TEMA 4: Segmentación de cauce

Objetivos del capítulo

- Conocer los principios básicos de la Segmentación de cauce
- Riesgos que pueden degradarlas prestaciones y las formas de mitigar sus efectos
- Implicaciones software y hardware en la segmentación de cauce
- Influencia en el diseño del repertorio de instrucciones
- Conocer los principios básicos de procesadores superescalares

Material complementario en:

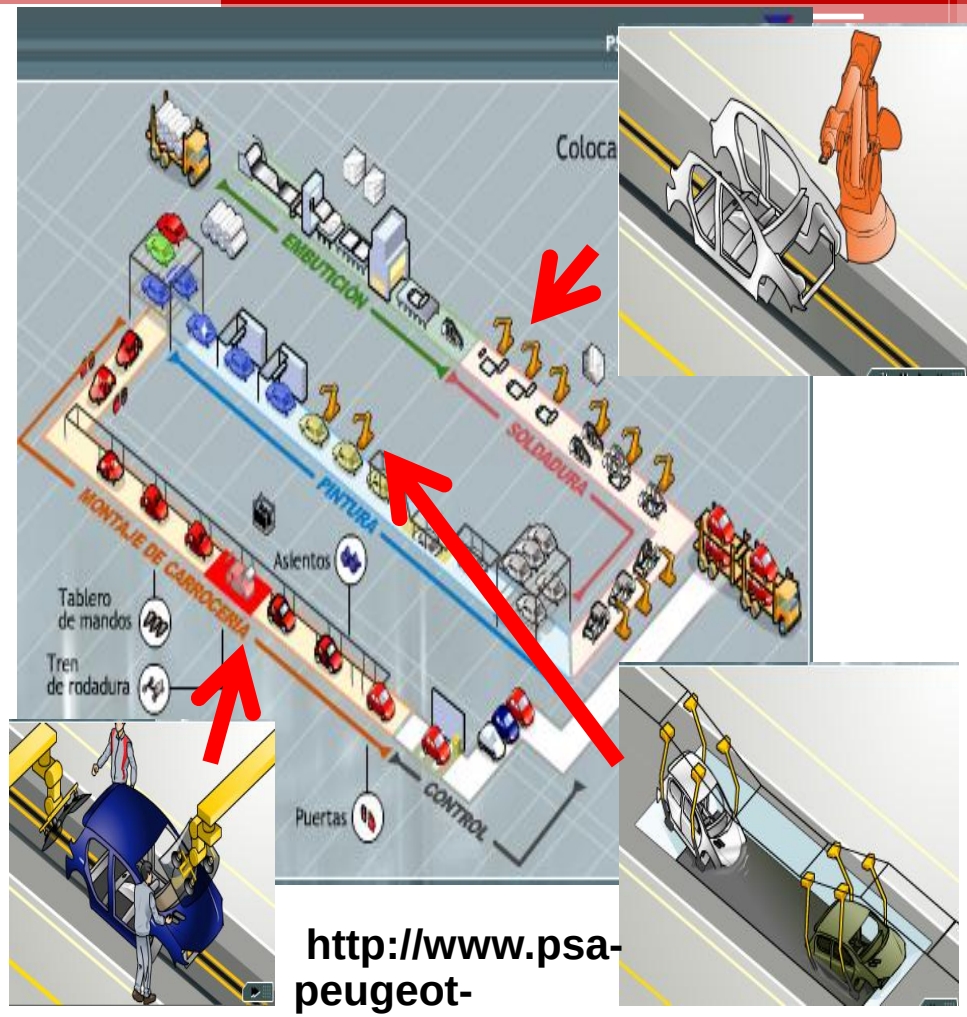
Organización de Computadores. C. Hamacher *et al*. 5ª Edición. McGraw Hill

Conceptos básicos Segmentación de Cauce

Aumentar prestaciones:

- ✓ Tecnología más rápida
- ✓ Reorganización del hardware : **Segmentación de cauce**
- ✓ Duplicación de hardware: **Procesador superescalar**

Fabricación en cadena: dividir una tarea larga y/o difícil en otras más cortas y fáciles de realizar



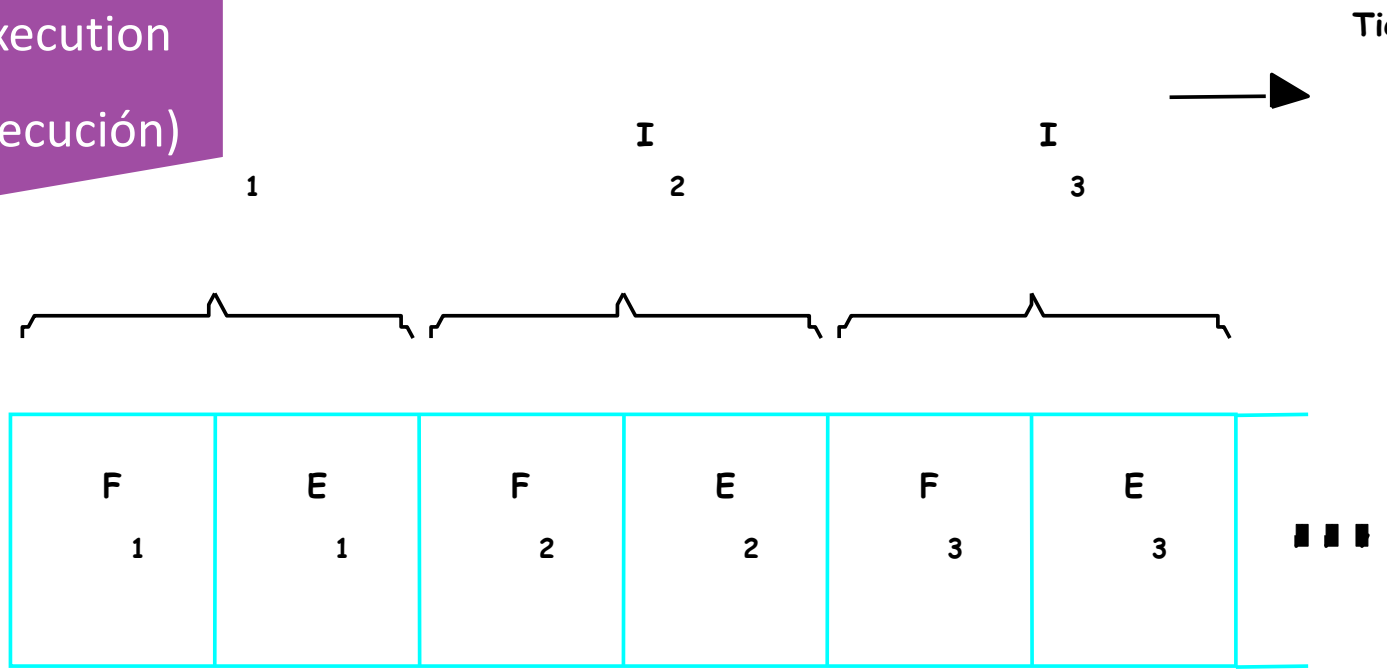
http://www.psa-peugeot-citroen.com/es/psa_grupo/visite/index.html

Segmentación de cauce

Ejemplo: un procesador segmentado en el que cada instrucción está compuesta de dos fases, una de captación (Fetch), y otra de ejecución (Execution)

Fetch (captación)

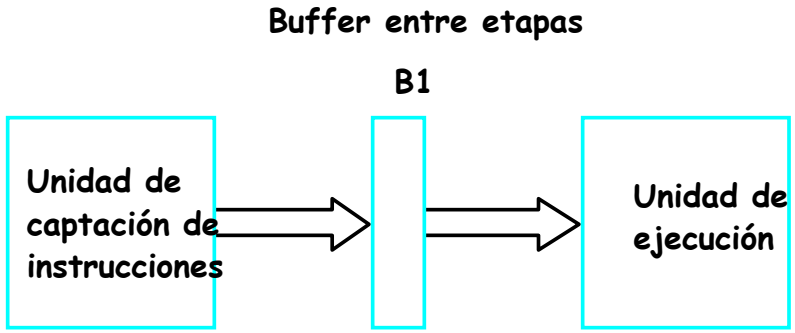
Execution (ejecución)



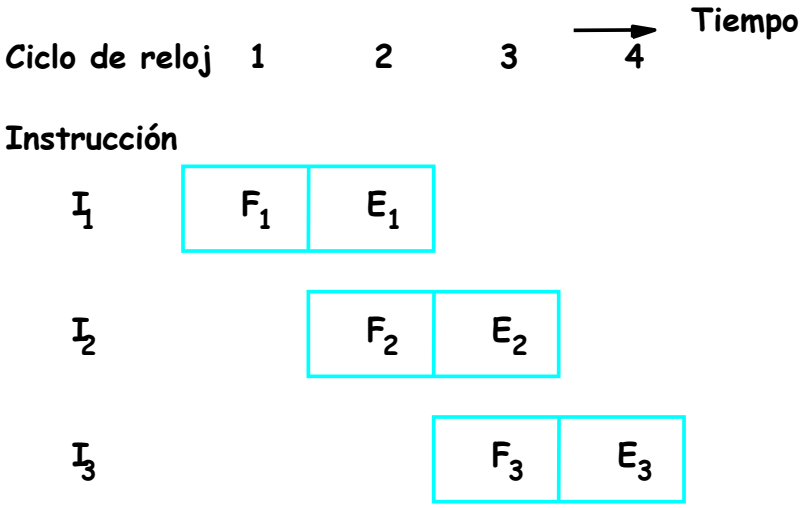
(a) Ejecución secuencial

Segmentación de cauce

Ejemplo: un procesador segmentado en el que cada instrucción está compuesta de dos fases, una de captación (Fetch), y otra de ejecución (Execution)

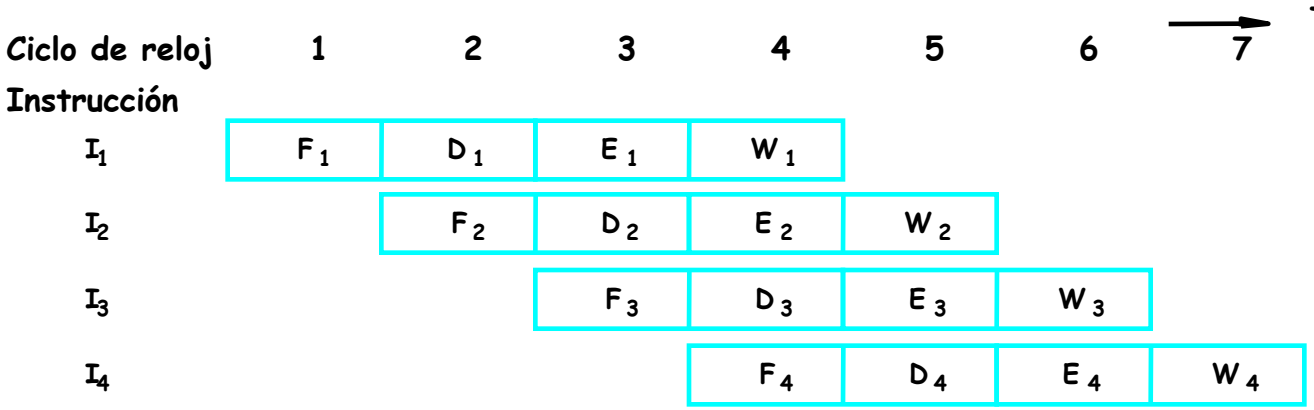


(b) Organización del hardware

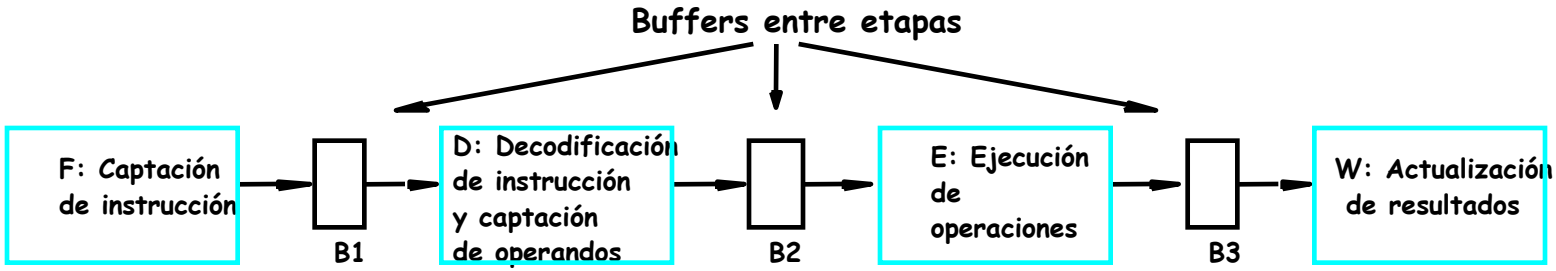


(c) Ejecución segmentada

Segmentación de cauce



(a) Ejecución de instrucciones en cuatro etapas



(b) Organización del hardware

Figura 8.2. Cauce de 4 etapas

Segmentación de cauce: memoria caché

- Cada etapa del cauce debe completarse en un ciclo de reloj
- La fase de captación debe acceder a memoria que es unas 10 veces más lento
- La caché nos permite captar instrucciones empleando un único ciclo de reloj
- El periodo de reloj se escoge de acuerdo a la etapa más larga del cauce

TEMA 4: Segmentación de cauce

Objetivos del capítulo

- Conocer los principios básicos de la Segmentación de cauce
- Riesgos que pueden degradarlas prestaciones y las formas de mitigar sus efectos
- Implicaciones software y hardware en la segmentación de cauce
- Influencia en el diseño del repertorio de instrucciones
- Conocer los principios básicos de procesadores superescalares

Material complementario en:

Organización de Computadores. C. Hamacher *et al*. 5ª Edición. McGraw Hill

Segmentación de cauce: prestaciones

- El incremento del prestaciones es proporcional al número de etapas
- Existen numerosos problemas que evitan que una etapa se complete en un ciclo
- **Riesgo**: cualquier condición que ocasione un atasco
- Tipos de riesgos:
  - ***Datos****: los operandos no están disponibles (ej: multiplicación)*
  - ***Instrucciones/control****: la instrucción no está disponible (ej: falló de caché)*
  - ***Estructurales****: dos instrucciones necesitan un mismo recurso (ej: captación+actualización)*

Segmentación de cauce: prestaciones

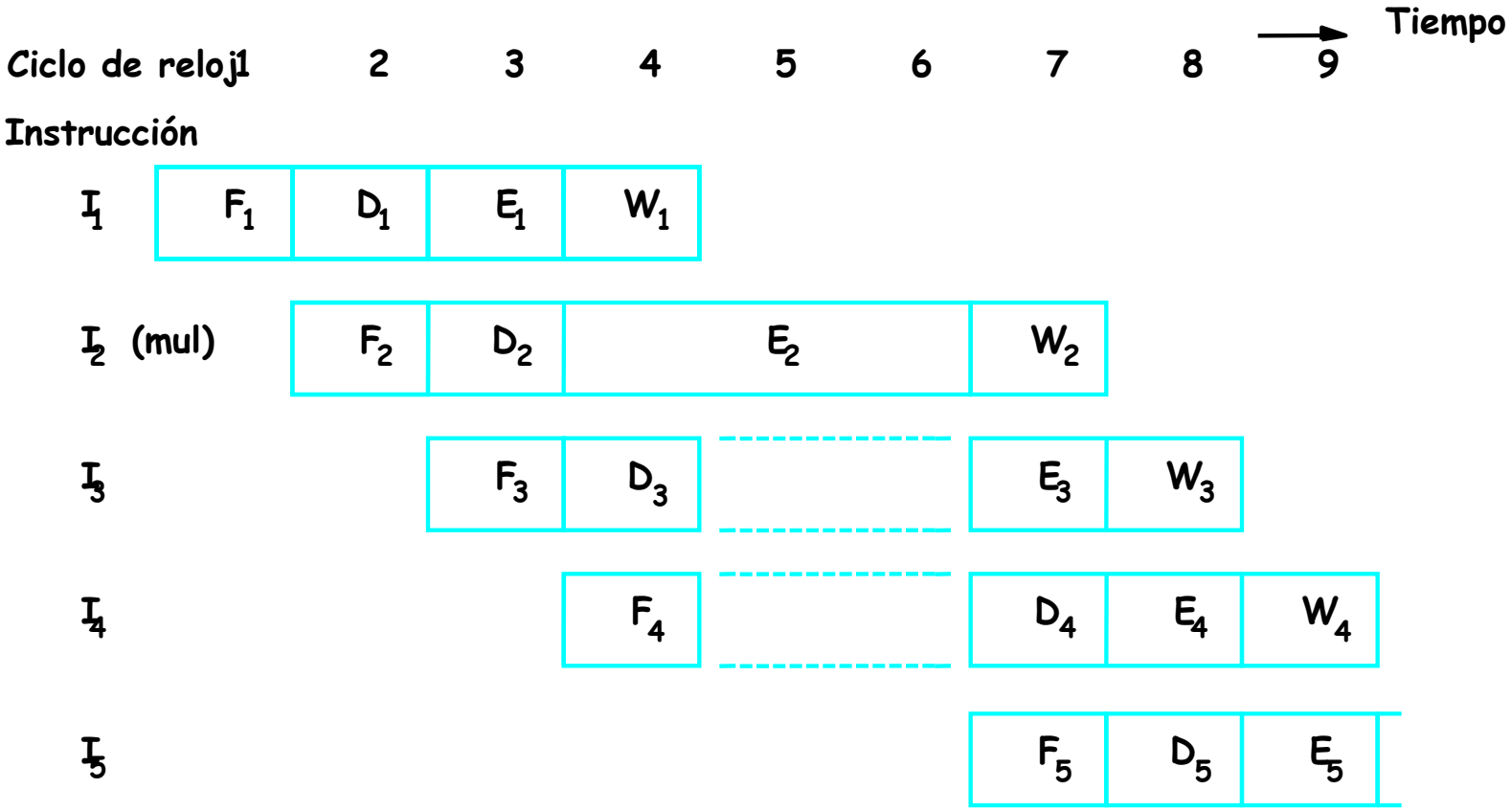
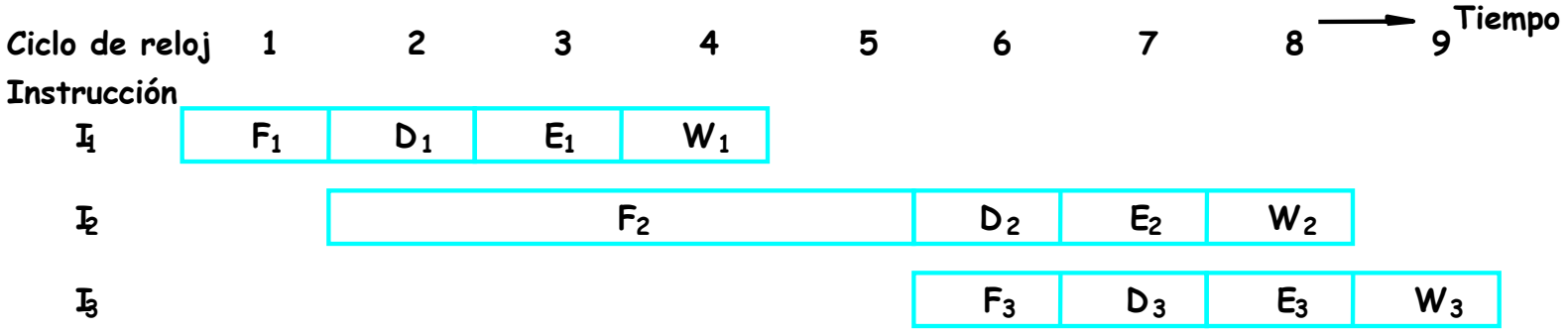


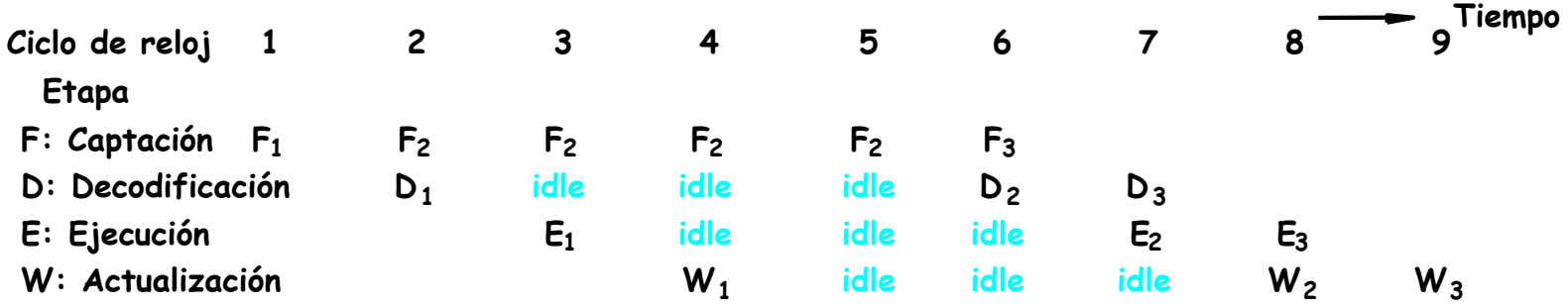
Figura 8.3. Efecto de la ejecución de una operación con más de un ciclo de reloj

RIESGO DE DATOS

Segmentación de cauce: prestaciones



(a) Etapas de la ejecución de una instrucción en ciclos de reloj sucesivos



(b) Funciones realizadas por cada etapa del procesador en ciclos de reloj sucesivos

Figura 8.4. Atasco en el cauce causado por un fallo de caché en F_2

RIESGO DE CONTROL

Segmentación de cauce: prestaciones

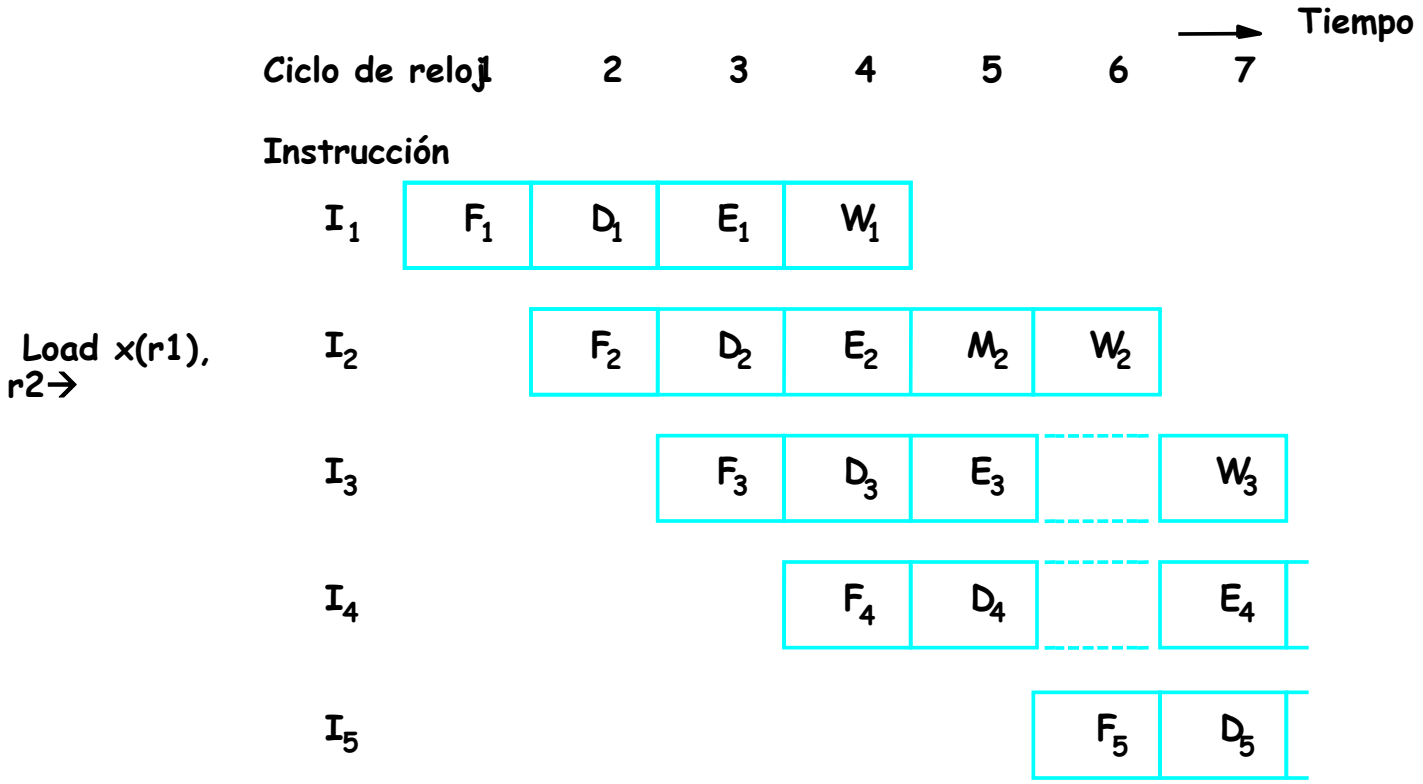


Figura 8.5. Efecto de una instrucción de carga en la temporización del cauce

RIESGO ESTRUCTURAL

Riesgos de datos

- Definición: situación en la que el cauce sufre un atasco debido a un retardo en la llegada de datos
- Dependencia entre instrucciones:
  - *las instrucciones independientes pueden ejecutarse concurrentemente*
  - *Las instrucciones dependientes sólo a veces pueden ejecutarse concurrentemente*

independencia

A ← 5 x C
B ← 20 + C

→ **resultado siempre correcto**

dependencia

A = 5

A ← 3 + A
B ← 4 x A

Resultado	Secuencial	Concurrente
correcto	B = 32	B = 32
incorrecto	imposible	B = 20

Riesgos de datos: ejemplo

Mul r2, r3, r4
Add r5, r4, r6

→ r4 hace falta en W_1 y D_2

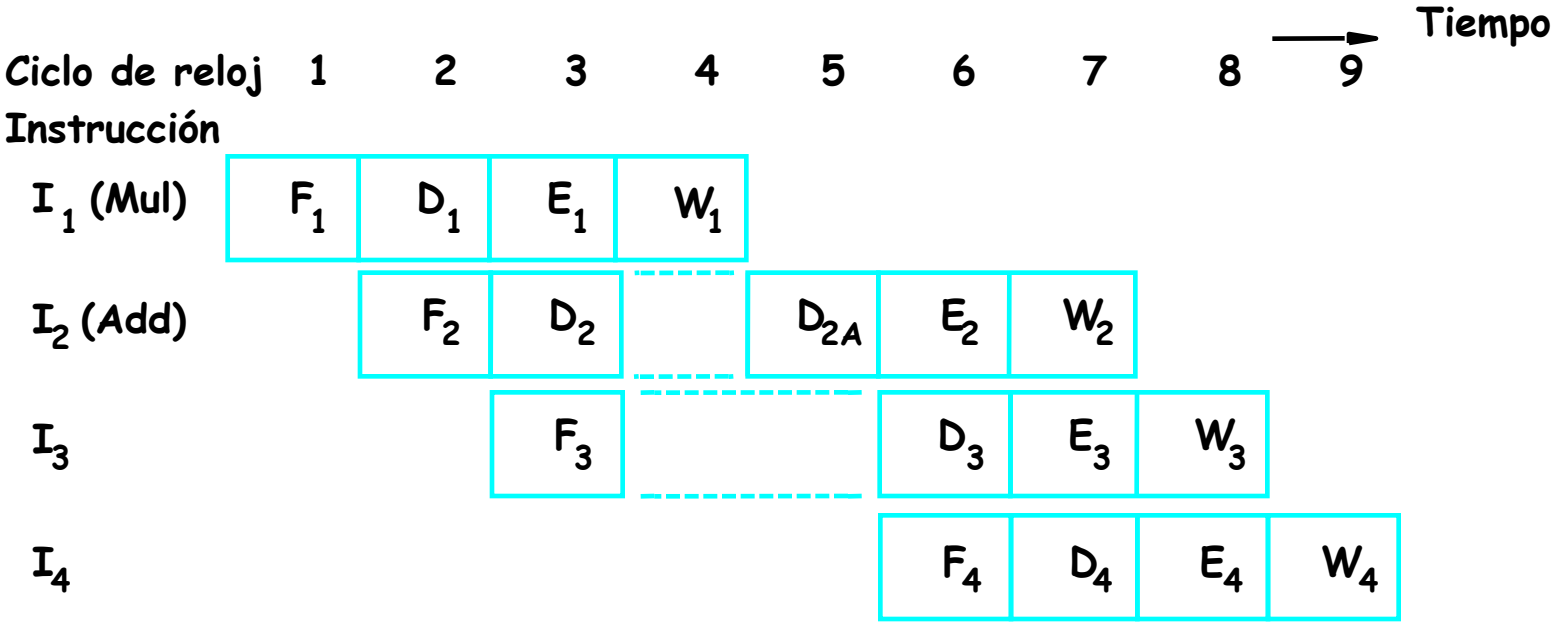
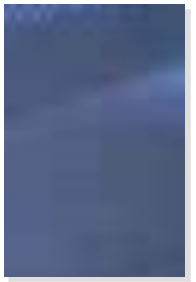


Figura 8.6. Atasco en el cauce ocasionado por la dependencia entre W_1 y D_2

TEMA 4: Segmentación de cauce

Objetivos del capítulo

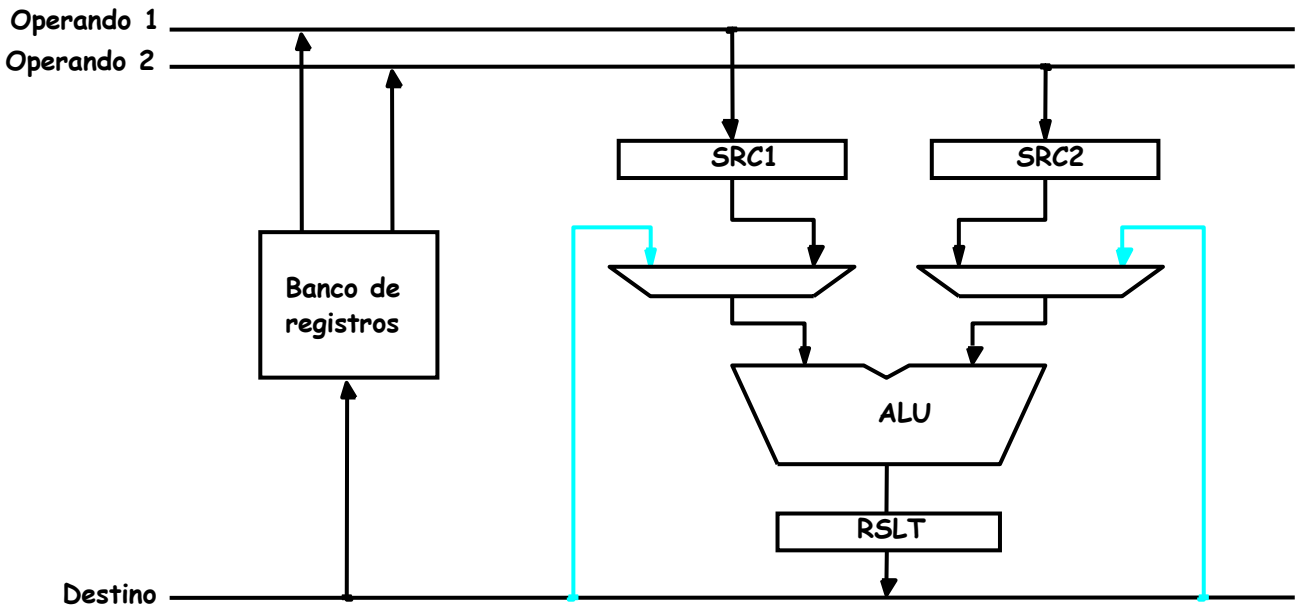
- Conocer los principios básicos de la Segmentación de cauce
- Riesgos que pueden degradarlas prestaciones y las formas de mitigar sus efectos
- Implicaciones software y hardware en la segmentación de cauce
- Influencia en el diseño del repertorio de instrucciones
- Conocer los principios básicos de procesadores superescalares



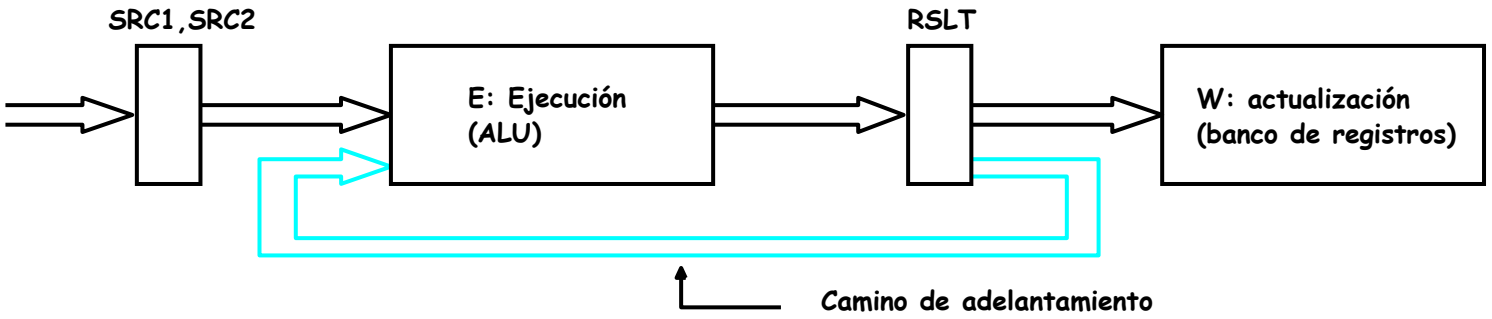
Material complementario en:

Organización de Computadores. C. Hamacher *et al*. 5ª Edición. McGraw Hill

Riesgos de datos: adelantamiento



(a) Camino de datos



(b) Situación de los registros fuente y resultado en el cauce del procesador

Figura 8.7. Adelantamiento de un operando en un procesador segmentado

Riesgos de datos: gestión software

- Las dependencias de datos las descubre el hardware al decodificar las instrucciones.
- Alternativamente podemos dejar que sea el compilador el que las resuelva:

  Mul r2, r3, r4
  NOP
  NOP
  Add r5, r4, r6

- Ventajas:
  - *Hardware más simple*
  - *Se pueden reorganizar las instrucciones para realizar trabajo útil en lugar de añadir NOP*
- Inconvenientes:
  - *Aumenta el tamaño del código*
  - *Las instrucciones NOP necesarias para una arquitectura pueden no hacer falta en otras*

Riesgos de datos: efectos colaterales

- Algunas instrucciones modifican registros no mencionados explícitamente:
  - *direccionamientos con automodificación*
  - *operaciones de pila*
  - *afectan al estado del procesador*
- Ejemplo:

  Add r1, r3 ← r3 = r3 + r2

  Adc r2, r4 ← r4 = r2 + r4 + acarreo
- Los efectos colaterales aumentan la complejidad del software o hardware encargado de detectarlos y corregirlos

Riesgos de instrucciones: salto incondicional

- Se producen cuando no está disponible a tiempo una instrucción (fallo de caché, instrucciones de salto,…)
- Tiempo perdido como consecuencia del salto: penalización de salto
- Importante averiguar la dirección de salto lo antes posible (Fig. 8.9)

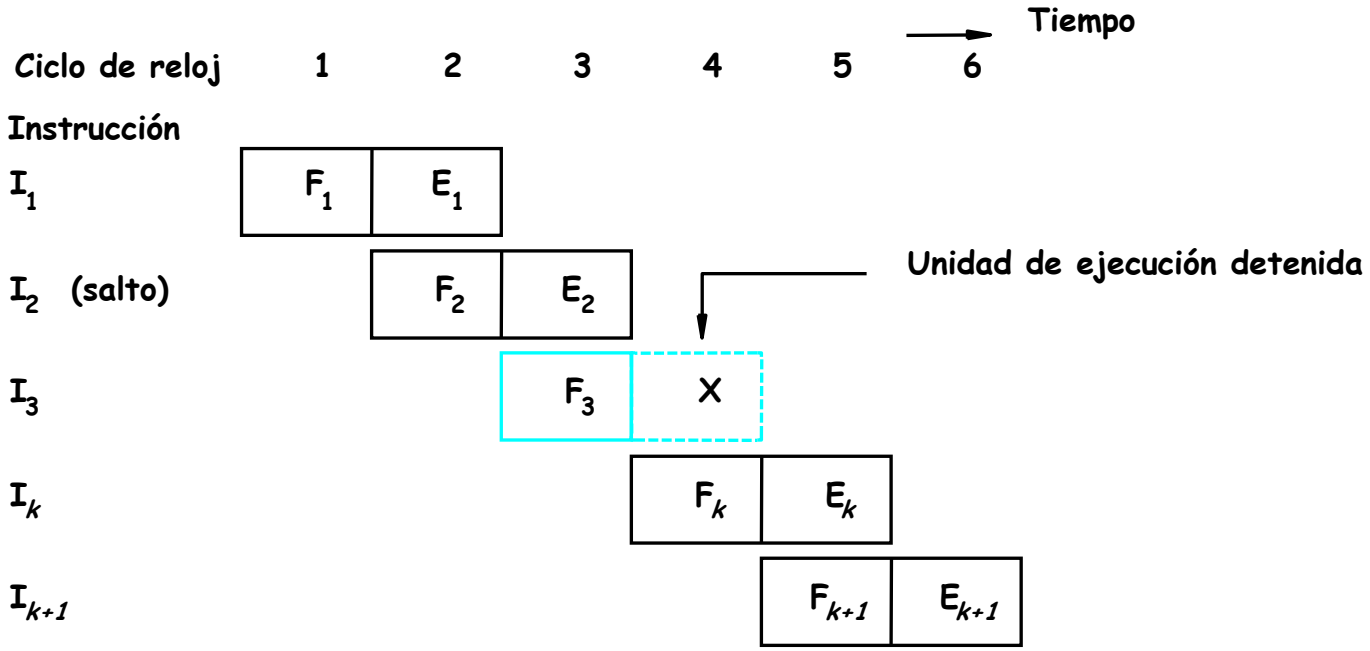
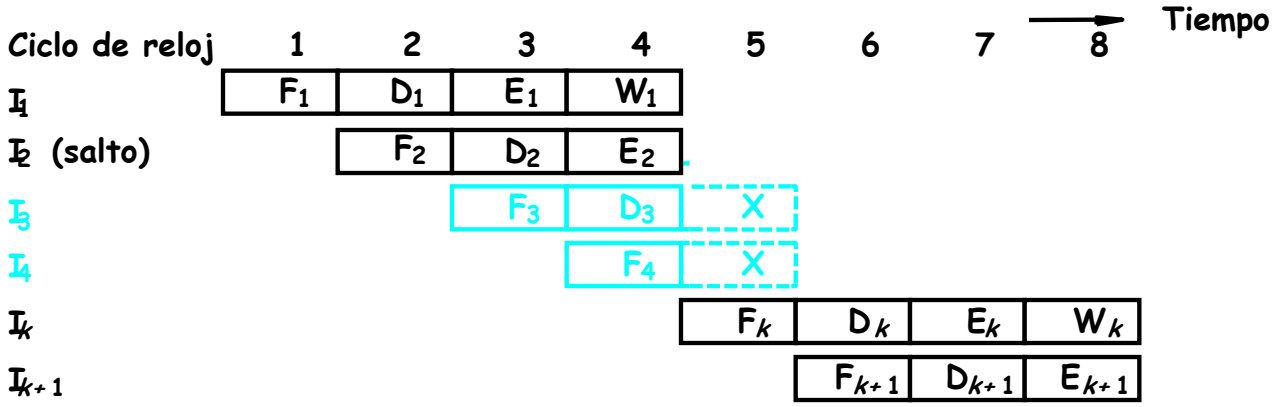
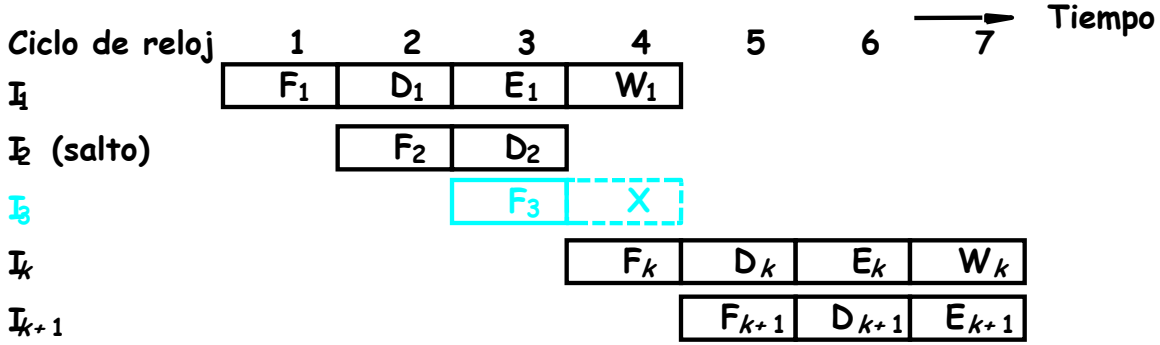


Figura 8.8. Ciclo perdido debido a un instrucción de salto

Riesgos de instrucciones: salto incondicional



(a) Dirección de salto calculada en la etapa de ejecución



(b) Dirección de salto calculada en la etapa de decodificación

Figura 8.9. Temporización de un salto

Riesgos de instrucciones: salto incondicional

- Para reducir el efecto de los fallos de caché se suelen captar instrucciones antes de que sean necesarias (**precaptación**) y se almacenan en una **cola de instrucciones** (Fig. 8.10)

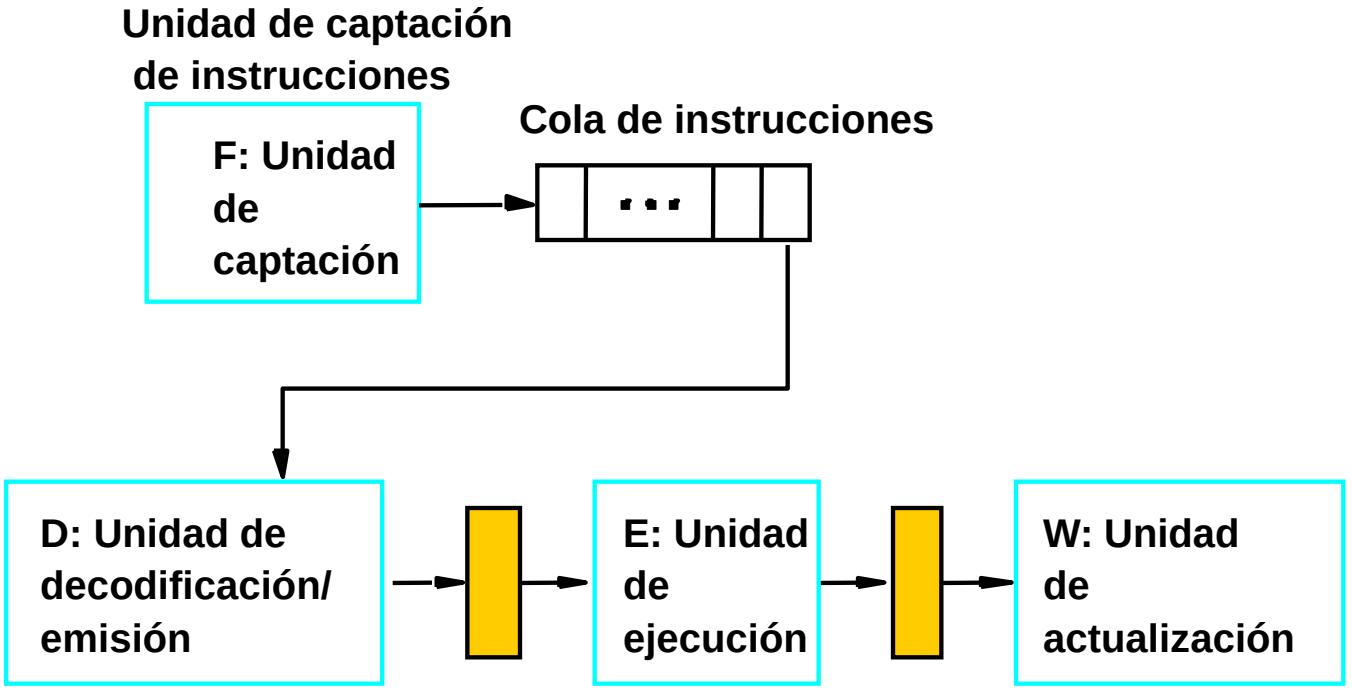


Figura 8.10. Uso de una cola de instrucciones en un segmento de 4 etapas

Riesgos de instrucciones: salto incondicional

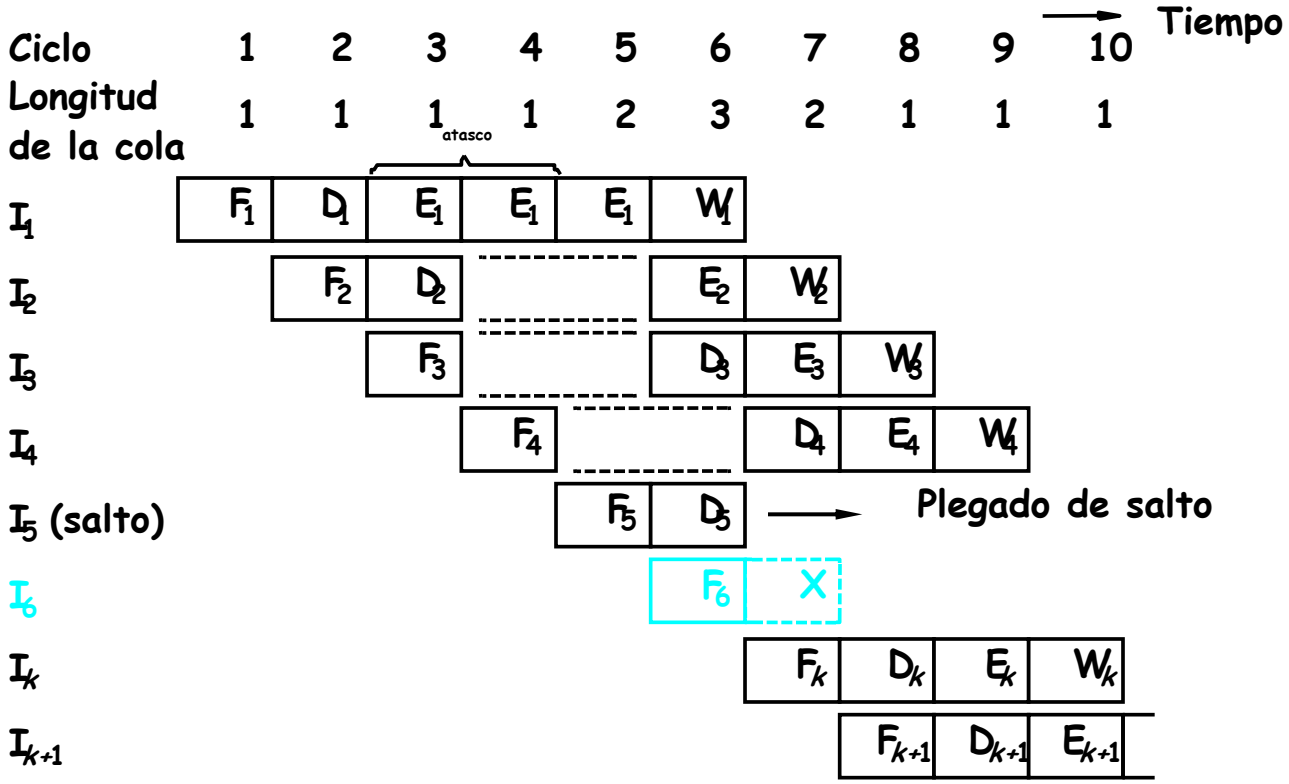


Figura 8.11. Temporización de un salto con cola de instrucciones. La dirección del salto se calcula en la etapa D

Riesgos de instrucciones: saltos condicionales

- Riesgo originado por la dependencia entre la condición de salto y el resultado de una operación previa
- Hueco de retardo de salto: instrucción/es que siguen a una instrucción de salto (depende del tiempo que tarde en calcularse la dirección - Fig. 8.9)
- Salto retardado: reordenar las instrucciones para ejecutar las instrucciones que siempre se captan (huecos)

loop	**shl**	**r1**
	dec	**r2**
	jnz	**loop**
next	**add**	**r1,r3**

(a) Programa original

loop	**dec**	**r2**
	jnz	**loop**
	shl	**r1**
next	**add**	**r1,r3**

(b) Programa reordenado

Figura 8.12. Reordenamiento de las instrucciones en salto retardado

Riesgos de instrucciones: saltos condicionales

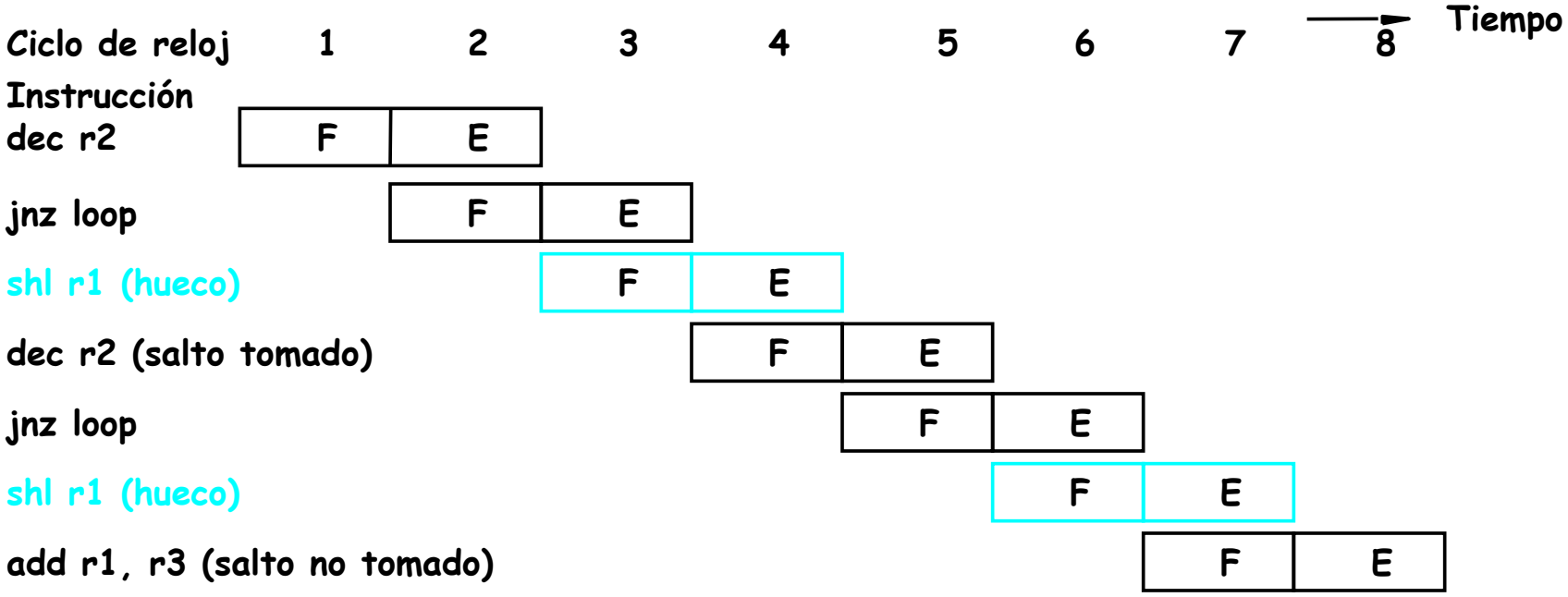


Figura 8.13. Ejecución mostrando un hueco de retardo lleno de las últimas dos iteraciones del programa de la Figura 8.12b

Riesgos de instrucciones: predicción de salto

- Más sencillo: asumir que el salto no se toma → seguir ejecutando secuencialmente hasta evaluar la condición
- **Ejecución especulativa**: ejecutar instrucciones sin haber verificado que forman parte de la secuencia correcta

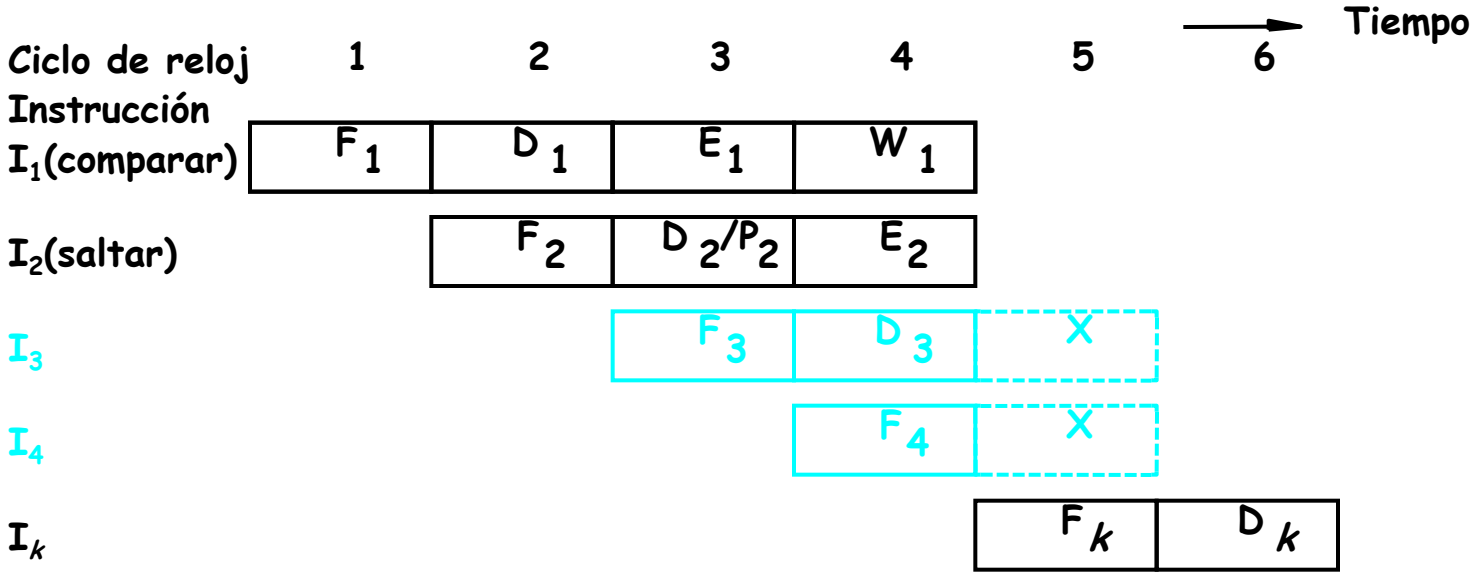


Figura 8.14. Temporización cuando la decisión de saltar se ha predicho incorrectamente como "no tomado"

Riesgos de instrucciones: predicción de salto

Tipos:

- **Estática:** se toma la misma decisión para cada tipo de instrucción (hardware/software)
- **Dinámica:** cambia según la historia de ejecución de un programa

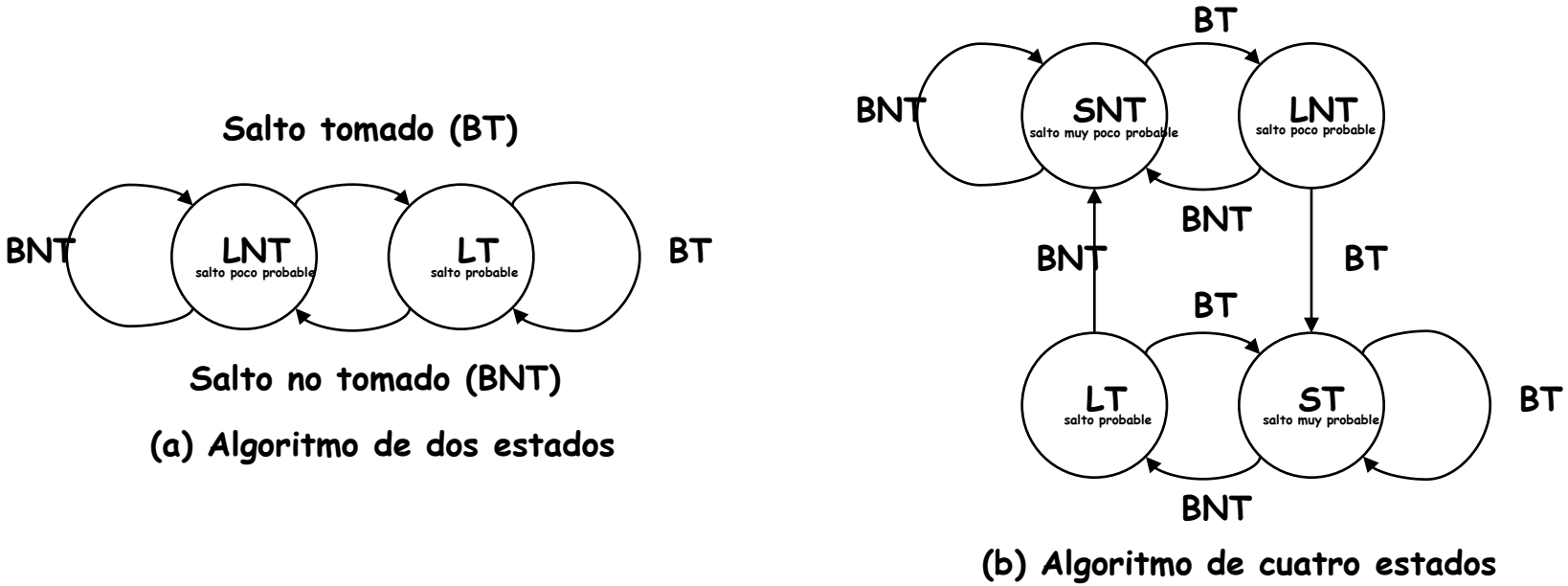
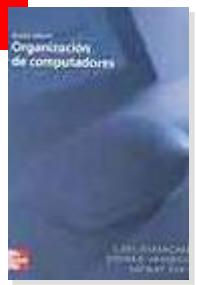


Figura 8.15. Representación de las máquinas de estados de los algoritmos de predicción de saltos

TEMA 4: Segmentación de cauce

Objetivos del capítulo

- Conocer los principios básicos de la Segmentación de cauce
- Riesgos que pueden degradarlas prestaciones y las formas de mitigar sus efectos
- Implicaciones software y hardware en la segmentación de cauce
- Influencia en el diseño del repertorio de instrucciones
- Conocer los principios básicos de procesadores superescalares



Material complementario en:

Organización de Computadores. C. Hamacher *et al*. 5ª Edición. McGraw Hill

Modos de direccionamiento

- Deberían ser eficientes y evitar los efectos colaterales, pero esto entra en conflicto con la utilidad
- Características empleadas en la actualidad (tres modos que cumplen estas condiciones: registro, indirecto con registro e indexado):
  - *El acceso a un operando no necesita más de un acceso a memoria*
  - *Sólo las instrucciones de carga y almacenamiento acceden a memoria*
  - *Los modos de direccionamiento no tienen efectos colaterales*
- Ejemplo: ver tabla siguiente y Fig. 8.16

Complejo	Sencillo
load x(r1), r2	add #x, r1, r2 load (r2), r2 load (r2), r2

Modos de direccionamiento

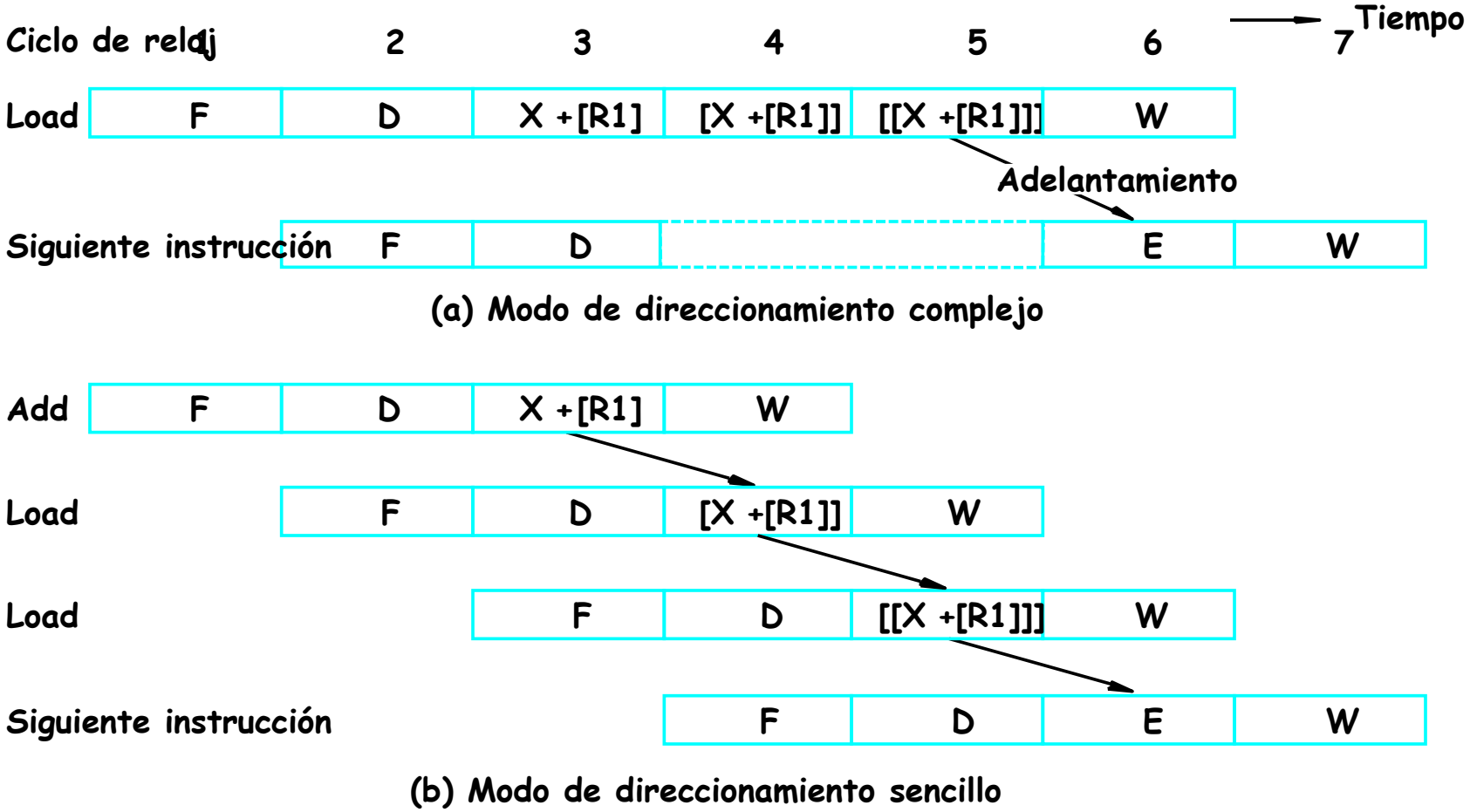


Figure 8.16. Operaciones equivalentes utilizando modos de direccionamiento sencillos y complejos

Códigos de condición

- Las dependencias introducidas por los bits de condición dificultan al compilador reordenar instrucciones
- Ejemplo: fragmento de la Figura 8.17 siguiendo el esquema la Figura 8.14 → adelanta la decisión de salto de E_2 a D_2

sumar	**r1, r2**
comparar	**R3, r4**
saltar = 0	

(a) **Fragmento de programa**

comparar	**r3, r4**
sumar	**r1, r2**
saltar = 0	

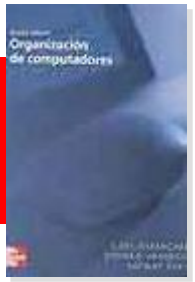
(b) **Fragmento reordenado**

Figura 8.17. Reordenamiento de instrucciones

TEMA 4: Segmentación de cauce

Objetivos del capítulo

- Conocer los principios básicos de la Segmentación de cauce
- Riesgos que pueden degradarlas prestaciones y las formas de mitigar sus efectos
- Implicaciones software y hardware en la segmentación de cauce
- Influencia en el diseño del repertorio de instrucciones
- Conocer los principios básicos de procesadores superescalares



Material complementario en:

Organización de Computadores. C. Hamacher *et al*. 5ª Edición. McGraw Hill

Funcionamiento superescalar

- Segmentación de cauce = concurrente + secuencial
- Superescalar = concurrente + **paralelo**
- Se han de duplicar unidades
- Se puede comenzar a ejecutar más de una instrucción por ciclo de reloj → **emisión múltiple**
- Rendimiento = más de una instrucción por ciclo
- Es fundamental poder captar instrucciones rápidamente → conexión ancha con memoria + caché + cola de instrucciones

Funcionamiento superescalar

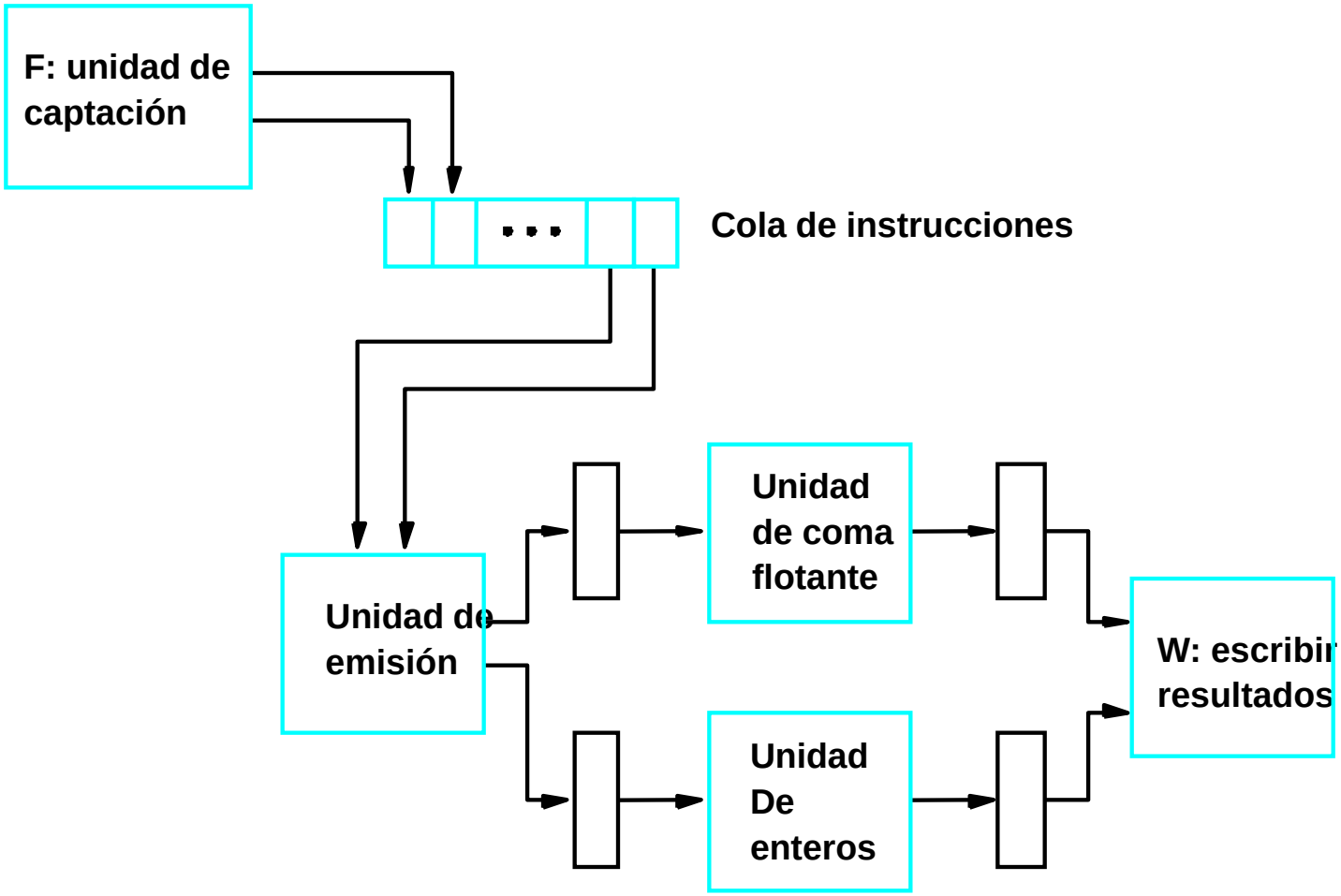


Figura 8.19. Procesador con dos unidades de ejecución

Funcionamiento superescalar

El efecto negativo de los riesgos es todavía más pronunciado

El compilador puede reordenar instrucciones para evitar riesgos

Ejemplo: mezcla de operaciones con enteros y en coma flotante de la Figura 8.20 siguiendo un esquema interno como el de la Figura 8.19

Funcionamiento superescalar

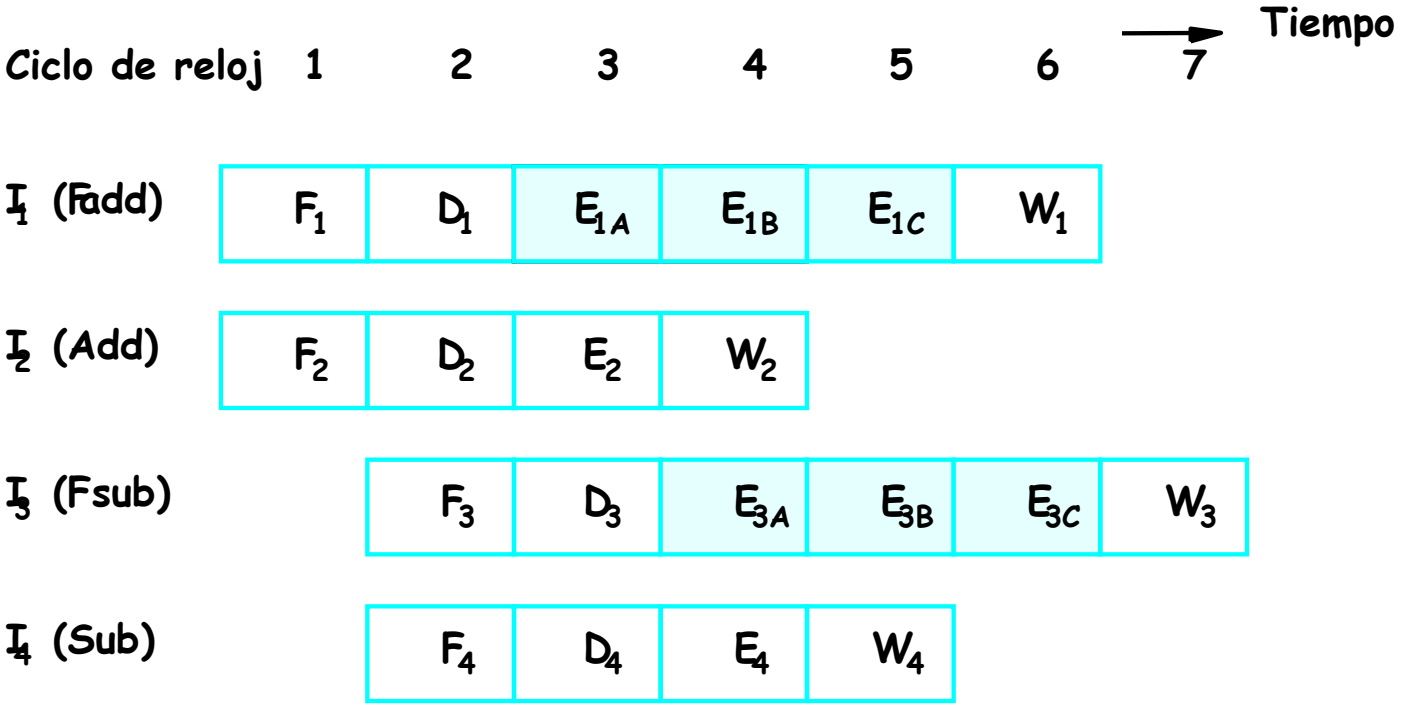
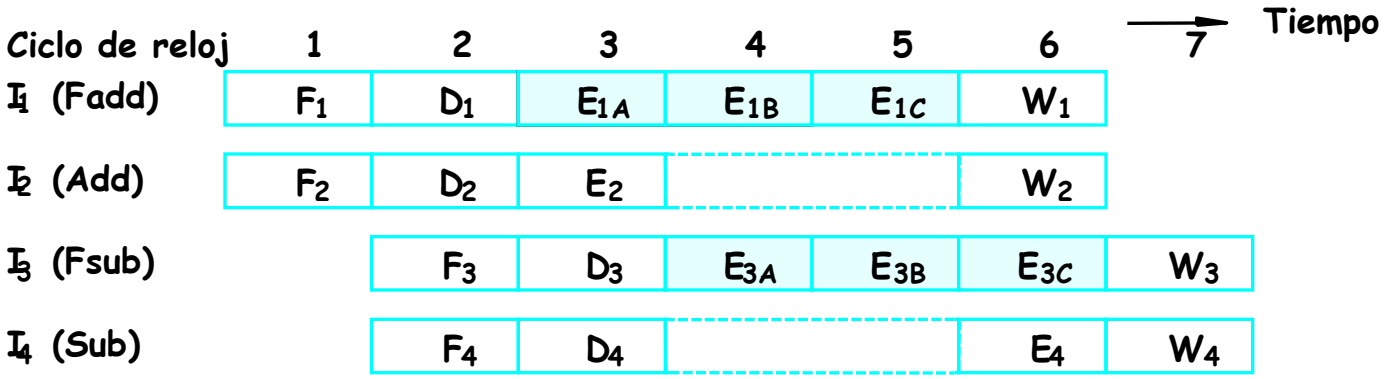


Figura 8.20. Ejemplo de flujo de ejecución de instrucciones en el procesador la figura 8.19, asumiendo que no existen riesgos.

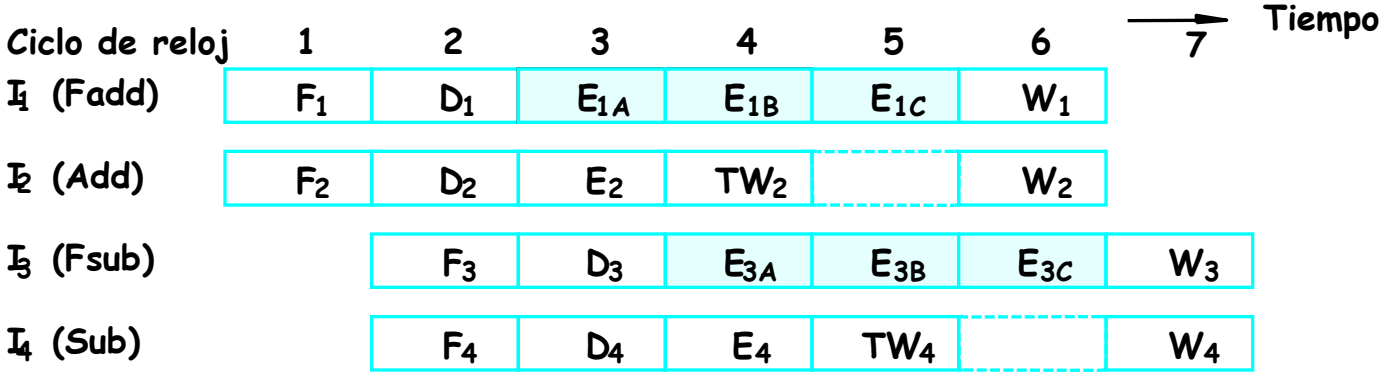
Ejecución desordenada

- Las instrucciones pueden emitirse en orden y terminar de forma desordenada
- Posibles complicaciones:
  - *Dependencia entre instrucciones (I_1 e I_2 en Fig. 8.20)*
  - *Excepciones*
- Tipos de excepciones:
  - *Imprecisa: alguna instrucción posterior a la que produce la excepción ha terminado de ejecutarse → deja el programa en un estado inconsistente*
  - *Precisa: no puede ocurrir lo anterior*
- Soluciones:
  - *Escritura retardada (Fig. 8.21a)*
  - *Uso de buffers temporales (Fig. 8.21b)*

Ejecución desordenada



(a) Escritura retardada



(b) Uso de registros temporales

Figura 8.21. Finalización de instrucciones según el orden del programa

Finalización de la ejecución

- La ejecución desordenada es conveniente para aprovechar los recursos
- Las instrucciones deberían completarse en el orden del programa para tener excepciones precisas
- Ambos objetivos contrapuestos pueden conseguirse mediante **renombramiento de registros** (ej: destino de I_2 es R_5 → $TW_2 = R_5$ durante ciclos 5 y 6, en Fig. 8.21b)
- Las unidades de control deben tener en cuenta la ejecución desordenada → **unidad de finalización** (**cola de reordenamiento**)
- Empareja las fases de **emisión** y **retirada** de cada instrucción

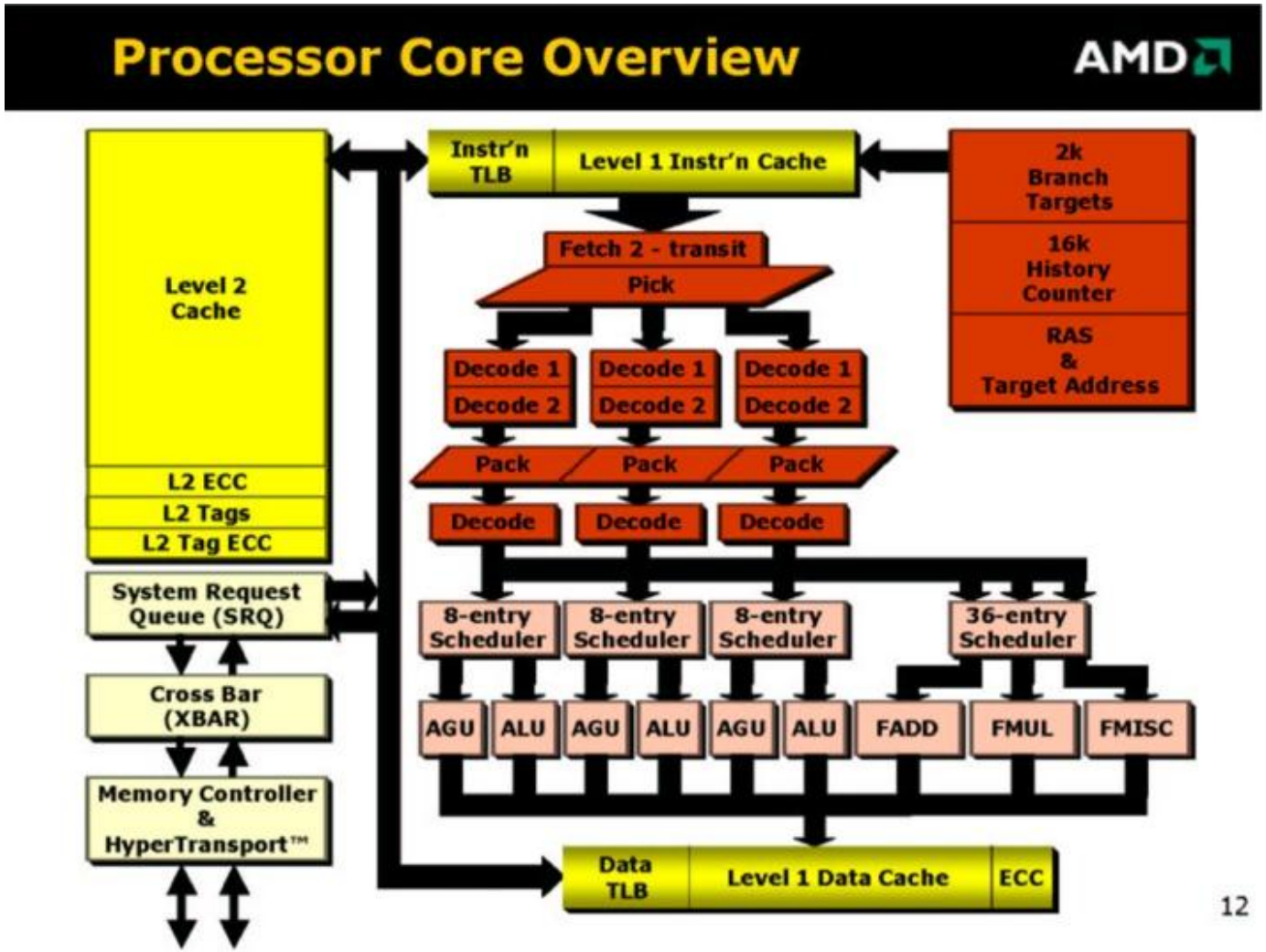
La operación de emisión

La unidad de emisión debe comprobar que hay recursos libres suficientes: unidades de ejecución, buffers temporales, huecos de la cola de reordenamiento,…

Objetivo: evitar bloqueos → situación que se da cuando dos instrucciones necesitan un recurso compartido para finalizar

Ejemplo: una instrucciones A utiliza un buffer temporal y necesita que B acabe para finalizar, y B necesita el mismo buffer temporal para acabar

Ejemplo: AMD Athlon 64



Consideraciones relativas a prestaciones

$$T = \frac{N \times S}{R}$$

T = Tiempo de ejecución
N = cantidad dinámica de instrucciones
S = número medio de ciclos de reloj que se tarda en ejecutar una instrucci
R = frecuencia de reloj

$$P_s = \frac{R}{S}$$

P_S = número de instrucciones ejecutadas por segundo (caso secuencial)
P_P = número de instrucciones ejecutadas por segundo (caso concurrente)

$$\delta_{fallo} = ((1 - h_i) + d(1 - h_d)) \times M_p$$

$$P_p = \frac{R}{T_I} = \frac{R}{1 + \delta_{fallo}}$$

$$P_s = \frac{R}{4 + \delta_{fallo}}$$

M_p = penalización por fallo en caché
h_i = probabilidad de acierto para instrucciones
h_d = probabilidad de acierto para datos
T_I = tiempo en el caso secuencial

Consideraciones relativas a prestaciones

Efectos de los riesgos de control

- Si el tiempo de la ALU es 2ns → frecuencia de reloj óptima 500MHz
- El procesador de la Fig. 8.2, en condiciones ideales Pp ≈ 500 MIPS

Numero de etapas del cauce segmentado

- **La teoría sugiere que a mayor numero de etapas mayor rendimiento**
- **Al aumentar el número de etapas también crece la probabilidad de sufrir un atasco**