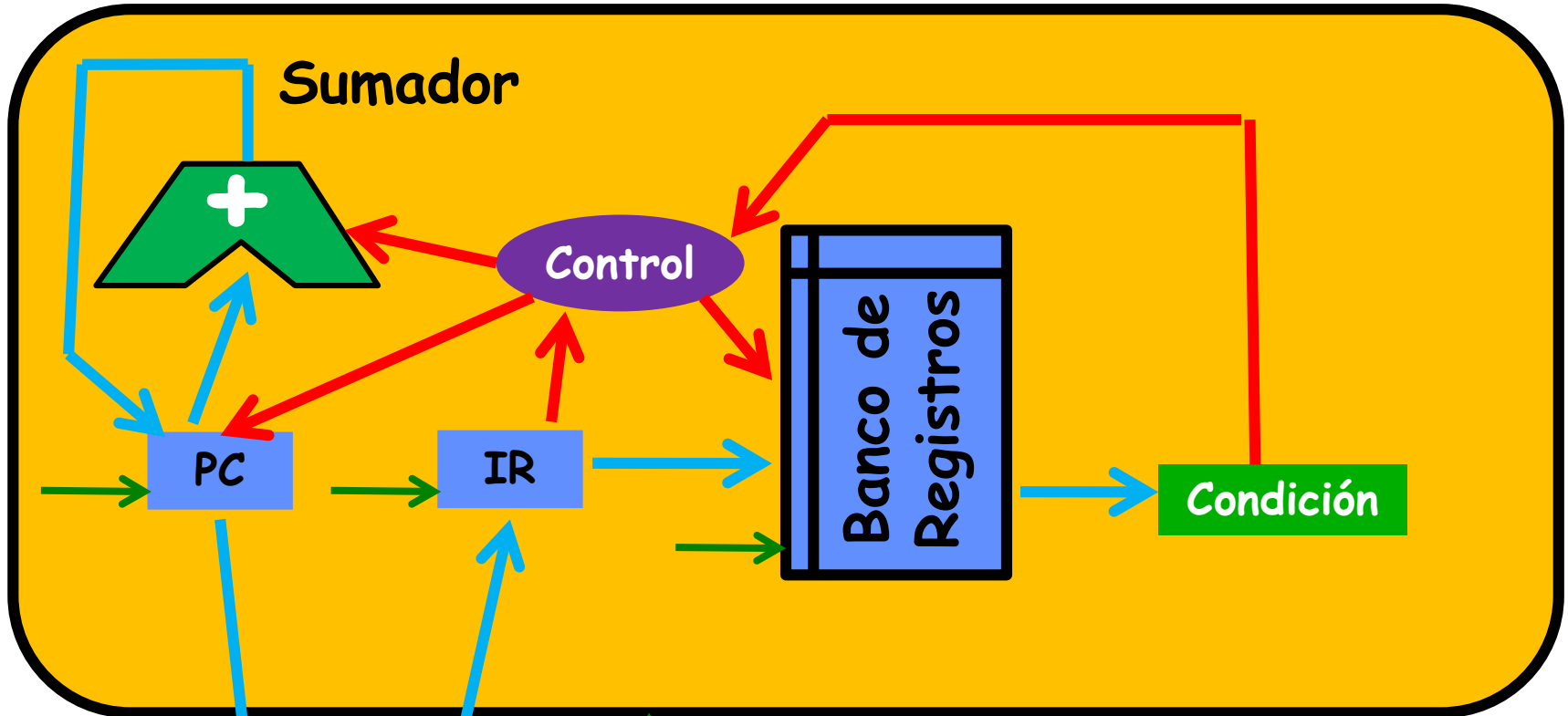
# Ejecución de una instrucción de almacenamiento en memoria



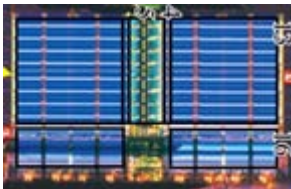
→ RUTA DE DATOS  
→ RUTA DE CONTROL  
→ RELOJ



# Ejecución de una instrucción de salto



Memoria de Programa  
(L1 iCache)

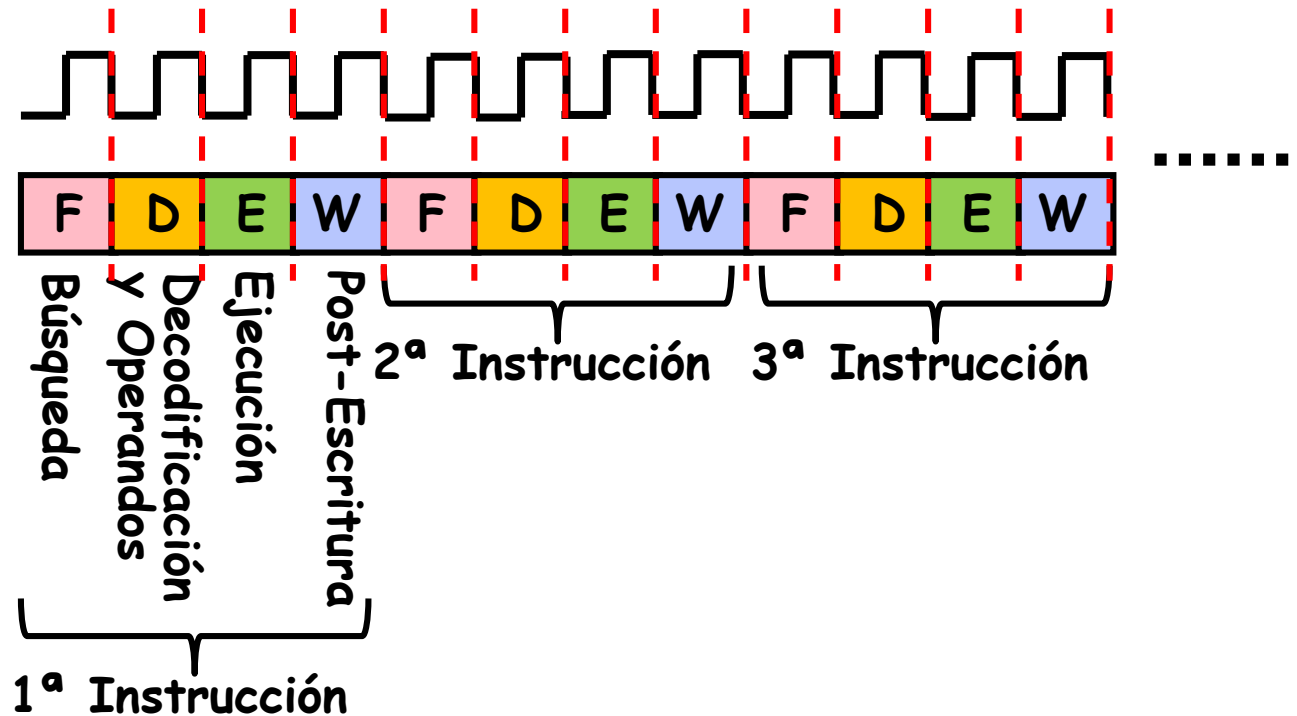
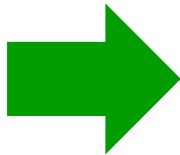


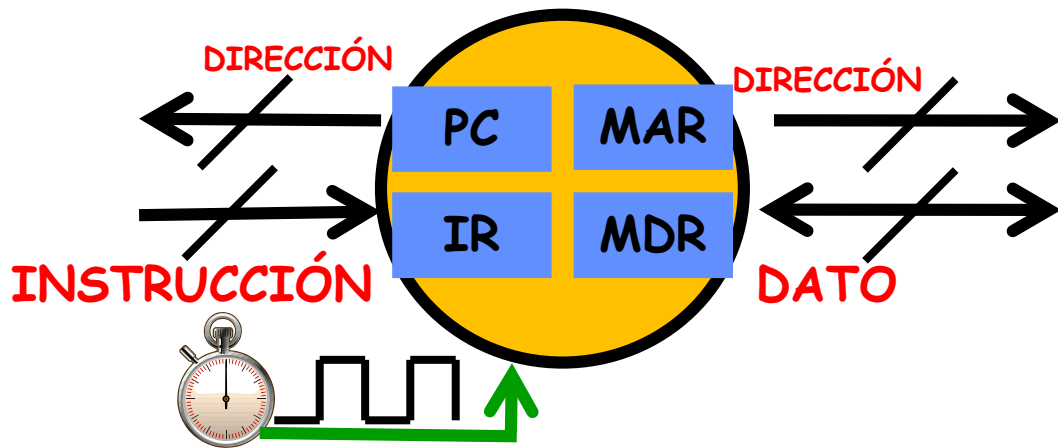


# Ejecución MULTICICLO de instrucciones

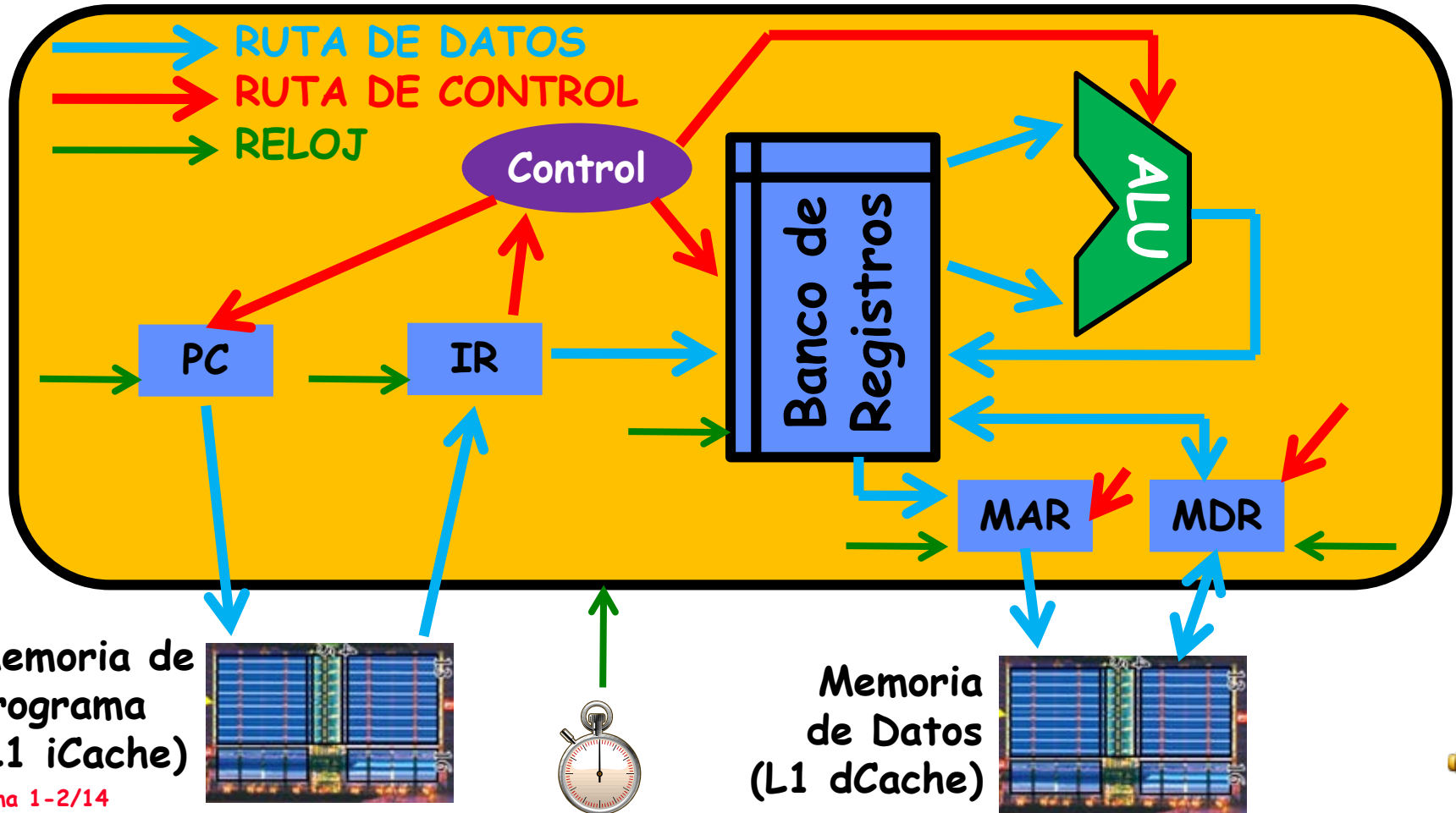


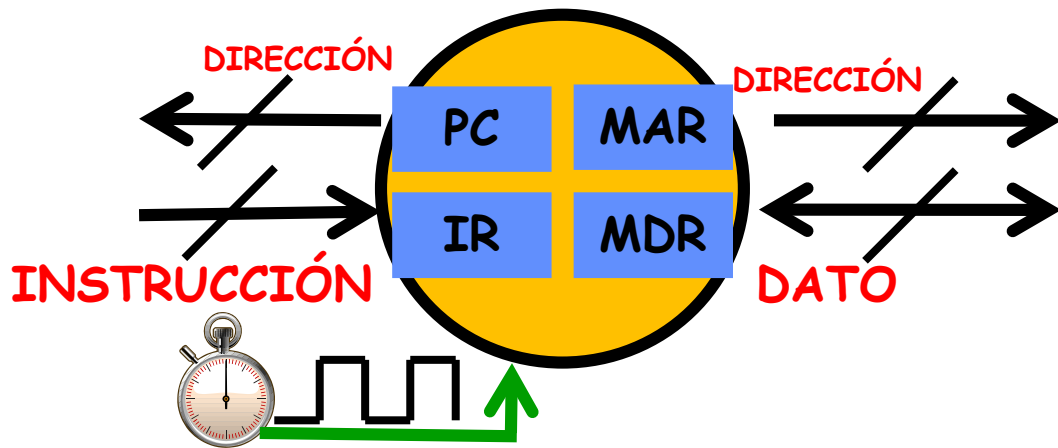
Señal de reloj  
del procesador



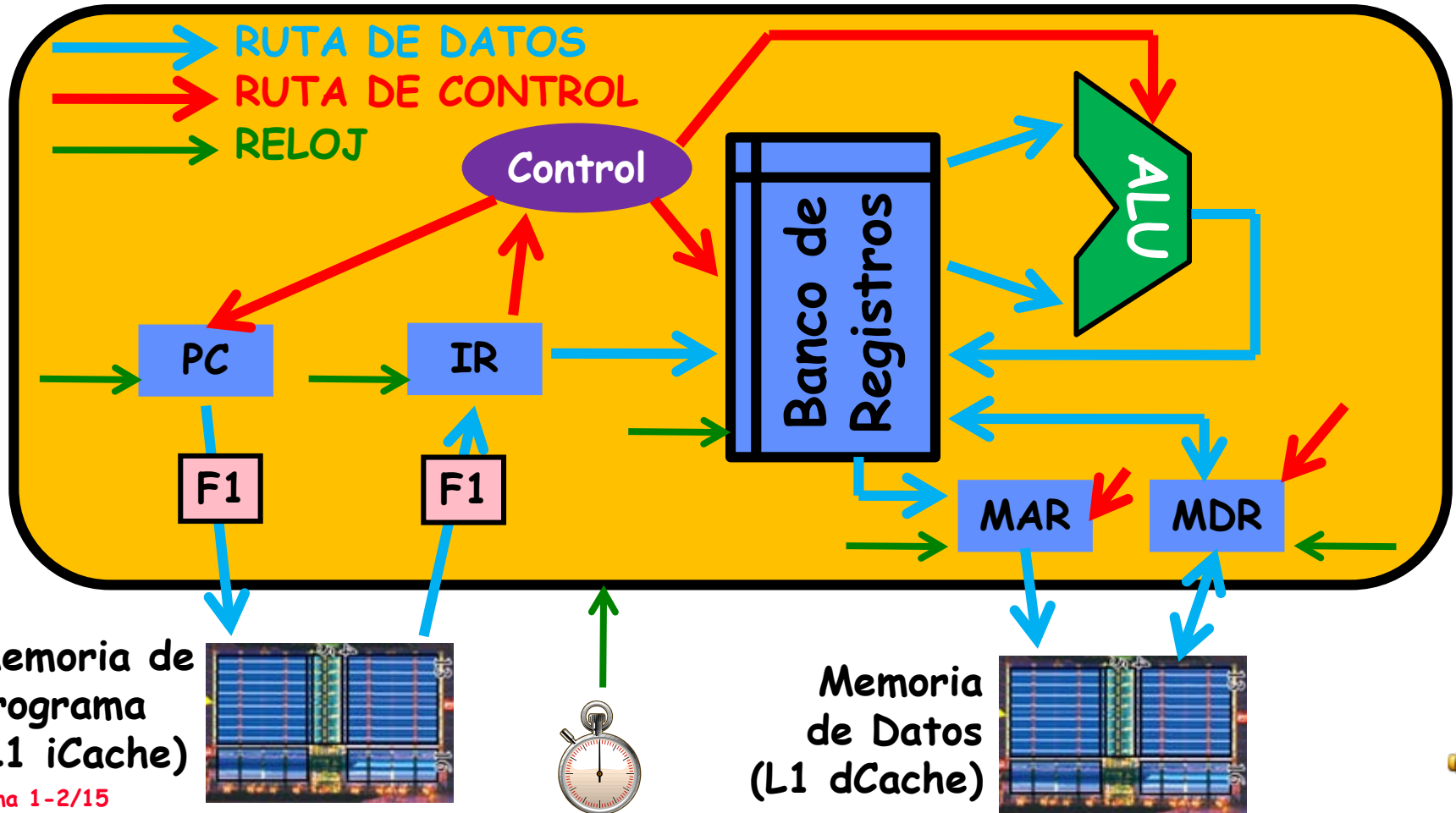


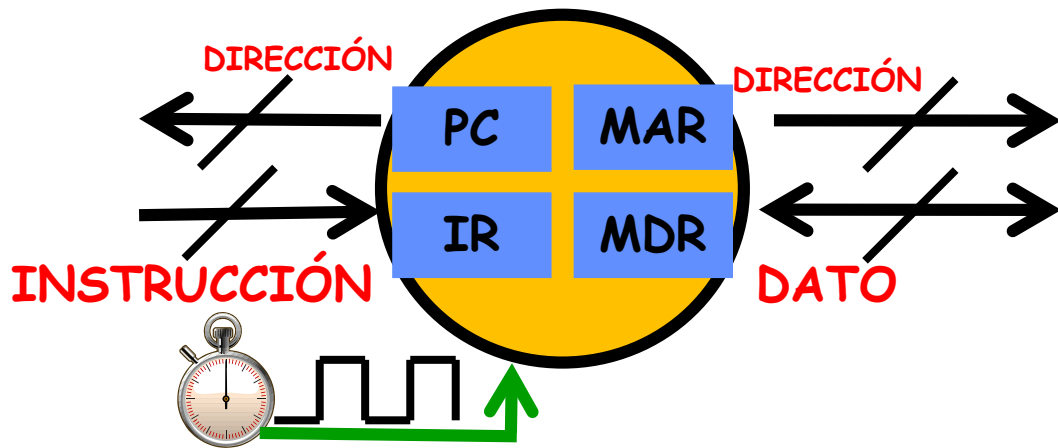
# Estructura interna de un procesador multiciclo



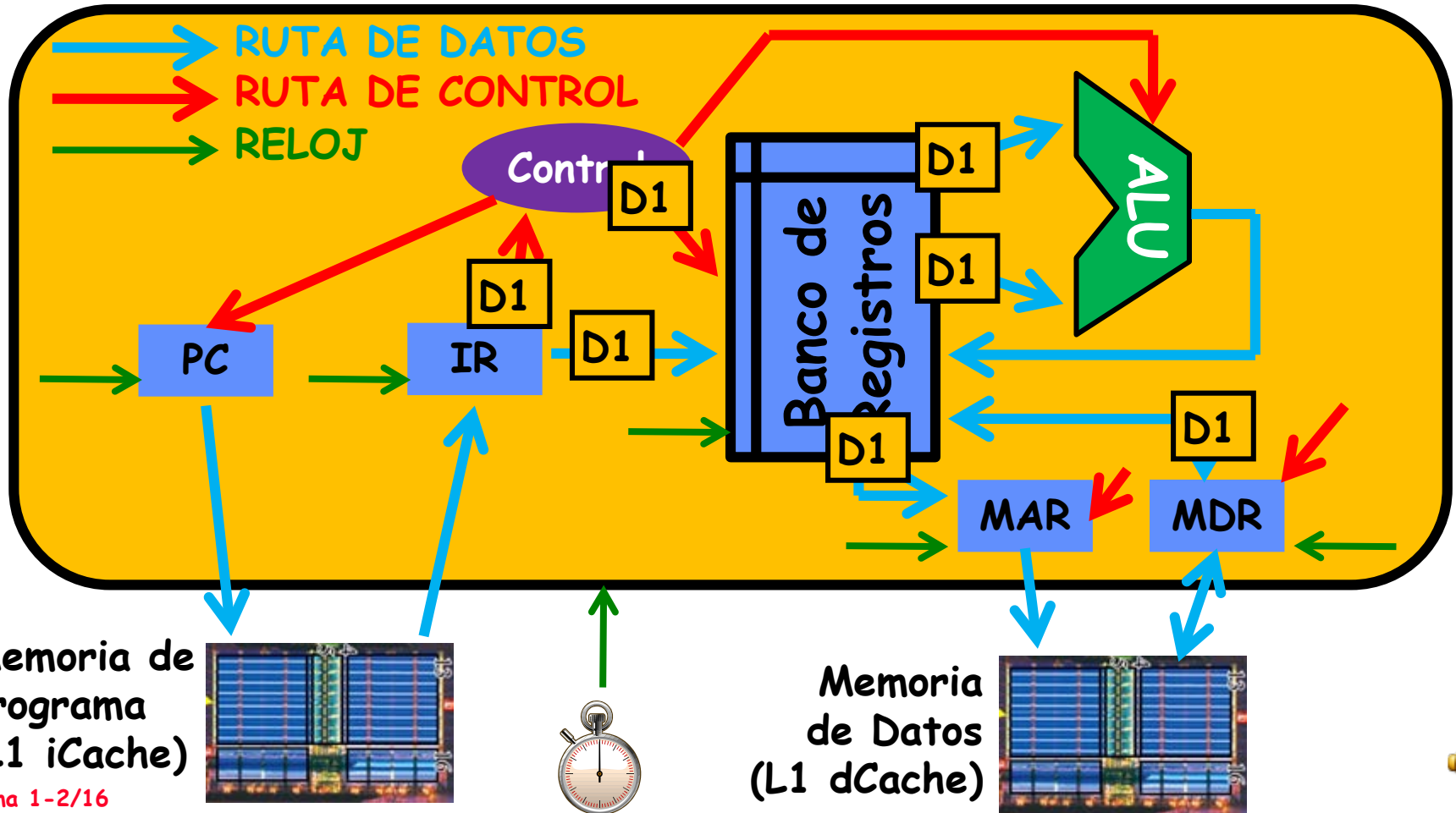


# Estructura interna de un procesador multiciclo: ETAPA F





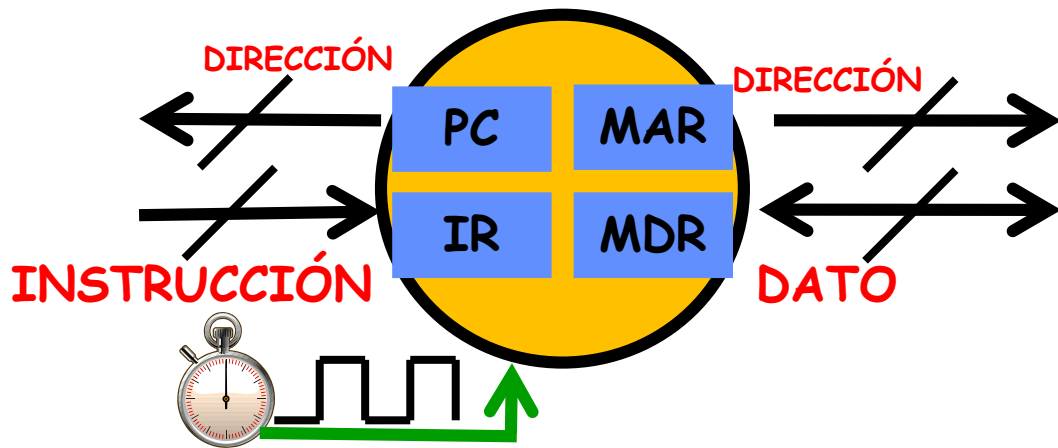
# Estructura interna de un procesador multiciclo: ETAPA D



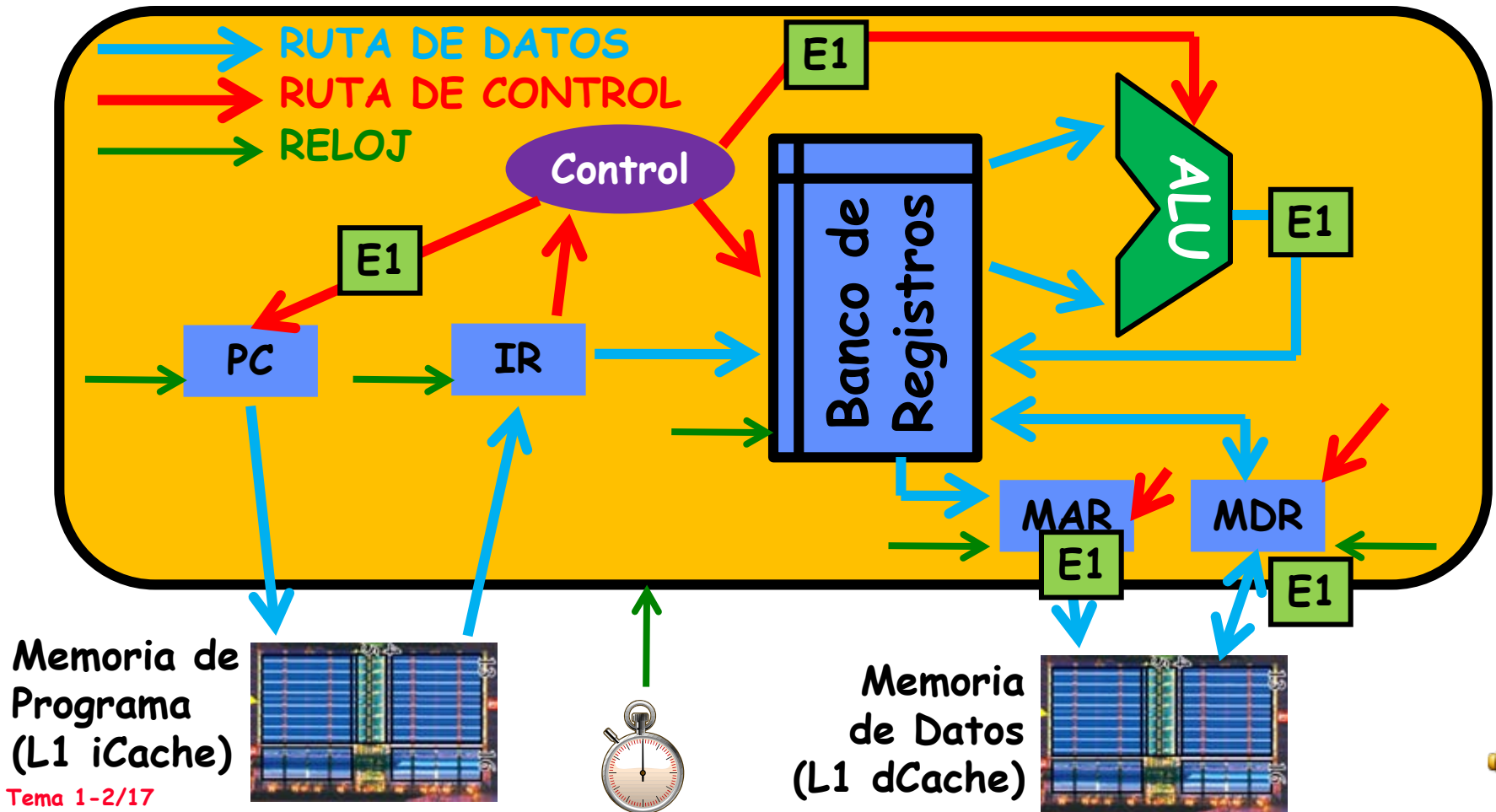
Memoria de Programa  
(L1 iCache)

Memoria de Datos  
(L1 dCache)

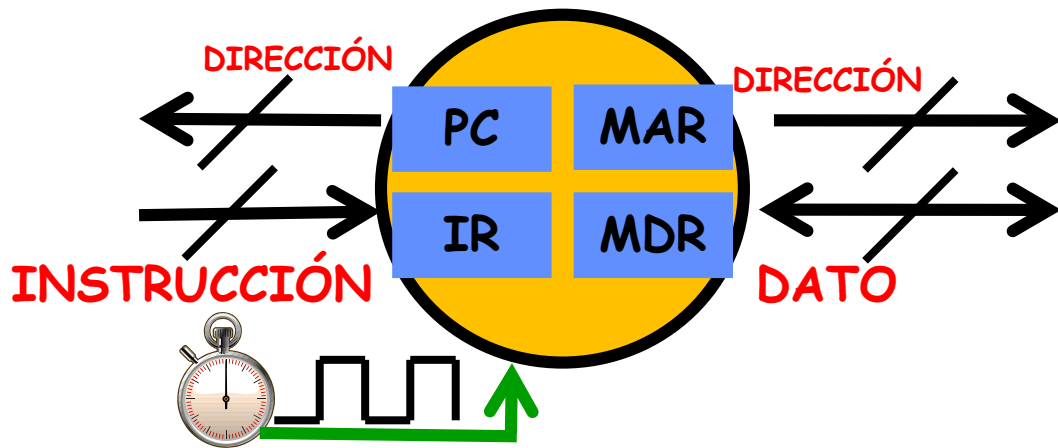




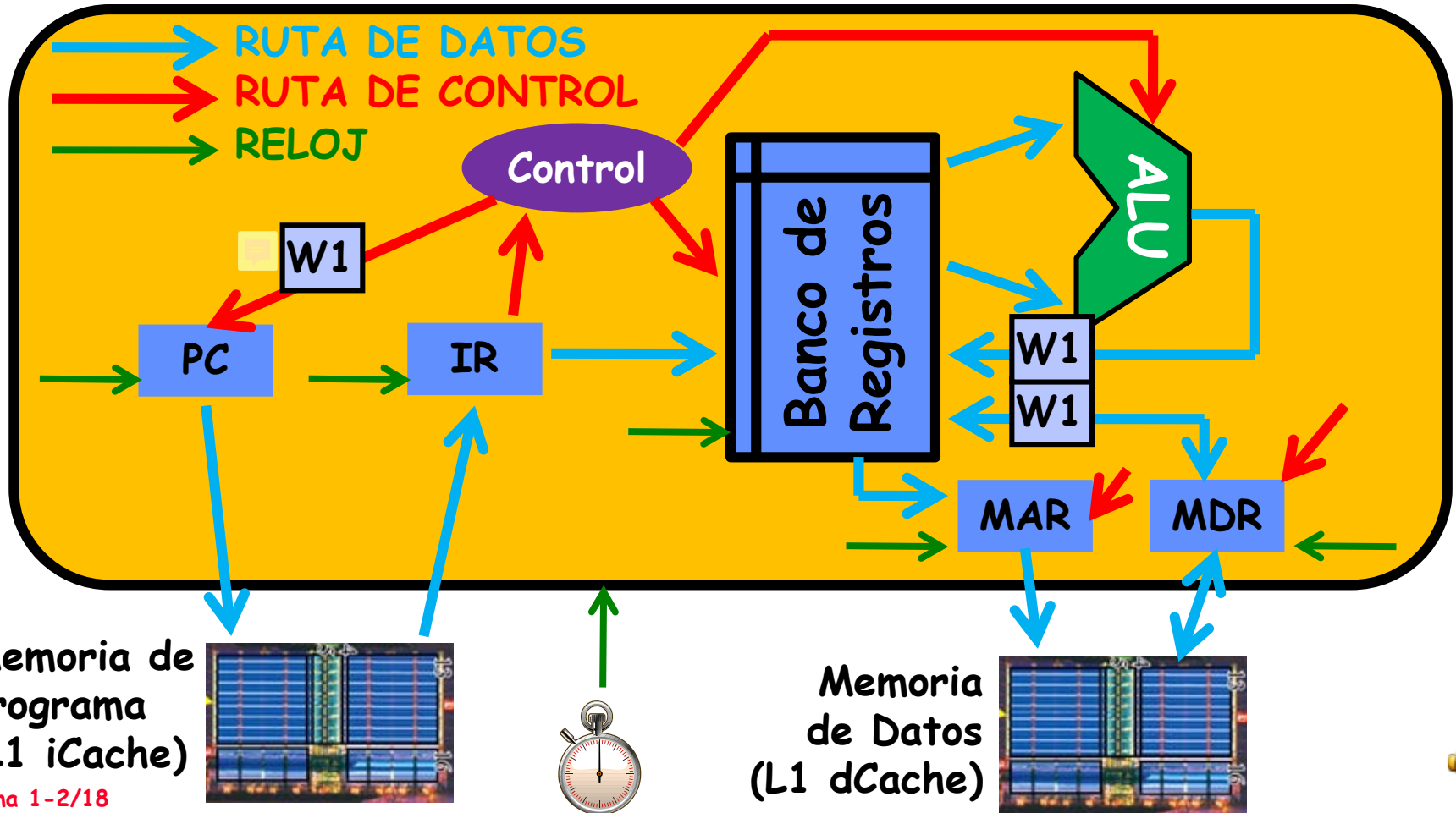
# Estructura interna de un procesador multiciclo: ETAPA E







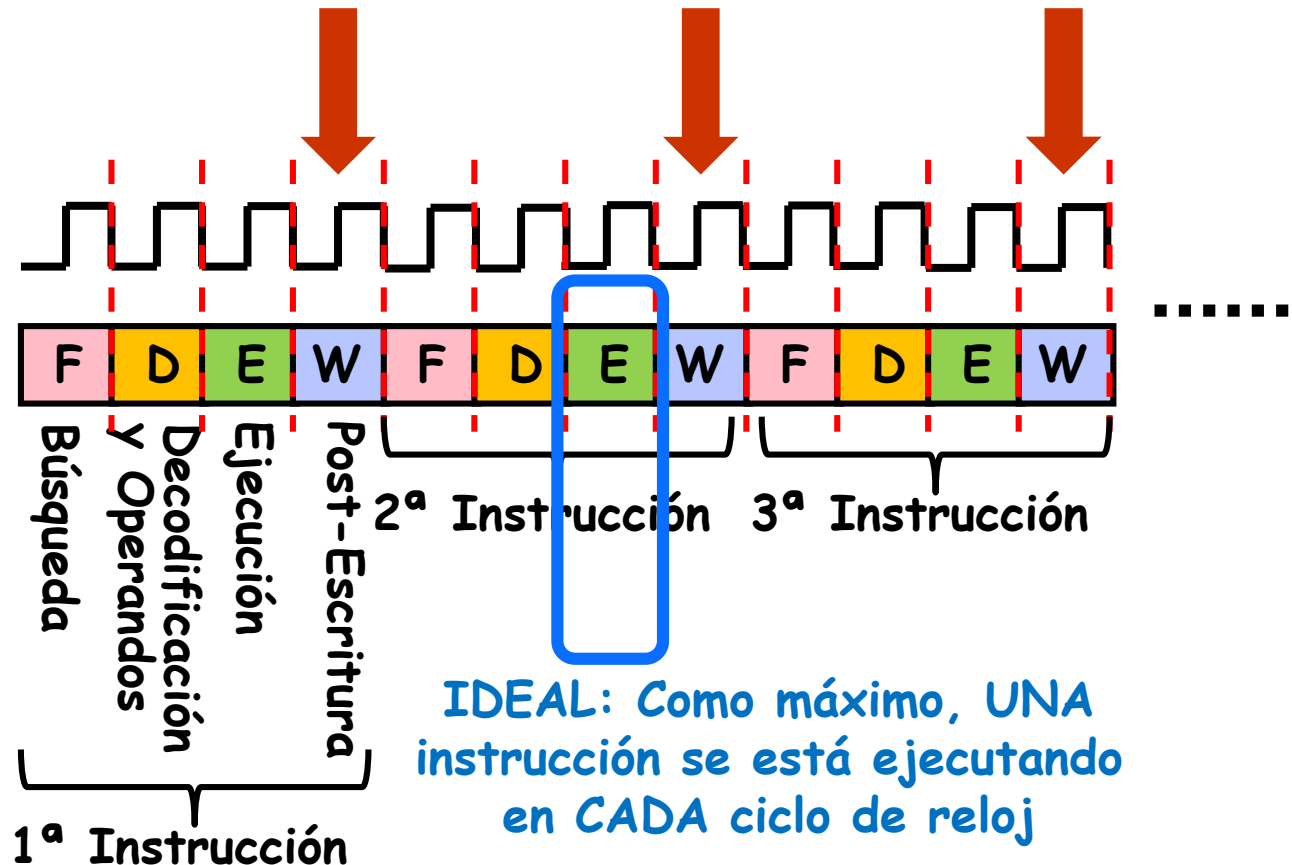
# Estructura interna de un procesador multiciclo: ETAPA W



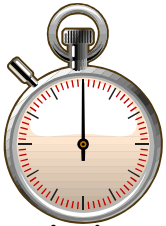
# Ejecución SECUENCIAL de instrucciones



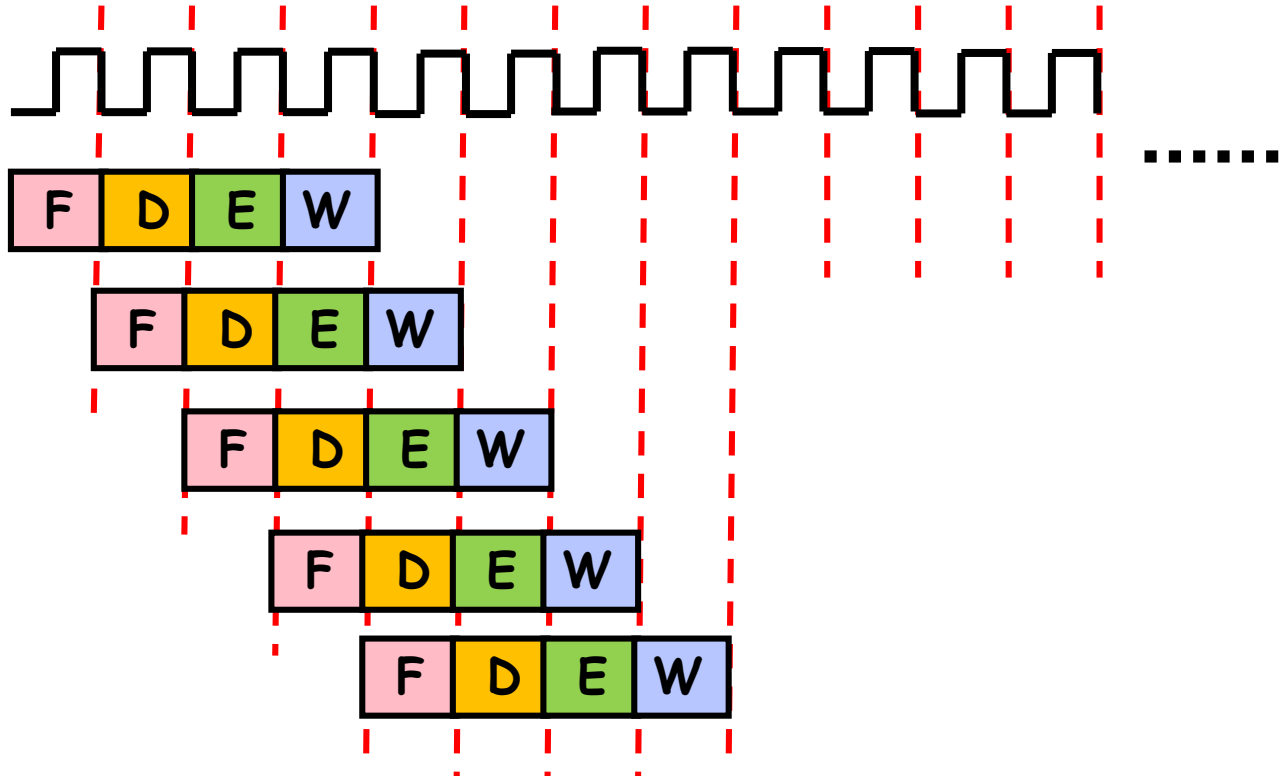
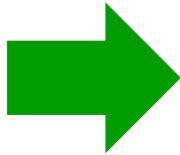
IDEAL: Cada CUATRO ciclos de reloj se termina UNA instrucción y se genera un resultado

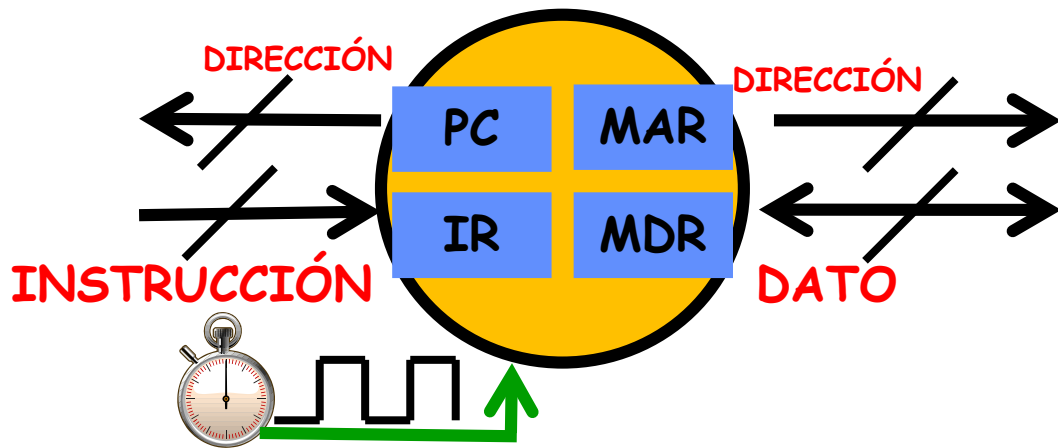


# Ejecución SEGMENTADA de instrucciones

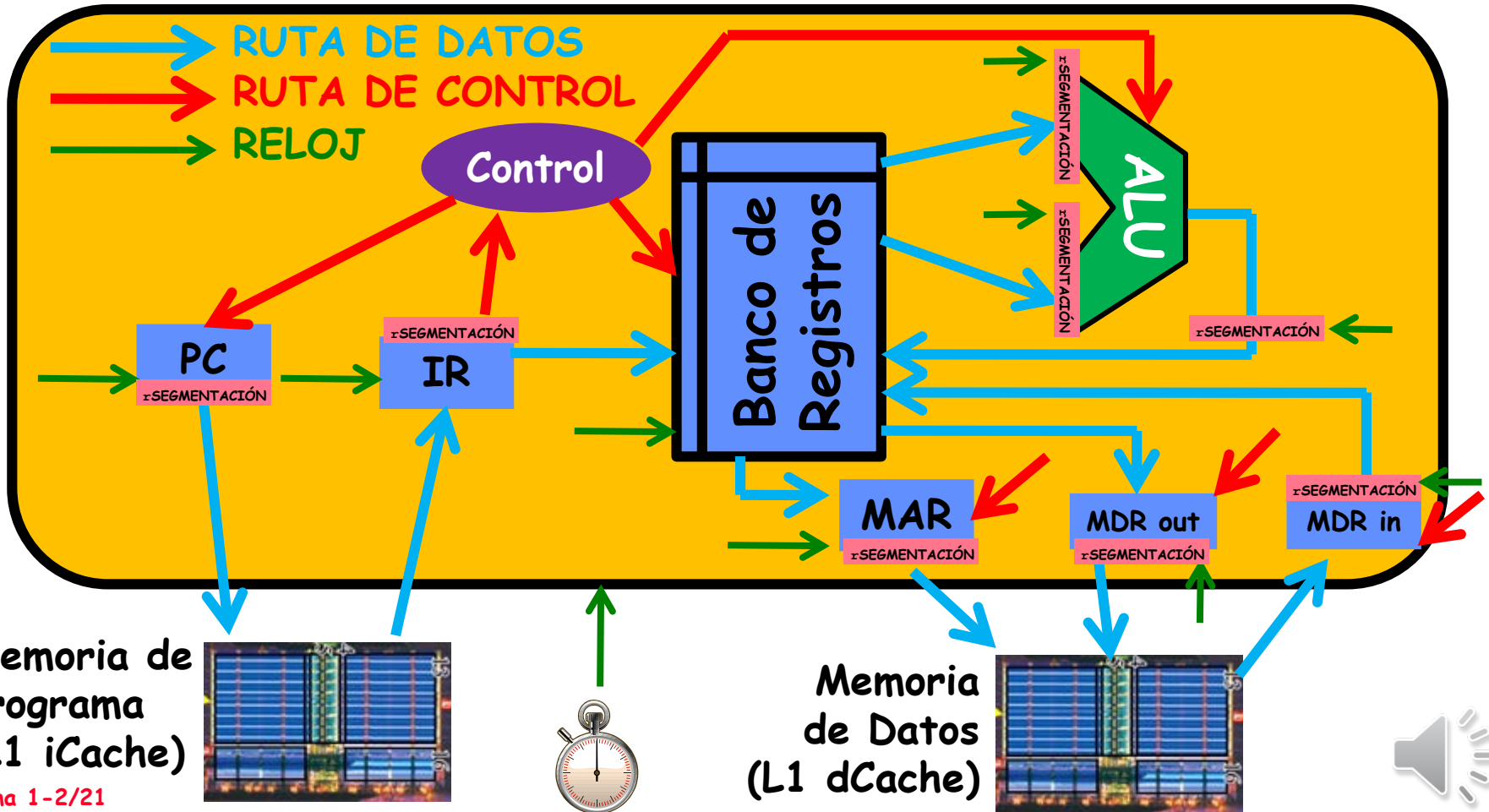


Señal de reloj  
del procesador





# Registros de segmentación



Memoria de Programa  
(L1 iCache)

Memoria de Datos  
(L1 dCache)

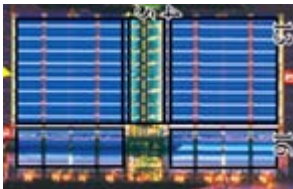




→ RUTA DE DATOS  
- - -> RUTA DE CONTROL  
- . - .> RELOJ



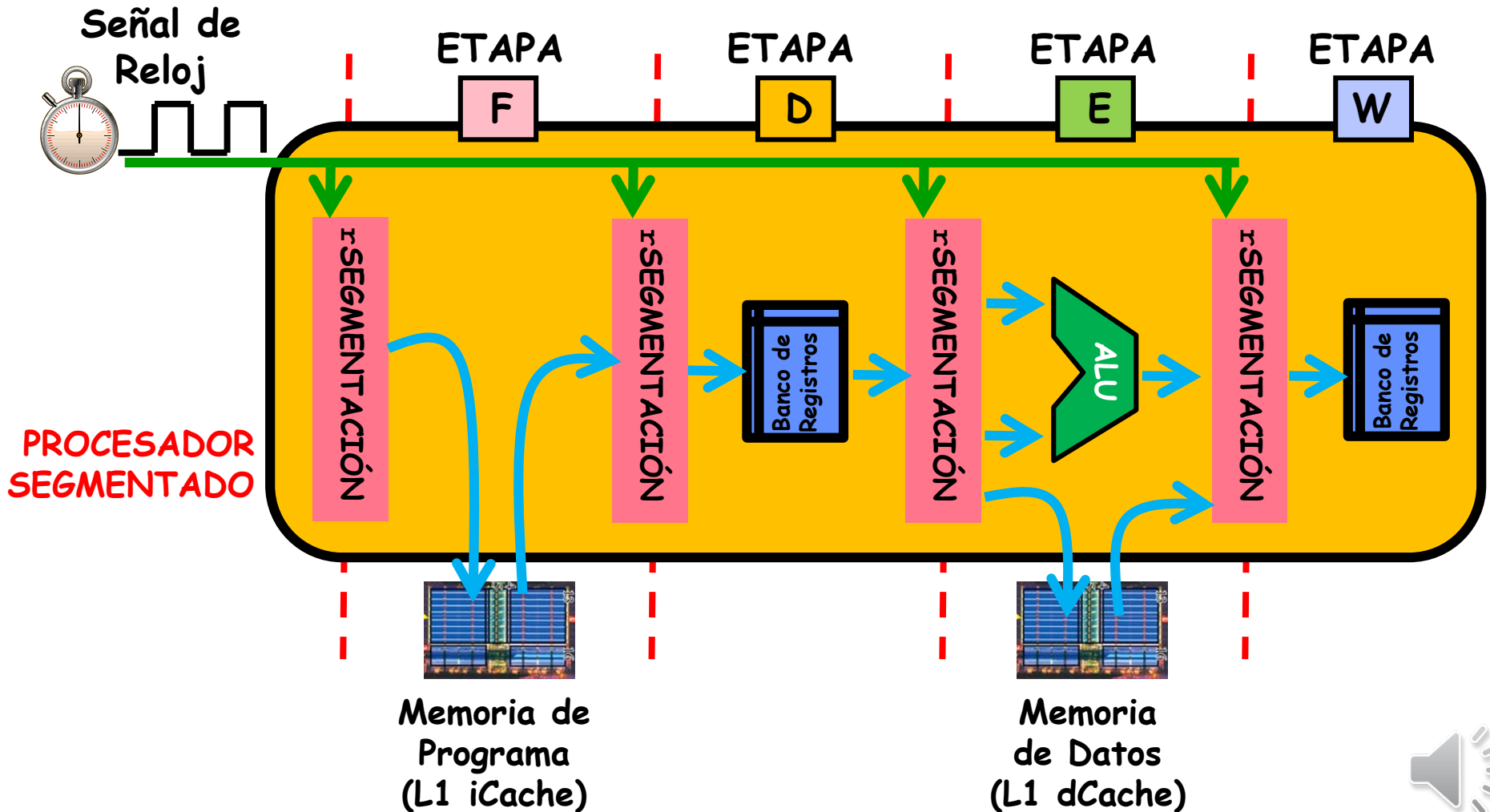
Memoria de  
Programa  
(L1 iCache)

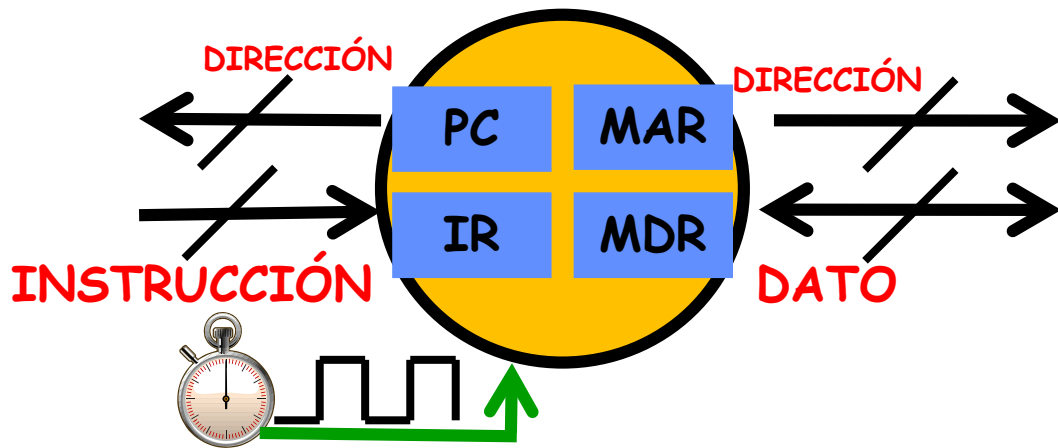


Memoria  
de Datos  
(L1 dCache)



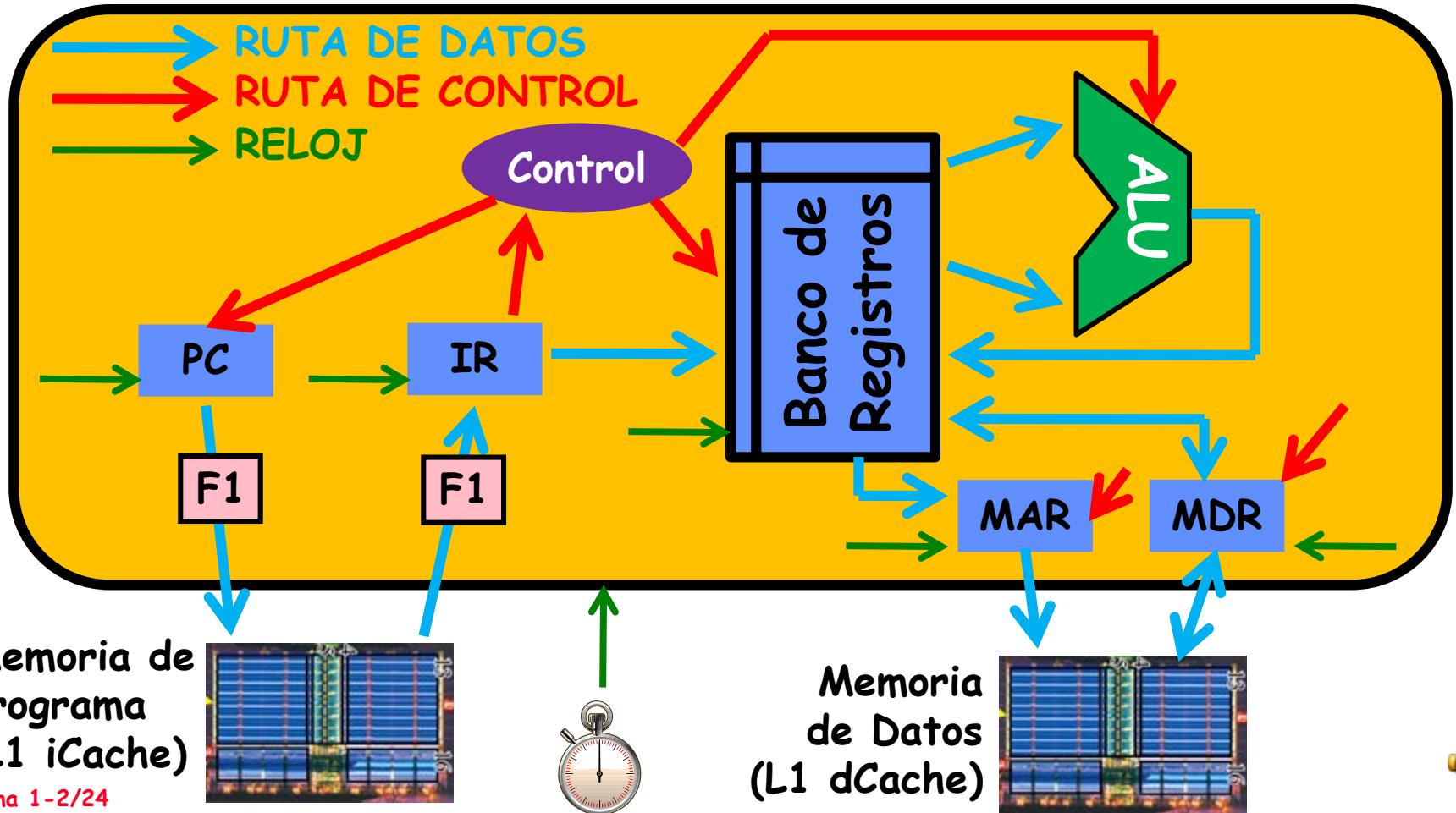
# Estructura interna simplificada de un procesador segmentado

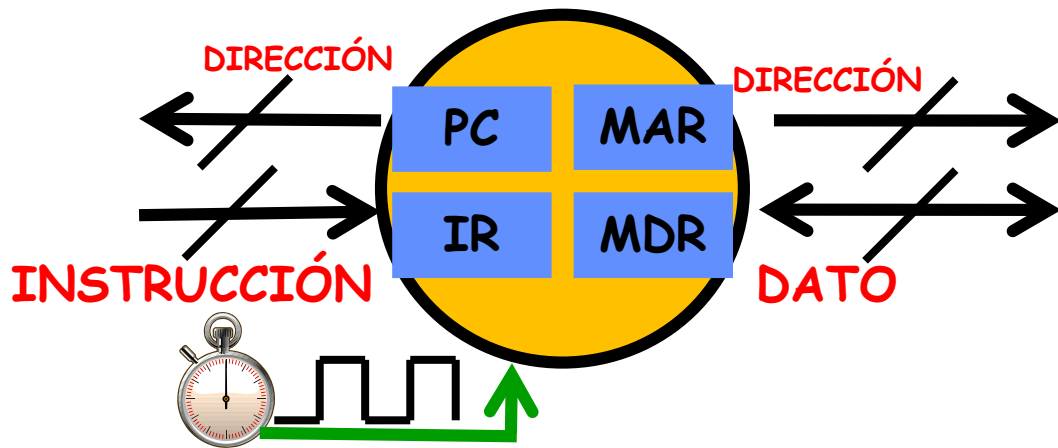




# Estructura interna de un procesador SEGMENTADO:

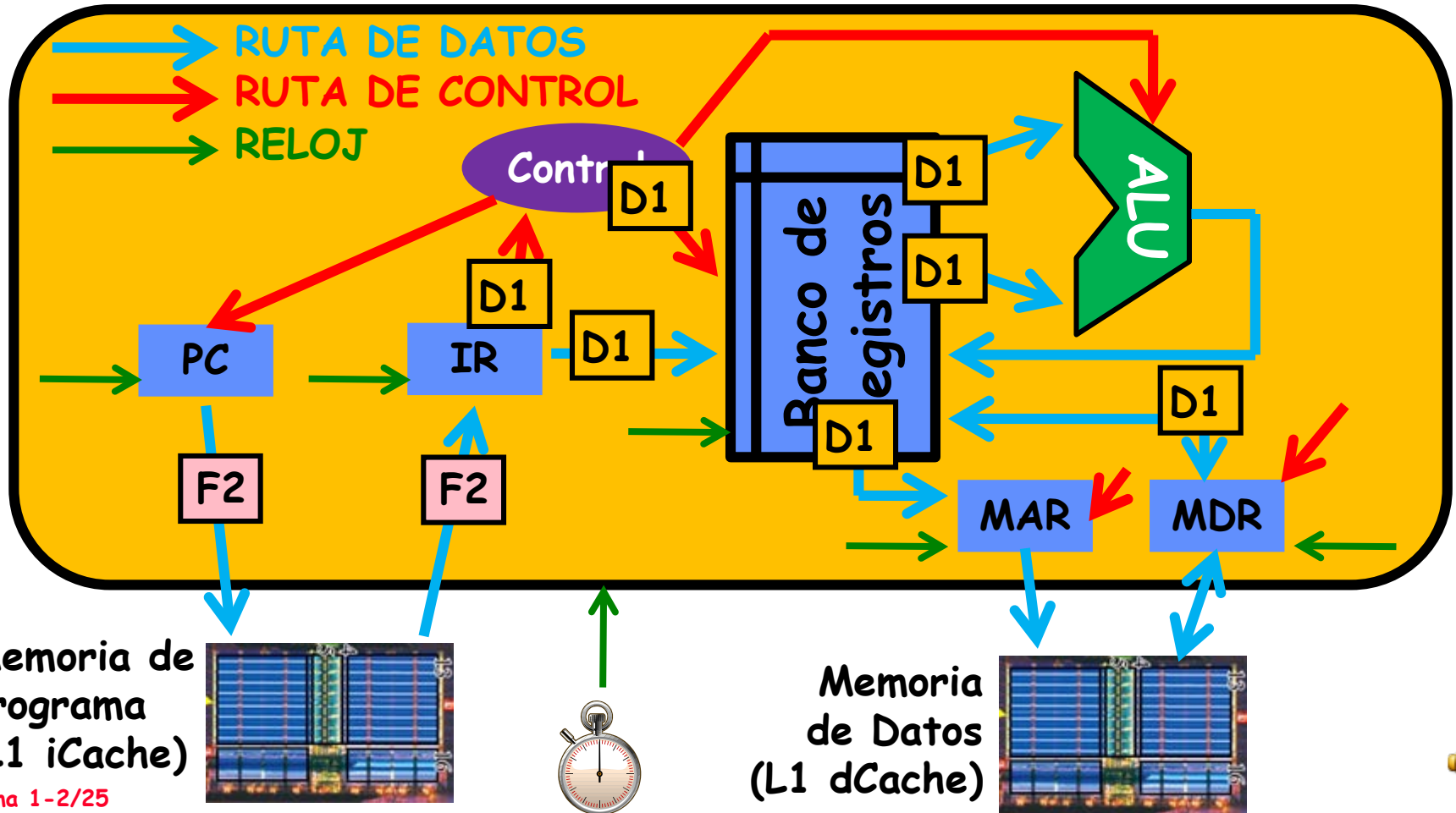
## ETAPA F1



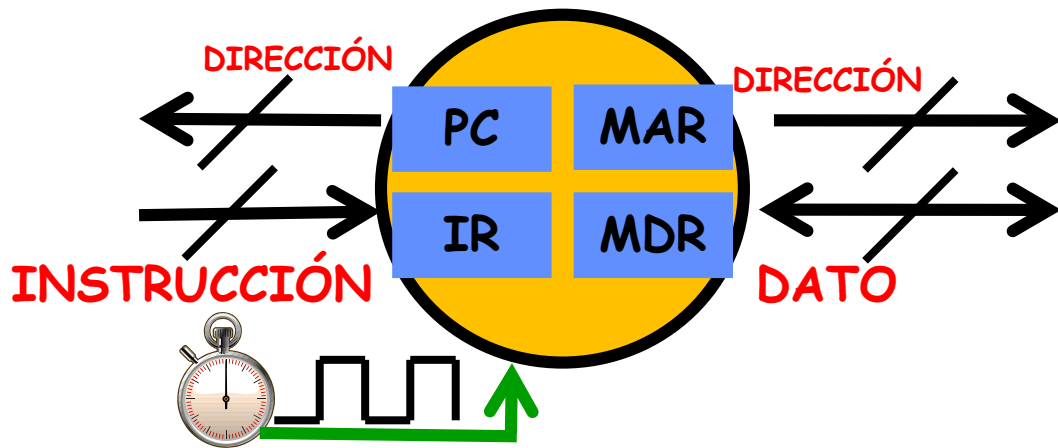


# Estructura interna de un procesador SEGMENTADO:

## ETAPAS F2+D1



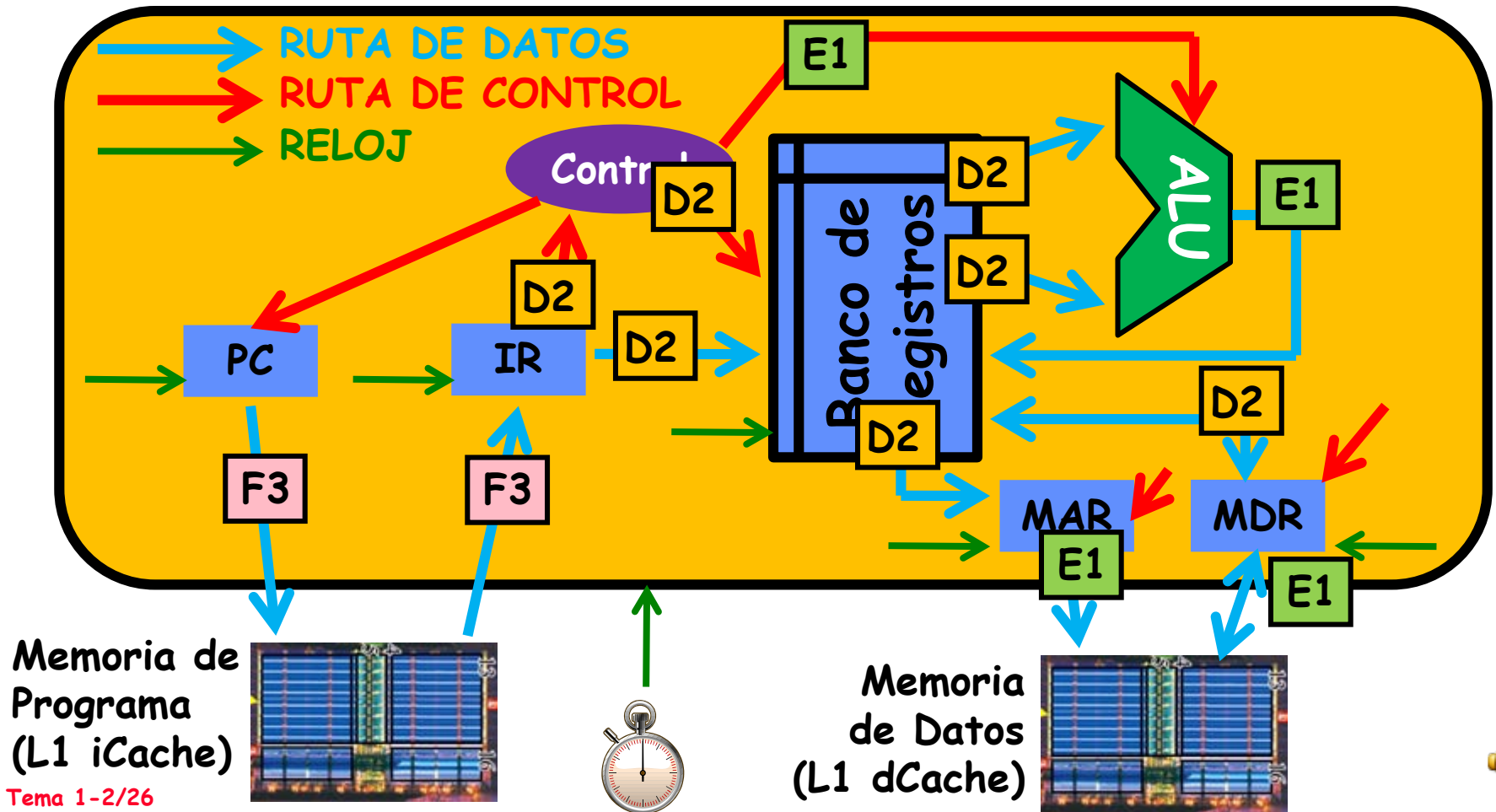


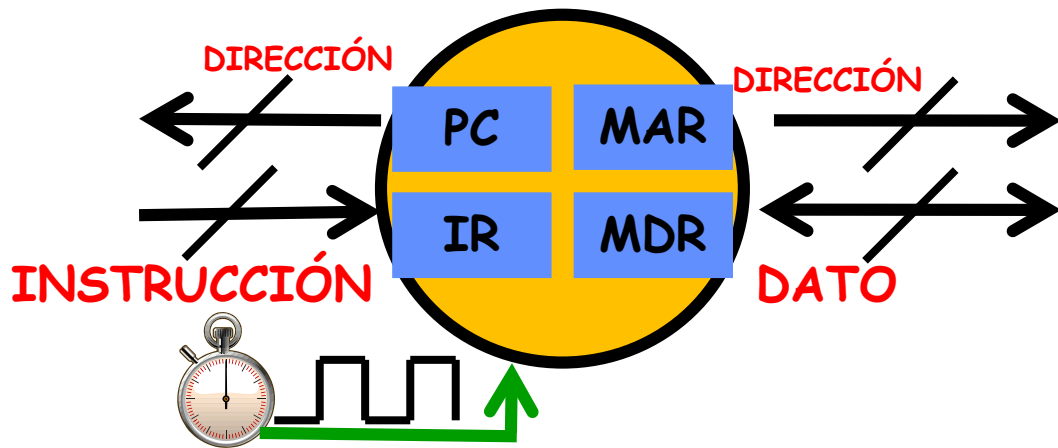


# Estructura interna de un procesador SEGMENTADO:

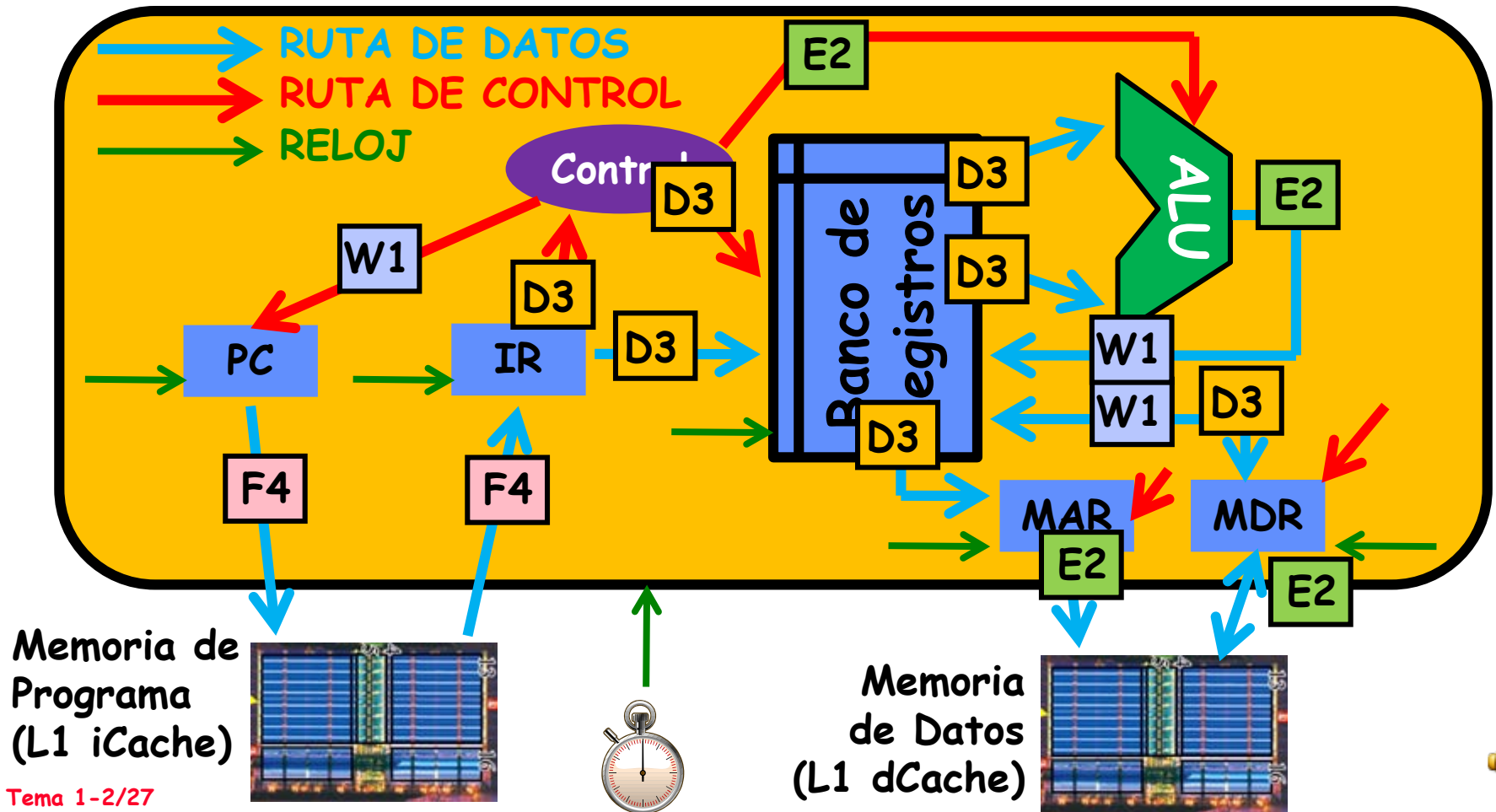
## ETAPAS

### F3+D2+E1





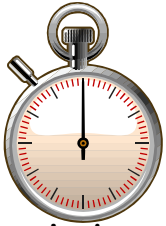
Estructura interna  
de un procesador  
**SEGMENTADO:**  
**ETAPAS**  
**F4+D3+E2+W1**



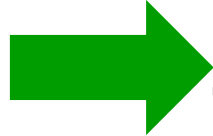
# Ejecución SEGMENTADA de instrucciones



IDEAL: En cada ciclo de reloj se termina UNA instrucción y se genera UN resultado



Señal de reloj del procesador



1ª Instrucción



2ª Instrucción



3ª Instrucción



4ª Instrucción



5ª Instrucción





IDEAL: Hasta CUATRO instrucciones se pueden estar ejecutando en PARALELO en cada ciclo de reloj





# NIOS II

**Nios® II**

	 <i>Nios II/f</i> “Fast”	<i>Nios II/s</i> “Standard”	 <i>Nios II/e</i> “Economy”
Pipeline	6-stage	5-stage	none
Max Frequency <sub>1</sub>	200 MHz	180 MHz	210 MHz
Max D-MIPS <sub>1</sub>	225	130	30
Size (4-input LUTs)	1800	1200	600
Branch Prediction	Dynamic	Static	no
I-Cache	Up to 64K	Up to 64K	no
D-Cache	Up to 64K	no	no

1. Characteristics in Stratix II 90nm FPGA





# Arquitectura de Computadores



## Tema 1-2. Procesadores Segmentados, Parte 2



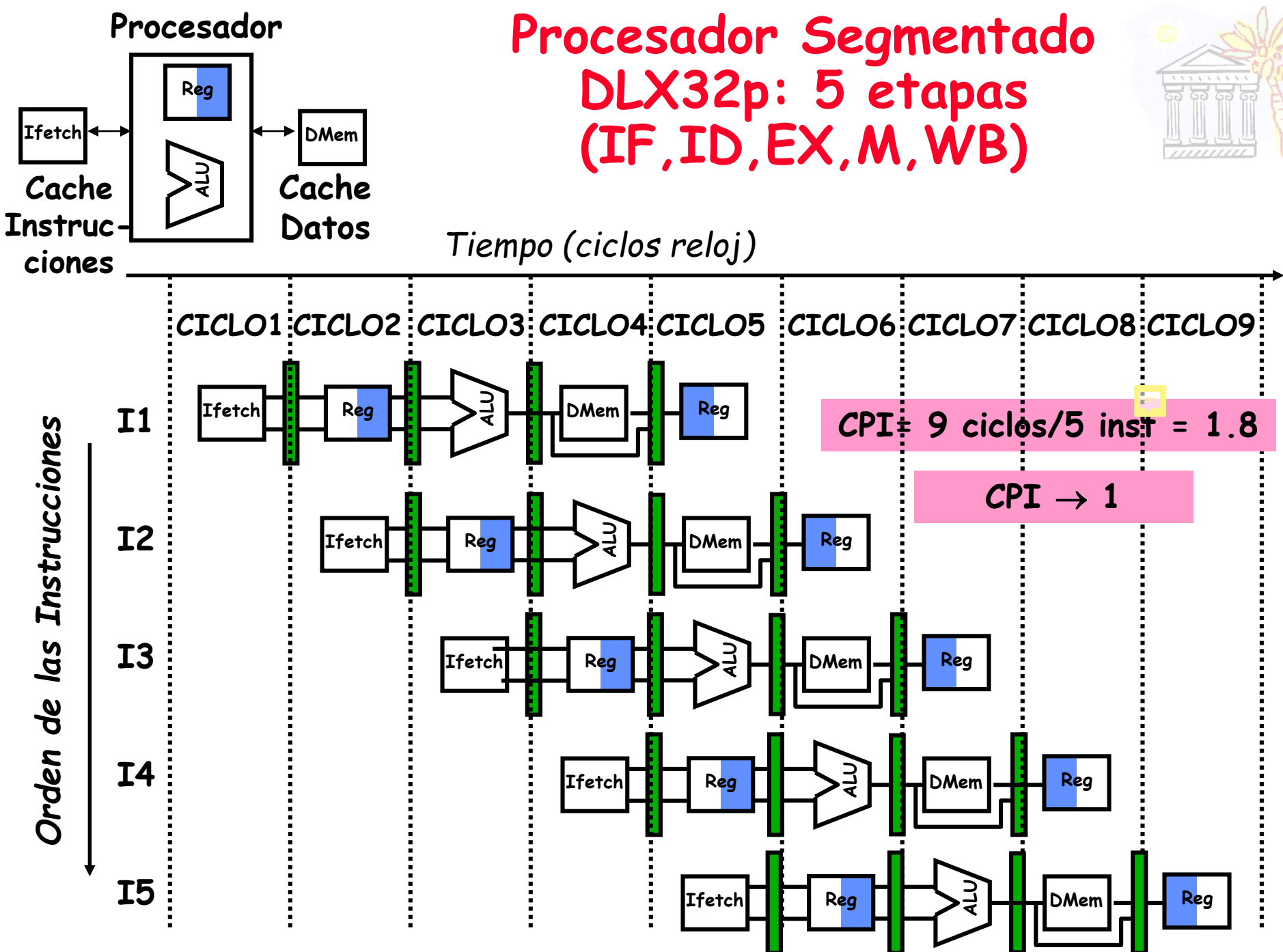
# Sumario

## • Parte 2

- Modelo segmentado del procesador DLX32p
  - » Cálculo de speed-up
  - » Dependencias estructurales
- Dependencias Estructurales
  - » Números de puertos de la cache L1
- Dependencias de datos
  - » Dependencias de datos verdaderas (RAW)
  - » Antidependencias de datos (WAR)
  - » Dependencias de salida (WAW)
  - » Anticipación en DLX32p
- Dependencias de control en DLX32p
- Procesador segmentado con unidades funcionales multicilo DLX32mf
  - » Latencia
  - » Intervalo de repetición
  - » Dependencias estructurales
  - » Dependencia datos
- Evaluación de prestaciones de MIPS R4000 con SPEC CPU 89

# Procesador Segmentado

## DLX32p: 5 etapas (IF, ID, EX, M, WB)



# Microarquitectura de DLX32p (arquitect=DLX)



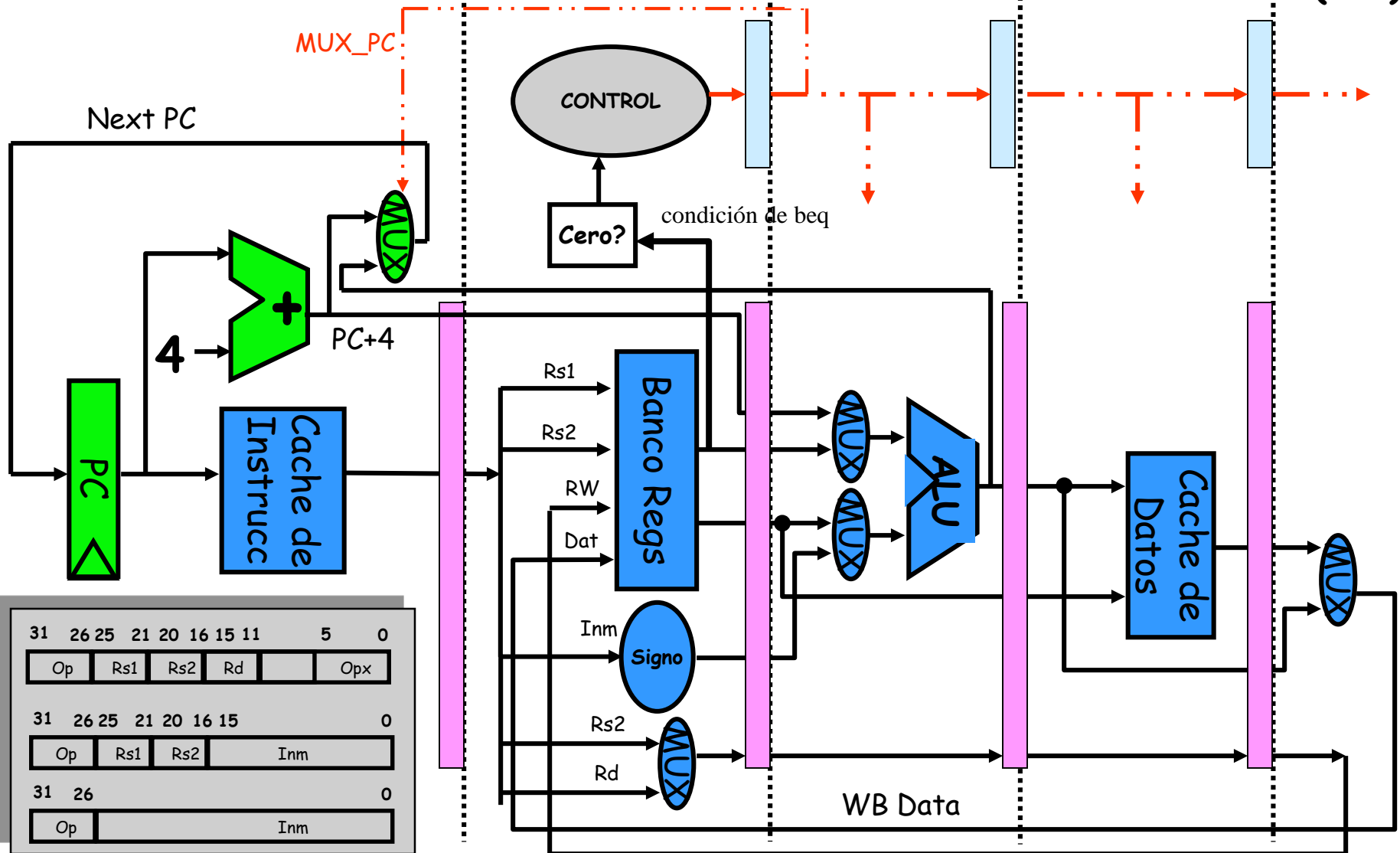
Búsqueda de la Instrucción (IF)

Decodificación y Búsqueda datos (ID)

Ejecución y Calc. Direc. (EX)

Acceso a Memoria (M)

Post-Escritura (WB)





# Ejercicio de Cálculo de Prestaciones



- Programa "benchmark":
  - loads: {5 ciclos , 40% instrucciones}
  - resto instrucciones: {4 ciclos , 60%}
- Procesador **DLX32mc** (multiciclo),  $T(\text{ciclo}) = 10\text{ns}$
- Procesador **DLX32p** (segmentado),  $T(\text{ciclo}) = 10\text{ns} + 1\text{ns}$
- ¿**Speed-Up** proporcionado por Segmentación?

$$\begin{aligned}\text{DLX32mc, } t_{\text{CPU}} &= T \times \text{CPI} \times N \\ &= 10 \text{ ns} \times (0.6 \times 4 + 0.4 \times 5) \times N \\ &= 44 \text{ ns}\end{aligned}$$

$$\text{DLX32p, } t_{\text{CPU}} = (10 + 1) \times 1 \times N = 11 \text{ ns} \times N$$

$$\text{SpeedUp} = 44 / 11 = 4X$$

- Nota. Se ha ignorado: llenado, vaciado, dependencias.

# Gran Problema:



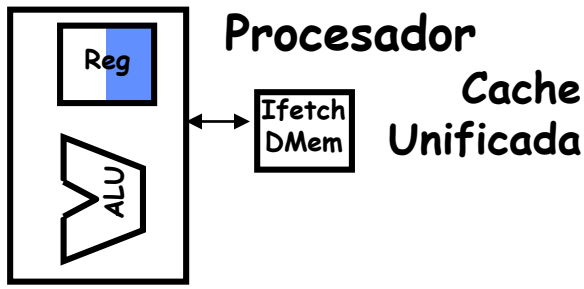
## "Riesgos / Penalizaciones" debidos a "Dependencias"

- **Problema:** no se consigue con la Segmentación el Speed-Up máximo ( $\propto$  número de segmentos)
  - ¿Por qué?
- **Penalización:** evento que se produce en el hardware y no permite que las instrucciones avancen en el cauce de ejecución al ritmo más eficiente
- **Dependencia:** colisión entre dos o más instrucciones que puede ocasionar una penalización
- **Tipos de Dependencias**
  - Estructurales
  - Datos
  - Control

# Dependencias Estructurales

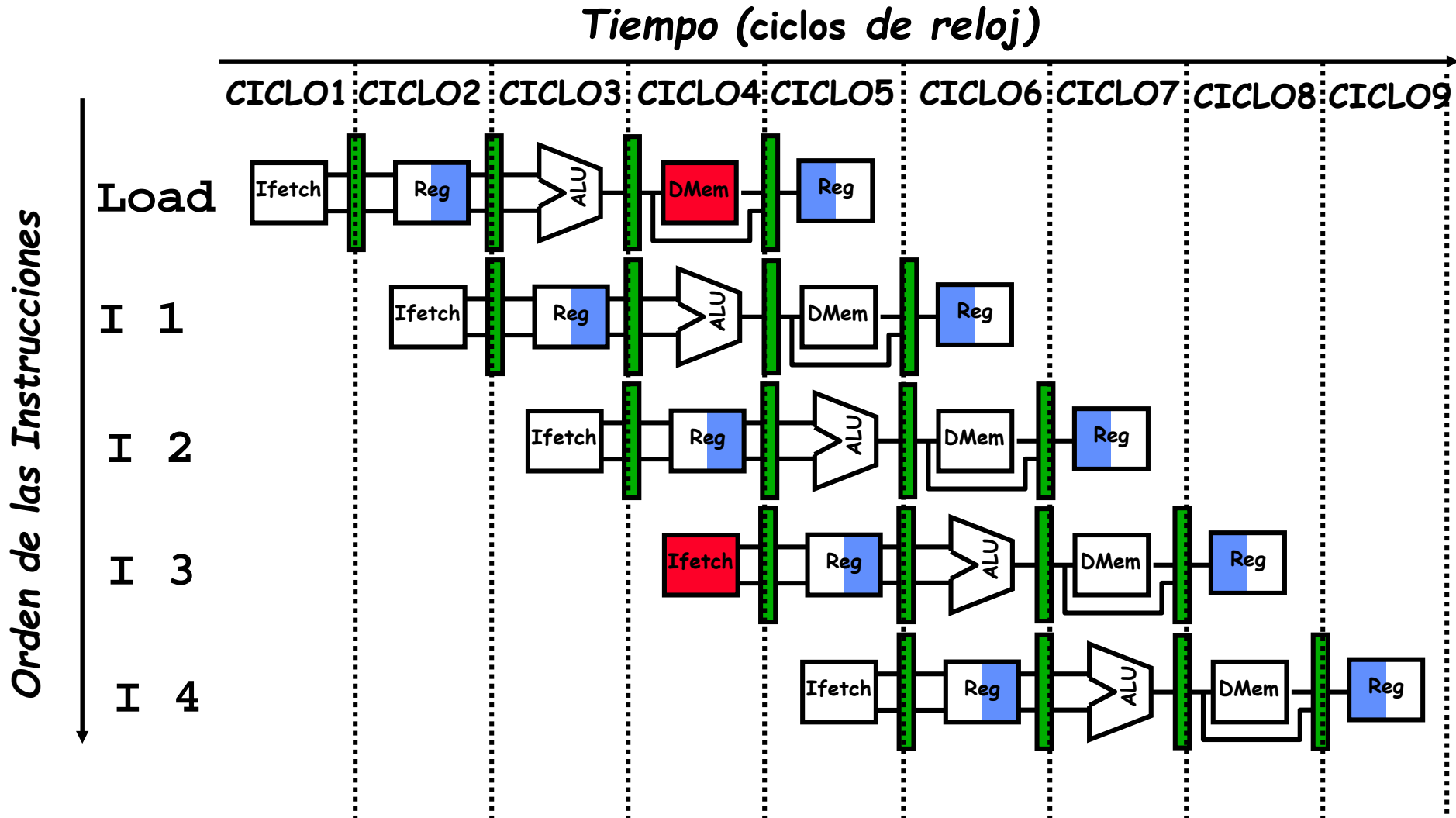


- ¿Cuándo?: varias instrucciones intentan utilizar a la vez un recurso hardware compartido
- ¿Técnicas de Optimización?
  - Duplicar hardware
  - Segmentar hardware
  - Reordenar instrucciones (trataremos próximamente)
- ¿Y si no se puede?... ¡ bloqueo ! = penalización
- A continuación ... Ejemplos

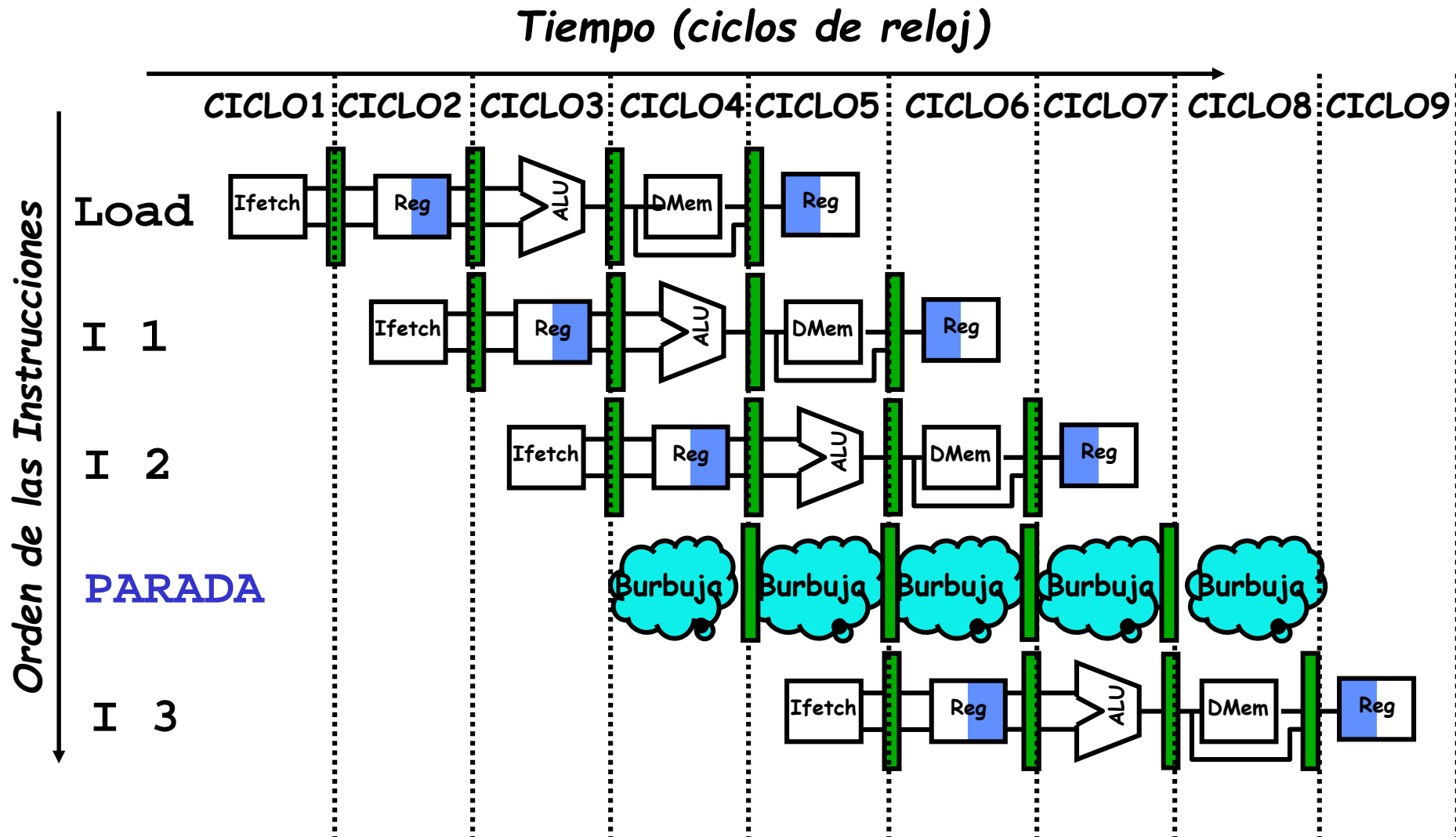


# Dependencia

## Estructural: un solo puerto de memoria



# Dependencia Estructural: un solo puerto de memoria





# Ecuación para el cálculo de la Aceleración "MC respecto Segmentado"

(suponer que todas las instrucciones toman el mismo número de ciclos en la implementación multiciclo )

$$CPI_{\text{segmentado}} = CPI_{\text{Ideal}} + \text{Num Prom CLK parada por Inst}$$

$$\text{Speedup} = \frac{CPI_{\text{Ideal}} \times \text{Num Segmentos}}{CPI_{\text{Ideal}} + CPI_{\text{de parada procesador segmentado}}} \times \frac{\text{Tiempo Ciclo}_{\text{multi-ciclo}}}{\text{Tiempo Ciclo}_{\text{segmentado}}}$$

**Caso Simple, CPI Ideal= 1:**

$$\text{Speedup} = \frac{\text{Num Segmentos}}{1 + CPI_{\text{de parada procesador segmentado}}} \times \frac{\text{Tiempo Ciclo}_{\text{multi-ciclo}}}{\text{Tiempo Ciclo}_{\text{segmentado}}}$$

# Ejercicio: speed-up de ... **Doble-Puerto vs. Puerto-Unico**



- Máquina A: Segmentada con Memoria de doble puerto ("Arquitectura Harvard")
- Máquina B: Segmentada con Memoria de puerto único,  $T_B = T_A / 1.05$  (B tiene mayor frecuencia que A)
- CPI Ideal = 1 para ambos
- Loads = 40% instrucciones totales

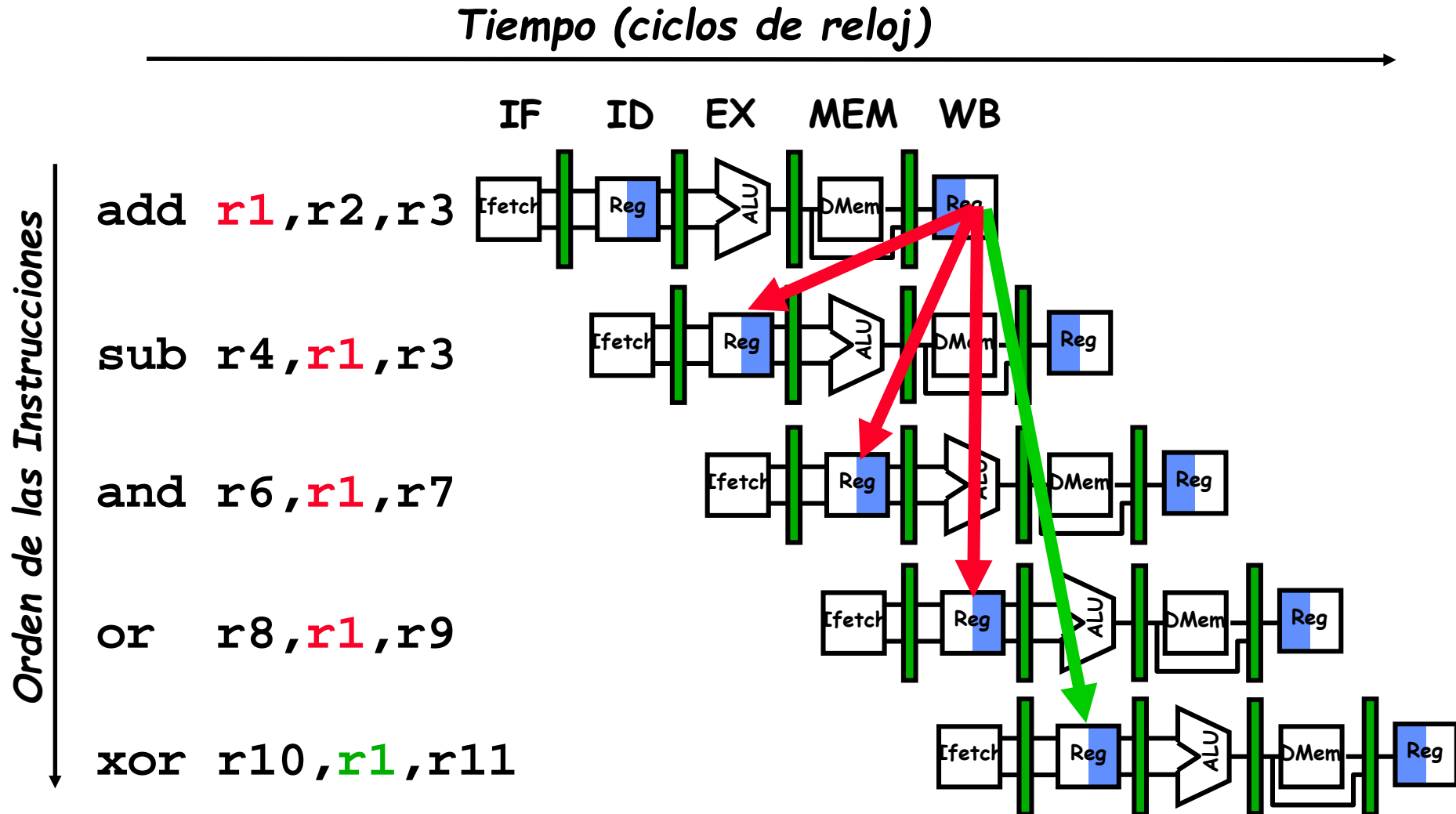
$$\begin{aligned}\text{SpeedUp}_A &= [N\_Segmentos / (1 + 0)] \times (T_{MC} / T_{SEG\_A}) \\ &= N\_Segmentos\end{aligned}$$

$$\begin{aligned}\text{SpeedUp}_B &= [N\_Segmentos / (1 + 0.4 \times 1)] \times [T_{MC} / (T_{SEG\_A} / 1.05)] \\ &= (N\_Segmentos / 1.4) \times 1.05 \\ &= 0.75 \times N\_Segmentos\end{aligned}$$

$$\text{SpeedUp}_A / \text{SpeedUp}_B = N\_Segmentos / (0.75 \times N\_Segmentos) = 1.33X$$

- Máquina A es 1.33X más rápida, aunque sea de frecuencia inferior

# Dependencias de Datos (en "r1")





# 3 Tipos de Dependencias de Datos (1)



- **RAW: Lectura después de escritura**

    I: add **r1**, r2, r3 (escribe después)  
    J: sub r4, **r1**, r3 (lee antes)

“**Dependencia Verdadera**” debido a necesidad de comunicación de datos

- Aparece en Procesadores Segmentados con ejecución en orden o fuera de orden
- Optimizaciones para mejorar la eficiencia del software:
  - Banco de registro que lee y escribe en 1 ciclo
  - Reordenamiento
  - Anticipaciones

# 3 Tipos de Dependencias de Datos (2)

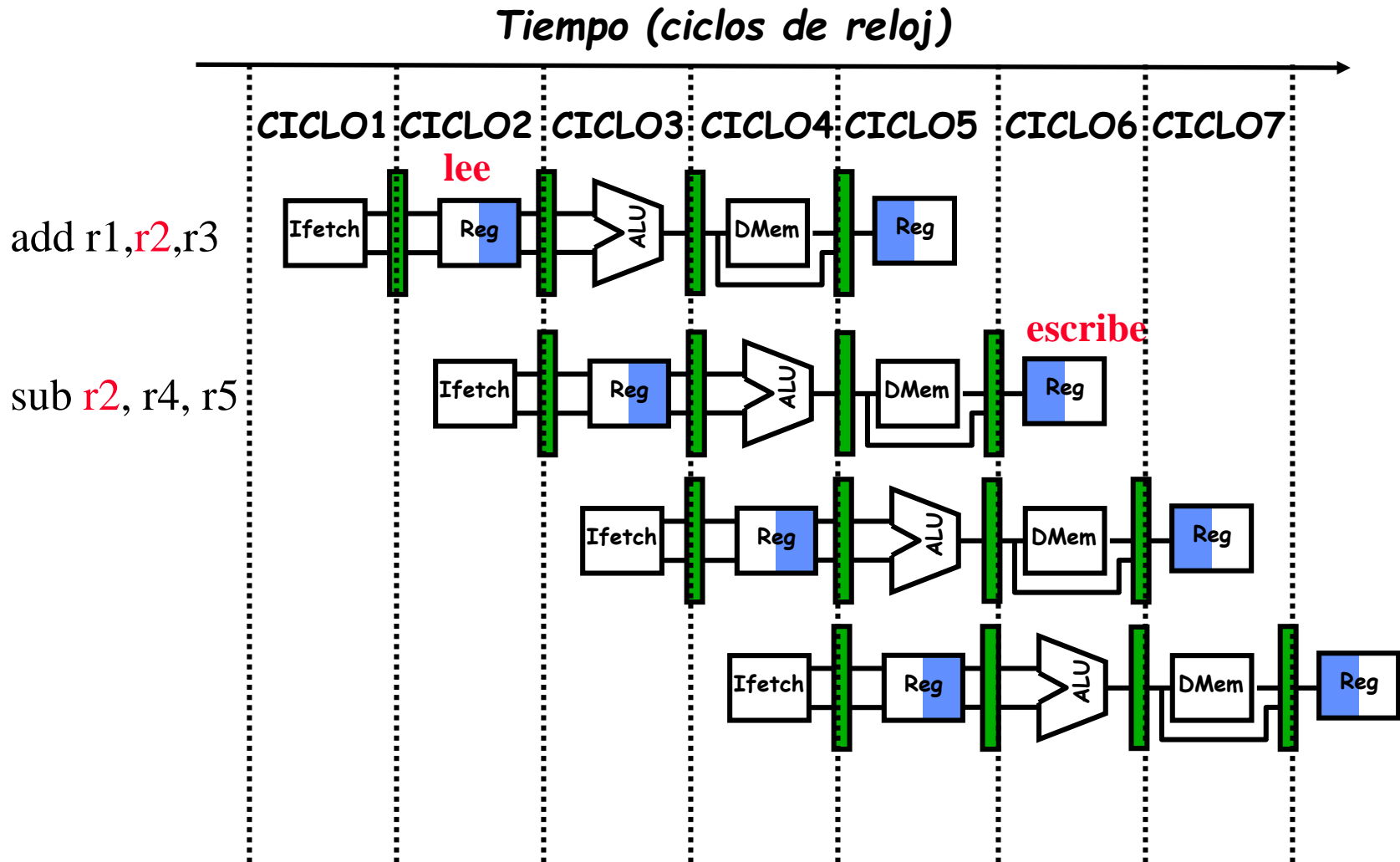


- **WAR: Escritura después de lectura**

↪ I: sub r4,**r1**,r3 (lee después)  
J: add **r1**,r2,r3 (escribe antes)  
K: mul r6,r1,r7

- “**Anti-Dependencia**” debido a la reutilización de “**r1**” (el compilador no tiene registros disponibles)
- ¿Puede ocurrir en el DLX32p?. Razona la respuesta ...

# Respuesta: No existen dependencias WAR en DLX32p !



# 3 Tipos de Dependencias de Datos (3)

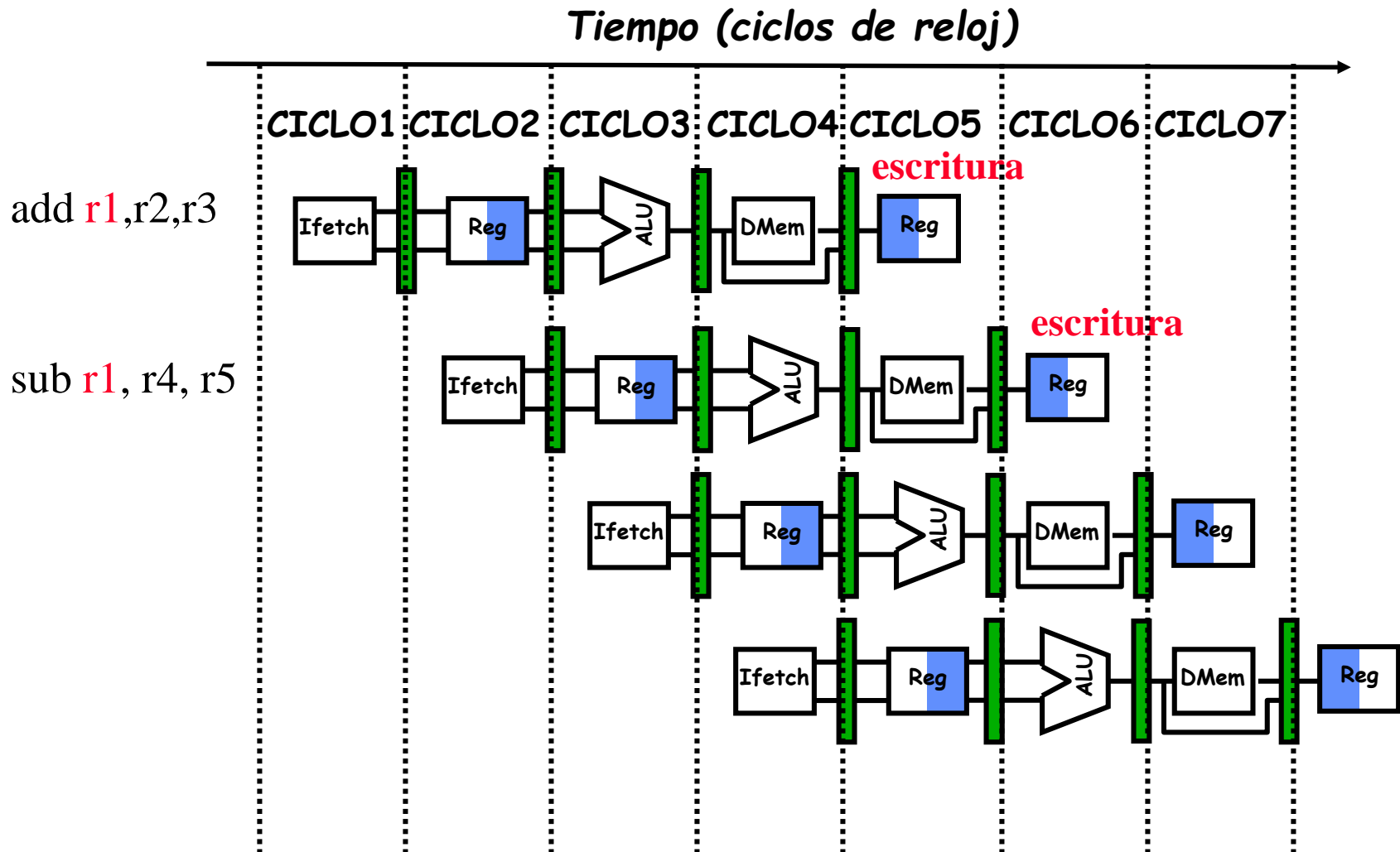


- **WAW: Escritura después de escritura**

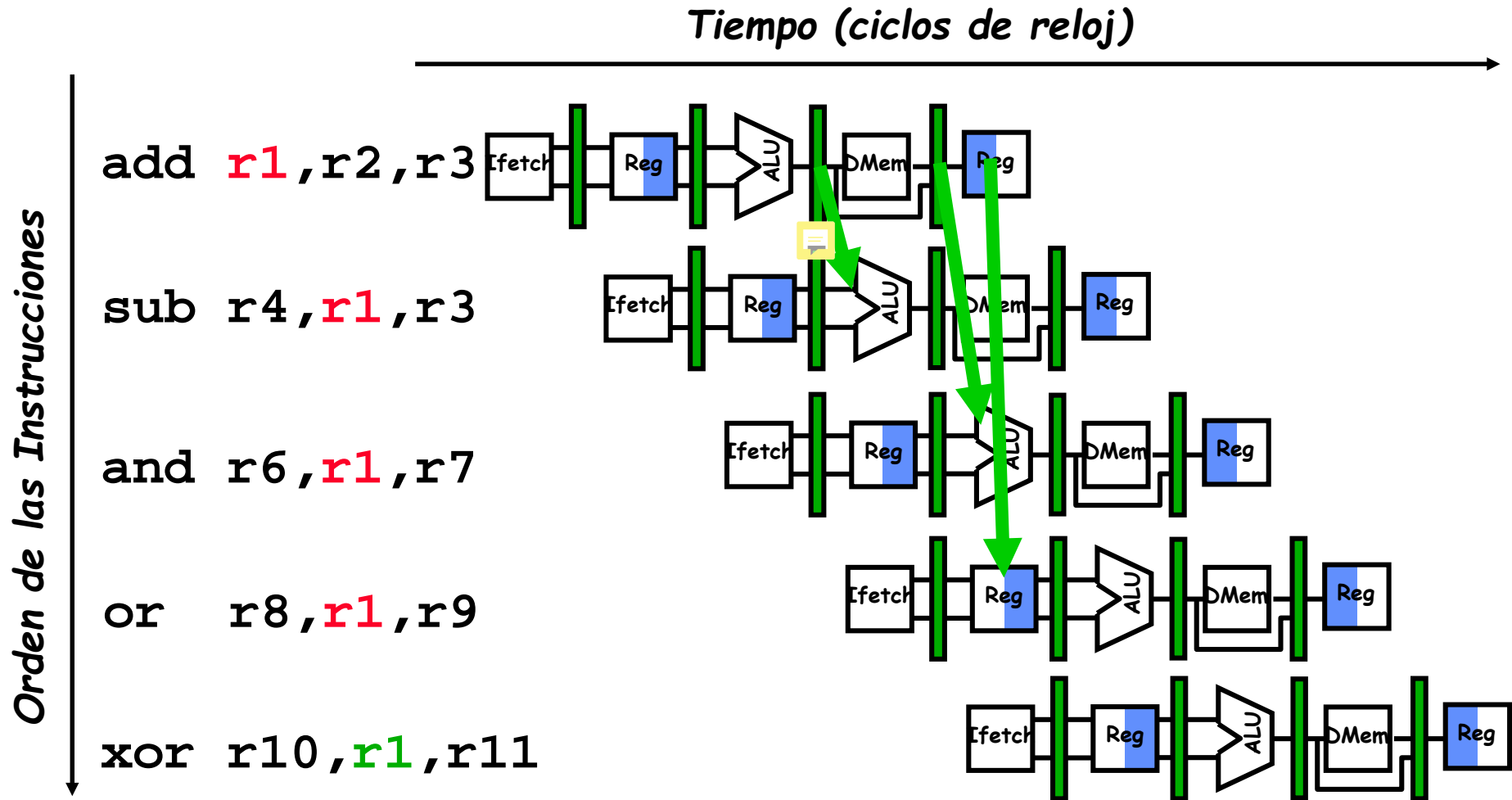
↪ I: sub **r1**,r4,r3 (escribe después)  
↪ J: add **r1**,r2,r3 (escribe antes)  
K: mul r6,r1,r7

- “**Dependencia de Salida**” debido a la reutilización de “**r1**” (el compilador no tiene registros disponibles)
- ¿Puede ocurrir en el DLX32p?. Razona la respuesta  
...

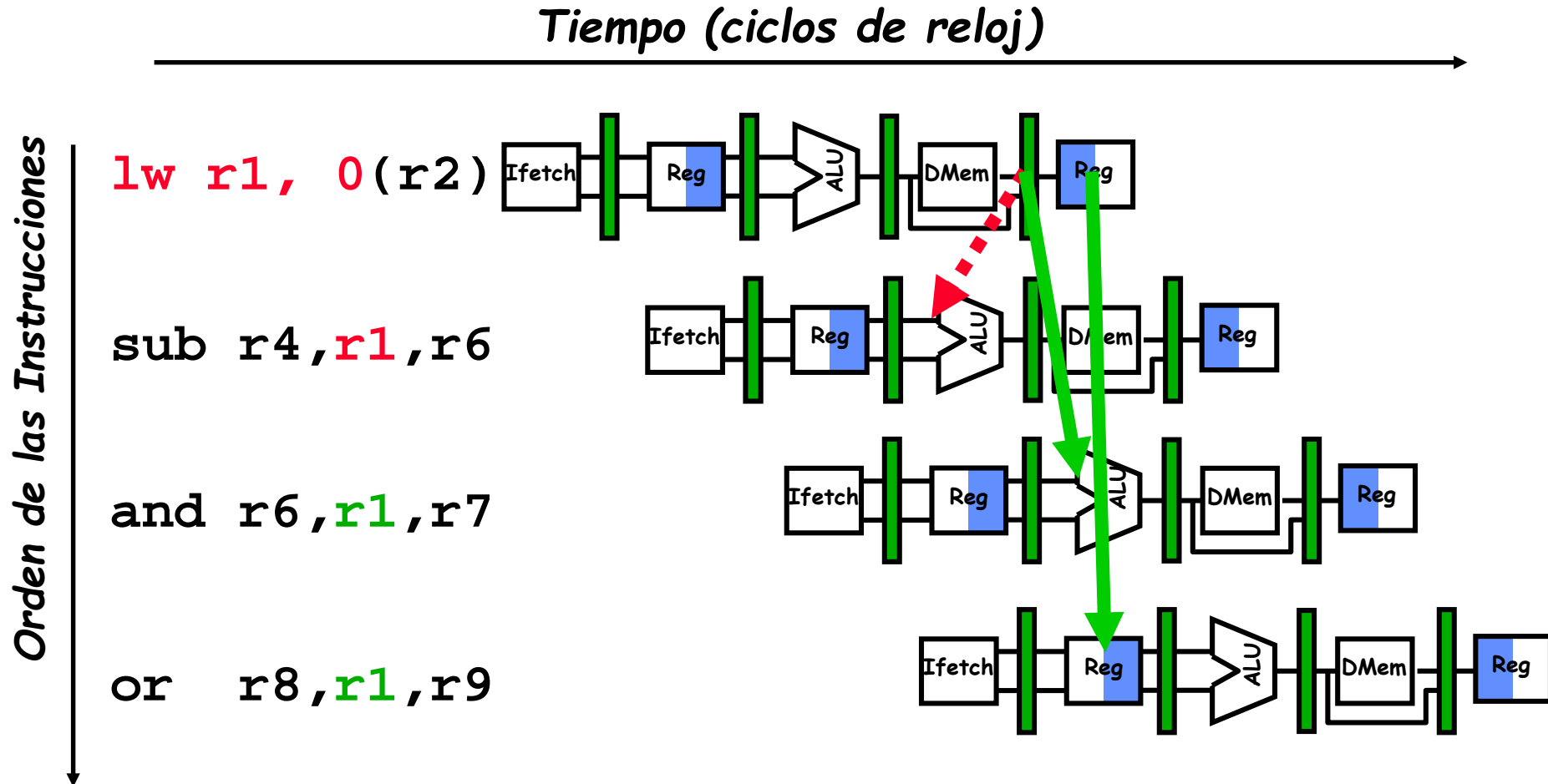
# Respuesta: No existen dependencias WAW en DLX32p



# Optimización: Anticipación para evitar Dependencias Verdaderas RAW

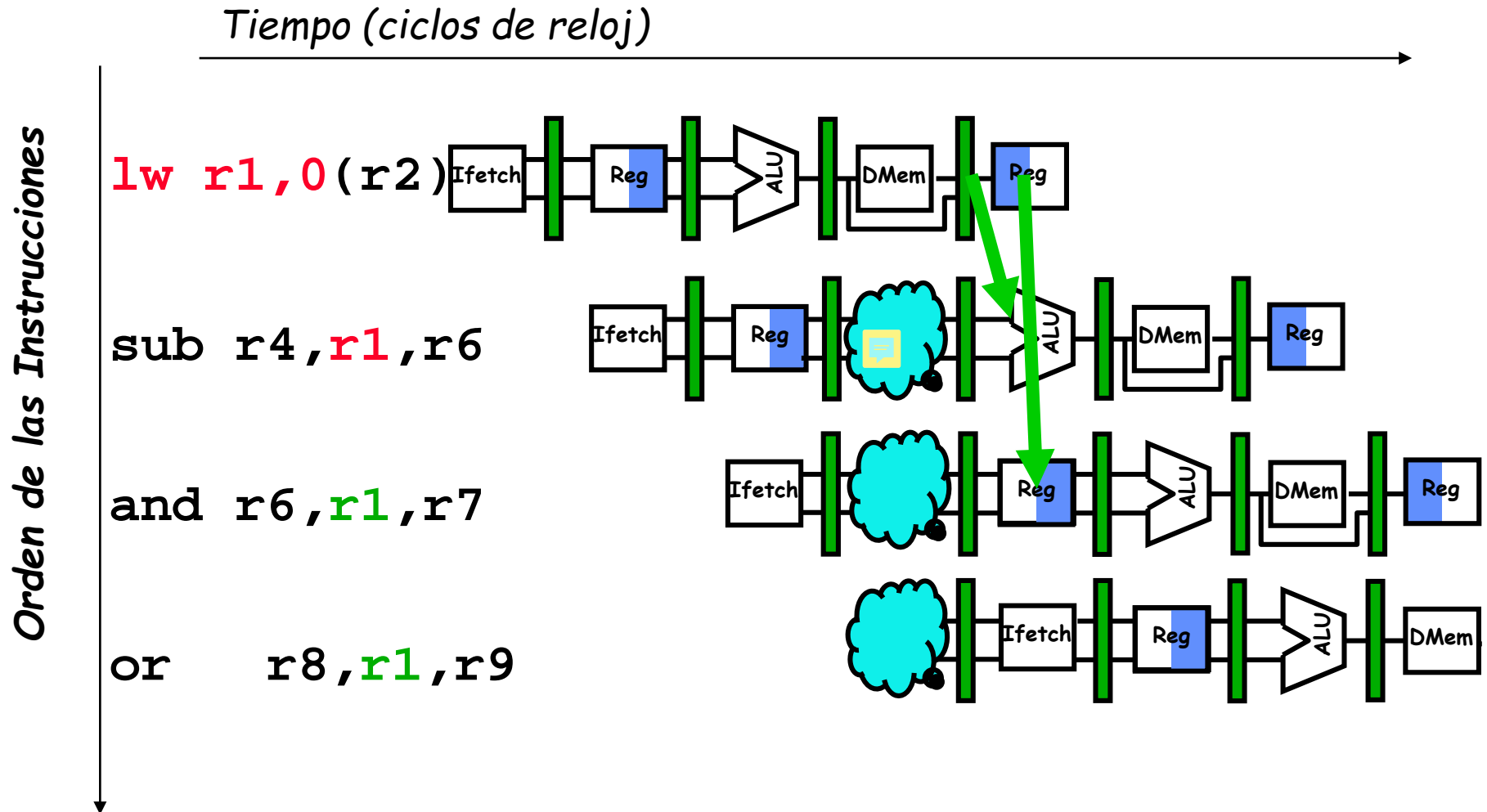


# Dependencias de Datos que no resuelve la Anticipación





# Solución: Burbujas





# Dependencias de Control



## Consecuencias: penalizaciones por cambio en el flujo de instrucciones

- Objetivo prestacional: conseguir el menor impacto debido al vaciado y llenado de la ruta de datos segmentada
- Inconvenientes de las instrucciones de salto:
  - Se tarda en decodificar instrucciones
  - Se tarda en computar el contenido del siguiente PC (NextPC)
  - En saltos condicionales, se tarda en computar la condición
- Problemas con saltos originados por Interrupciones y Excepciones
  - ¿Cuándo se debe interrumpir el flujo de instrucciones?
  - ¿Cuánta información debemos guardar para restaurar el estado del procesador después de resuelta la interrupción?

# Microarquitectura de DLX32p



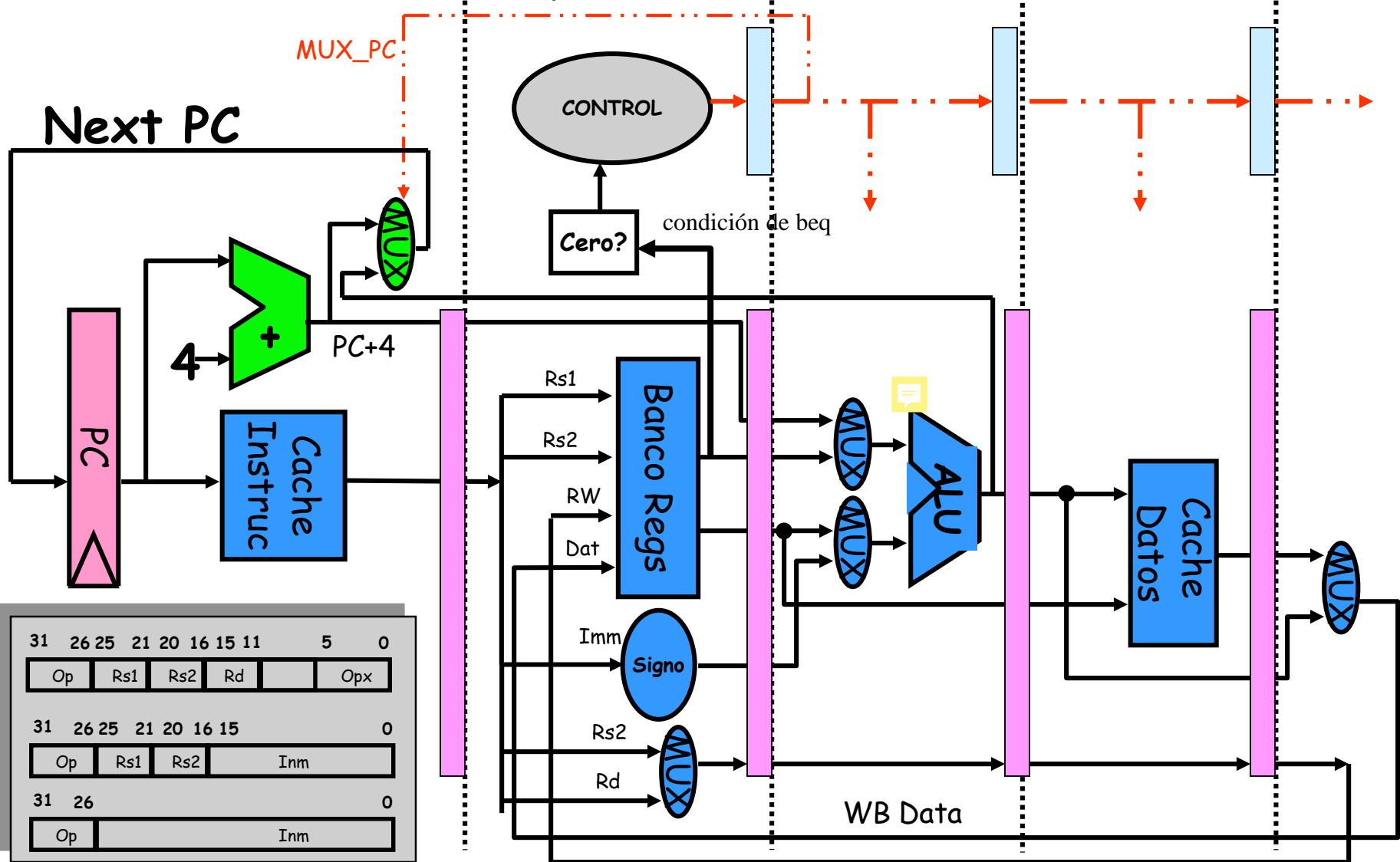
Búsqueda de la Instrucción

Decodificación  
Búsqueda datos

Ejecución  
Calc. Direc.

Acceso a Memoria

Write Back



# Riesgos de Control para Saltos Condicionales



- Solución 1 (más sencilla): PARAR EL FLUJO
- DLX32p original: retardo de saltos condicionales/incondicionales = 2 ciclos
- Si  $CPI = 1$ , 30% de saltos condicionales, 2 ciclos de penalización => nuevo  $CPI = 1.6$  !

	ciclo1	ciclo2	ciclo3	ciclo4	ciclo5	ciclo6	ciclo7	ciclo8	ciclo9
Instr. Salto Cond.	IF	ID	EX	MEM	WB				
Siguiente		IF!	IF!	IF	ID	EX	MEM	WB	
Siguiente +1		PENALIZACIÓN			IF	ID	EX	MEM	WB

# Riesgos de Control para Saltos Condicionales



## • Solución 2:

- Calcular NextPC en ID incluyendo un nuevo sumador
- Penalización saltos tomados: 1 ciclo
- Penalización saltos no-tomados: 0 ciclo

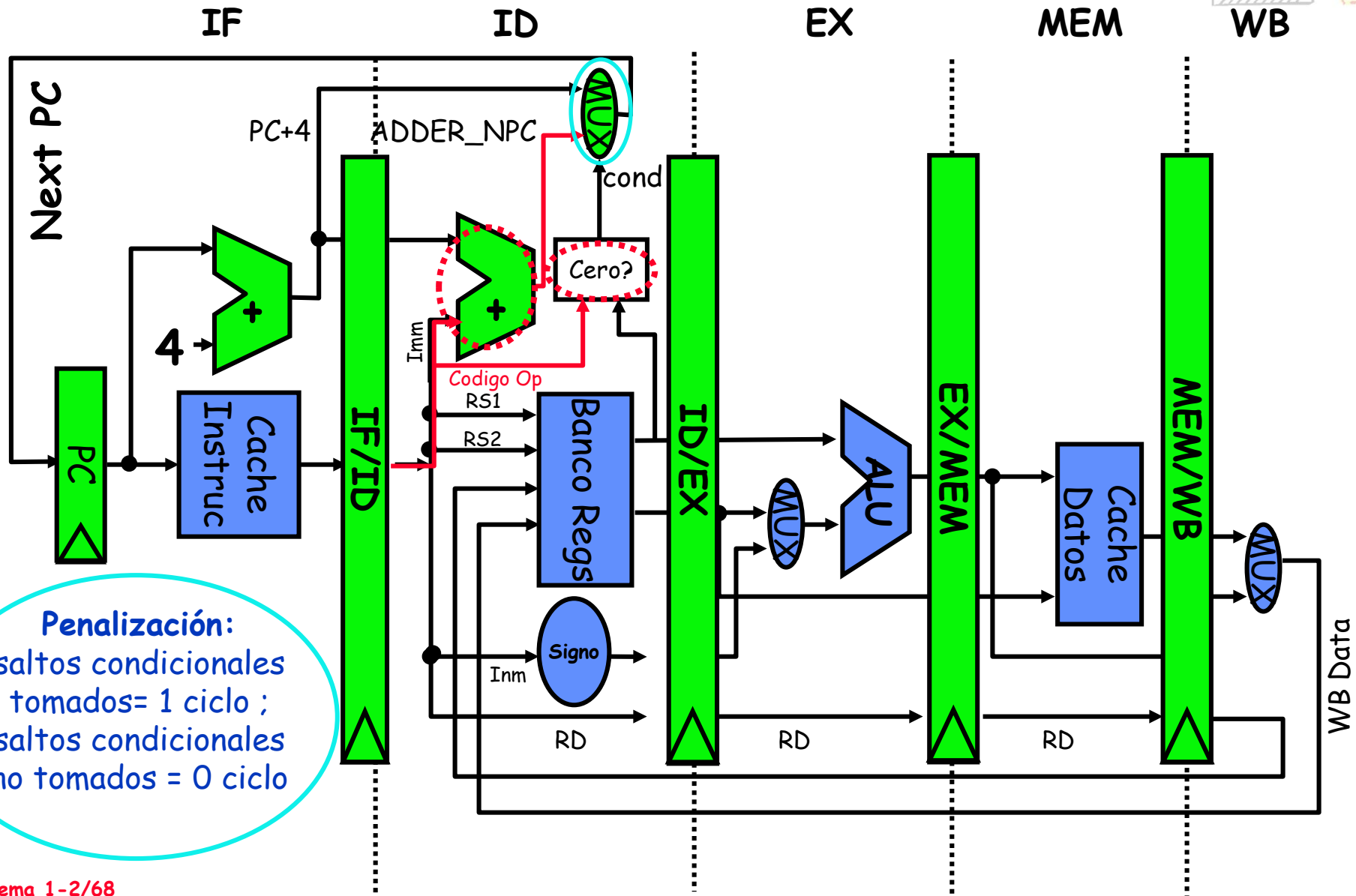
	ciclo1	ciclo2	ciclo3	ciclo4	ciclo5	ciclo6	ciclo7	ciclo8	ciclo9
Instr. Salto	IF	ID	EX	MEM	WB				
Cond.		IF!	IF'	ID	EX	MEM	WB		
Destino del Salto				IF	ID	EX	MEM	WB	
Destino Salto									

**PENALIZACIÓN= 1  
CICLO**

	ciclo1	ciclo2	ciclo3	ciclo4	ciclo5	ciclo6	ciclo7	ciclo8	ciclo9
Instr. Salto	IF	ID	EX	MEM	WB				
Cond.		IF	ID	EX	MEM	WB			
Siguiente									
Destino Salto			IF	ID	EX	MEM	WB		

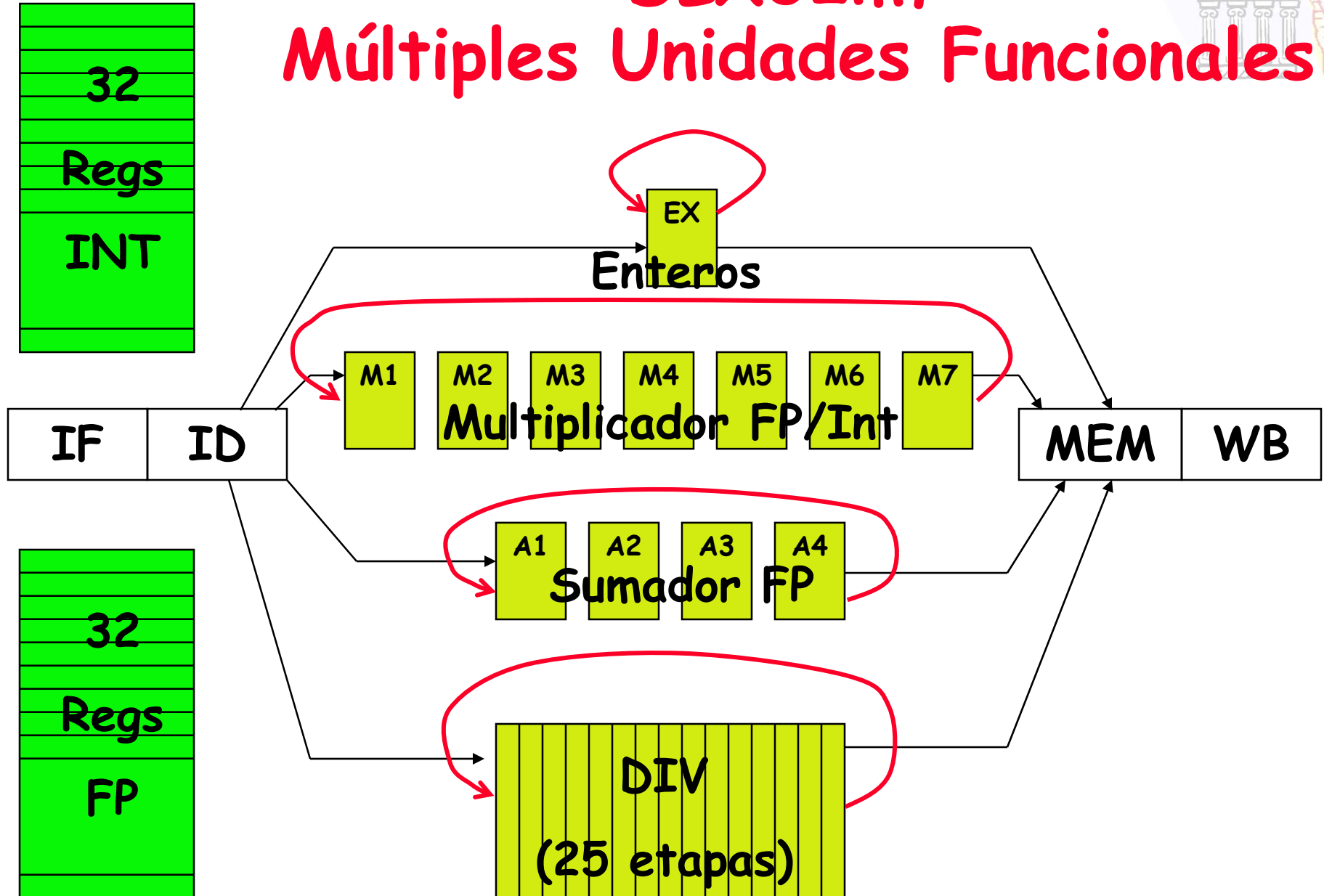
**PENALIZACIÓN= 0  
CICLO**

# DLX32p (Solución 2)





# DLX32mf: Múltiples Unidades Funcionales



# Características generales de las U.F.

## MultiCICLO en procesadores segmentados



- **LATENCIA** de una unidad funcional: número de ciclos de reloj transcurridos entre que una unidad funcional empieza a generar un resultado y la primera instrucción que puede utilizar el resultado finalmente generado (en la misma u otra unidad funcional)
- **INTERVALO DE REPETICIÓN**: número de ciclos transcurridos entre el inicio de 2 operaciones sucesivas en una misma unidad funcional
- **Número de segmentos de una U.F. = latencia + 1**

DLX32mf: Unidad Funcional	Latencia (ciclos)	Intervalo Repetición (ciclos)
ALU Enteros	0	1
Memoria de datos (LW)	1 (por “burbuja”)	1
Suma FP	3	1
Multiplicación	6	1
División (no segmentada)	24	25

# Dependencias Estructurales en Operaciones Multiciclo



## • Dependencias Estructurales

- Debido a las distintas latencias de las unidades funcionales.  
Solución: insertar burbuja hasta que se resuelva la dependencia
- **Dependencia WB**: Debido a que el número de escrituras en banco de registros es mayor que 1

Ciclos en los que las instrucciones que crean "dependencia estructural WB" se encuentran en etapa ID, y se conoce el número de ciclos que le queda para llegar a la etapa WB

Instrucciones DLX  
para operaciones  
en coma flotante

Instrucción	1	2	3	4	5	6	7	8	9	10	11
MULTD F0,F4,F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
...		IF	ID	EX	MEM	WB					
...			IF	ID	EX	MEM	WB				
ADDD F2,F4,F6				IF	ID	A1	A2	A3	A4	MEM	WB
...					IF	ID	EX	MEM	WB		
...						IF	ID	EX	MEM	WB	
LD F8, 0(R2)							IF	ID	EX	MEM	WB



# Dependencias de Datos y Anticipación en Operaciones Multiciclo



- Tipos de dependencias de datos
  - RAW: el problema aquí es más influyente. Solución: anticipación donde se pueda, o burbujas.
  - WAW: debido a que instrucciones posteriores a una dada tienen menor latencia de ejecución. Para el DLX32mf, esta dependencia puede influir negativamente. Solución: parar el cauce; o dejar evolucionar la segunda instrucción además de no dejar que la primera instrucción actualice el estado del procesador (registros o memoria de datos).
  - WAR: no existe en DLX32mf, ya que todas las instrucciones tardan lo mismo en leer los datos (etapa ID) aunque tarden un número diferente de ciclos en escribir los resultados.

# Dependencias de Datos de instrucciones mult Ciclo en DLX32mf



Aparición de Dependencia RAW porque **DATOS** requeridos aquí

MULTD	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
ADDD		IF	ID	A1	A2	A3	A4	MEM	WB		
LD			IF	ID	EX	MEM	WB				
SD				IF	ID	EX	MEM	WB			

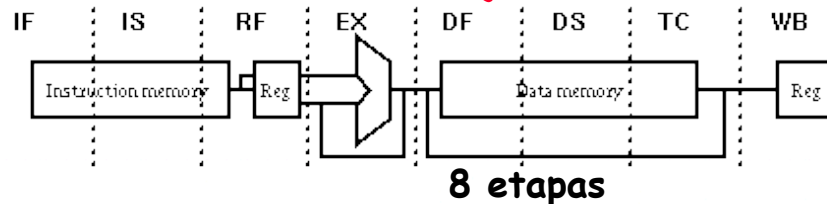
**DATOS** disponibles (escribir)

NO PUEDE aparecer una Dependencia WAR

Aparición de Dependencia WAW

# Prestaciones de MIPS R4000 (similar a DLX32mf)

CPI superior a 1 (CPI ideal=1), causas:



[https://es.wikipedia.org/wiki/MIPS\\_\(procesador\)](https://es.wikipedia.org/wiki/MIPS_(procesador))

## MIPS R4000 Integer Pipeline

The MIPS R4000 architecture's use of superpipelining increases load and branch delays, and also the amount of forwarding required among the stages.

### Load Delays:

If the result of a load is used in the next instruction, it causes a 2 cycle stall in the R4000 pipeline. This is due to the fact that the result of the load is not available to later instructions until the DS stage.

Instruction	1	2	3	4	5	6	7	8	9	10	11
LW R1,(R2)	IF	IS	RF	EX	DF	DS	TC	WB			
ADD R3,R4,R1		IF	IS	RF	stall	stall	EX	DF	DS	TC	WB

**Penalizaciones por Loads (1 o 2 ciclos)**

### Branch Delays:

Taken branches result in a 3 cycle stall, as condition evaluation and branch target computation occur in the EX stage.

Instruction	1	2	3	4	5	6	7	8	9
Branch	IF	IS	RF	EX	DF	DS	TC	WB	
Delay Slot		IF	IS	RF	EX	DF	DS	TC	WB
Stall			stall	stall	stall	stall	stall	stall	stall
Stall				stall	stall	stall	stall	stall	stall
Branch Target					IF	IS	RF	EX	DF

**Penalizaciones por Saltos Tomados (2 ciclos + slots no llenos por salto retardado)**

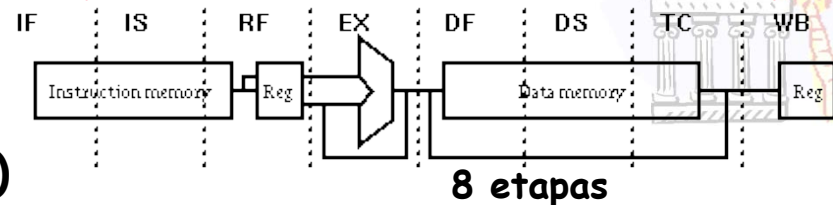
An untaken branch results in no stalls, it simply uses a one cycle delay slot.

Instruction	1	2	3	4	5	6	7	8	9
Branch	IF	IS	RF	EX	DF	DS	TC	WB	
Delay Slot		IF	IS	RF	EX	DF	DS	TC	WB
Branch Instruction + 2			IF	IS	RF	EX	DF	DS	TC
Branch Instruction + 3				IF	IS	RF	EX	DF	DS

**Penalizaciones por Saltos No Tomados (slots no llenos por salto retardado)**

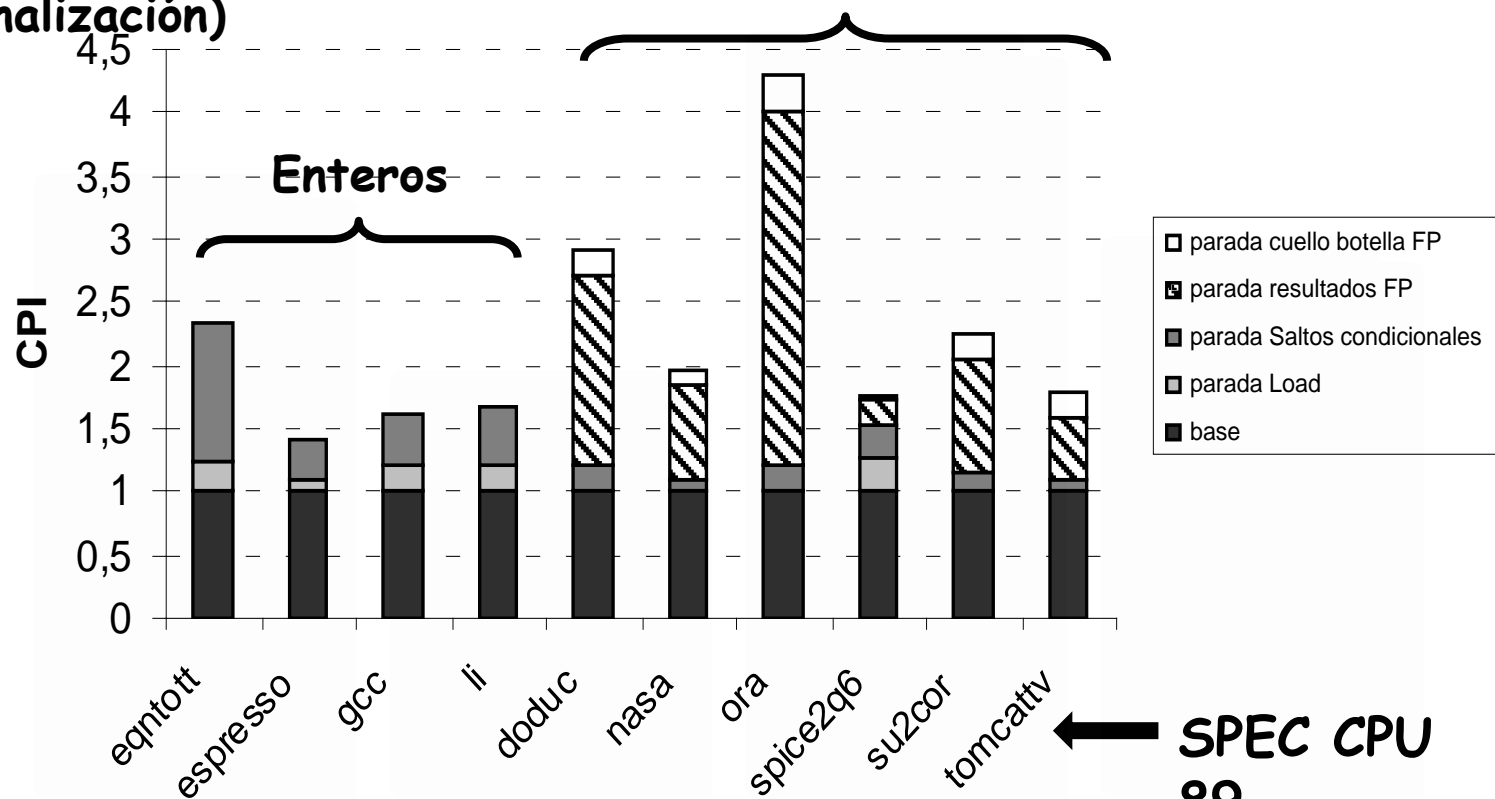
- Penalizaciones por Resultados FP: dependencia RAW (latencia)
- Penalizaciones por Riesgos Estructurales FP: hardware insuficiente, cuello de botella en unidad funcional (paralelismo reduciría esta penalización)

# Prestaciones de MIPS R4000 (similar a DLX32mf)



• CPI superior a 1 (CPI ideal=1), causas:

- Penalizaciones por Loads (1 o 2 ciclos)
- Penalizaciones por Saltos (2 ciclos + slots no llenos por salto retardado)
- Penalizaciones por Resultados FP: dependencia RAW (latencia)
- Penalizaciones por Riesgos Estructurales FP: hardware insuficiente, cuello de botella en unidad funcional (paralelismo reduciría esta penalización)



# Análisis de Prestaciones de Procesadores con Operaciones Multiciclo (R4000)



- Conclusiones sobre SPEC cpu 89 fp ([HP03]pp.A57)

- Número promedio de ciclos de parada por instrucción = 0.65 - 3.2

- » 82% (mayoritario) ocasionado por las "burbujas" insertadas en la ruta de datos debido a las dependencias de resultados en operaciones FP.
    - » 18% (restante) se origina en: saltos, LD, dependencias estructurales, comparaciones.