Procesadores Segmentados y Superescalares

Motivación:

- ✓ ¿Qué implican la segmentación y el paralelismo en el procesador?
- ✓ ¿Qué mejoras de rendimiento son alcanzables con ellas?
- ✓ ¿Qué problemas surgen con la segmentación y como se enfrentan?
- ✓ ¿Qué técnicas derivan de la segmentación?
- ✓ ¿Cómo se organiza un procesador superescalar?

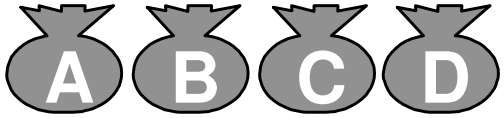
Procesadores Segmentados y Superescalares

- Segmentación y Paralelismo
- Segmentación de un Procesador
- Parámetros fundamentales de rendimiento con segmentación
- Tipos de Segmentación
- Riesgos de la Segmentación. Rendimiento real obtenido
  - ✓ Riesgos estructurales
  - ✓ Riesgos por dependencias de datos
  - ✓ Riesgos de control
- Segmentación avanzada. Procesadores Superescalares

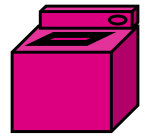
Segmentación *(Pipelining)*

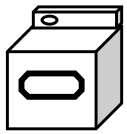
- Ejemplo: hacer la colada

  (cuatro cargas de ropa para lavar, secar, planchar y colocar)

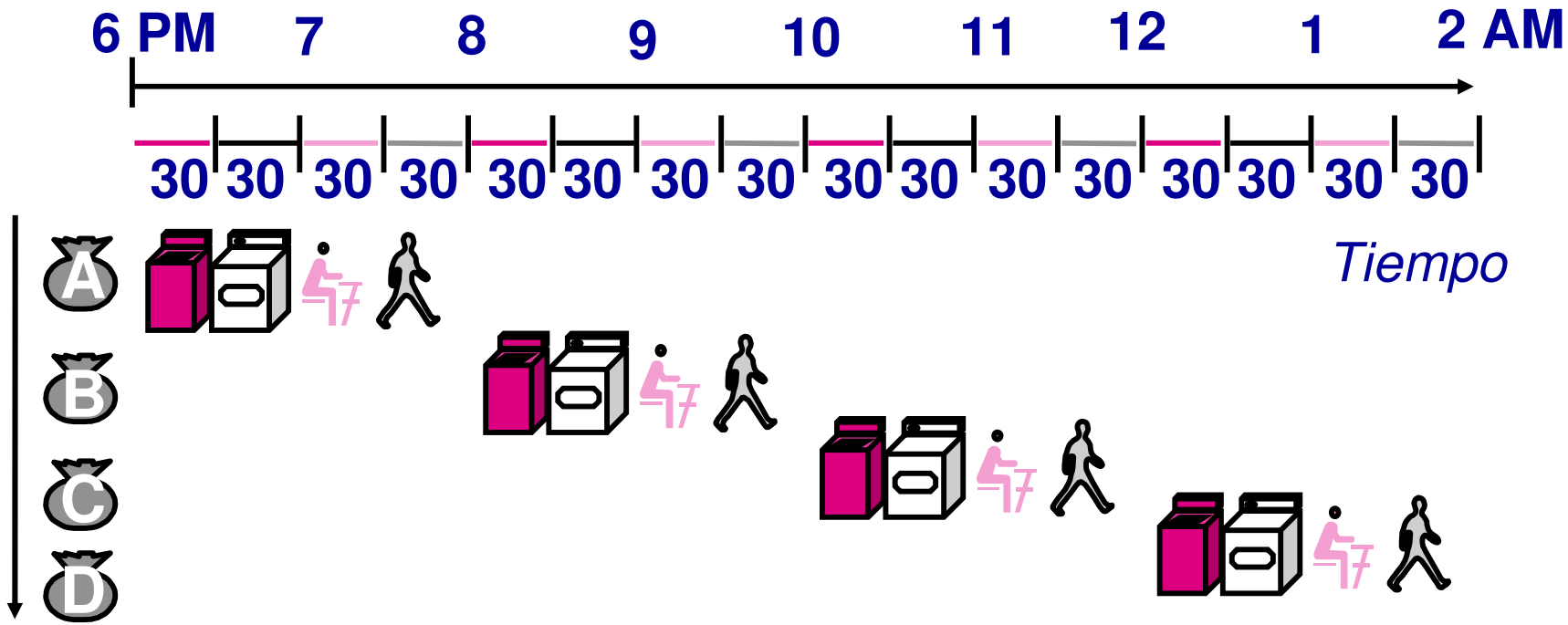


- ✓ Lavar lleva 30 minutos
- ✓ Secar lleva 30 minutos
- ✓ Planchar lleva 30 minutos
- ✓ Colocar lleva 30 minutos



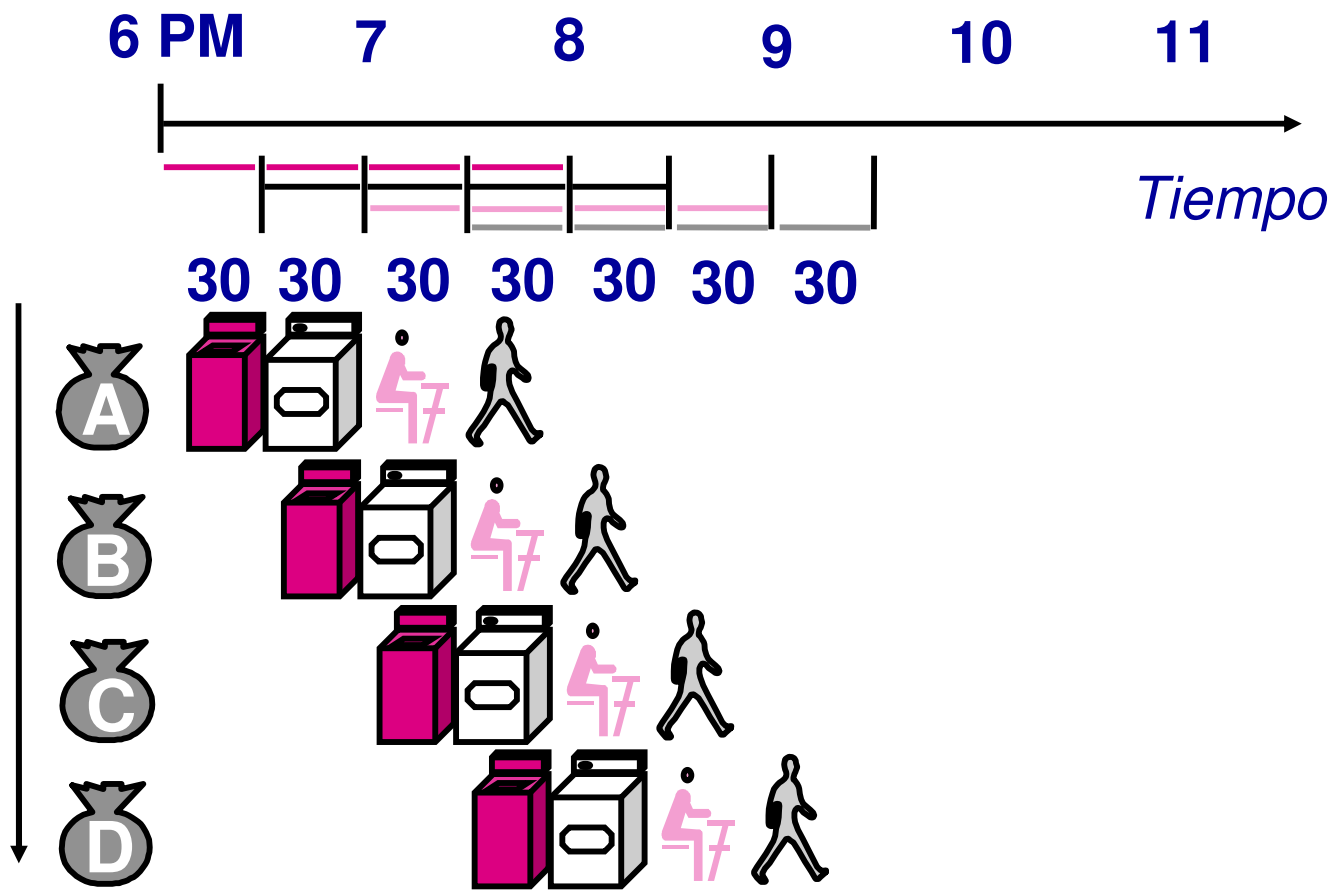


Colada secuencial



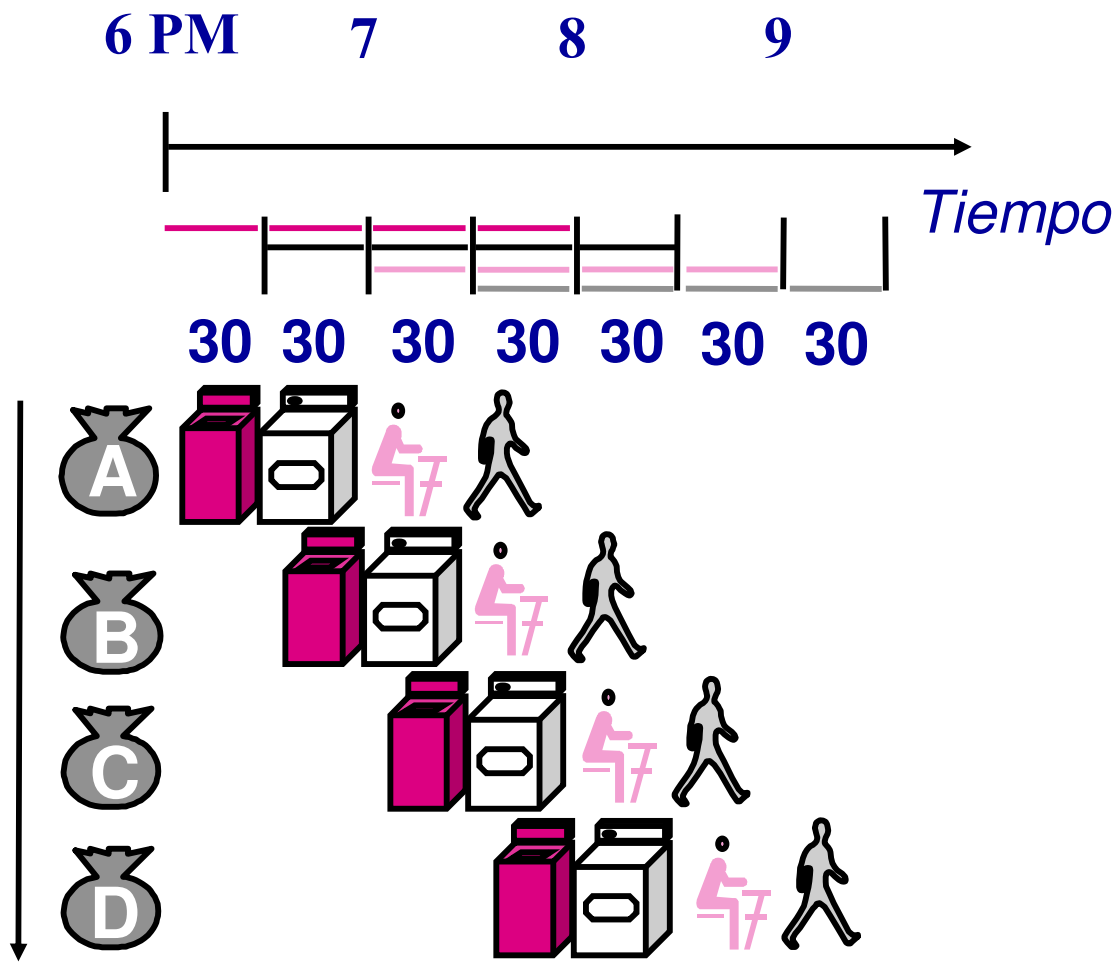
- ✓ La colada secuencial para cuatro cargas lleva 8 horas
- ✓ Si aplicásemos la técnica de segmentación ¿cuánto llevaría?

Colada con Segmentación



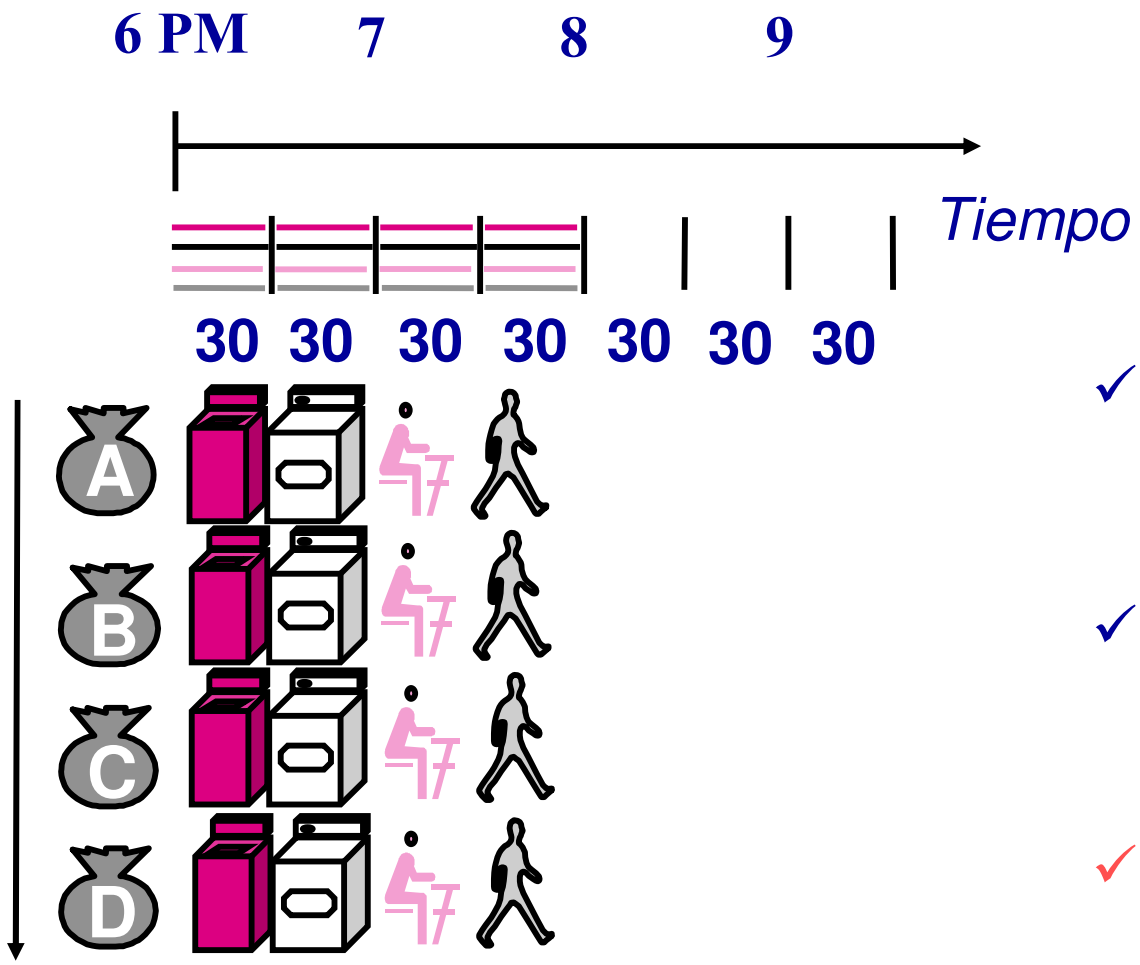
✓ La colada con segmentación para cuatro cargas lleva 3,5 horas

Segmentación: conclusiones



- ✓ La segmentación no reduce la latencia, sino que aumenta la productividad
- ✓ Las múltiples tareas concurrentes utilizan diferentes recursos
- ✓ Ganancia potencial =

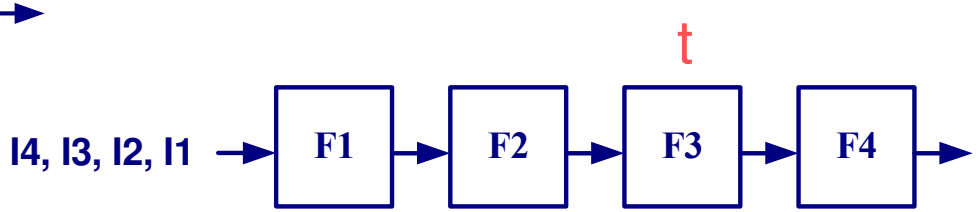
Paralelismo



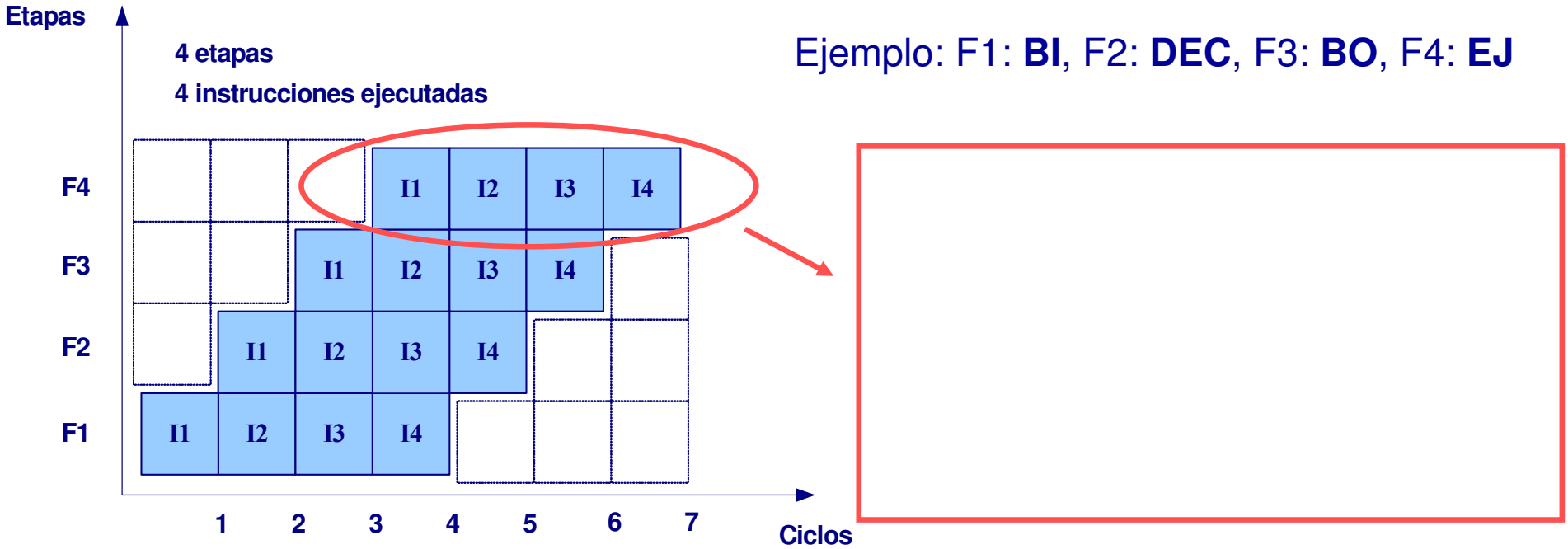
- ✓ El paralelismo no reduce la latencia, sino que aumenta la productividad
- ✓ Las múltiples tareas concurrentes utilizan diferentes recursos
- ✓ Ganancia potencial =

Segmentación de un Procesador

- Cauce de ejecución de instrucciones segmentado



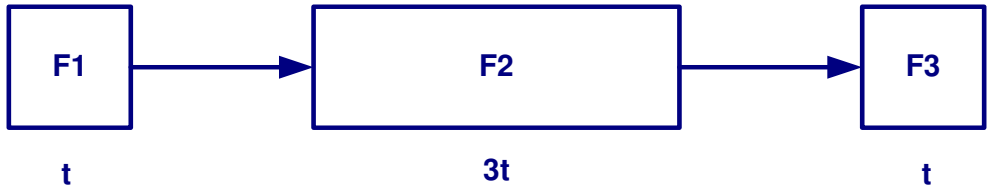
Ejemplo: F1: **BI**, F2: **DEC**, F3: **BO**, F4: **EJ**



Segmentación de un Procesador

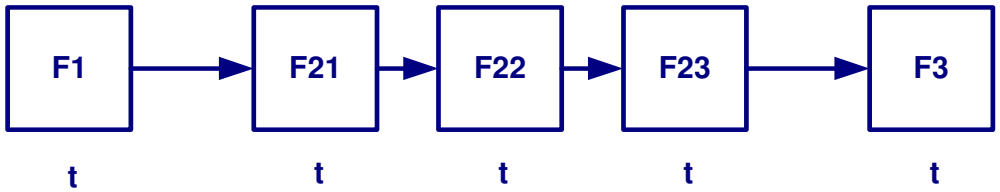
- Nivel al que se explota el paralelismo
  - ✓ Entre instrucciones de un flujo secuencial
  - ✓
- Ventaja importante respecto a otras técnicas (vectorización / multiprocesamiento)
  - ✓ Invisible al programador
- Aspectos importantes
  - ✓ Equilibrado del cauce de ejecución de instrucciones
    - Tiempos de etapas diferentes ➔ etapa cuello de botella ➔
    - G =
  - ✓ Implementación del cauce de ejecución de instrucciones
    - Funcionamiento síncrono

Equilibrado de un Cauce Segmentado



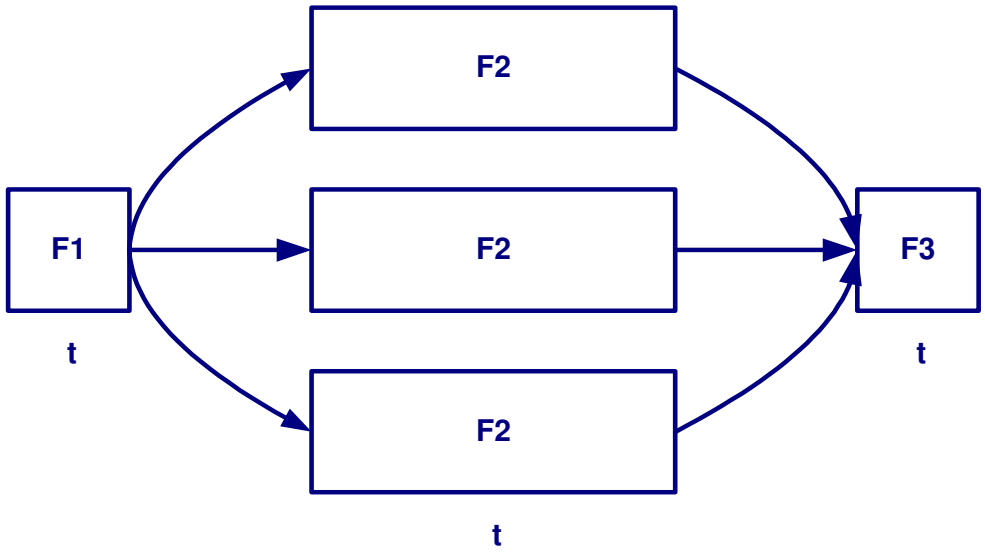
F2 = cuello de botella

G =



Cauce Equilibrado

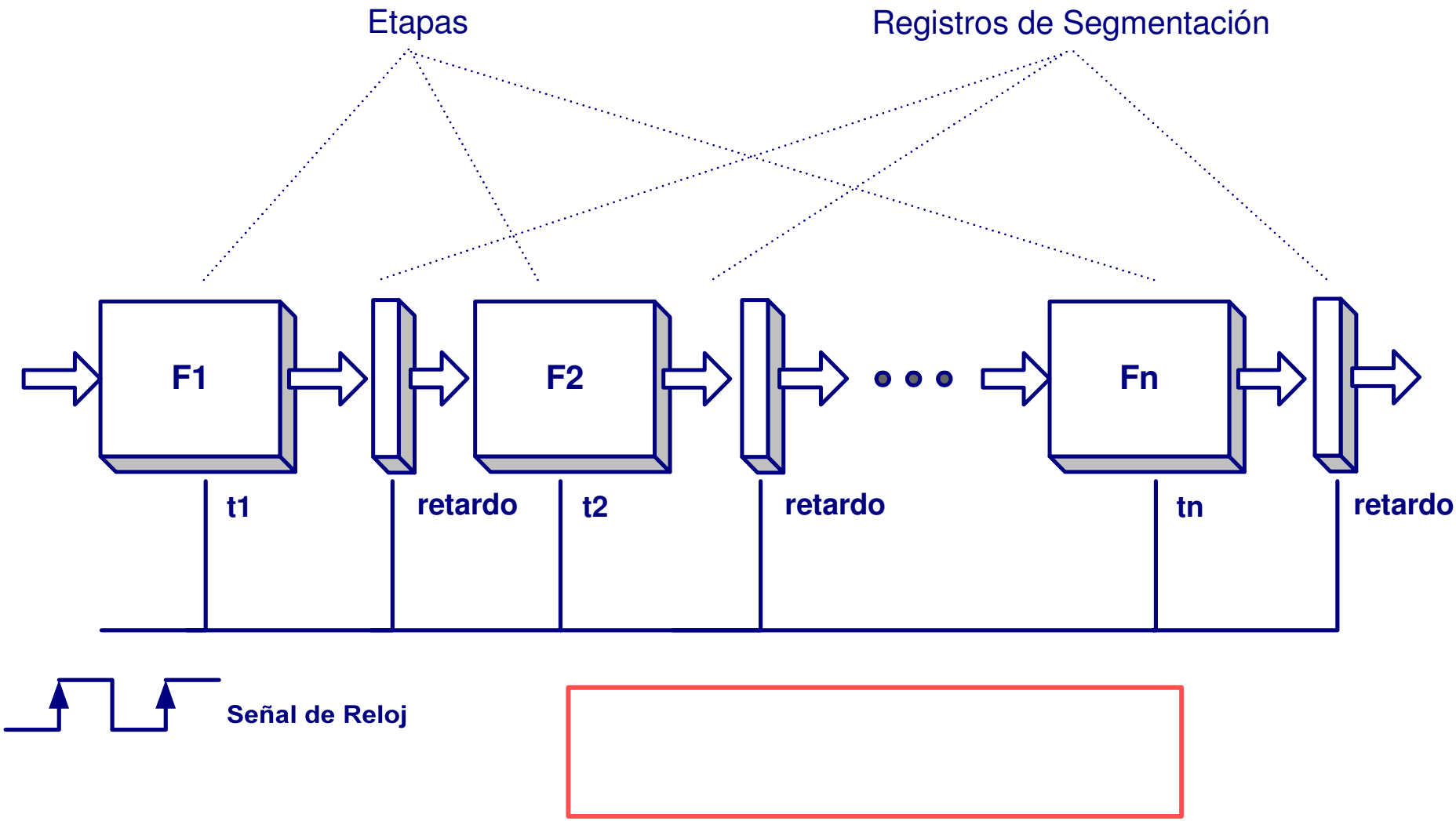
G =



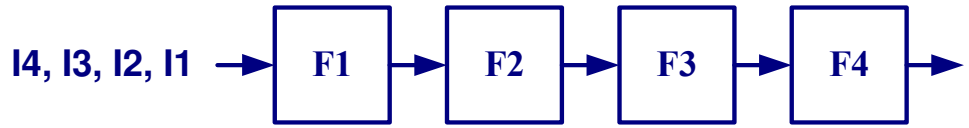
Cauce Equilibrado

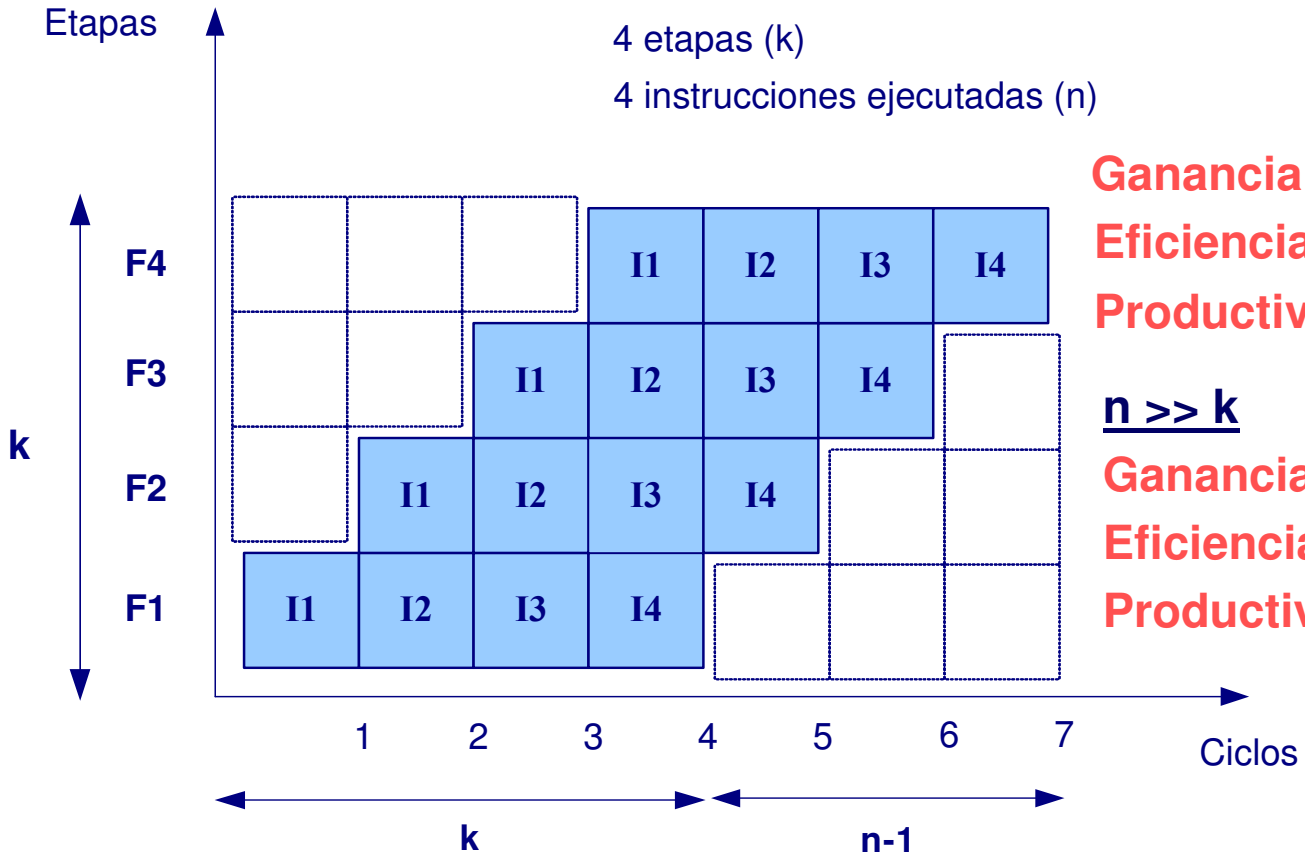
G =

Implementación de un Cauce Segmentado



Parámetros fundamentales de Rendimiento con Segmentación





Ganancia =

Eficiencia =

Productividad =

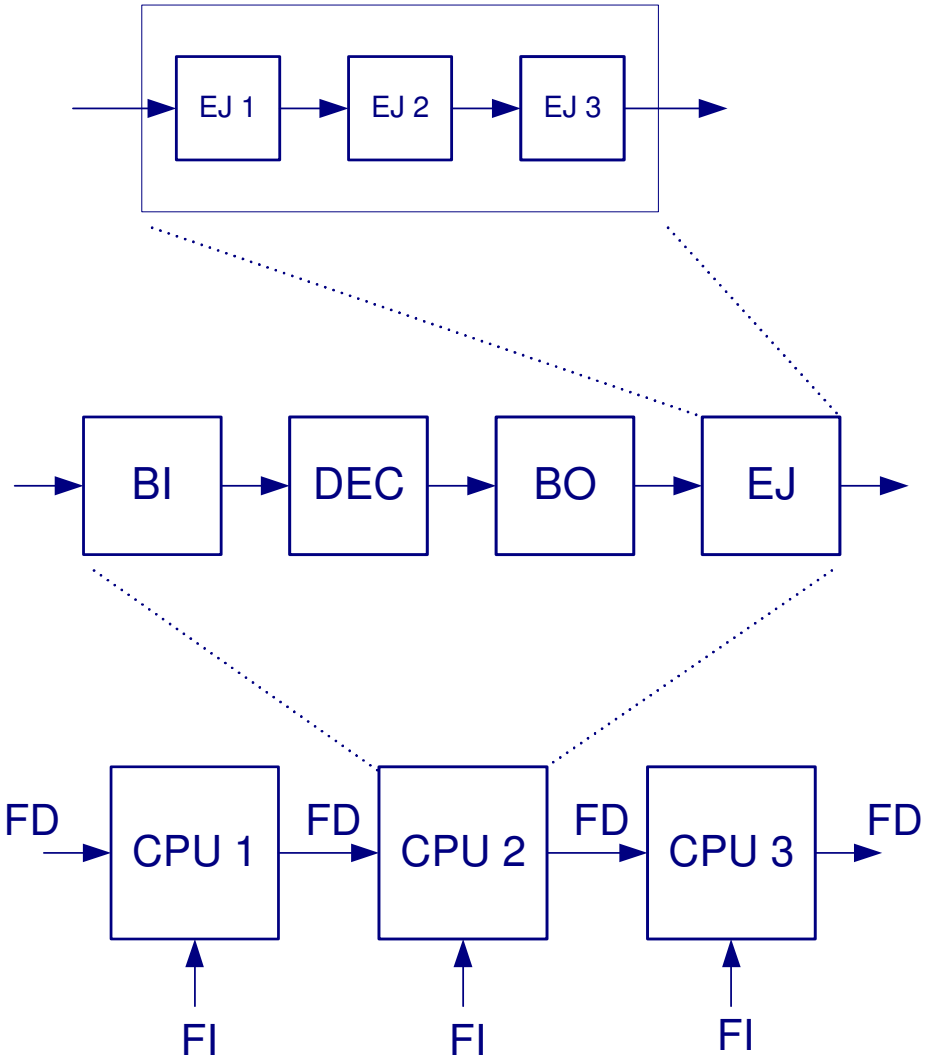
n >> k

Ganancia =

Eficiencia =

Productividad =

Tipos de Segmentación Encauzada



Encauzamiento Aritmético
(Segmentación de ALU)

Encauzamiento de Instrucciones
(Segmentación de CPU)

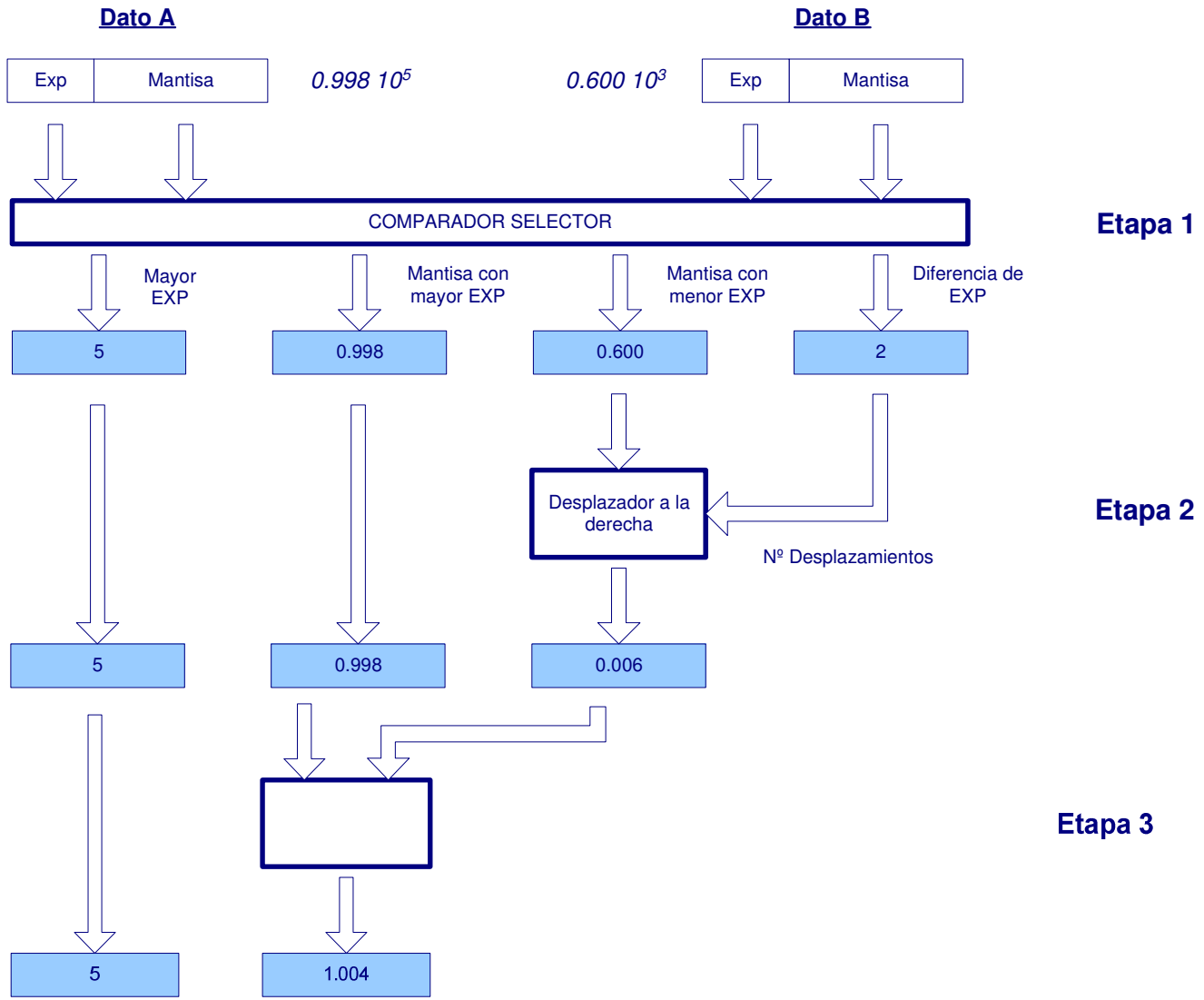
Encauzamiento de procesadores
(Arquitectura MISD)

FI =
FD =

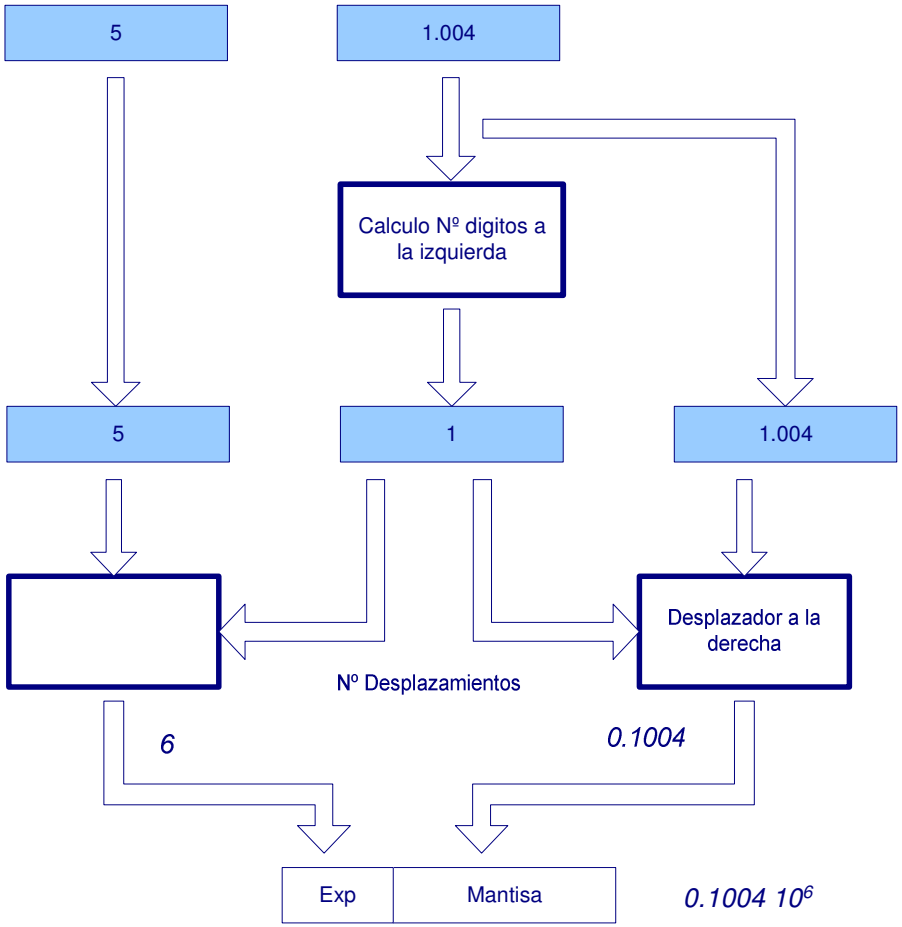
Ejemplo de Unidad Aritmética Segmentada

- Suma en punto flotante
  - ✓ Resta de exponentes y selección del dato de menor exponente
  - ✓ Desplazamiento a la derecha de la mantisa con menor exp. hasta igualar exp.
  - ✓ Suma de las mantisas
- Normalización del resultado
  - ✓ Obtención del nº de dígitos significativos a la izquierda del punto decimal
  - ✓ Desplazamiento a la derecha de la mantisa suma (incrementando el exponente) tantos lugares como nº de dígitos significativos

Unidad de Suma en Punto Flotante Segmentada



Unidad de Suma en Punto Flotante Segmentada



Etapa 4

Etapa 5

Resultado

Riesgos en la Segmentación

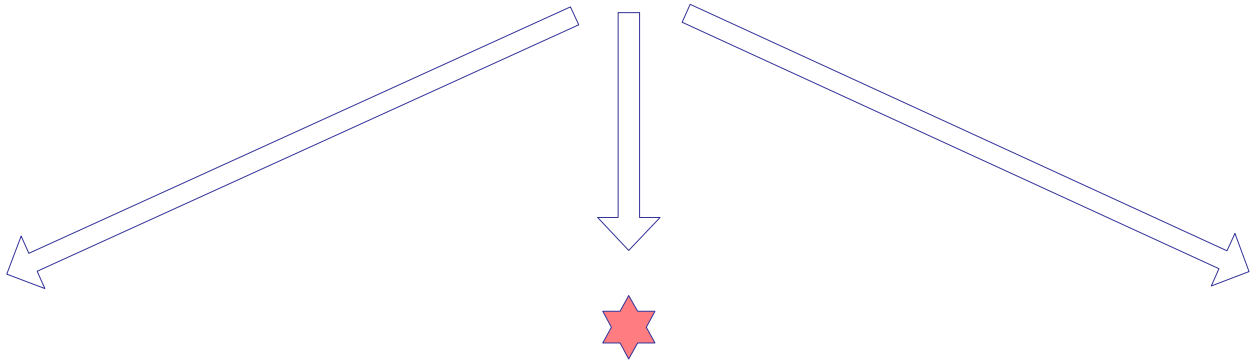
- Tipos de riesgos
- Soluciones
- Rendimiento real teniendo en cuenta los riesgos
- Ejemplo de procesador segmentado sencillo
- Riesgos Estructurales. Causas
- Riesgos por Dependencias de Datos. Tipos y soluciones
- Riesgos de Control. Tipos y soluciones

Tipos de Riesgos

- Riesgos Estructurales
  - ✓ Conflictos en el uso de los recursos del cauce segmentado por parte de las multiples instrucciones cuya ejecución se solapa (recursos insuficientemente replicados o UF no segmentadas)
- Riesgos por Dependencias de Datos
  - ✓ Problemas derivados de dependencias entre instrucciones cuya ejecución se solapa
- Riesgos de Control
  - ✓ Problemas derivados de las instrucciones de control de flujo, debido al desconocimiento temporal de la siguiente instrucción a ejecutar

Consecuencias a minimizar:

Soluciones



Resolución Estática

Desarrollada por el Compilador

Ejemplos: MIPS

Resolución Combinada

Desarrollada en parte por el compilador y en parte por Hardware

Ejemplos: MIPS R-line

Resolución Dinámica

Desarrollada por Hardware adicional

Ejemplos: La mayoría

Tendencia

Rendimiento real teniendo en cuenta los riesgos

$$G = \frac{\qquad\qquad}{\qquad\qquad}$$

Profundidad Segmentación =

Ciclos Detención por Instrucción = *

Ejemplo de procesador segmentado sencillo

- Procesador segmentado lineal de 5 etapas (MIPS R3000)

BUS	D-L	EJE	MEM	ESC

- Etapas:

BUS	Busqueda de Instrucción
D-L	Decodificación y Lectura de Registros
EJE	Ejecución en la ALU (operación + calculo de direcciones)
MEM	Acceso a Memoria (lectura o escritura)
ESC	Escritura en Registro

- Todas las instrucciones tardan el mismo tiempo y "atraviesan" todas las etapas (las utilicen o no)
- Arquitectura de Carga/Almacenamiento (el resto de instrucciones opera sobre registros)

Ejemplo de procesador segmentado sencillo

Ciclo de Reloj	1	2	3	4	5	6	7	8	9 ...
Instrucción i	BUS	D-L	EJE	MEM	ESC				
Instrucción i+1		BUS	D-L	EJE	MEM	ESC			
Instrucción i+2			BUS	D-L	EJE	MEM	ESC		
Instrucción i+3				BUS	D-L	EJE	MEM	ESC	
Instrucción i+4					BUS	D-L	EJE	MEM	ESC

Riesgos Estructurales

- Causas:
  - ✓ ***Recursos insuficientes***: *los recursos no se han replicado lo suficiente como para permitir la ejecución solapada de todas las combinaciones de instrucciones sin dar lugar a pérdida de ciclos*
  - ✓ ***Unidades Funcionales no segmentadas***: *no es posible iniciar una secuencia de instrucciones en la que varias consecutivas utilicen esa unidad funcional sin esperas y por tanto pérdida de ciclos*

Riesgos por recursos insuficientes

Ciclo de Reloj	1	2	3	4	5	6	7	8	9 ...
Instrucción i	BUS	D-L	EJE	MEM	ESC				
Instrucción i+1		BUS	D-L	EJE	MEM	ESC			
Instrucción i+2			BUS	D-L	EJE	MEM	ESC		
Instrucción i+3				BUS	D-L	EJE	MEM	ESC	
Instrucción i+4					BUS	D-L	EJE	MEM	ESC

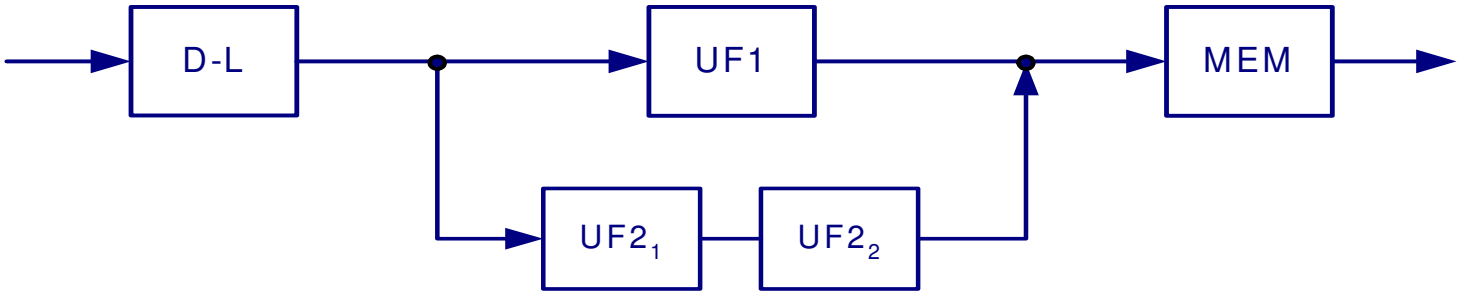
- Requerimientos: En cada ciclo debe efectuarse ...
  - ✓ lectura de una instrucción
  - ✓ lectura o escritura de un dato
  - ✓ lectura del contenido de dos registros
  - ✓ escritura de un registro
  - ✓ operación de ALU y cálculo del nuevo valor del PC *(en la etapa BUS)*

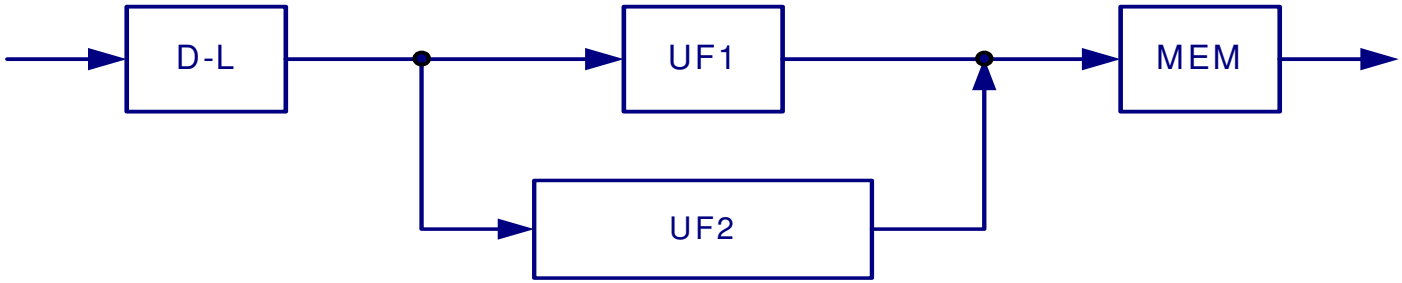
Riesgos por recursos insuficientes

Ciclo de Reloj	1	2	3	4	5	6	7	8	9 ...
No Carga/Alm.	BUS	D-L	EJE	MEM	ESC				
Instrucción i+1		BUS	D-L	EJE	MEM	ESC			
Instrucción i+2			BUS	D-L	EJE	MEM	ESC		
Instrucción i+3				BUS	D-L	EJE	MEM	ESC	
Instrucción i+4					BUS	D-L	EJE	MEM	ESC

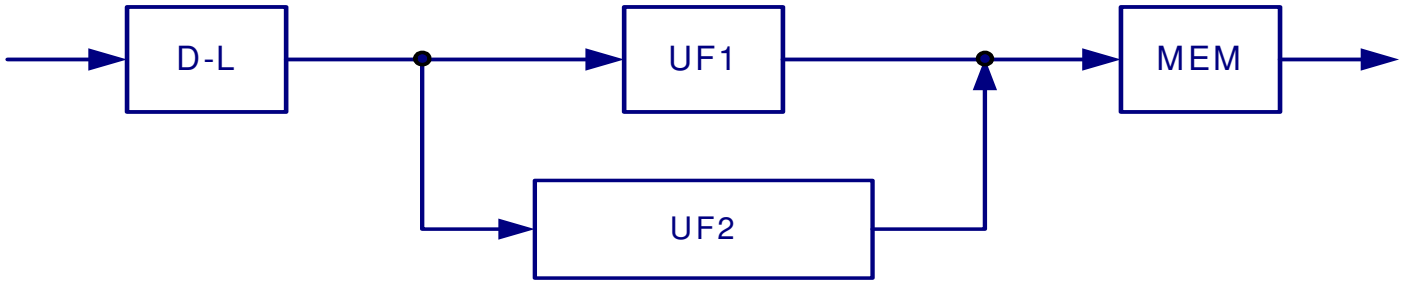
Ciclo de Reloj	1	2	3	4	5	6	7	8	9 ...
Carga/Almac.	BUS	D-L	EJE	MEM	ESC				
Instrucción i+1		BUS	D-L	EJE	MEM	ESC			
Instrucción i+2			BUS	D-L	EJE	MEM	ESC		
Instrucción i+3					BUS	D-L	EJE	MEM	ESC
Instrucción i+4						BUS	D-L	EJE	MEM	ESC

Riesgos por Unidades Funcionales no segmentadas





Riesgos por Unidades Funcionales no segmentadas

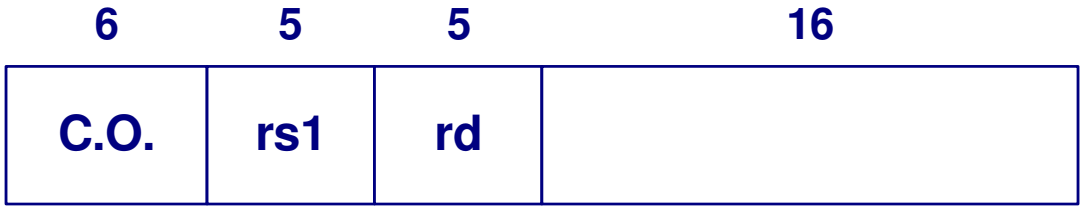


- No se pueden soportar 2 instrucciones que necesiten UF2 en 2 ciclos consecutivos
- La frecuencia de instrucciones que puede soportar la unidad no segmentada está limitada:

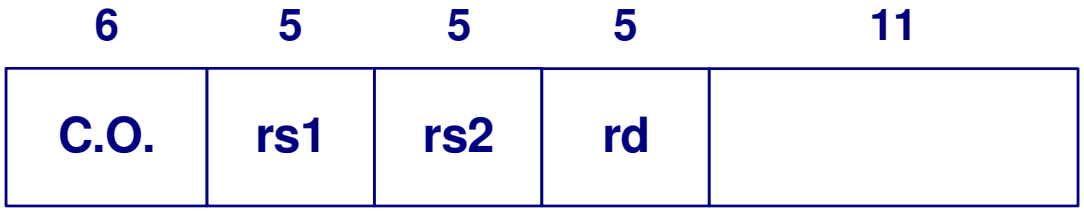
f_{max} =

f_{max} =

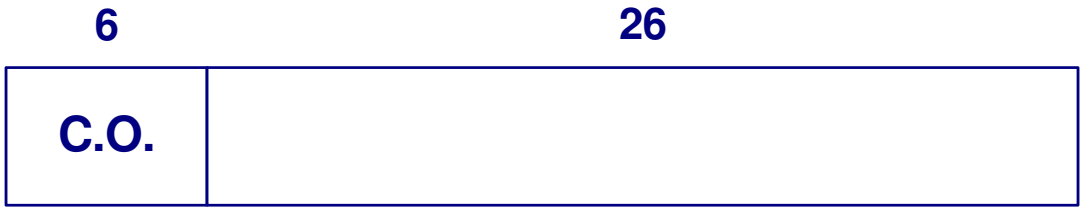
Formatos de Instrucción en el MIPS R3000



Carga /Almacenamiento
Saltos Condicionales



Operaciones de ALU



Saltos Incondicionales

Riesgos por Dependencias de Datos

- Problema de la dependencia de datos
- Tipos de dependencias de datos
- Dependencias en el MIPS R3000
- Métodos de resolución
  - ✓ Resolución estática (en tiempo de compilación)
  - ✓ Resolución dinámica (mediante hardware adicional)

Problema de la Dependencia de Datos

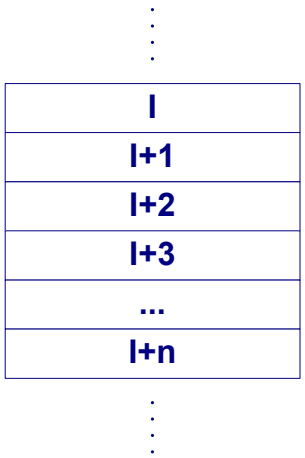
- Las Lecturas/Escrituras sobre una misma variable (registro) deben efectuarse en el orden que especifica el programa secuencial

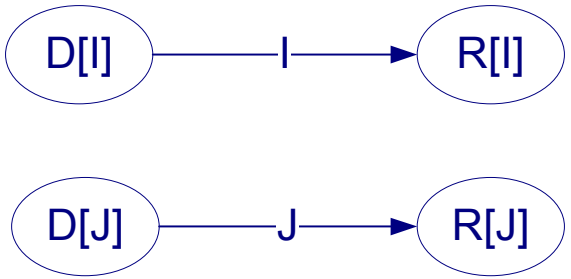
  - Dominio de una Instrucción (**D[I]**):

    Conjunto de variables (registros) sobre las que la instrucción realiza lecturas

  - Rango de una Instrucción (**R[I]**):

    Conjunto de variables (registros) sobre las que la instrucción realiza escrituras

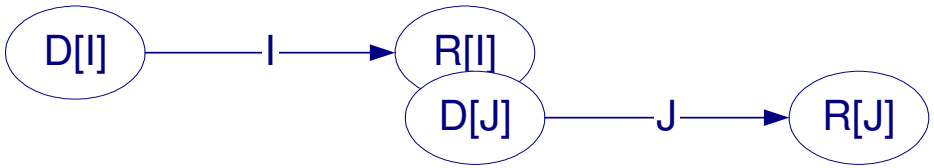




J =

Tipos de Dependencias de Datos

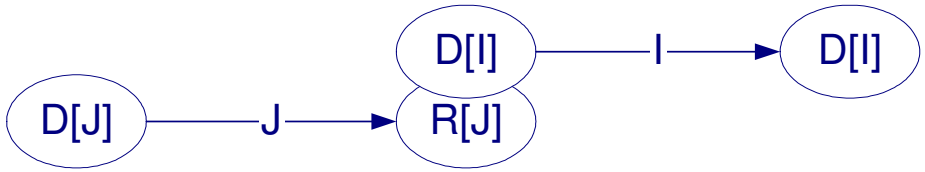
- **Lectura despues de Escritura** (Verdadera)



Ejemplo:

R1 = ...
... = R1

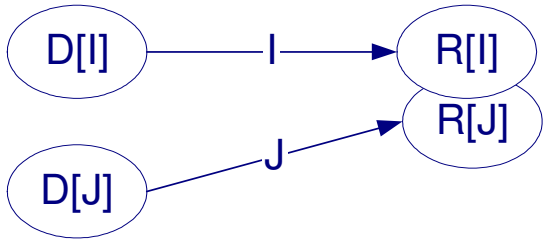
- **Escritura despues de Lectura** (Antidependencia)



Ejemplo:

... = R1
R1 = ...

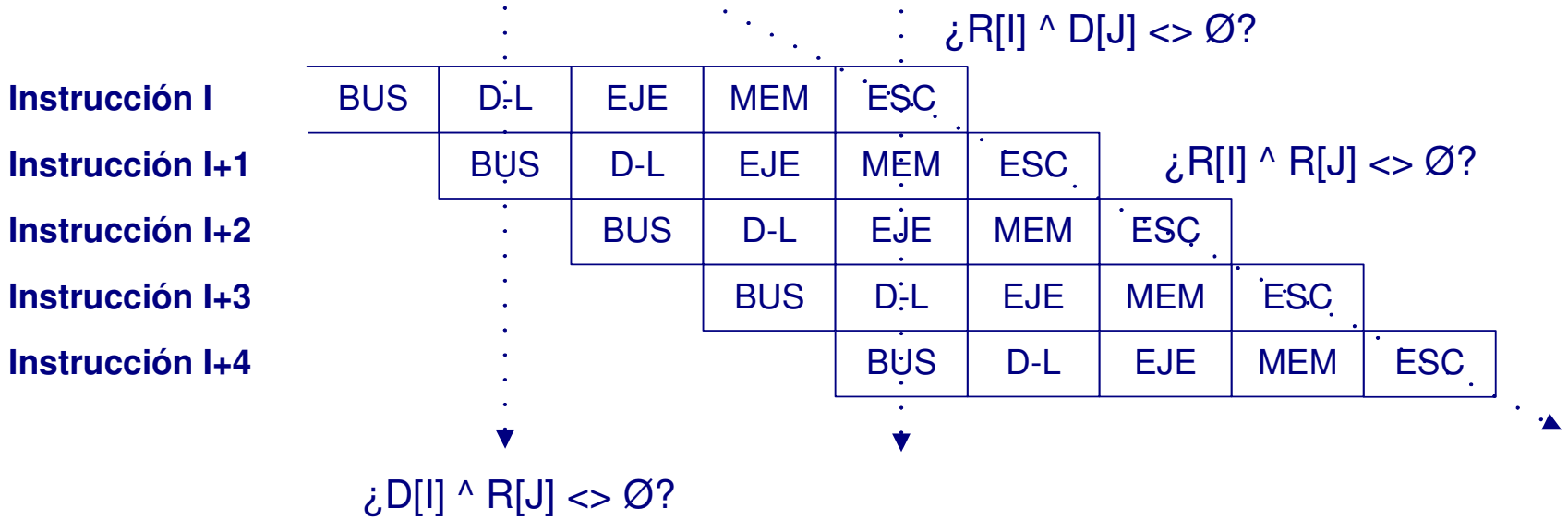
- **Escritura despues de Escritura** (de salida)



Ejemplo:

R1 = ...
R1 = ...

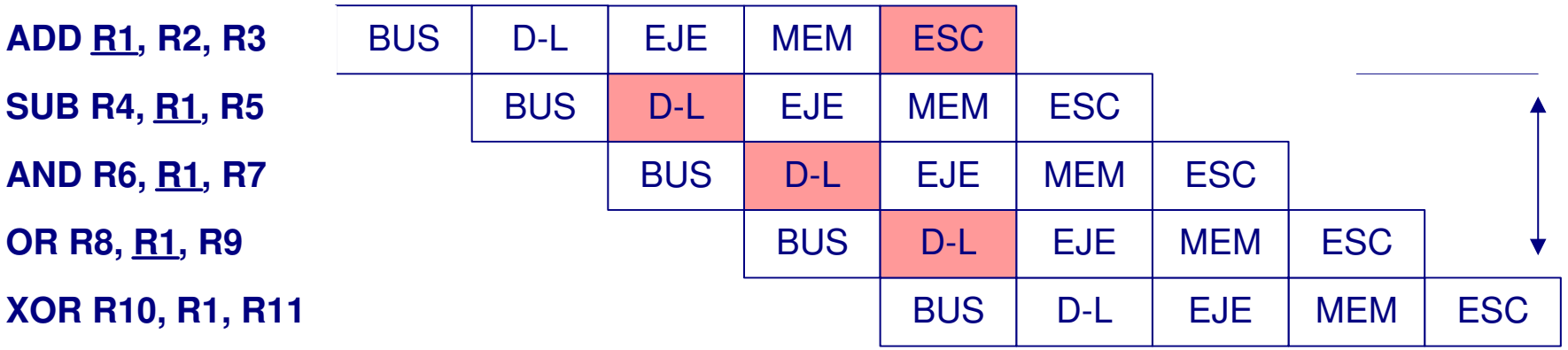
Dependencias en el MIPS R3000



- **Escritura despues de Lectura** → ¿Puede alguna instrucción I+n escribir antes de que I lea? →
- **Escritura despues de Escritura** → ¿Puede alguna instrucción I+n escribir antes de que I escriba? →
- **Lectura despues de Escritura** → ¿Puede alguna instrucción I+n leer antes de que I escriba? →

Ejemplo de Dependencia LDE

Formato de Instrucciones MIPS:



➡ **Número máximo de instrucciones afectadas =**

Resolución Estática (I)

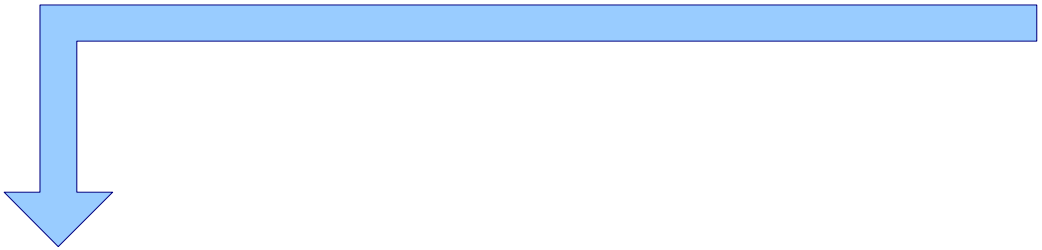
- **Inserción de Instrucciones NOP**

ADD R1, R2, R3	BUS	D-L	EJE	MEM	ESC			
NOP		BUS	D-L	EJE	MEM	ESC		
NOP			BUS	D-L	EJE	MEM	ESC	
NOP				BUS	D-L	EJE	MEM	ESC
OR R8, R1, R9					BUS	D-L	EJE	MEM
SUB R4, R1, R5								
AND R6, R1, R7								
XOR R10, R1, R11								

Resolución Estática (II)

- **Reordenación de Código**

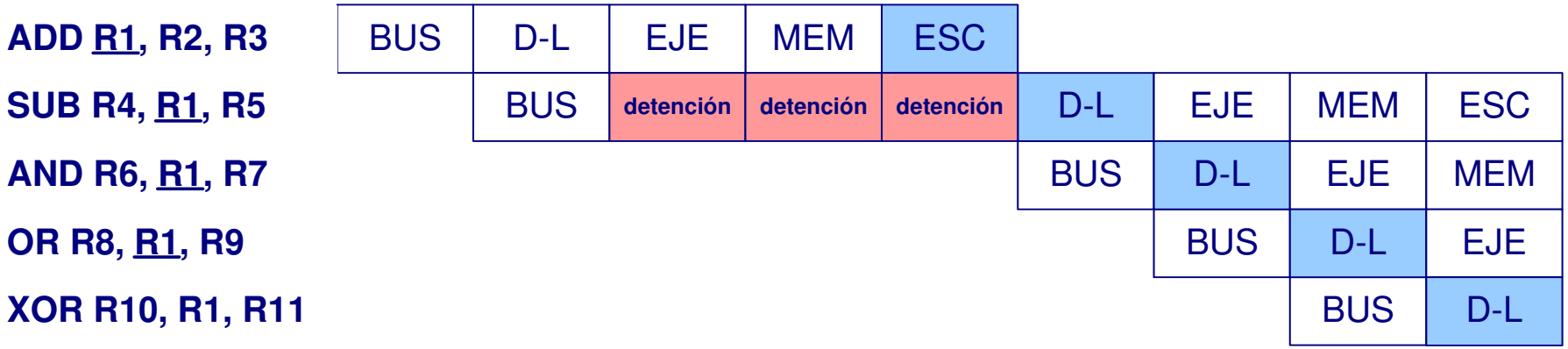
ADD R1, R2, R3
SUB R4, R1, R5
AND R6, R1, R7
OR R8, R1, R9
XOR R10, R1, R11
AND R12, R12, R13
SUB R14, R15, R14
ADD R16, R17, R18



ADD R1, R2, R3	BUS	D-L	EJE	MEM	ESC							
AND R12, R12, R13		BUS	D-L	EJE	MEM	ESC						
SUB R14, R15, R14			BUS	D-L	EJE	MEM	ESC					
ADD R16, R17, R18				BUS	D-L	EJE	MEM	ESC				
SUB R4, R1, R5					BUS	D-L	EJE	MEM	ESC			
AND R6, R1, R7												
OR R8, R1, R9												
XOR R10, R1, R11												

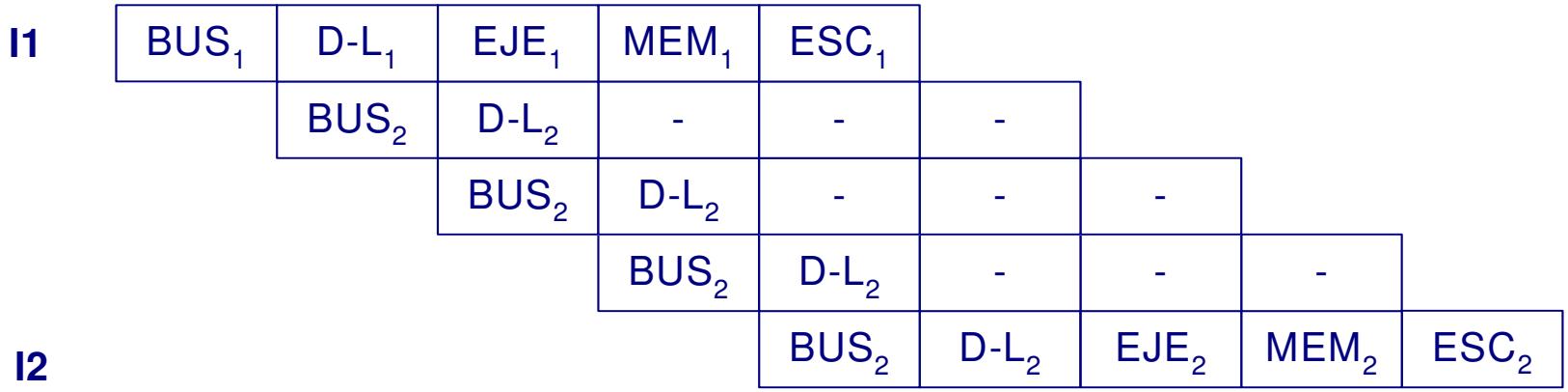
Resolución mediante Hardware (I)

Detención del cauce



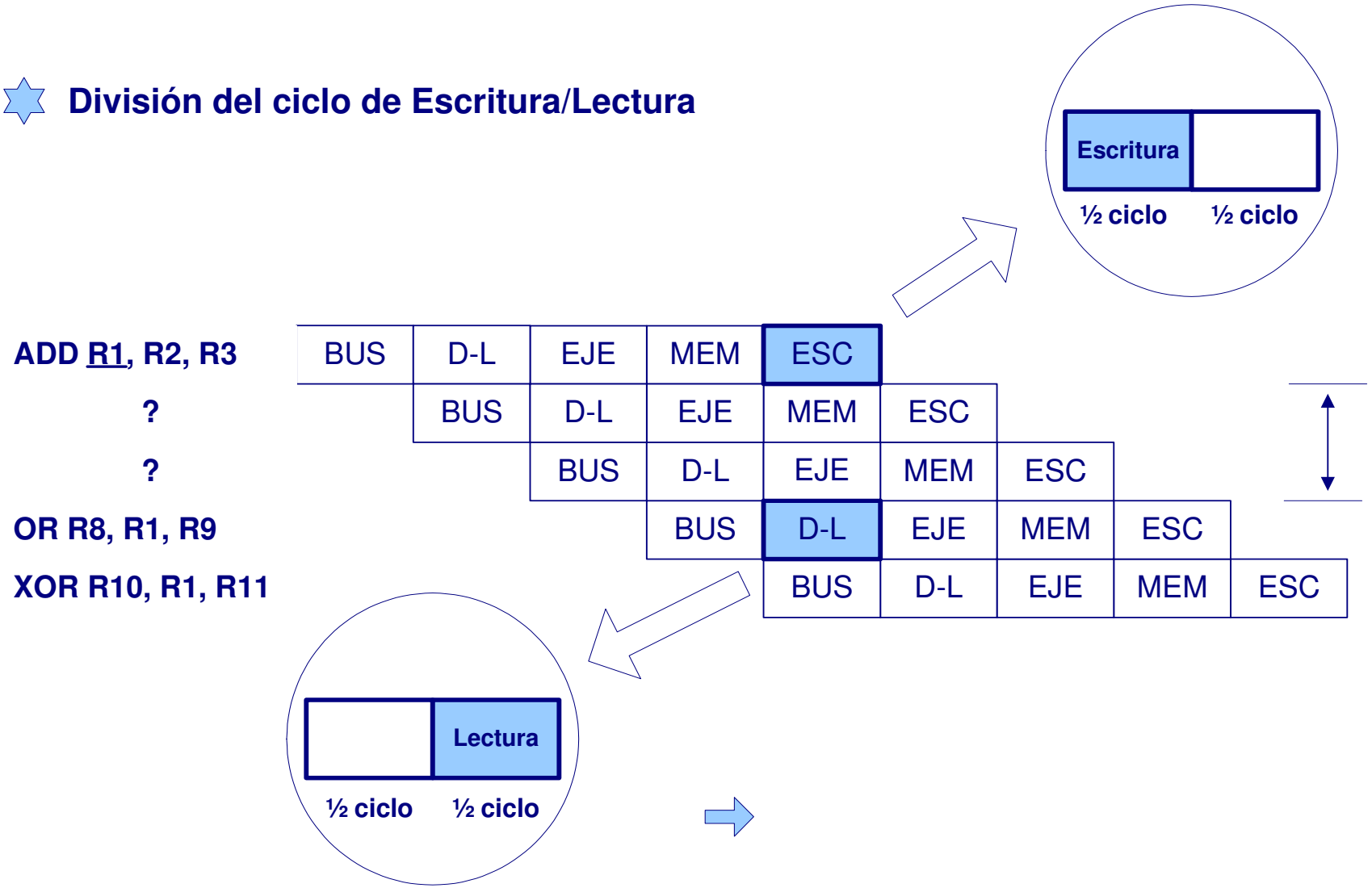
Alternativa Hardware mas sencilla:

Control de Dependencias



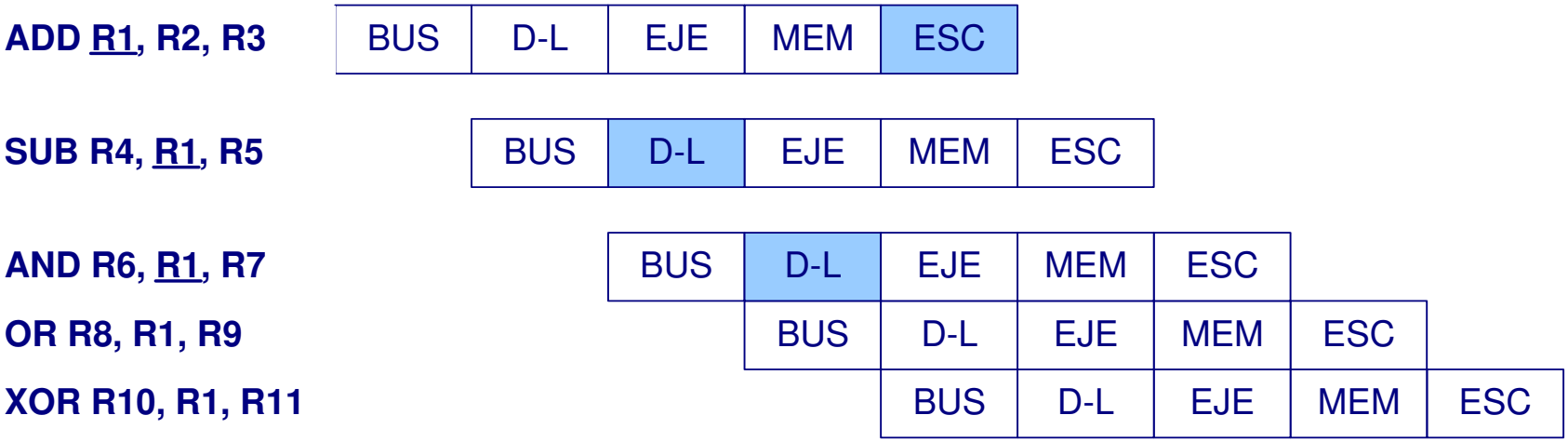
Resolución mediante Hardware (II)

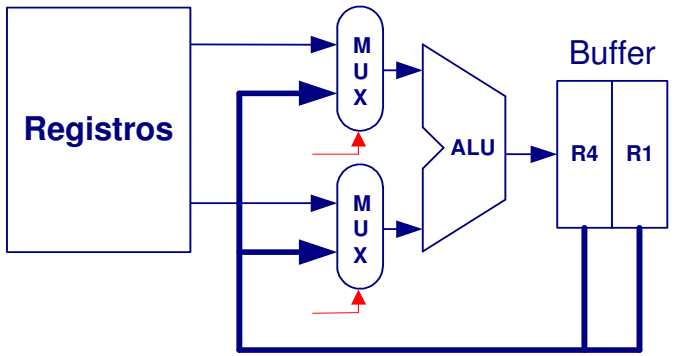
- **División del ciclo de Escritura/Lectura**



Resolución mediante Hardware (III)

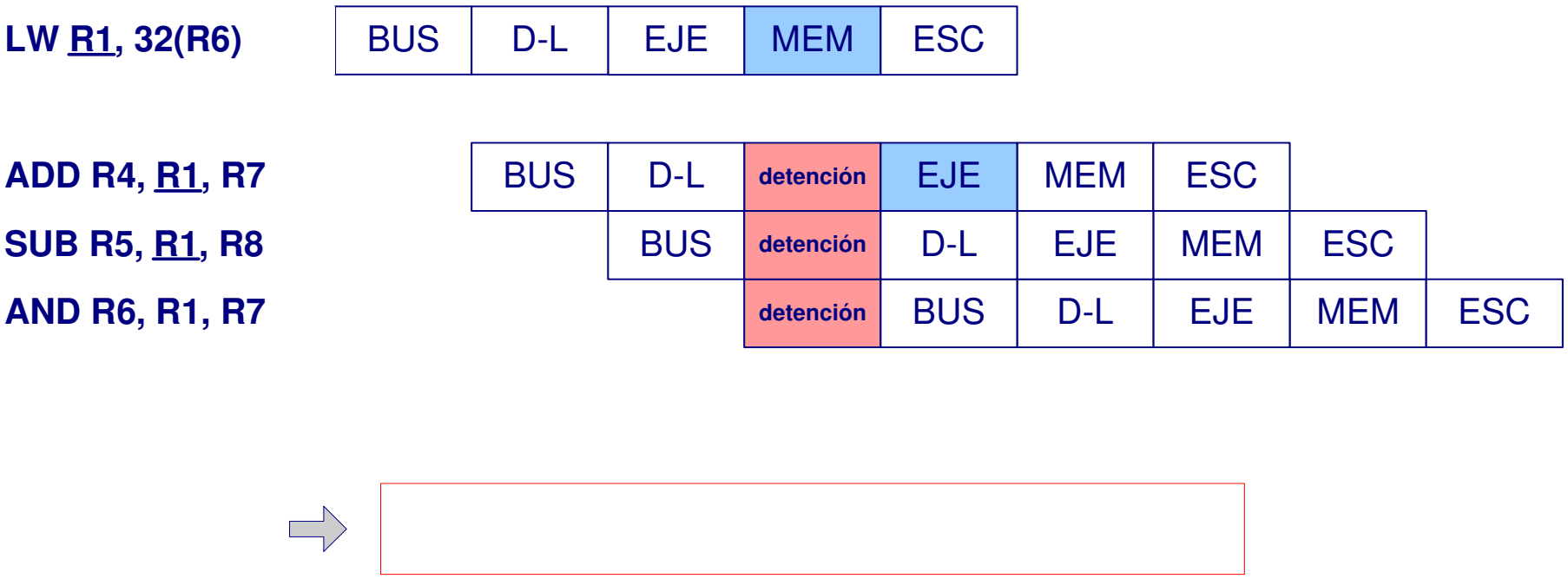
- **Técnica de Adelantamiento o Desvío** *(Forwarding, Bypassing)*





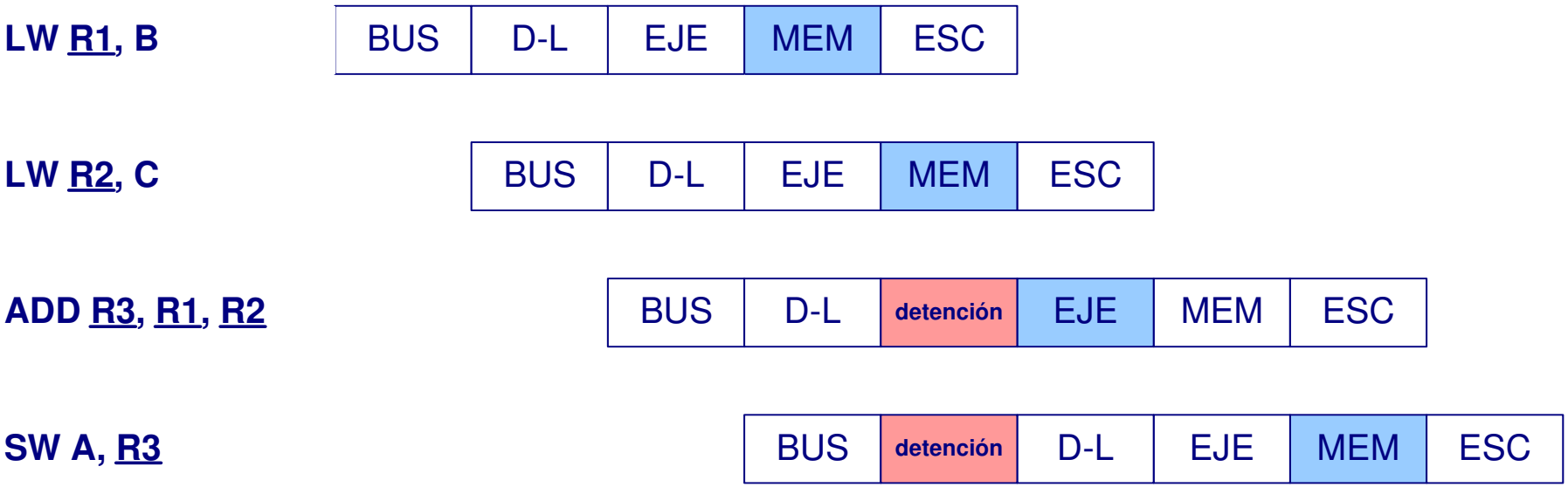
Instrucciones de Carga (I)

Efecto de una instrucción de carga en la segmentación:



Instrucciones de Carga (II)

Secuencia de código para la operación **A=B+C** :

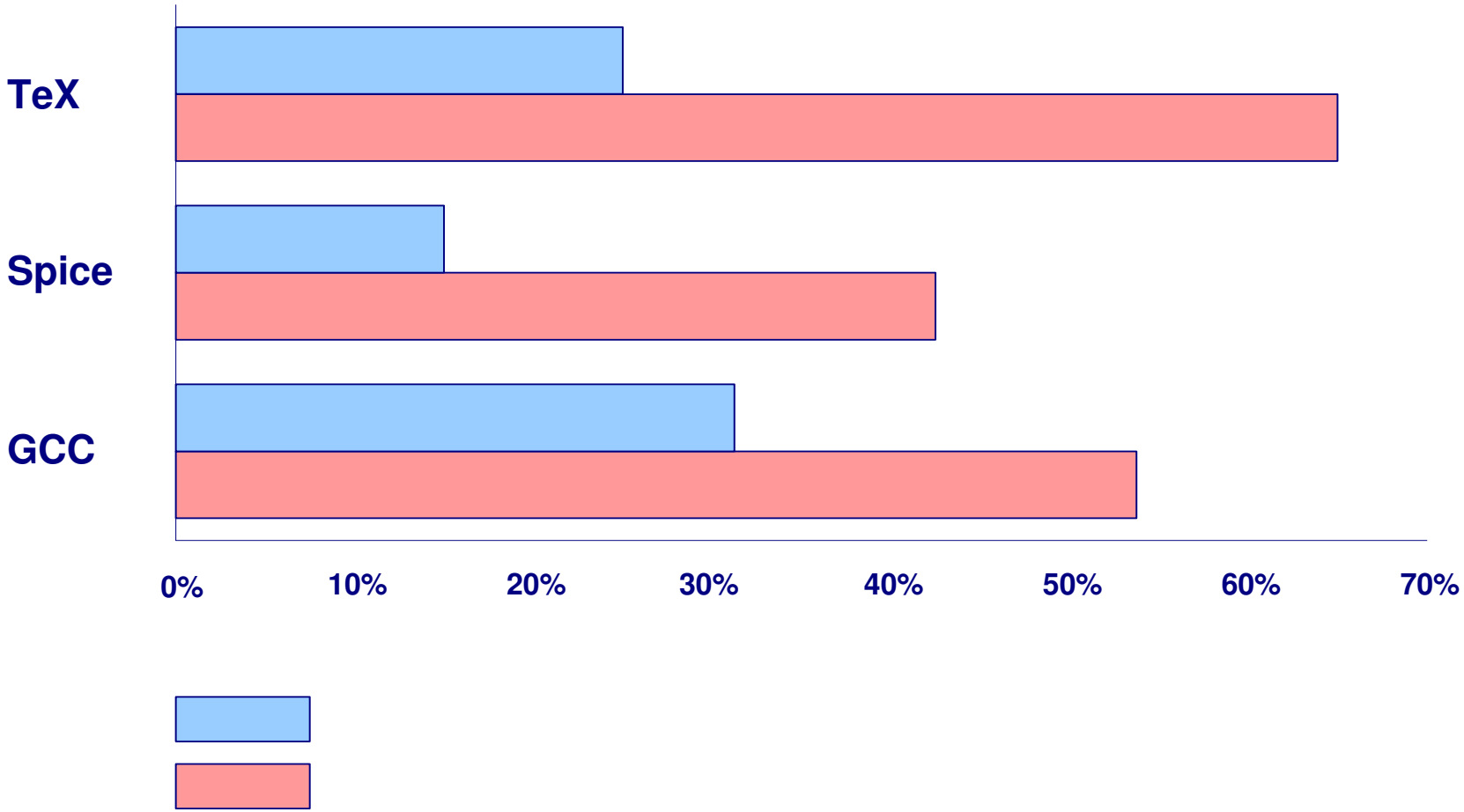


Instrucciones de Carga (II)

- Planificación de la Segmentación:
  - ✓
- Carga retardada:
  - ✓
- Retardo de carga / Hueco de retardo de carga:
  - ✓

Instrucciones de Carga (III)

Porcentaje de cargas que causan detención con la segmentación:



Instrucciones de Carga (IV)

No Dependencia	LW R1, 45(R2)
	ADD R5, R6, R7
	SUB R8, R6, R7
	OR R9, R6, R7
		No hay problema, porque no existe dependencia de R1 en las tres siguientes instrucciones

Dependencia que requiere detención	LW R1, 45(R2)
	ADD R5, R1, R7
	SUB R8, R6, R7
	OR R9, R6, R7
		El Hardware detecta el uso de R1 en la instrucción ADD y detiene tanto a ella como a las siguientes antes de que comienze la etapa EJE de ADD

Dependencia superada por adelantamiento	LW R1, 45(R2)
	ADD R5, R6, R7
	SUB R8, R1, R7
	OR R9, R6, R7
		El Hardware detecta el uso de R1 en la instrucción SUB y adelanta el resultado de la carga a la ALU en el instante en el que comienza la etapa EJE de SUB

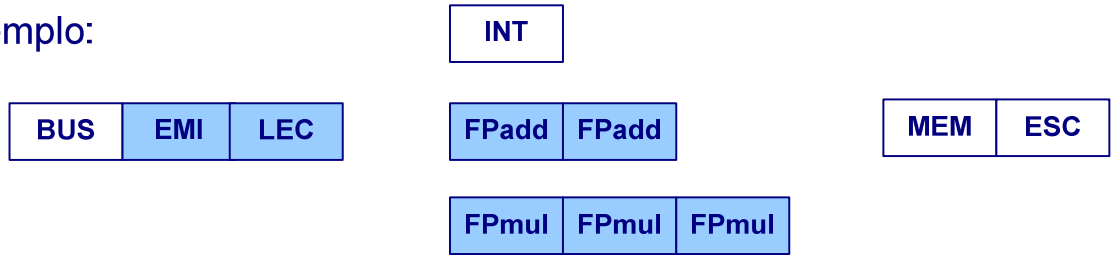
Dependencia con accesos ordenados	LW R1, 45(R2)
	ADD R5, R6, R7
	SUB R8, R6, R7
	OR R9, R1, R7
		No se requiere ninguna acción, porque la lectura de R1 por parte de la instrucción OR tiene lugar en la segunda mitad del ciclo en la etapa D-L, mientras que la escritura de dato cargado en R1 tiene lugar en la primera mitad del ciclo

Resolución mediante Hardware (IV)

Ejecución fuera de orden con Marcador *(Scoreboarding)*

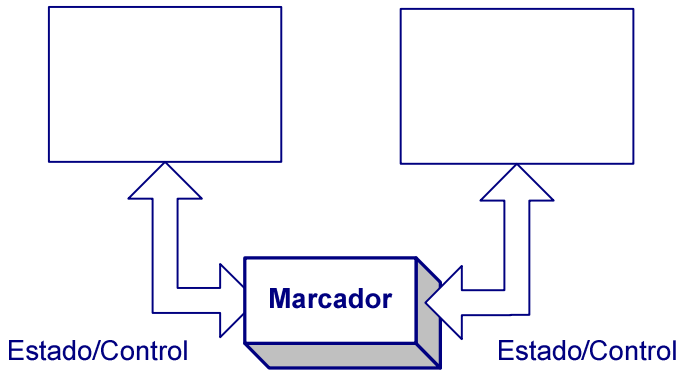
- solo se detienen las instrucciones que dependen, y no las posteriores
- la etapa D-L se divide en Emisión (EMI) y Lectura de Registros (LEC)
- necesidad de múltiples unidades funcionales de ejecución

Ejemplo:




```
LW    F4, 0(R2)
ADDF  F3, F4, F6
MULF  F7, F1, F8
```

- las instrucciones pueden emitirse y/o completarse fuera de orden
- aparecen problemas de dependencias EDL y EDE
- el marcador lleva cuenta del estado de todas las instrucciones, unidades funcionales y registros, y controla la emisión y detención de instrucciones



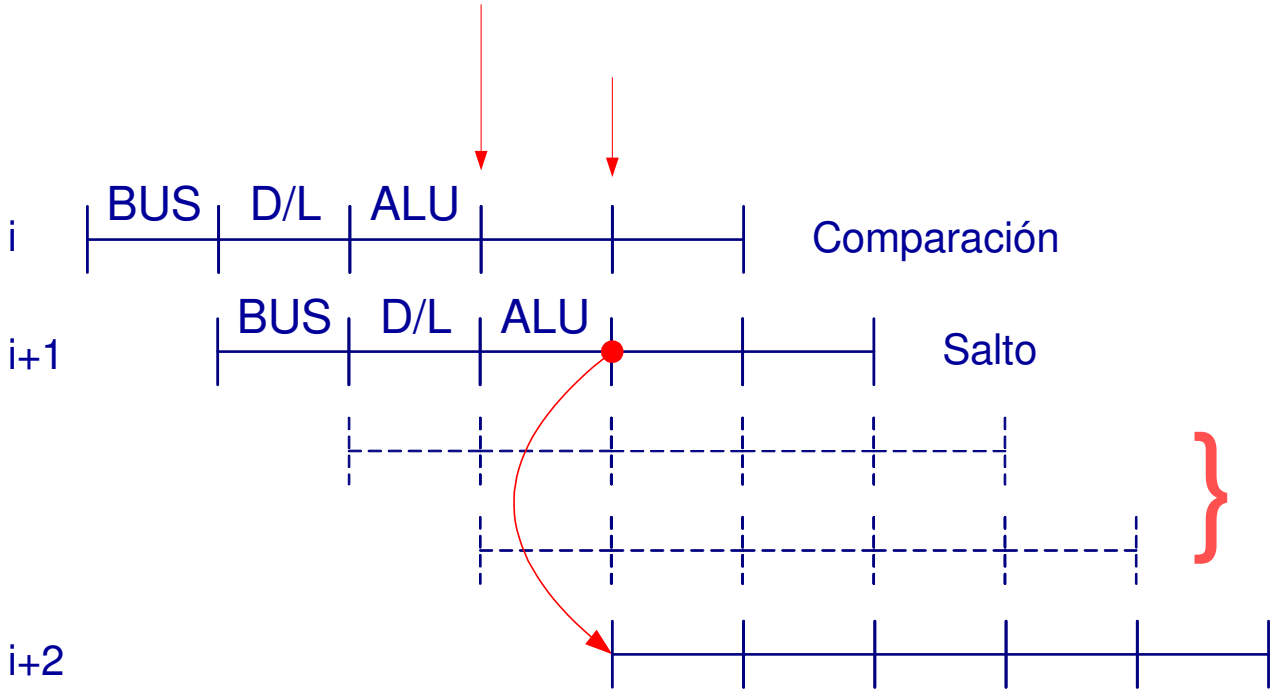
Riesgos de Control

- Problemas con el Control de Flujo
- Saltos en el MIPS R3000
- Métodos de Resolución
  - ✓ Reducción de la Latencia
  - ✓ Predicción de Salto
  - ✓ Salto Retardado

Problemas con el Control de Flujo

- Provocados por las rupturas en la ejecución secuencial de los programas:
  - ✓
  - ✓
- Se traducen en una indeterminación a la hora de emitir las siguientes Instrucciones a ejecutar mientras no se conoce el nuevo punto de ejecución (el resultado del salto)

Saltos en el MIPS R3000

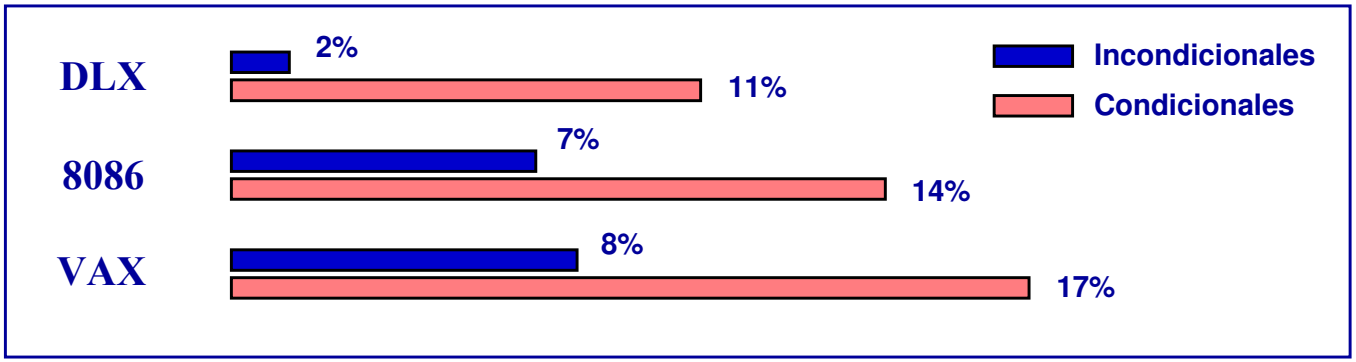


- ✓ **La dirección efectiva del salto no se conoce hasta después de dos ciclos**

  (

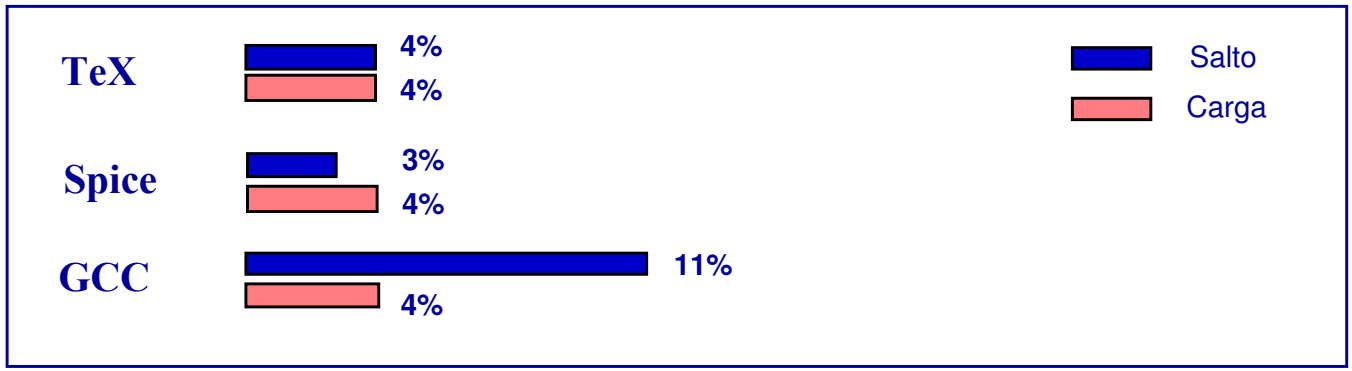
- ✓ **Las instrucciones de salto representan entre el 15 y el 30% del total de instrucciones**

Instrucciones de Control



Frecuencia de instrucciones de control en varias arquitecturas (53% de saltos condicionales efectivos en DLX)

Retardos de Salto



Porcentaje de los ciclos de reloj en los retardos (de carga y salto) frente a los consumidos por ejecución de Instrucciones

Métodos de Resolución

- Parada del cauce (
- Reducción de la Latencia
  - ✓ Anticipar las operaciones que se puedan dentro del proceso de ejecución de las instrucciones de salto
- Efectuar la búsqueda de Instrucciones por las dos ramas
  - ✓ Ahorro del ciclo de búsqueda de instrucción una vez resuelto el salto
- Predicción de Salto
  - ✓ Estática
  - ✓ Dinámica
- Salto Retardado
  - ✓ Hacer visible la latencia al nivel de lenguaje máquina

  (

Reducción de la Latencia

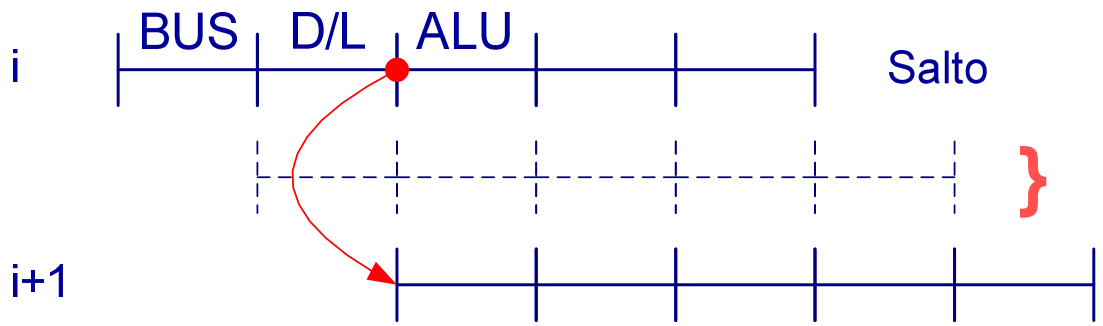
- Averiguar antes si el salto es o no efectivo
- Calcular antes la dirección de destino del salto

Ejemplo:

- ✓ Instrucción de salto que chequea únicamente si el valor de un registro es 0

  ⇒
- ✓ Adición de un sumador a la etapa D/L

  ⇒
- ✓ **Resultado:** reducción de la latencia (HUECO DE RETARDO DE SALTO) a



Predicción de Salto

- Tipos de predicción:
  - ✓ Estática (
  - ✓ Dinámica (
- Predicción SALTO NO EFECTIVO
  - ✓ Se continúa la búsqueda de instrucciones normalmente
  - ✓ Si el salto NO es finalmente efectivo ⇒
  - ✓ Si el salto es finalmente efectivo ⇒
- Predicción SALTO EFECTIVO
  - ✓ Se calcula la dirección destino y se comienza la búsqueda a partir de ella
  - ✓ Si el salto es finalmente efectivo ⇒
  - ✓ Si el salto NO es finalmente efectivo ⇒
  - ✓ Inútil en nuestro cauce (
  - ✓ De aplicación en caso de tener condiciones de salto complejas

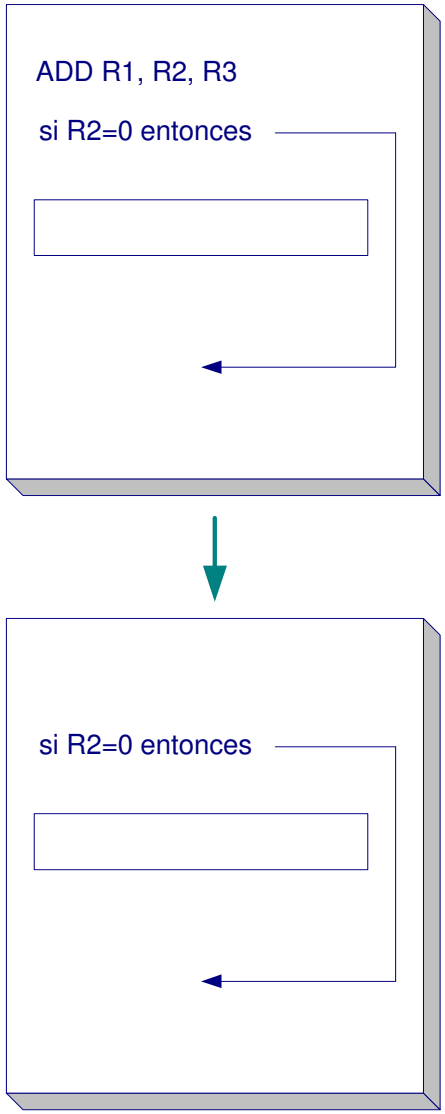
Predicción SALTO NO EFECTIVO

Ciclo de Reloj	1	2	3	4	5	6	7	8	9 ...
Salto NO Efec.	BUS	D-L	EJE	MEM	ESC				
Instrucción i+1		BUS	D-L	EJE	MEM	ESC			
Instrucción i+2			BUS	D-L	EJE	MEM	ESC		
Instrucción i+3				BUS	D-L	EJE	MEM	ESC	
Instrucción i+4					BUS	D-L	EJE	MEM	ESC

Ciclo de Reloj	1	2	3	4	5	6	7	8	9 ...
Salto Efectivo	BUS	D-L	EJE	MEM	ESC				
Instrucción i+1		BUS	BUS	D-L	EJE	MEM	ESC		
Instrucción i+2				BUS	D-L	EJE	MEM	ESC	
Instrucción i+3					BUS	D-L	EJE	MEM	ESC
Instrucción i+4						BUS	D-L	EJE	MEM	ESC

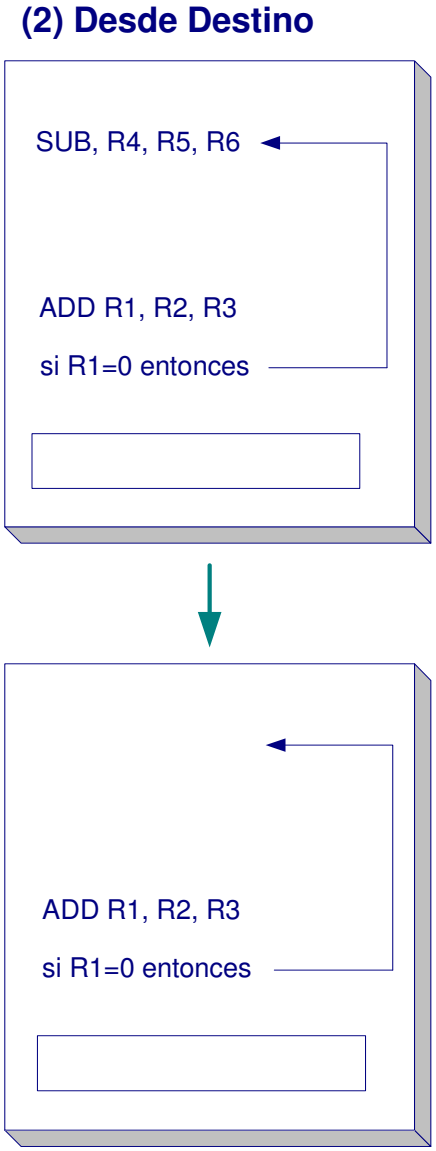
Salto Retardado

(1) Desde Antes



- ✓ **El hueco se rellena con instrucciones anteriores a la de salto**
- ✓ **Los saltos NO deben depender de las instrucciones replanificadas**
- ✓

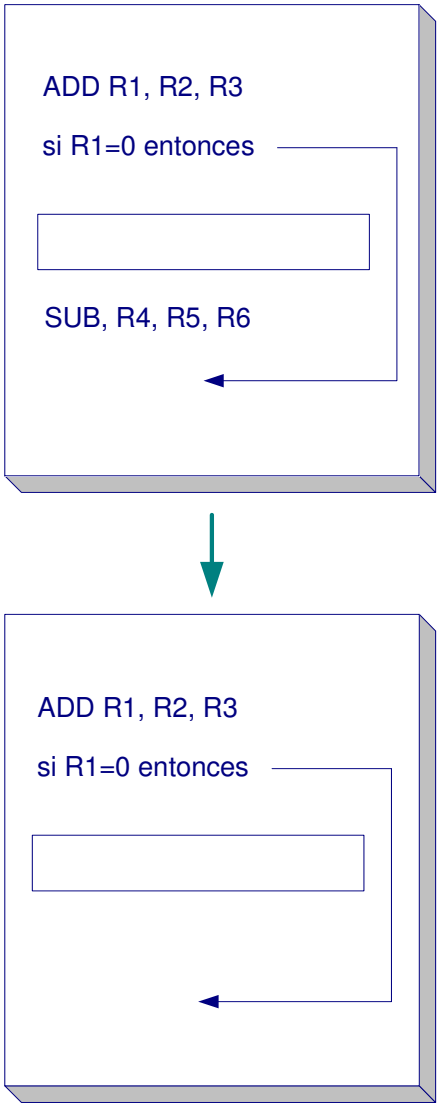
Salto Retardado



- ✓ **El hueco se rellena con instrucciones a partir de la dirección de salto**
- ✓ **La ejecución de las instrucciones replanificadas cuando NO es efectivo el salto solo debe implicar trabajo desperdiciado, NUNCA resultados incorrectos**
- ✓ **Puede ser necesario duplicar instrucciones**
- ✓
- ✓

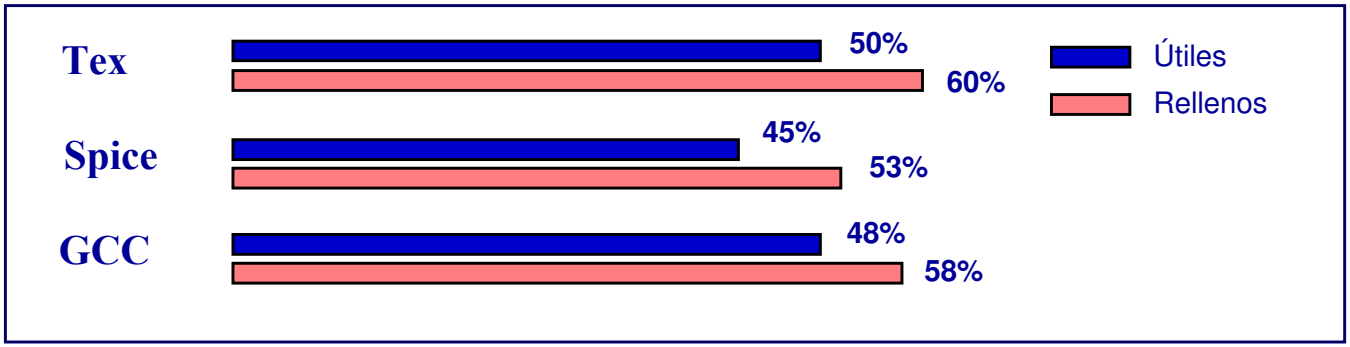
Salto Retardado

(3) Desde Siguiente



- ✓ **El hueco se rellena con instrucciones que siguen a la de salto**
- ✓ **La ejecución de las instrucciones replanificadas cuando es efectivo el salto solo debe implicar trabajo desperdiciado, NUNCA resultados incorrectos**
- ✓

Salto Retardado



Frecuencia con la que se rellenan los huecos de retardo de salto y frecuencia con la que son útiles para la computación

Problema de Riesgos de Control

A partir de las gráficas y datos relativos a riesgos de control para la arquitectura del procesador segmentado visto en clase, y considerando un CPI ideal sin riesgos de 1, se pide:

1) ¿Cuál es la frecuencia total de instrucciones de control?
2) ¿Qué porcentaje de las instrucciones de control cambia realmente el contador de programa?
3) ¿Qué porcentaje medio de saltos retardados hace realmente trabajo útil?
4) Rellenar la tabla siguiente relativa a los costes asociados a varios esquemas de planificación de salto:

Esquema de planificación	Penalización de salto	CPI efectivo o real	Aceleración de la segmentación	Aceleración respecto a la estrategia de detención
1 - Detención	2			
2 - Predicción “salto efectivo”	1			
3 - Predicción “salto no efectivo”	1			
4 - Salto retardado	1			

Solución:

1) (11% de saltos condicionales + 2% de saltos incondicionales)

2)

53% de saltos condicionales efectivos ➔ total de saltos efectivos = (11 * 0.53 + 2)/13 = 0.6

3)

frecuencia media de saltos retardados que realizan trabajo util = (0.50 + 0.45 + 0.48) / 3 = 0.48

4) Aceleración = CPI Ideal * Prof. Seg. / (CPI Ideal + frecuencia saltos * penalización salto)
= 1 x 5 / CPI real
= 5 / CPI real

Detención ➔ CPI real = 1 + 0.13*2 = 1.26 ➔ Aceleración = 5/1.26 =
Pred. efectivo ➔ CPI real = 1 + 0.13*1 = 1.13 ➔ Aceleración =
Pred. no efectivo ➔ CPI real = 1 + (0.13*0.60)*1 = 1.08 ➔ Aceleración =

Salto retardado:

Frecuencia media de saltos retardados que provocan penalización =
1 - frecuencia media de saltos retardados que realizan trabajo util = 1 - 0.48 = 0.52

➔ CPI real = 1 + (0.13*0.52)*1 = 1.07 ➔ Aceleración =

Aceleración de la estrategia n respecto a la 1 = Aceleración de n / Aceleración de 1:

➔ Aceleración 1/1 = 3.97 / 3.97 =
Aceleración 2/1 = 4.43 / 3.97 =
Aceleración 3/1 =
Aceleración 4/1 =

Segmentación Avanzada

- Técnicas que se apoyan en la segmentación
- Procesadores Superescalares
  - ✓ Familias
  - ✓ Tipos
  - ✓ Ejemplos

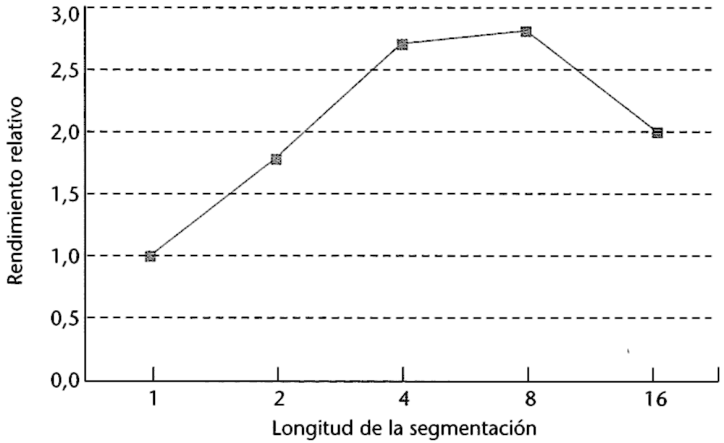
Técnicas que se apoyan en la segmentación

Tiempo $_{CPU}$ = NI x CPI x T $\Rightarrow$

$f_{reloj}\uparrow \Rightarrow$

CPI $\downarrow \Rightarrow$

- Procesadores Supersegmentados
  - ✓ aumento del número de etapas
    $\Rightarrow$

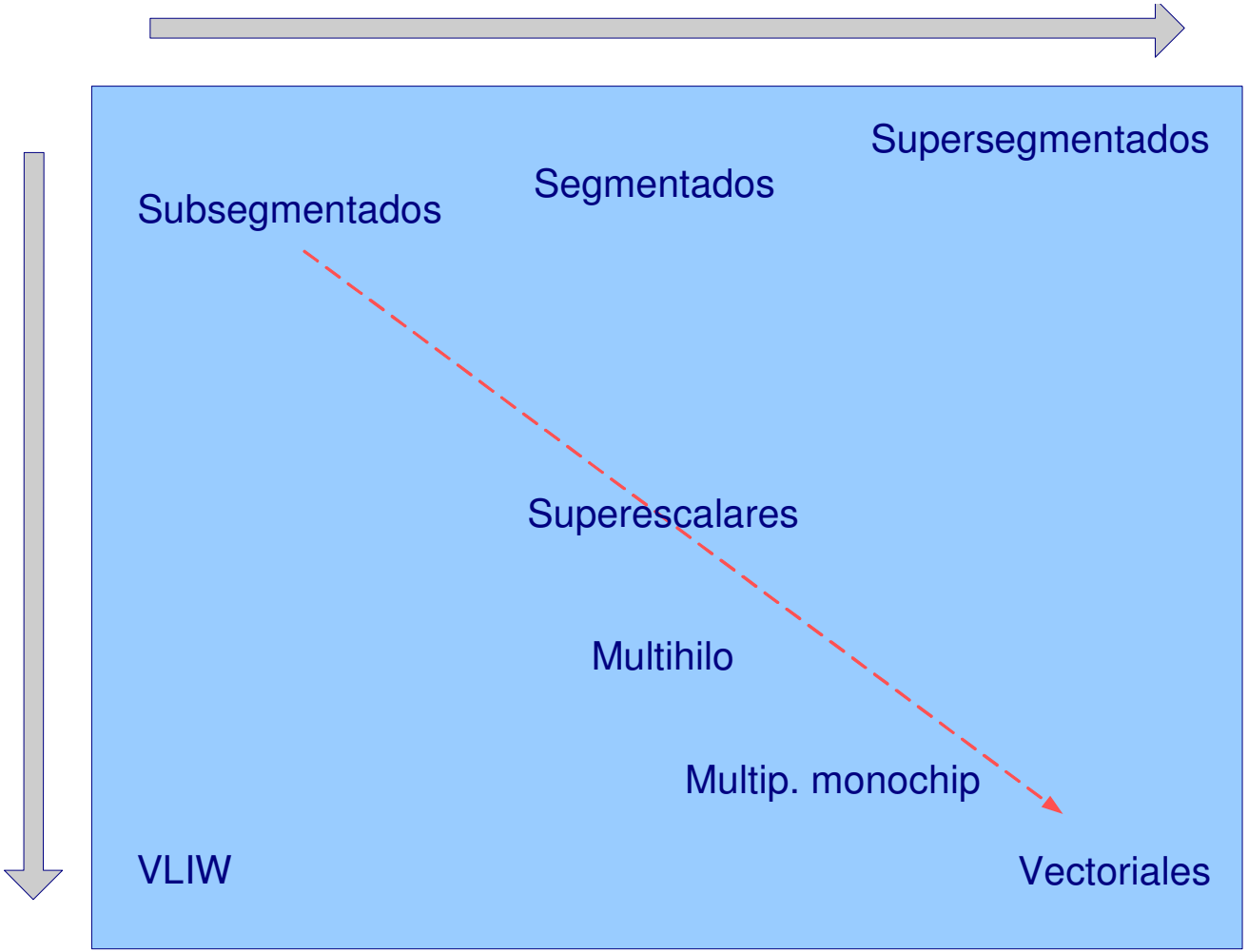


- Procesadores Superescalares
  - ✓ se emite más de una instrucción por ciclo de reloj $\Rightarrow$
  - ✓ las instrucciones emitidas a la vez deberían ser independientes
  - ✓ el hardware se encarga de la planificación (dinámica) de instrucciones

Técnicas que se apoyan en la segmentación

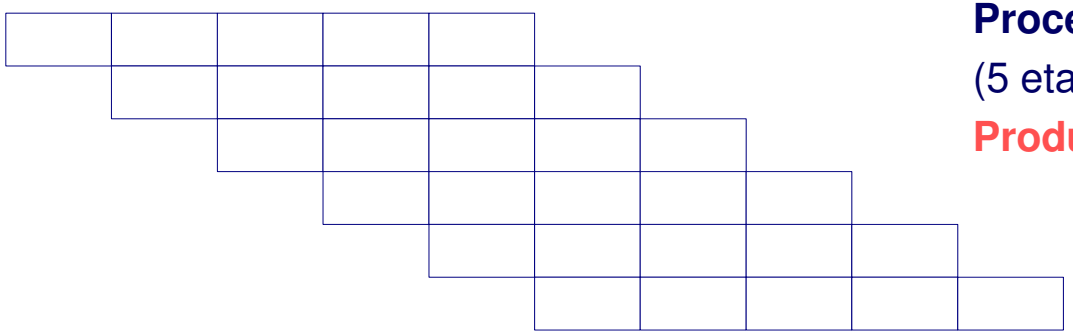
- Procesadores VLIW *(Very Long Instruction Word)*
  - ✓ varias operaciones independientes en instrucciones muy largas ⇒
  - ✓ el compilador se encarga de la planificación (estática) de instrucciones
  - ✓ frecuencias de trabajo bajas para limitar el AB de memoria ⇒
  - ✓ Ejemplo:
- Procesadores Vectoriales
  - ✓ mejoran el rendimiento cuando se procesan vectores o matrices
  - ✓ combinan las ventajas de Supersegmentadas y VLIW ⇒
- Procesadores Multihilo
  - ✓ ejecución concurrente de varios hilos en los cauces ⇒
  - ✓ mayor aprovechamiento de los recursos del cauce
  - ✓ Ejemplo:
- Multiprocesadores monochip
  - ✓ varios procesadores en un mismo chip ⇒
  - ✓ procesadores más simples ⇒
  - ✓ ejecución en paralelo de los hilos de una aplicación
  - ✓ Ejemplo:

Técnicas que se apoyan en la segmentación



P. Superescalares frente a Supersegmentados

t

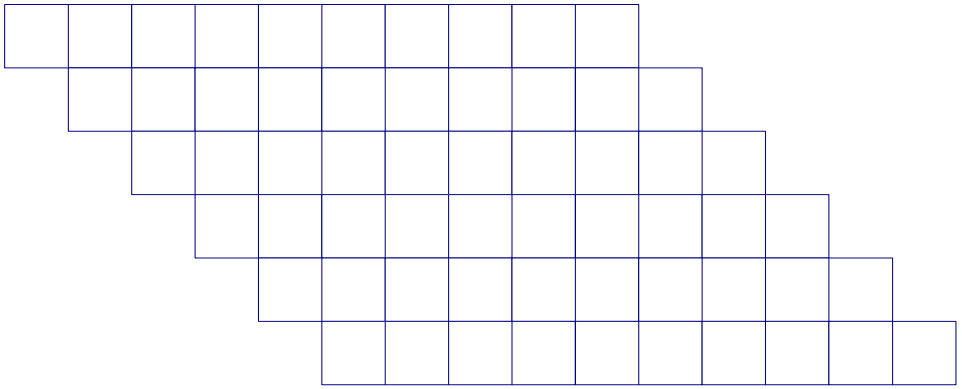


Procesador Segmentado

(5 etapas)

Productividad:

t/2

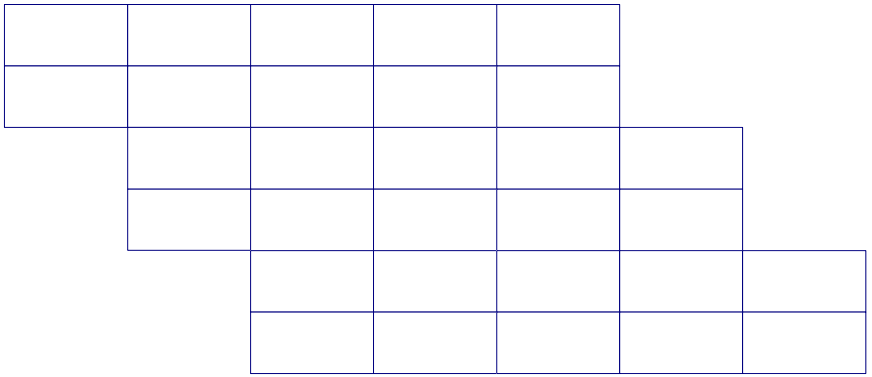


Procesador Supersegmentado

(10 etapas)

Productividad:

t



Procesador Superescalar

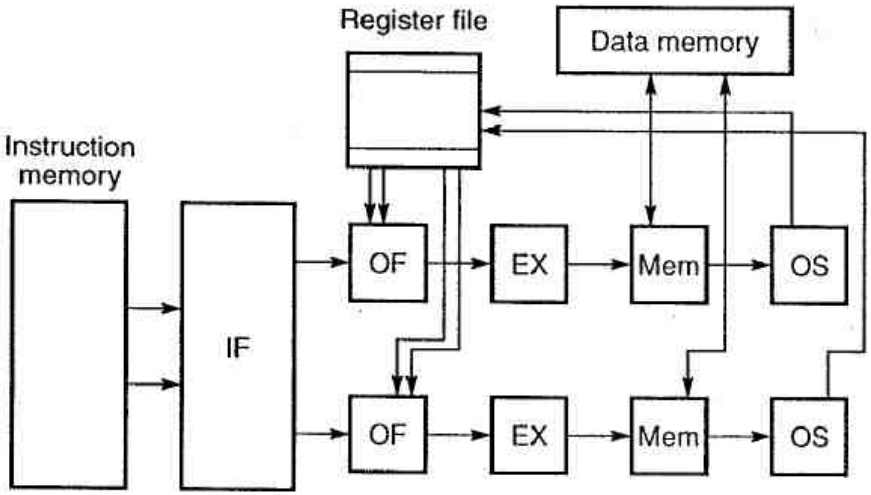
(orden 2 y 5 etapas en cada cauce)

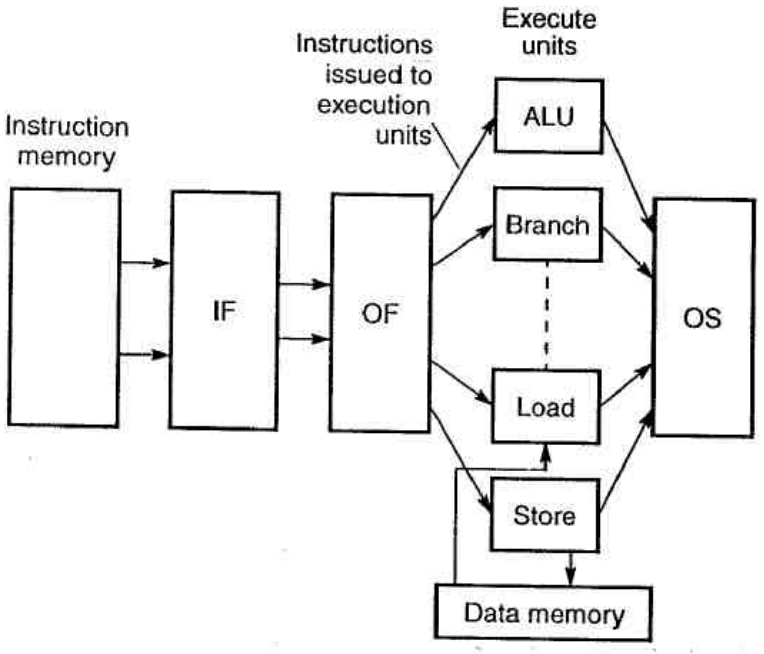
Productividad:

Procesadores Superescalares

- Características
  - ✓ ejecución de varias instrucciones en paralelo
  - ✓ emisión de varias instrucciones por ciclo
  - ✓ productividad de mas de 1 instrucción por ciclo ⇒
  - ✓
- Requisitos
  - ✓ existencia de paralelismo en el código a nivel de instrucciones
  - ✓ existencia de paralelismo en el hardware (para poder explotar el anterior)
  - ⇒
- Limitaciones
  - ✓ RIESGOS: estructurales / dependencia de datos / control
  - ✓ mayor concurrencia ⇒ conflictos ↑ (ganancia real mas lejos de la ideal)
  - ✓ mayor influencia de los huecos de retardo (carga / salto)
  - ⇒

Tipos de procesadores superescalares



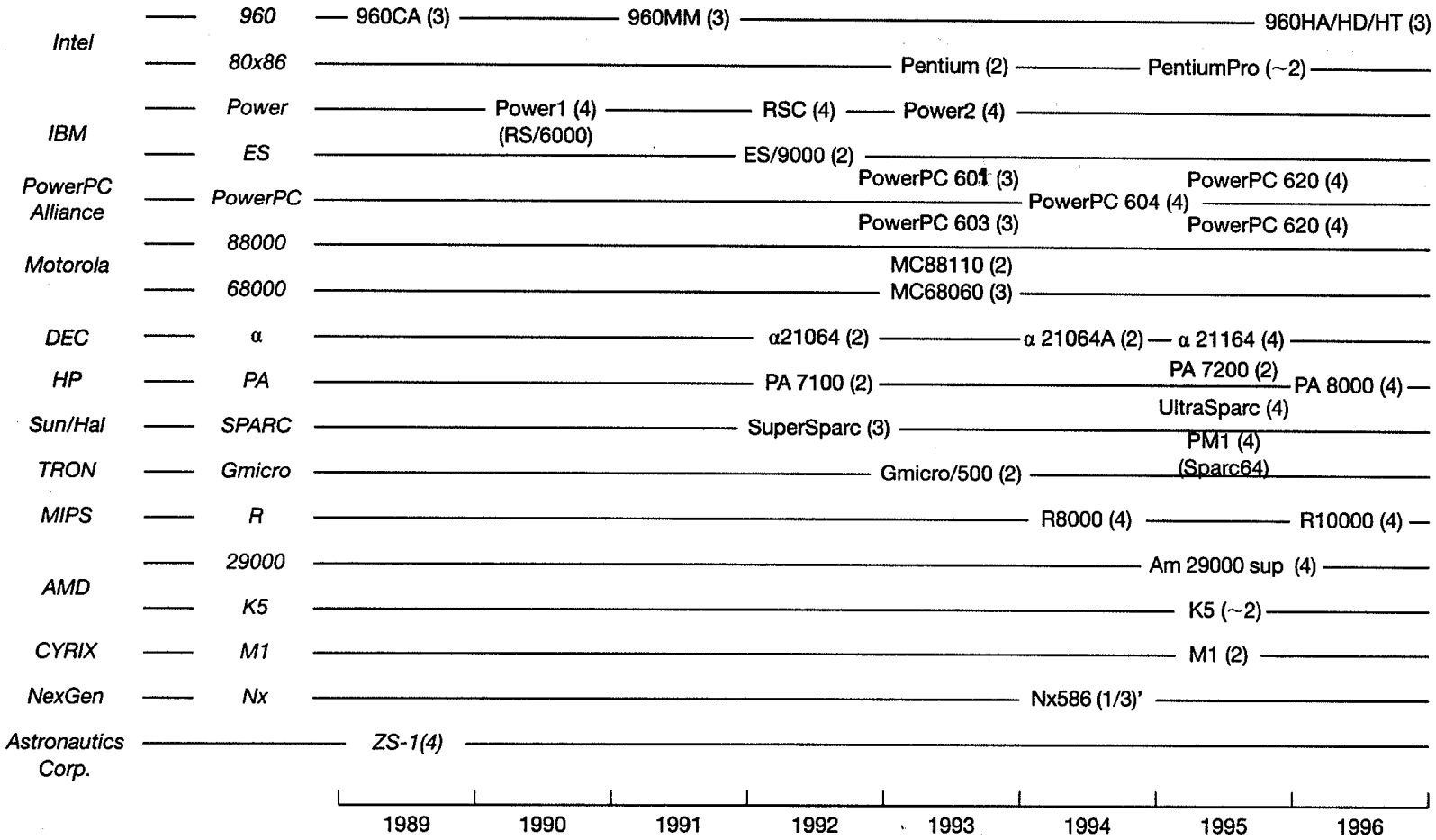


IF =

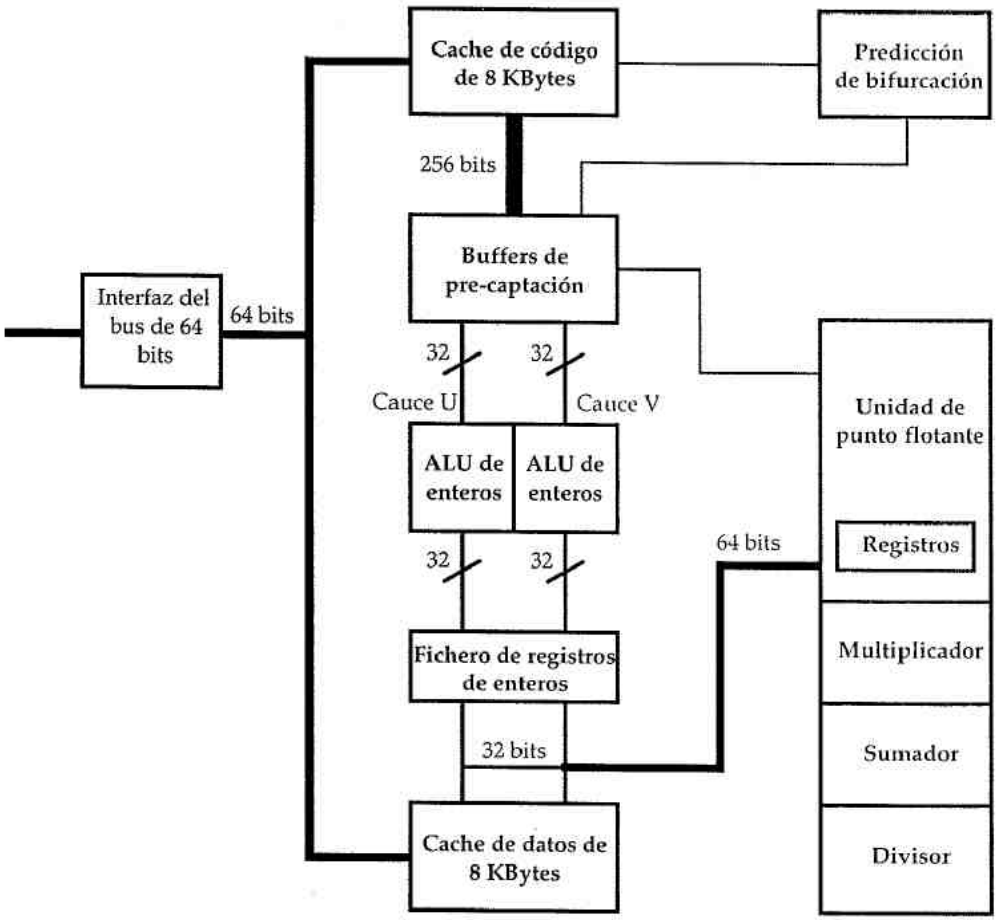
OF =

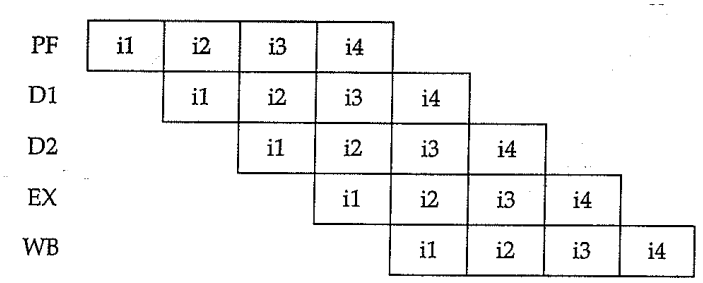
OS =

Familias de procesadores superescalares

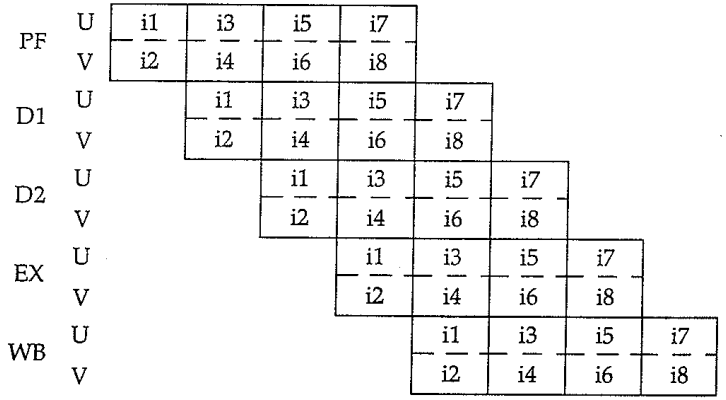


Ejemplos de procesadores superescalares





(a) Cauce del 80486



(b) Cauce del Pentium

PENTIUM (

PF =

D1 / D2 =

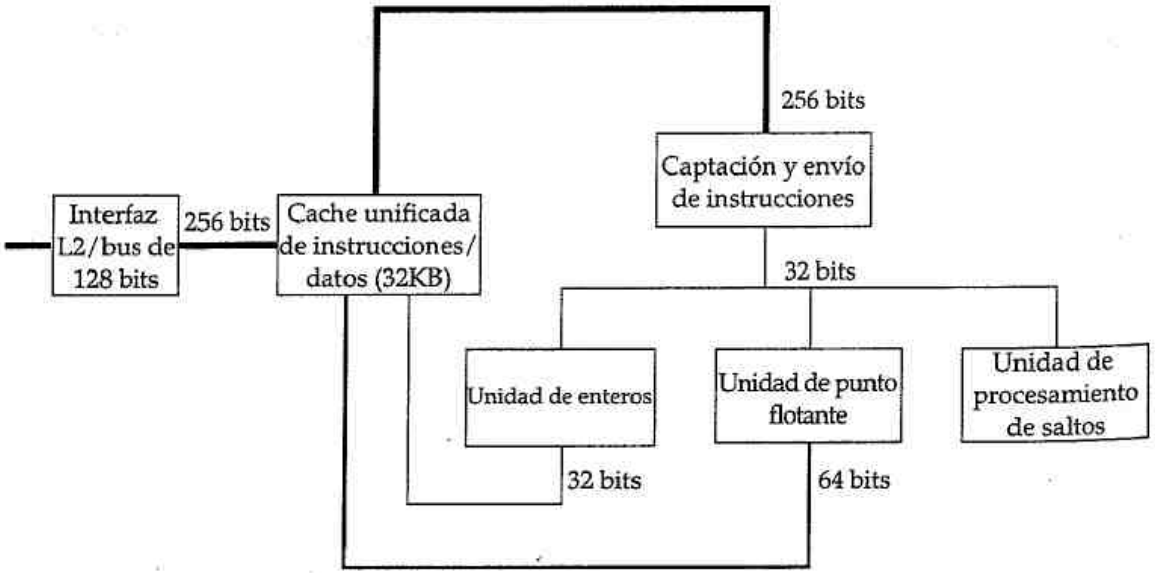
EX =

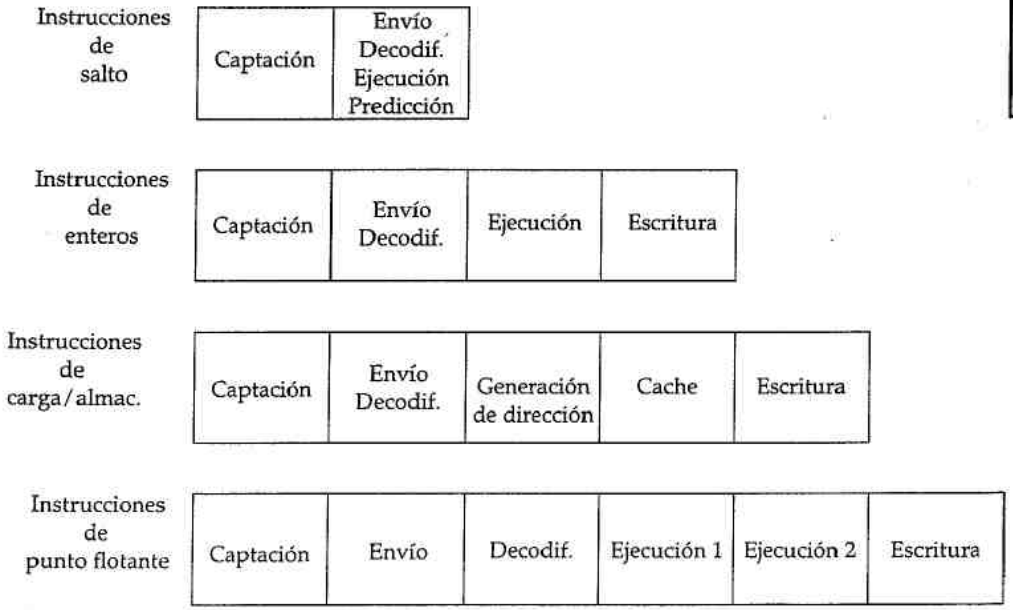
WB =

Ejemplos de procesadores superescalares

PowerPC 601

(



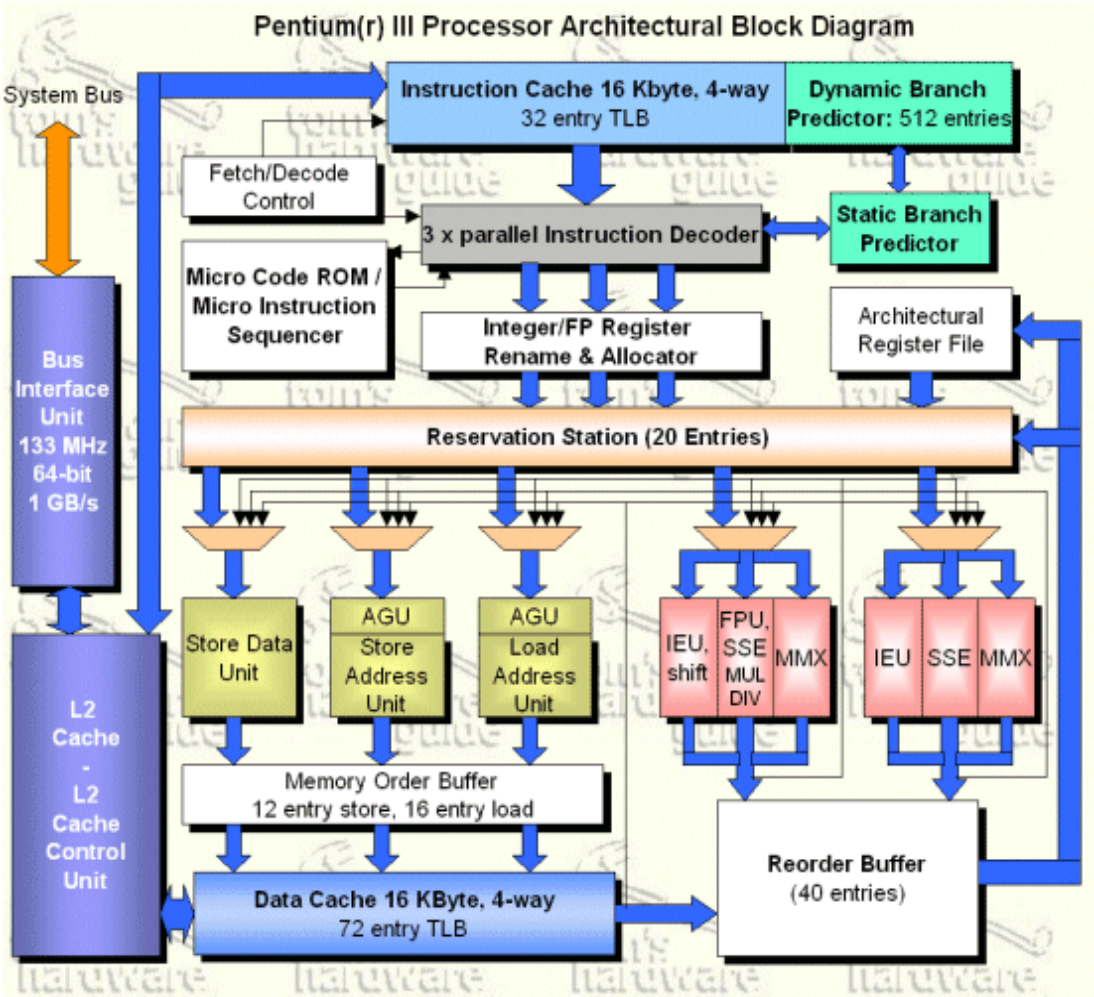


Ejemplos de procesadores superescalares

Processor	System ship	Maximum current CR (MHz)	Power (W)	Transistors (M)	Window size	Rename registers (int/FP)	Issue rate: maximum/ memory/ integer/FP/ branch	Branch-predict buffer	Pipe stages (int/load)
MIPS R14000	2000	400	25	7	48	32/32	4/1/2/2/1	2K × 2	6
UltraSPARC III	2001	900	65	29	N.A.	None	4/1/4/3/1	16K × 2	14/15
Pentium III	2000	1000	30	24	40	Total: 40	3/2/2/1/1	512 entries	12/14
Pentium 4	2001	1700	64	42	126	Total: 128	3/2/3/2/1	4K × 2	22/24
HP PA 8600	2001	552	60	130	56	Total: 56	4/2/2/2/1	2K × 2	7/9
Alpha 21264B	2001	833	75	15	80	41/41	4/2/4/2/1	multilevel (see p. 207)	7/9
PowerPC 7400 (G4)	2000	450	5	7	5	6/6	3/1/2/1/1	512 × 2	4/5
AMD Athlon	2001	1330	76	37	72	36/36	3/2/3/3/1	4K × 9	9/11
IBM Power3-II	2000	450	36	23	32	16/24	4/2/2/2/2	2K × 2	7/8

Ejemplos de procesadores superescalares

Pentium III (1999)
3 Inst. emitidas / ciclo
11 Unidades Funcionales
2 ALUs + 3 FPUs



Ejemplos de procesadores superescalares

AMD K7 Athlon (1999)
3 Inst. emitidas / ciclo
9 Unidades Funcionales
3 ALUs + 3 FPUs