

# Arquitectura de Computadores



## TEMA 2

ILP, Planificación dinámica,  
Predicción de saltos, Especulación

**D**EPARTAMENTO DE  
**A**RQUITECTURA DE **C**OMPUTADORES  
Y **A**UTOMÁTICA

Curso 2020-2021

# Contenidos

---

- ❑ Introducción: ILP
- ❑ Técnicas SW: Compilador
- ❑ Planificación dinámica. Algoritmo de Tomasulo.
- ❑ Tratamiento de dependencias de control: Predicción de saltos
  - Tratamiento de saltos
  - Técnicas de predicción: Estáticas y dinámicas
- ❑ Especulación
- ❑ Bibliografía
  - Básica: Capítulo 3 de Hennessy & Patterson, 5th ed., 2012
  - Complementaria: Capítulo 3 de Dubois, Annavaram & Stenstöm, 2012

# Introducción

- ❑ **OBJETIVO:** Ejecutar el mayor numero de instrucciones por ciclo
- ❑ **Obtener el máximo numero de instrucciones independientes**

$$CPI = CPI\ ideal + Penaliz.\ Media\ por\ Instr.\ (paradas\ "pipe")$$

¿Qué técnicas conocemos?

- Conflictos de recursos - Riesgos estructurales → Replicación/segmentación
- Dependencias de datos → Cortocircuitos
- Dependencias de control ( Un salto cada 4-7 instrucciones) → Saltos retardados

## Mecanismos para explotar ILP

- Basados en HW en tiempo de ejecución (dinámicos). Ej Intel Core, AMD, IBM, ARM...  
Toda la información disponible en ejecución  
Código independiente de la implementación
- Basados en SW en tiempo de compilación (estáticos). Ej Itanium  
Dependencias de memoria muy difíciles de determinar

## ❑ Paralelismo a nivel de instrucción ILP


- ❑ Es la técnica consistente en explotar paralelismo entre instrucciones próximas en la secuencia
- ❑ El bloque básico:
  - Un bloque básico (BB) es una secuencia de código sin saltos. Un solo punto de entrada y salida.
  - Típicamente muy pequeño: solo de 4 a 7 instrucciones
  - ... y con fuertes dependencias entre ellas
- ❑ El camino es explotar ILP entre varios BB
- ❑ El caso más simple: paralelismo a nivel de bucle

```
for ( i =1000; i>0; i--)
```

$$x(i) = x(i) + s;$$
  - Todas las iteraciones son independientes (saltos)

# Introducción

## ❑ Técnicas para explotar ILP

|  | Técnica  | Reduce  |
|--|--|---|
| <b>Dinámicas</b><br>  | Planificación Dinámica                               | Paradas por riesgos de datos                      |
|  | Predicción dinámica de saltos                        | Paradas por riesgos de control                    |
|  | Lanzamiento múltiple                                 | CPI Ideal   |
|  | Varias instrucciones por ciclo                       |   |
|  | Especulación   | Riesgos de datos y control                        |
|  | Dynamic memory disambiguation                        | Paradas por riesgos de datos en memoria           |
| <b>Estáticas</b><br> | Desenrollado de bucles                               | Paradas por riesgos de control                    |
|  | Planificación por el compilador                      | Paradas por riesgos de datos                      |
|  | Software pipelining                                  | CPI Ideal y Paradas por riesgos de datos          |
|  | Predicción estática y Especulación por el Compilador | CPI Ideal, paradas por riesgos de datos y control |

# Dependencias

---

- ❑ Determinar las dependencias es crítico para obtener el máximo paralelismo

¿Cuáles hay? ¿A qué recursos afectan?

Las dependencias son propias de los programas

- o La presencia de una dependencia indica la posibilidad de aparición de un riesgo, pero la aparición de éste y la posible parada depende de las características del "pipe"
- o Las dependencias
  - Indican la posibilidad de un riesgo
  - Determinan el orden de cálculo de los resultados
  - Imponen un límite al paralelismo que es posible obtener

## □ Tipos de Dependencias

- Dependencias de datos
  - Dependencia verdadera (LDE)
  - Dependencias de nombre
    - Antidependencia (EDL)
    - Dependencia de salida (EDE)
- Dependencias de control

## □ Dependencia verdadera (LDE)

- La instrucción j depende de i
  - i produce un resultado que usa j
  - j depende de k y k depende de i

|    |       |          |
|----|-------|----------|
| i: | L.D   | F0,0(R1) |
| j: | ADD.D | F4,F0,F2 |

## □ Dependencias de nombre ( Reutilización de los registros )

- o Dos instrucciones i y j donde i precede a j presentan dependencias de nombre en las siguientes situaciones:

- o Antidependencia WAR (EDL)

- La instrucción j escribe ( Reg o memoria) antes de que i lea.

ADD.D F4,**F0**,F2

L.D **F0**, -8(R1)

- o Dependencia de salida WAW (EDE)

- Las instrucciones i y j escriben el mismo reg. o memoria

ADD.D **F4**,F0,F2

SUB.D **F4**,F3,F2

## □ ILP y Dependencias de datos

- o Los mecanismos de ejecución deben preservar el comportamiento del programa. Mismo resultado que en ejecución secuencial
- o Explotar todo el paralelismo posible sin afectar al resultado de la ejecución
- o Para las dependencias de nombre eliminar la dependencia usando otros "nombres"



## □ Dependencias de control

o Cada instrucción depende de un conjunto de saltos y en general esta dependencia debe preservarse para preservar el comportamiento del programa

```
if P1 (  
    S1;  
);  
if P2 (  
    S2;  
)
```

S1 depende de P1 ; S2 depende de P2

Pero... las dependencias de control pueden violarse, es decir, se pueden ejecutar instrucciones no debidas **si esto no afecta** al resultado correcto del programa (Ej. "delay slot" en los saltos retardados)

**LO IMPORTANTE: el comportamiento de las excepciones y el flujo de datos deben preservarse**

## ❑ Dependencias de control y Excepciones

- o Comportamiento de excepciones se debe preservar. Cualquier cambio en el orden de ejecución no debe cambiar cómo las excepciones son atendidas en la ejecución.

DADDU R2,R3,R4

BEQZ R2,L1

LW R1,0(R2)

L1: --- ---

- o LW no se puede mover antes de BEQZ ( posible fallo de pagina )

## ❑ Dependencias de control y flujo de datos

- o Se debe mantener el flujo de datos entre instrucciones productoras y consumidoras de datos.

DADDU R1,R2,R3

BEQZ R4,L1

DSUBU R1,R5,R6

L1: --- ---

OR R7,R1,R8

- o OR usa el resultado de DADDU o DSUBU dependiendo del comportamiento del salto. El flujo de datos se debe preservar.

# Dependencias

## ❑ Dependencias de datos

Fáciles de determinar para registros

Difíciles para direcciones de memoria

¿Son el mismo dato 100(R4) y 20(R6)?

En dos iteraciones diferentes 20(R6) y 20(R6) ¿son el mismo dato?

Implica: Más registros para evitar dependencias de nombre

Identificar dependencias entre load y store para permitir su reordenación

## ❑ Dependencias de control

En general:

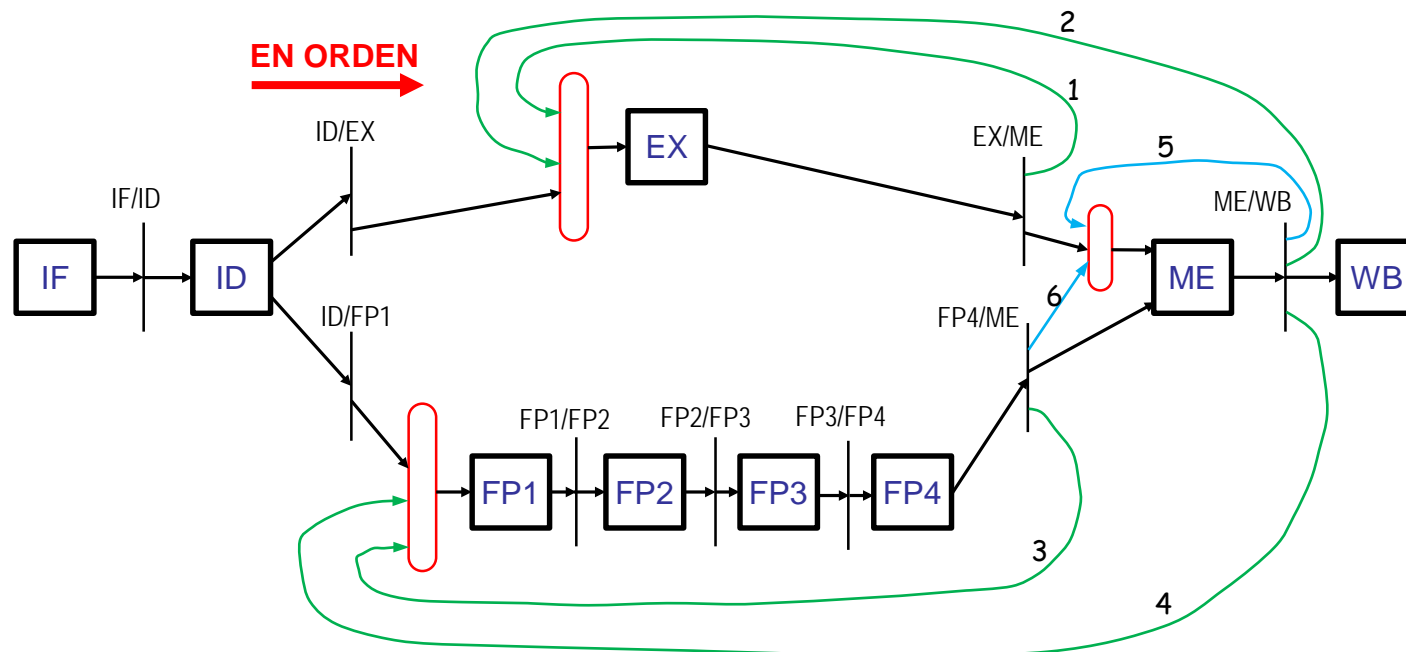
- Una instrucción dependiente de un salto no puede moverse antes del salto
- Una instrucción no dependiente de un salto no puede moverse después del salto

Efecto de las dependencias de control sobre el orden de las excepciones y el flujo de datos

**SOLUCIÓN : HW + SW (PROCESADOR + COMPILADOR)**

# Técnicas SW para explotar ILP

- ❑ Modelo de computador para el estudio (muy similar al estudiado en EC)
  - o Lanzamiento a ejecución **en orden**. Detección de riesgos en ID. Si riesgo, congelar instrucción en ID.
  - o Saltos resueltos en ID
  - o Etapa de ejecución con varias UFs de diferente duración. Posibilidad de finalización en desorden.
    - EX: load, store, int
    - FP: cálculos en FP (segmentada en 4 etapas)
  - o Bancos de registros separados para INT y FP



## Red de anticipación de operandos (by-pass)

1: Resultado de op int

2: Res. de op int o load INT

3: Res. de op FP

4: Resultado de op FP o load FP

5: Res. de op int, load INT → store int  
Res. de op FP, load FP → store FP

6: Res. de op FP → store FP

# Técnicas SW para explotar ILP

- ❑ Latencia de uso: número mínimo de ciclos entre una instrucción que produce un resultado y otra que lo consume

| Instrucción que produce resultado | Instrucción que usa el resultado | Latencia de uso |
|-----------------------------------|----------------------------------|-----------------|
| FP aritmética                     | FP aritmética                    | 3               |
| FP aritmética                     | STORE FP                         | 2               |
| LOAD FP                           | FP aritmética                    | 1               |
| LOAD FP                           | STORE FP                         | 0               |
| LOAD Int                          | Int aritmética                   | 1               |
| Int aritmética                    | Int aritmética                   | 0               |

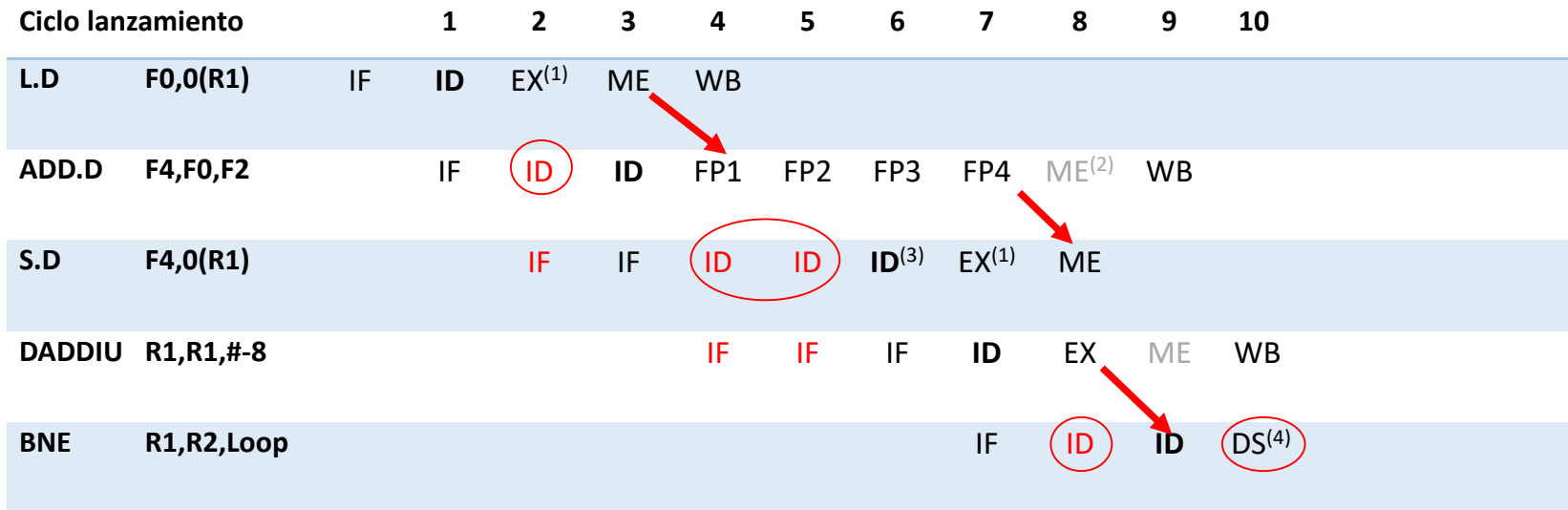
# Técnicas SW para explotar ILP

## □ Un programa: Bucle simple

```
for ( i = 1000; i > 0; i--)
    x (i) = x (i) + s ;
```

### Código maquina MIPS

```
Loop: L.D      F0,0(R1)    ; F0 ← elemento array
      ADD.D    F4,F0,F2    ; sumar escalar en F2
      S.D      F4,0(R1)    ; almacenar resultado
      DADDIU   R1,R1,#-8   ; decrementar puntero
      BNE      R1,R2,Loop  ; repetir si R1 ≠ R2
```



(1) Cálculo de la dirección de memoria efectiva

(2) Pasa por la etapa ME, pero no hay un acceso real a memoria

(3) El SD no avanza mientras su R. fuente coincida con alguno de los R. destino en FP1 o FP2. Si la coincidencia es con FP3, el SD ya puede avanzar a EX, cuando llegue a ME recibirá el operando por forwarding desde FP4 (camino 6).

(4) Delay Slot

# Técnicas SW para explotar ILP

## □ Rendimiento en una iteración

|       |        | <u>Ciclo de<br/>lanzamiento</u> |                           |
|-------|--------|---------------------------------|---------------------------|
| Loop: | L.D    | F0,0(R1)                        | 1                         |
|       | Espera |                                 | 2 ; ADD.D bloqueado en ID |
|       | ADD.D  | F4,F0,F2                        | 3                         |
|       | Espera |                                 | 4 ; S.D bloqueado en ID   |
|       | Espera |                                 | 5 ; S.D bloqueado en ID   |
|       | S.D    | F4,0(R1)                        | 6                         |
|       | DADDIU | R1,R1,#-8                       | 7                         |
|       | Espera |                                 | 8 ; BNE bloqueado en ID   |
|       | BNE    | R1,R2,Loop                      | 9                         |
|       | Espera |                                 | 10 ; Salto retardado      |

- El rendimiento se degrada por las dependencias, a pesar del by-pass de operandos
- El CPI ideal de 1 se duplica (10 ciclos para lanzar 5 instrucciones):  
CPI = 2

# Técnicas SW para explotar ILP

## ❑ Planificación de instrucciones: Reordenar para ocultar latencias

|       |        |            |         |                                       |
|-------|--------|------------|---------|---------------------------------------|
| Loop: | L.D    | F0,0(R1)   | Ciclo 1 | Reordenamiento para ocultar latencias |
|       | DADDIU | R1,R1,#-8  | 2       |                                       |
|       | ADD.D  | F4,F0,F2   | 3       | 6 ciclos por iteración. CPI ~ 1       |
|       | Espera |            | 4       |                                       |
|       | BNE    | R1,R2,Loop | 5       | 2 ciclos de overhead por el salto     |
|       | S.D    | F4,8(R1)   | 6       |                                       |

## ❑ Desenrollado 4 veces para más paralelismo (elimina saltos)

|       |        |             |  |
|-------|--------|-------------|--|
| Loop: | L.D    | F0,0(R1)    |  |
|       | ADD.D  | F4,F0,F2    | Expone más paralelismo y elimina saltos            |
|       | S.D    | F4,0(R1)    |  |
|       | L.D    | F6,-8(R1)   |  |
|       | ADD.D  | F8,F6,F2    | Se elimina 3 saltos y 3 decrementos                |
|       | S.D    | F8,-8(R1)   |  |
|       | L.D    | F10,-16(R1) |  |
|       | ADD.D  | F12,F10,F2  | Permanecen dependencias y paradas:                 |
|       | S.D    | F12,-16(R1) | Hallar ciclos por iteración y CPI                  |
|       | L.D    | F14,-24(R1) |  |
|       | ADD.D  | F16,F14,F2  |  |
|       | S.D    | F16,-24(R1) | <u>MÁS REGITROS!! Renombrado por el Compilador</u> |
|       | DADDIU | R1,R1,#-32  | (Imprescindible ??)                                |
|       | BNE    | R1,R2,Loop  |  |



## ❑ Desenrollado + Planificación

```
Loop:  L.D      F0,0(R1)
        L.D      F6,-8(R1)
        L.D      F10,-16(R1)
        L.D      F14,-24(R1)
        ADD.D    F4,F0,F2
        ADD.D    F8,F6,F2
        ADD.D    F12,F10,F2
        ADD.D    F16,F14,F2
        S.D      F4,0(R1)
        S.D      F8,-8(R1)
        DADDIU   R1,R1,#-32
        S.D      F12,16(R1)          ; 16-32= -16
        BNE     R1,R2,Loop
        S.D      F16,8(R1)          ; 8-32 = -24
```

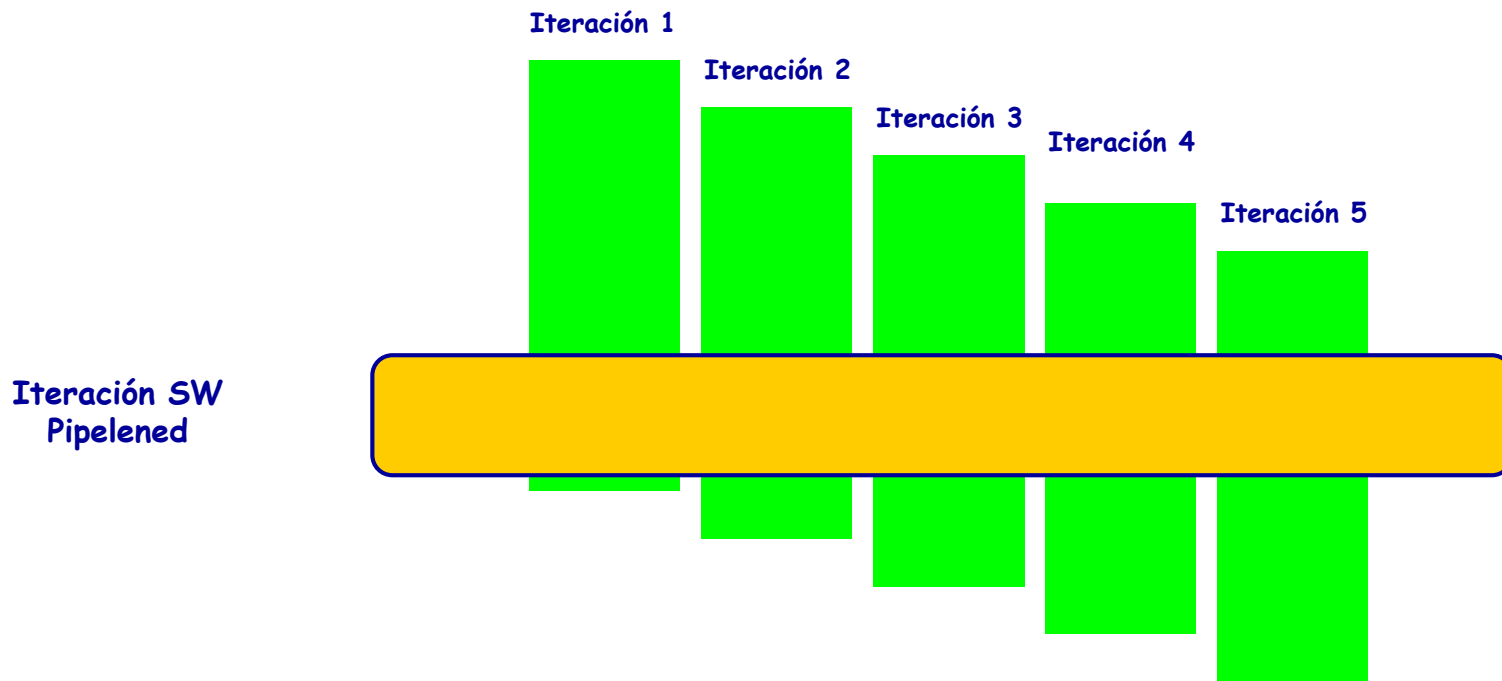
- ✓ Mover S.D después de DADDIU: ojo al valor de R1
- ✓ 14 instrucciones lazadas en 14 ciclos: 3.5 ciclos por iteración
- ✓ CPI = 1
- ✓ Más registros (Imprescindible !!)

El compilador planifica para minimizar los riesgos y eliminar las paradas del "pipe"

## ❑ Software “pipelining”

- Idea:

Si las diferentes iteraciones de un bucle son independientes, tomemos instrucciones de diferentes iteraciones para aumentar el ILP  
Reorganiza los bucles de manera que cada instrucción pertenece a una iteración diferente



# Técnicas SW para explotar ILP

## ❑ Software “pipelining”: Ejemplo en pseudo-código de alto nivel.

### ○ Procesar un vector de 126 componentes

// Bucle original

```
for (i=125; i>=0; i--) {
```

```
    f0=x(i);
```

```
    f4=f0+f2;
```

```
    x(i) = f4
```

```
}
```

Aumenta distancia  
entre instrucciones  
dependientes

// Versión SW Pipeline

```
f0=x(125);           (cabecera)
```

```
f4=f0+f2;
```

```
f0=x(124);
```

```
for (i=125; i>1; i--) { //bucle sw pl  
    x(i) = f4;           //Store x(i)
```

```
    // Add x(i-1)
```

```
    //Load x(i-2)
```

```
(cola)
```

```
    f4=f0+f2;
```

```
    f0=x(i-2) }
```

```
x(1)=f4;
```

```
f4=f0+f2;
```

```
x(0)=f4;
```

## □ Software “pipelining”

Antes: Unrolled 3 veces

|    |        |             |
|----|--------|-------------|
| 1  | L.D    | F0,0(R1)    |
| 2  | ADD.D  | F4,F0,F2    |
| 3  | S.D    | F4,0(R1)    |
| 4  | L.D    | F6,-8(R1)   |
| 5  | ADD.D  | F8,F6,F2    |
| 6  | S.D    | F8,-8(R1)   |
| 7  | L.D    | F10,-16(R1) |
| 8  | ADD.D  | F12,F10,F2  |
| 9  | S.D    | F12,-16(R1) |
| 10 | DADDIU | R1,R1,#-24  |
| 11 | BNE    | R1,R2,LOOP  |

Después: Software Pipelined

|   |        |                           |
|---|--------|---------------------------|
| 1 | S.D    | F4,0(R1) ; Stores x[i]    |
| 2 | ADD.D  | F4,F0,F2 ; Adds to x[i-1] |
| 3 | L.D    | F0,-16(R1) ; Loads x[i-2] |
| 4 | DADDIU | R1,R1,#-8                 |
| 5 | BNE    | R1,R2,LOOP; R2 = cte. 8   |

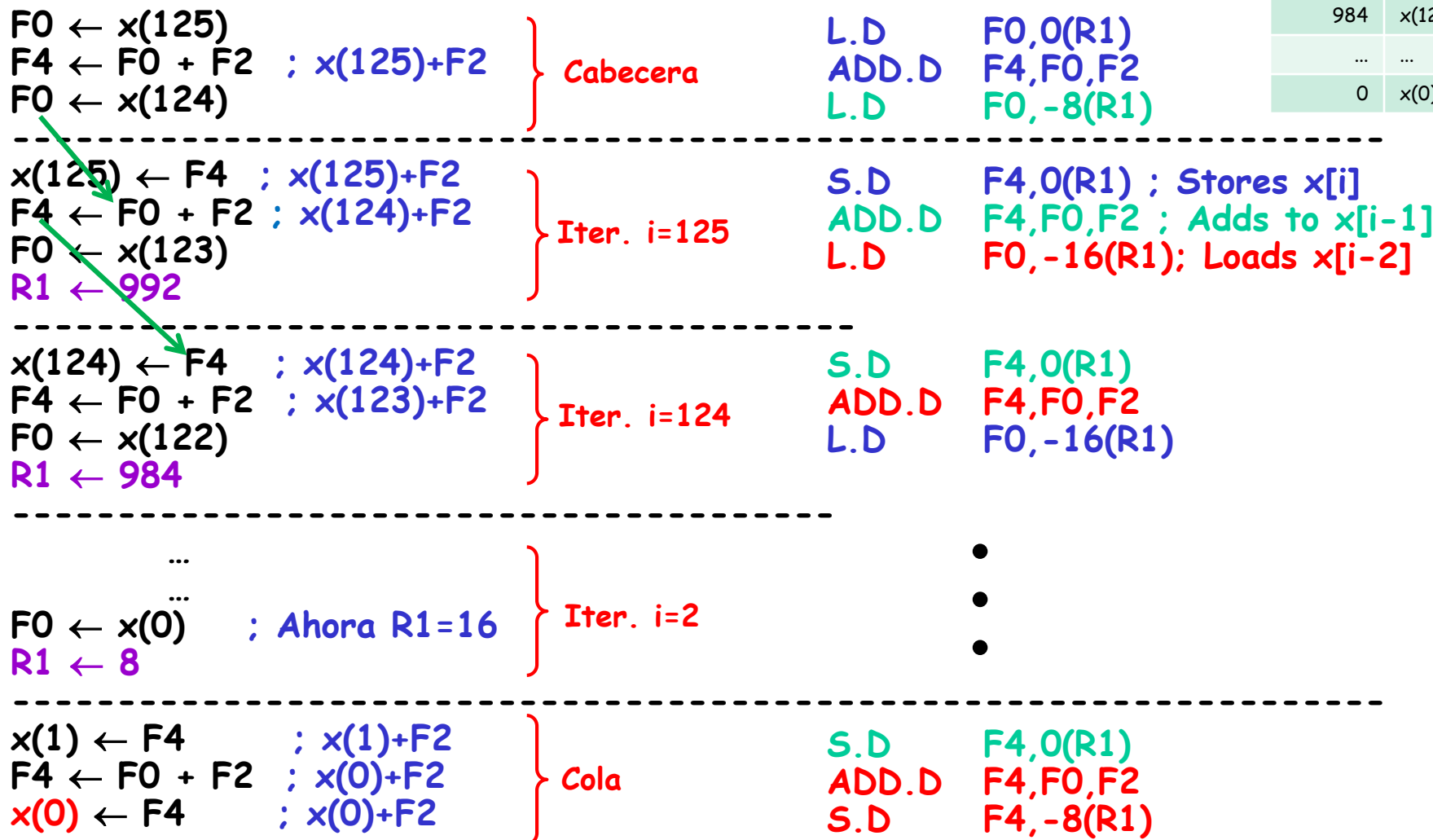
Es necesario código previo al bucle (cabecera) y posterior al bucle (cola)

- ✓ Loop unrolling simbólico
  - Maximiza la distancia resultado-uso
  - Menor tamaño del código

# Técnicas SW para explotar ILP

## Ejecución SW pipelined (suposición R1=1000)

| Dir  | Mem | Dato   |
|------|-----|--------|
| 1000 |     | x(125) |
| 992  |     | x(124) |
| 984  |     | x(123) |
| ...  |     | ...    |
| 0    |     | x(0)   |



## ❑ Comparación

- Loop Unrolling
  - Bloque grande para planificar
  - Reduce el numero de saltos
  - Incrementa el tamaño del código
  - Tiene que incluir iteraciones extra
  - Presión sobre el uso de registros
- Software Pipelining
  - No hay dependencias en el cuerpo del bucle
  - No reduce el numero de saltos
  - Necesita inicio y finalización especial

# Tratamiento de dependencias de datos en ejecución

## ❑ Planificación dinámica : Procesador.

Modifica la secuencia de instrucciones resolviendo las dependencias en tiempo de ejecución. Disponibilidad de más unidades funcionales. Código válido para diferentes implementaciones

## ❑ Problema : Lanzamiento de instrucciones en orden.

|       |            |           |                                  |
|-------|------------|-----------|----------------------------------|
| DIV.D | F0,F2,F4   | <b>S1</b> |                                  |
| ADD.D | F10,F0,F8  | <b>S2</b> | S2 depende de S1                 |
| SUB.D | F12,F8,F14 | <b>S3</b> | S3 es independiente de las demás |

La etapa ID bloquea la ejecución en S2 hasta que se resuelve la dependencia (F0 disponible) y SUB.D no puede ejecutarse.

## ❑ Solución : Dividir la etapa ID en dos etapas diferenciadas.

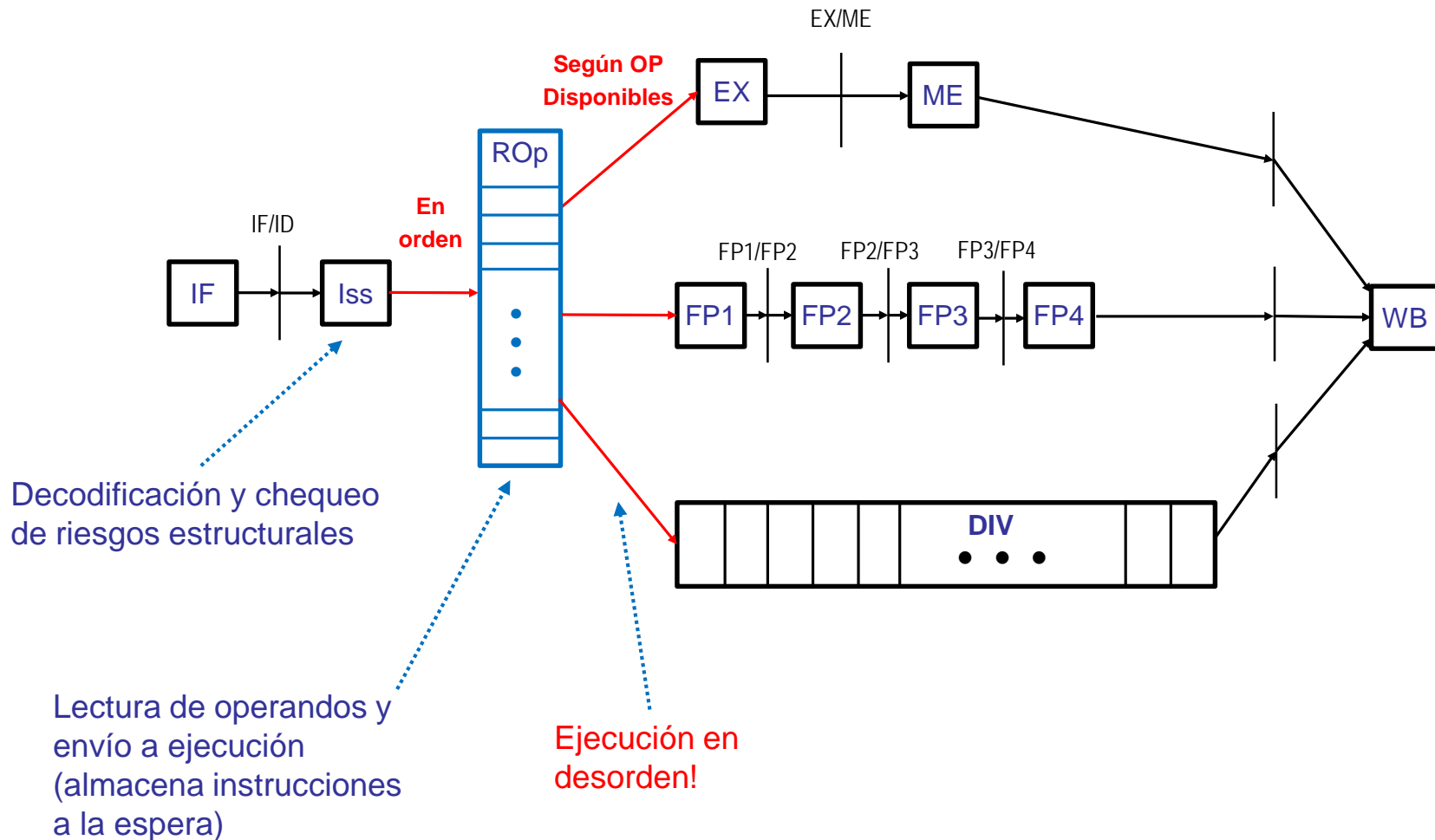
**Issue:** Decodifica y chequea riesgos estructurales.

**Lectura de operandos:** Chequea disponibilidad de operandos. Debe implementarse para permitir el flujo de instrucciones.

**Ejecución fuera de orden → ¿Finalización fuera de orden?**

# Planificación dinámica: Ventajas...Problemas...Soluciones

## ❑ Etapas Issue (Iss) y Lectura de Operandos (ROp) separadas



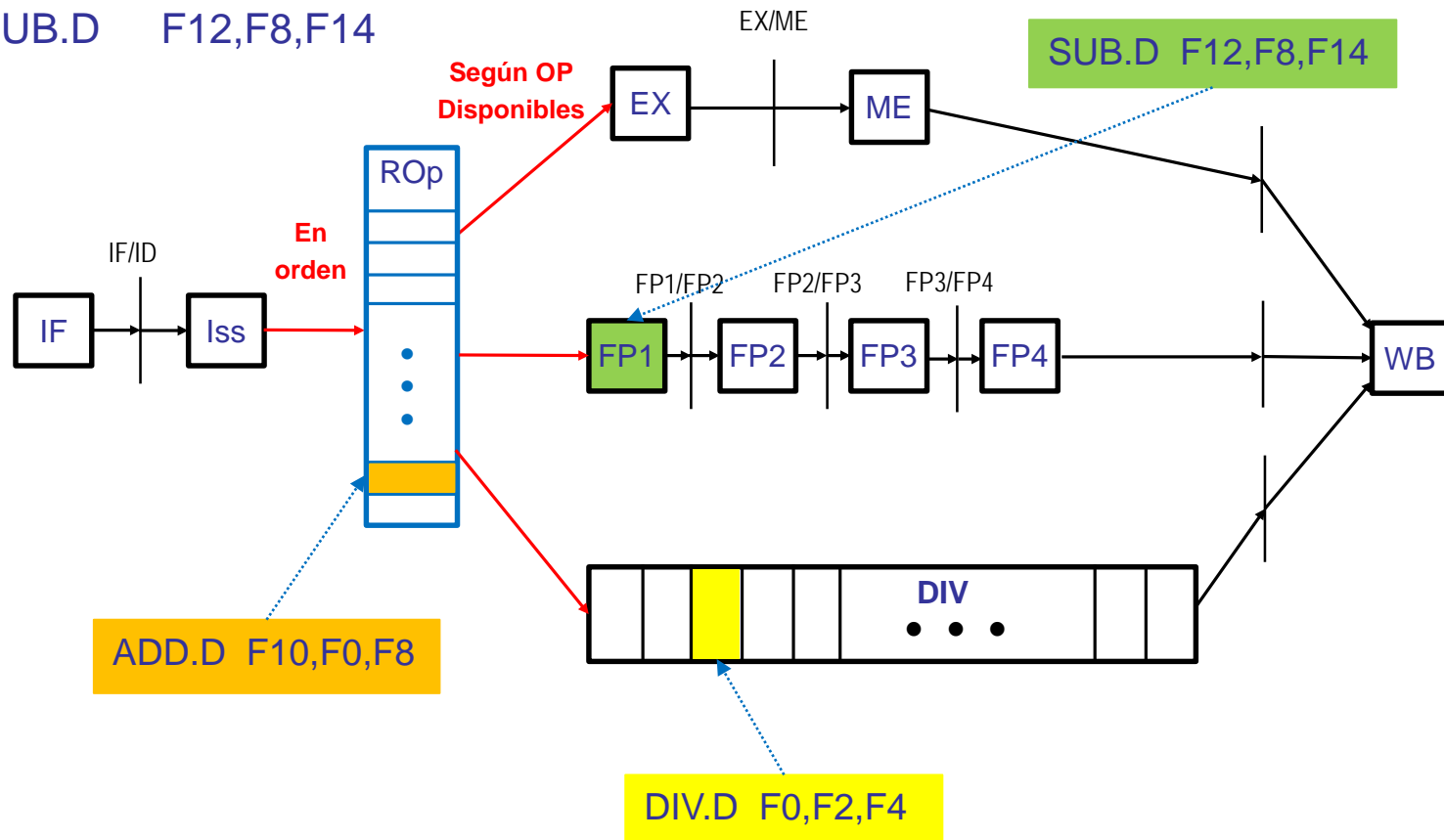


# Planificación dinámica: Ventajas...Problemas...Soluciones

## ❑ Ejecución de la secuencia

DIV.D    F0,F2,F4  
ADD.D    F10,F0,F8  
SUB.D    F12,F8,F14

Adelanta!



SUB.D acabará antes que DIV.D y que ADD.D: Se evita el bloqueo.



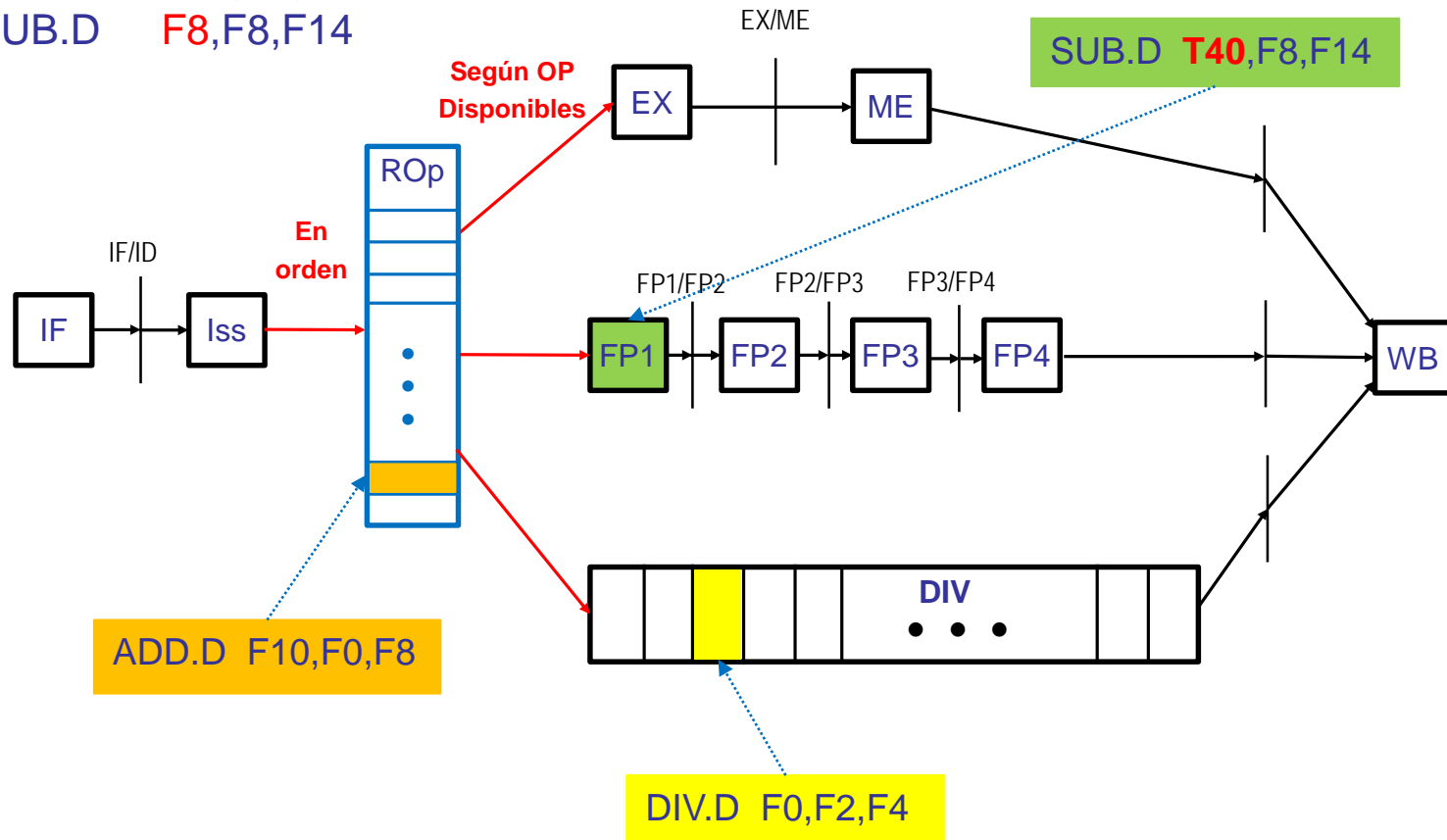
# Planificación dinámica: Ventajas...Problemas...Soluciones

□ Y si la secuencia es ...

DIV.D    F0,F2,F4  
ADD.D    F10,F0,F8  
SUB.D    F8,F8,F14

Podría el Hw suplantar el reg destino de SUB.D?

IDEA: Renombramiento Dinámico



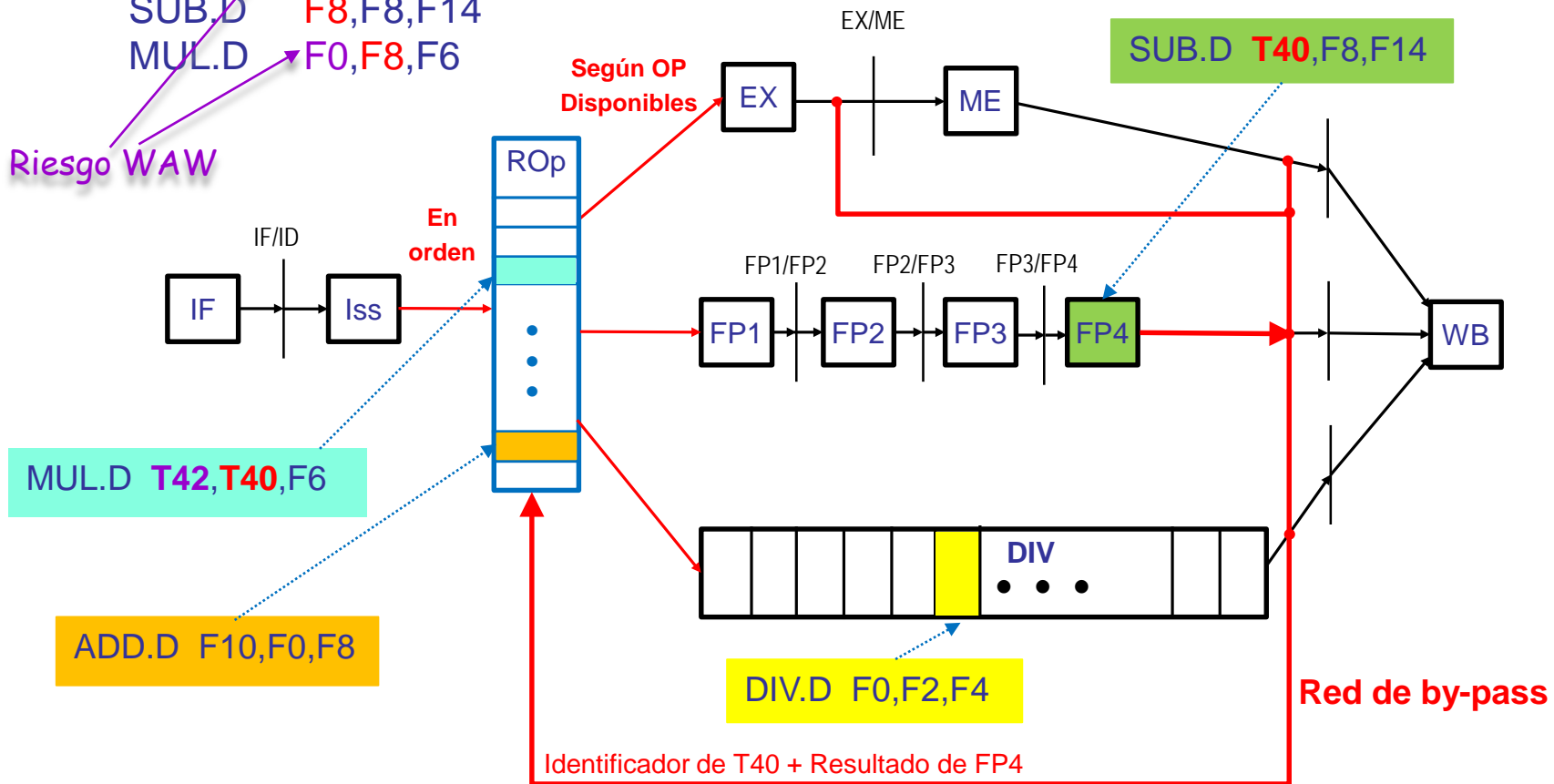
Las instrucciones posteriores al SUB.D que tengan F8 como operando, por supuesto deben usar el valor almacenado en T40! (Hasta cuando?)

# Planificación dinámica: Ventajas...Problemas...Soluciones

## ❑ Y si la secuencia es ...

DIV.D F0,F2,F4  
ADD.D F10,F0,F8  
SUB.D F8,F8,F14  
MUL.D F0,F8,F6

Riesgo WAW



Las instrucciones en la etapa ROp pueden obtener sus operandos a través de la red de by-pass. En el ejemplo, MUL.D está a la espera del valor de T40.

Una instrucción posterior a MUL.D que referencie F0, leerá el resultado de MUL.D, incluso aunque DIV.D acabe más tarde.

# Planificación dinámica: Ventajas...Problemas...Soluciones

- ❑ Múltiples alternativas para implementar las ideas esbozadas en las anteriores transparencias.
- ❑ Esfuerzos desde los años 60
  - o Scoreboard (CDC 6600):
    - Limita el efecto de dependencias RAW (no bloqueo en lanzamiento)
  - o Algoritmo de Tomasulo
    - También limita el efecto de dependencias RAW
    - Además, introduce el concepto de "renombramiento dinámico de registros" → elimina el efecto de dependencias WAR y WAW
    - Complejidad hardware considerable para la época: éxito modesto



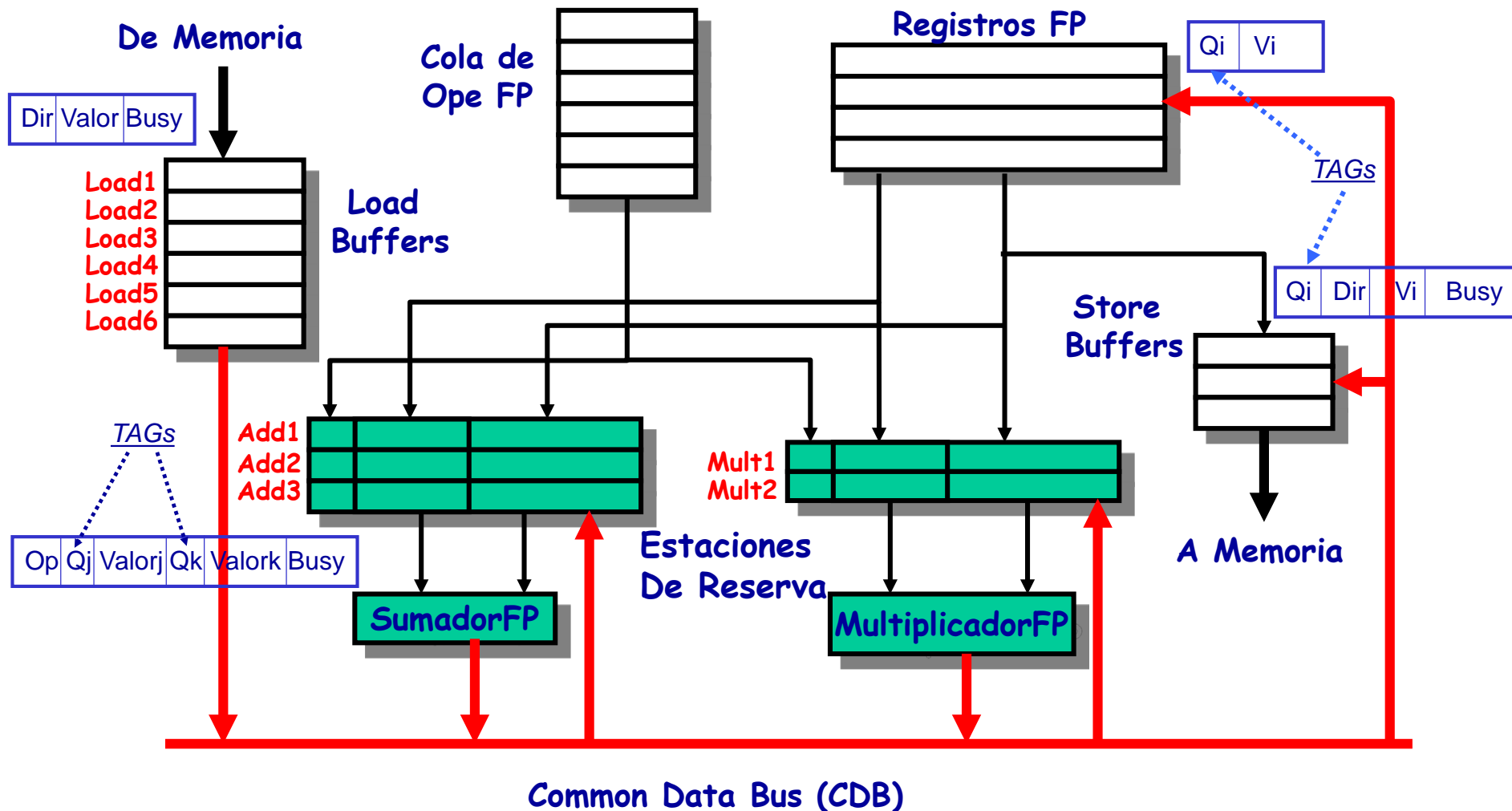
Evolución: Ley de Moore

- Idea implementada (con diversos matices) en casi todos los  $\mu$ -procesadores modernos

# Planificación Dinámica: Tomasulo

- ❑ (IBM 360/91, año 1967)
- ❑ Elimina dinámicamente los riesgos WAW y WAR mediante el renombrado de registros
- ❑ Motivación:
  - Arquitectura 360, **4 registros FP** (Pocos registros → muchas dependencias de nombre)
  - Arquitectura RM
  - UF segmentadas Add(3), Mul(2), Load (6), Stores (3)
  - Requerimientos de diseño: Arquitectura segmentada del alto rendimiento, pero compatible hacia abajo con el resto de máquinas de la familia 360 → no se pueden añadir más registros de FP.
- ❑ Solo consideraremos las operaciones de PF en la explicación
- ❑ Ej: Procesadores que usan mecanismos basados en este algoritmo  
Alpha 21264, Pentium III-4-Core generations, AMD Opteron - Athlon, Zen. IBM Power 4-5-7-8-9...

# Planificación Dinámica : Tomasulo



Excepciones imprecisas

Issue en orden-- Ejecución fuera de orden—Finalización fuera de orden

## ❑ Componentes de las ER

- Op: Operación a realizar
- Valorj, Valork: Valores de los operandos fuente (si están calculados)
- Qj, Qk: N° de la ER que está produciendo los operandos fuente. Notar:  $Qj, Qk=0 \Rightarrow$  operando calculado.
- Busy: Indica ER ocupada

## ❑ Store Buffers

- Vi: Valor que debe ser almacenado en memoria (si está calculado).
- Dir: Dirección de almacenamiento
- Qi: N° de la ER que está produciendo el resultado. Si  $Qi=0 \Rightarrow$  valor calculado

## ❑ Registros FP

- Vi: Valor almacenado en el registro (puede estar obsoleto)
- Qi: N° de la ER que está produciendo el valor a almacenar en el registro. Si  $Qi=0 \Rightarrow$  no hay ninguna ER que vaya a almacenar un nuevo valor en el reg (el valor del registro no está obsoleto)

## ❑ Load buffers

- Valor: Valor leído en la memoria (si ya se ha completado la lectura).
- Dir: Dirección de lectura



# Planificación Dinámica : Tomasulo

## □ Tres estados para una instrucción en el algoritmo

### 1. Issue

- Toma la instrucción (COD  $F_i, F_j, F_k$ ) de la cola de instrucciones.
- Envía la instrucción a la Estación de Reserva (ER) correspondiente si hay entradas disponibles (caso contrario  $\rightarrow$  bloqueo).
- Envía los operandos, si están disponibles, o identificación (tag) de la(s) ER(s) donde están las instrucciones que los van a generar.
- En load/store: envía la instrucción si hay buffer libre.

(En suma: Copiar "Tag+Valor" de registros fuente sobre campos "Tag+Valor" de ER)

- Marca tag de registro destino ( $F_i$ ) con ID de la ER donde se aloja la instrucción

### 2. Ejecución

- Monitoriza CDB para disponibilidad de operandos. Cuando los operandos están listos manda ejecutar. **Resuelve riesgos LDE**

### 3. Escritura de resultados

- Vía CDB en registros y ER(s) que esperan el resultado: Comparación de tags.
- Marca ER como libre.

**No chequea riesgos EDE ni EDL (eliminados por el renombrado dinámico)**

En un bus normal: dato + destino, " va a "

Sin embargo, en CDB: dato + fuente, " viene de "

## □ Ejemplo: Ejecución de la secuencia:

(S1): COD1 F2, ---, ---

(S2): COD2 F4, ---, ---

(S3): ADD.D F0, F2, F4

Suposiciones: S1 y S2 ya lanzadas a ejecución, S3 se va a lanzar

Estado inicial de registros:

|    | TAG | VALOR |
|----|-----|-------|
| F0 | ??  | ??    |
| F2 | Y   | ??    |
| F4 | Z   | ??    |

Instrucción S1  
lanzada a ER Y

Instrucción S2  
lanzada a ER Z

# Planificación Dinámica : Tomasulo

## □ Ejemplo: Ejecución de la secuencia:

(S1): COD1 F2, ---, ---

(S2): COD2 F4, ---, ---

(S3): ADD.D F0, F2, F4

Paso 1: Lanzamiento de ADD.D a la ER X (X es una ER libre de la UF de Suma/Resta de Punto Flotante)

### Estación de reserva

|   | Busy | OP | TAGj | Valorj | TAGk | Valork |
|---|------|----|------|--------|------|--------|
| X | yes  | +  | Y    | ??     | Z    | ??     |

### Estado de registros

|    | TAG | VALOR |
|----|-----|-------|
| F0 | X   | ??    |
| F2 | Y   | ??    |
| F4 | Z   | ??    |

## □ Ejemplo: Ejecución de la secuencia:

(S1): COD1 F2, ---, ---

(S2): COD2 F4, ---, ---

(S3): ADD.D F0, F2, F4

Paso 2: Esperar a que se generen los operandos

a) Escritura del resultado de S1 sobre el CDB: (Y, 22.57)

### Estación de reserva

|   | Busy | OP | TAGj | Valorj | TAGk | Valork |
|---|------|----|------|--------|------|--------|
| X | yes  | +  | 0    | 22.57  | Z    | ??     |

### Estado de registros

|    | TAG | VALOR |
|----|-----|-------|
| F0 | X   | ??    |
| F2 | 0   | 22.57 |
| F4 | Z   | ??    |

b) Escritura del resultado de S2 sobre el CDB: (Z, 3.2)

### Estación de reserva

|   | Busy | OP | TAGj | Valorj | TAGk | Valork |
|---|------|----|------|--------|------|--------|
| X | yes  | +  | 0    | 22.57  | 0    | 3.2    |

### Estado de registros

|    | TAG | VALOR |
|----|-----|-------|
| F0 | X   | ??    |
| F2 | 0   | 22.57 |
| F4 | 0   | 3.2   |

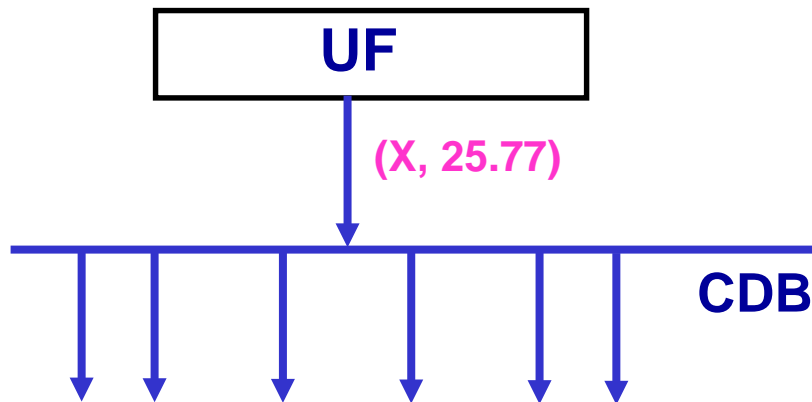
## □ Ejemplo: Ejecución de la secuencia:

(S1): COD1 F2, ---, ---

(S2): COD2 F4, ---, ---

(S3): ADD.D F0, F2, F4

Paso 3: Ejecutar operación y escribir resultado sobre el CDB



A todas las ER, Store Buffers y Registros que tengan la marca X

| Estado de registros |     |       |
|---------------------|-----|-------|
|                     | TAG | VALOR |
| F0                  | 0   | 25.77 |
| F2                  | 0   | 22.57 |
| F4                  | 0   | 3.2   |

# Planificación Dinámica:TOMASULO

- ❑ **Ejemplo** T ejecución: L.D 2 ciclos, ADD.D y SUB.D 2 ciclos, MUL.D 10 ciclos, DIV.D 40 ciclos  
 Memoria segmentada: **acepta comenzar 1 acceso/ciclo**

| Instr |     | J   | K  | Issue | Ejecución | escritura |
|-------|-----|-----|----|-------|-----------|-----------|
| L.D   | F6  | 34+ | R2 |       |           |           |
| L.D   | F2  | 45+ | R3 |       |           |           |
| MUL.D | F0  | F2  | F4 |       |           |           |
| SUB.D | F8  | F6  | F2 |       |           |           |
| DIV.D | F10 | F0  | F6 |       |           |           |
| ADD.D | F6  | F8  | F2 |       |           |           |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 |         |           |
| Load2 |         |           |
| Load3 |         |           |

## Estado de ER

Operación

Qj y Qk: ER produciendo operandos

Vj y Vk: valores de los operandos

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 |         |           |          |          |              |              |
|        | Mul2 |         |           |          |          |              |              |

## Estado de REG

|    | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... |
|----|----|----|----|----|----|-----|-----|-----|
| Qi |    |    |    |    |    |     |     |     |
| Vi |    |    |    |    |    |     |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 1

| Instr |     | J   | K  | Issue | Ejecución | escritura |
|-------|-----|-----|----|-------|-----------|-----------|
| L.D   | F6  | 34+ | R2 | 1     |           |           |
| L.D   | F2  | 45+ | R3 |       |           |           |
| MUL.D | F0  | F2  | F4 |       |           |           |
| SUB.D | F8  | F6  | F2 |       |           |           |
| DIV.D | F10 | F0  | F6 |       |           |           |
| ADD.D | F6  | F8  | F2 |       |           |           |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | SI      | 34+R2     |
| Load2 | NO      |           |
| Load3 | NO      |           |

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 | NO      |           |          |          |              |              |
|        | Add2 | NO      |           |          |          |              |              |
|        | Add3 | NO      |           |          |          |              |              |
|        | Mul1 | NO      |           |          |          |              |              |
|        | Mul2 | NO      |           |          |          |              |              |

Estado de REG

|    | F0 | F2 | F4 | F6    | F8 | F10 | F12 | ... |
|----|----|----|----|-------|----|-----|-----|-----|
| Qi |    |    |    | Load1 |    |     |     |     |
| Vi |    |    |    |       |    |     |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 2

| Instr |     | J   | K  | Issue | Ejecución | escritura |
|-------|-----|-----|----|-------|-----------|-----------|
| L.D   | F6  | 34+ | R2 | 1     |           |           |
| L.D   | F2  | 45+ | R3 | 2     |           |           |
| MUL.D | F0  | F2  | F4 |       |           |           |
| SUB.D | F8  | F6  | F2 |       |           |           |
| DIV.D | F10 | F0  | F6 |       |           |           |
| ADD.D | F6  | F8  | F2 |       |           |           |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | SI      | 34+R2     |
| Load2 | SI      | 45+R3     |
| Load3 | NO      |           |

No hay bloqueo

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 | NO      |           |          |          |              |              |
|        | Add2 | NO      |           |          |          |              |              |
|        | Add3 | NO      |           |          |          |              |              |
|        | Mul1 | NO      |           |          |          |              |              |
|        | Mul2 | NO      |           |          |          |              |              |

Estado de REG

|    | F0 | F2    | F4 | F6    | F8 | F10 | F12 | ... |
|----|----|-------|----|-------|----|-----|-----|-----|
| Qi |    | Load2 |    | Load1 |    |     |     |     |
| Vi |    |       |    |       |    |     |     |     |



# Planificación Dinámica:TOMASULO

## ❑ Ciclo 3

| Instr |     | J   | K  | Issue | Ejecución | escritura |
|-------|-----|-----|----|-------|-----------|-----------|
| L.D   | F6  | 34+ | R2 | 1     | 2-3       |           |
| L.D   | F2  | 45+ | R3 | 2     |           |           |
| MUL.D | F0  | F2  | F4 | 3     |           |           |
| SUB.D | F8  | F6  | F2 |       |           |           |
| DIV.D | F10 | F0  | F6 |       |           |           |
| ADD.D | F6  | F8  | F2 |       |           |           |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | SI      | 34+R2     |
| Load2 | SI      | 45+R3     |
| Load3 | NO      |           |

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 | NO      |           |          |          |              |              |
|        | Add2 | NO      |           |          |          |              |              |
|        | Add3 | NO      |           |          |          |              |              |
|        | Mul1 | SI      | Mult      |          | R(F4)    | Load2        |              |
|        | Mul2 | NO      |           |          |          |              |              |

Estado de REG

|    | F0   | F2    | F4 | F6    | F8 | F10 | F12 | ... |
|----|------|-------|----|-------|----|-----|-----|-----|
| Qi | Mul1 | Load2 |    | Load1 |    |     |     |     |
| Vi |      |       |    |       |    |     |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 4

| Instr |     | J   | K  | Issue | Ejecución | escritura |
|-------|-----|-----|----|-------|-----------|-----------|
| L.D   | F6  | 34+ | R2 | 1     | 2-3       | 4         |
| L.D   | F2  | 45+ | R3 | 2     | 3-4       |           |
| MUL.D | F0  | F2  | F4 | 3     |           |           |
| SUB.D | F8  | F6  | F2 | 4     |           |           |
| DIV.D | F10 | F0  | F6 |       |           |           |
| ADD.D | F6  | F8  | F2 |       |           |           |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | NO      |           |
| Load2 | SI      | 45+R3     |
| Load3 | NO      |           |

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 | SI      | Subd      | M(34+R2) |          |              | Load2        |
|        | Add2 | NO      |           |          |          |              |              |
|        | Add3 | NO      |           |          |          |              |              |
|        | Mul1 | SI      | Mult      |          | R(F4)    | Load2        |              |
|        | Mul2 | NO      |           |          |          |              |              |

Estado de REG

|    | F0   | F2    | F4 | F6       | F8   | F10 | F12 | ... |
|----|------|-------|----|----------|------|-----|-----|-----|
| Qi | Mul1 | Load2 |    |          | Add1 |     |     |     |
| Vi |      |       |    | M(34+R2) |      |     |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 6

| Instr |     | J   | K  | Issue | Ejecución | escritura |
|-------|-----|-----|----|-------|-----------|-----------|
| L.D   | F6  | 34+ | R2 | 1     | 2-3       | 4         |
| L.D   | F2  | 45+ | R3 | 2     | 3-4       | 5         |
| MUL.D | F0  | F2  | F4 | 3     |           |           |
| SUB.D | F8  | F6  | F2 | 4     |           |           |
| DIV.D | F10 | F0  | F6 | 5     |           |           |
| ADD.D | F6  | F8  | F2 | 6     |           |           |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | NO      |           |
| Load2 | NO      |           |
| Load3 | NO      |           |

Ningún bloqueo

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
| 2      | Add1 | SI      | Subd      | M(34+R2) | M(45+R3) |              |              |
|        | Add2 | SI      | Addd      |          | M(45+R3) | Add1         |              |
|        | Add3 | NO      |           |          |          |              |              |
| 10     | Mul1 | SI      | Mult      | M(45+R3) | R(F4)    |              |              |
|        | Mul2 | SI      | Divd      |          | M(34+R2) | Mul1         |              |

Estado de REG

|    | F0   | F2       | F4 | F6   | F8   | F10  | F12 | ... |
|----|------|----------|----|------|------|------|-----|-----|
| Qi | Mul1 |          |    | Add2 | Add1 | Mul2 |     |     |
| Vi |      | M(45+R3) |    |      |      |      |     |     |

F6 reasignado

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 8

| Instur |     | J   | K  | Issue | Ejecución | escritura |
|--------|-----|-----|----|-------|-----------|-----------|
| L.D    | F6  | 34+ | R2 | 1     | 2-3       | 4         |
| L.D    | F2  | 45+ | R3 | 2     | 3-4       | 5         |
| MUL.D  | F0  | F2  | F4 | 3     |           |           |
| SUB.D  | F8  | F6  | F2 | 4     | 6-7       | 8         |
| DIV.D  | F10 | F0  | F6 | 5     |           |           |
| ADD.D  | F6  | F8  | F2 | 6     |           |           |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | NO      |           |
| Load2 | NO      |           |
| Load3 | NO      |           |

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 | NO      |           |          |          |              |              |
| 2      | Add2 | SI      | Addd      | M()-M()  | M(45+R3) |              |              |
|        | Add3 | NO      |           |          |          |              |              |
| 7      | Mul1 | SI      | Mult      | M(45+R3) | R(F4)    |              |              |
|        | Mul2 | SI      | Divd      |          | M(34+R2) | Mul1         |              |

Estado de REG

|    | F0   | F2       | F4 | F6   | F8      | F10  | F12 | ... |
|----|------|----------|----|------|---------|------|-----|-----|
| Qi | Mul1 |          |    | Add2 |         | Mul2 |     |     |
| Vi |      | M(45+R3) |    |      | M()-M() |      |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 13

| Instr |     | J   | K  | Issue | Ejecución | escritura |
|-------|-----|-----|----|-------|-----------|-----------|
| L.D   | F6  | 34+ | R2 | 1     | 2-3       | 4         |
| L.D   | F2  | 45+ | R3 | 2     | 3-4       | 5         |
| MUL.D | F0  | F2  | F4 | 3     |           |           |
| SUB.D | F8  | F6  | F2 | 4     | 6-7       | 8         |
| DIV.D | F10 | F0  | F6 | 5     |           |           |
| ADD.D | F6  | F8  | F2 | 6     | 9-10      | 11        |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | NO      |           |
| Load2 | NO      |           |
| Load3 | NO      |           |

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 | NO      |           |          |          |              |              |
|        | Add2 | NO      |           |          |          |              |              |
|        | Add3 | NO      |           |          |          |              |              |
| 3      | Mul1 | SI      | Mult      | M(45+R3) | R(F4)    |              |              |
|        | Mul2 | SI      | Divd      |          | M(34+R2) | Mul1         |              |

Estado de REG

|    | F0   | F2       | F4 | F6     | F8      | F10  | F12 | ... |
|----|------|----------|----|--------|---------|------|-----|-----|
| Qi | Mul1 |          |    |        |         | Mul2 |     |     |
| Vi |      | M(45+R3) |    | F8+M() | M()-M() |      |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 16

| Instr |     | J   | K  | Issue | Ejecución | escritura |
|-------|-----|-----|----|-------|-----------|-----------|
| L.D   | F6  | 34+ | R2 | 1     | 2-3       | 4         |
| L.D   | F2  | 45+ | R3 | 2     | 3-4       | 5         |
| MUL.D | F0  | F2  | F4 | 3     | 6-15      | 16        |
| SUB.D | F8  | F6  | F2 | 4     | 6-7       | 8         |
| DIV.D | F10 | F0  | F6 | 5     |           |           |
| ADD.D | F6  | F8  | F2 | 6     | 9-10      | 11        |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | NO      |           |
| Load2 | NO      |           |
| Load3 | NO      |           |

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 | NO      |           |          |          |              |              |
|        | Add2 | NO      |           |          |          |              |              |
|        | Add3 | NO      |           |          |          |              |              |
|        | Mul1 | NO      |           |          |          |              |              |
| 40     | Mul2 | SI      | Divd      | M()*F4   | M(34+R2) |              |              |

Estado de REG

|    | F0     | F2       | F4 | F6     | F8      | F10  | F12 | ... |
|----|--------|----------|----|--------|---------|------|-----|-----|
| Qi |        |          |    |        |         | Mul2 |     |     |
| Vi | M()*F4 | M(45+R3) |    | F8+M() | M()-M() |      |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 57

Finalización en desorden ☒ Excepciones

| Instr |     | J   | K  | Issue | Ejecución. | escritura |
|-------|-----|-----|----|-------|------------|-----------|
| L.D   | F6  | 34+ | R2 | 1     | 2-3        | 4         |
| L.D   | F2  | 45+ | R3 | 2     | 3-4        | 5         |
| MUL.D | F0  | F2  | F4 | 3     | 6-15       | 16        |
| SUB.D | F8  | F6  | F2 | 4     | 6-7        | 8         |
| DIV.D | F10 | F0  | F6 | 5     | 17-56      | 57        |
| ADD.D | F6  | F8  | F2 | 6     | 9-10       | 11        |

|       | Ocupada | Dirección |
|-------|---------|-----------|
| Load1 | NO      |           |
| Load2 | NO      |           |
| Load3 | NO      |           |

Estado de ER

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P.J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 | NO      |           |          |          |              |              |
|        | Add2 | NO      |           |          |          |              |              |
|        | Add3 | NO      |           |          |          |              |              |
|        | Mul1 | NO      |           |          |          |              |              |
|        | Mul2 | NO      |           |          |          |              |              |

Estado de REG

|    | F0     | F2       | F4 | F6     | F8      | F10   | F12 | ... |
|----|--------|----------|----|--------|---------|-------|-----|-----|
| Qi |        |          |    |        |         |       |     |     |
| Vi | M()*F4 | M(45+R3) |    | F8+M() | M()-M() | F0/F6 |     |     |

## ❑ Renombrado dinámico en un unrolling

```
Loop: L.D      F0,0(R1)
      MUL.D    F4,F0,F2
      S.D      F4,0(R1)
      DADDIU   R1,R1,#-8
      BNE      R1,R2,Loop
```

Operación: vector F0 \* escalar F2

### Suposiciones:

Predicción: el salto se toma

MUL.D tarda 4 ciclos

Memoria: no segmentada, T acceso 1 ciclo

L.D en 1ª iteración tarda 8 ciclos (fallo). En 2ª iteración: tarda 1 ciclo (acierto)

Mostraremos dos iteraciones



# Planificación Dinámica:TOMASULO

## ❑ Bucle

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 |       |           |           |
| MUL.D | F4 | F0 | F2 |       |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |
| L.D   | F0 | 0  | R1 |       |           |           |
| MUL.D | F4 | F0 | F2 |       |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |

|        | Ocupada | Direc | Vi | Qi |
|--------|---------|-------|----|----|
| Load1  | NO      |       |    |    |
| Load2  | NO      |       |    |    |
| Load3  | NO      |       |    |    |
| Store1 | NO      |       |    |    |
| Store2 | NO      |       |    |    |
| Store3 | NO      |       |    |    |

### Estado de ER

Loop: L.D      F0,0(R1)  
 MUL.D    F4,F0,F2  
 S.D        F4,0(R1)  
 DADDIU   R1,R1,#-8  
 BNE       R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 |         |           |          |          |              |              |
|        | Mul2 |         |           |          |          |              |              |

### Estado de REG

|    | F0 | F2 | F4 | F6 | F8 | F10 | F12 | ... |
|----|----|----|----|----|----|-----|-----|-----|
| Qi |    |    |    |    |    |     |     |     |
| Vi |    |    |    |    |    |     |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 1

| Instr |    | J  | K  | Issue | ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     |           |           |
| MUL.D | F4 | F0 | F2 |       |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |
| L.D   | F0 | 0  | R1 |       |           |           |
| MUL.D | F4 | F0 | F2 |       |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |

|        | Ocupada | Direc | Vi | Qi |
|--------|---------|-------|----|----|
| Load1  | SI      | 80    |    |    |
| Load2  | NO      |       |    |    |
| Load3  | NO      |       |    |    |
| Store1 | NO      |       |    |    |
| Store2 | NO      |       |    |    |
| Store3 | NO      |       |    |    |

### Estado de ER

Loop: L.D F0,0(R1)  
 MUL.D F4,F0,F2  
 S.D F4,0(R1)  
 DADDIU R1,R1,#-8  
 BNE R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 |         |           |          |          |              |              |
|        | Mul2 |         |           |          |          |              |              |

### Estado de REG R1=80

|    | F0    | F2 | F4 | F6 | F8 | F10 | F12 | ... |
|----|-------|----|----|----|----|-----|-----|-----|
| Qi | Load1 |    |    |    |    |     |     |     |
| Vi |       |    |    |    |    |     |     |     |

Ojo latencia del primer load

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 2

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     |           |           |
| MUL.D | F4 | F0 | F2 | 2     |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |
| L.D   | F0 | 0  | R1 |       |           |           |
| MUL.D | F4 | F0 | F2 |       |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |

|        | Ocupada | Direc | Vi | Qi |
|--------|---------|-------|----|----|
| Load1  | SI      | 80    |    |    |
| Load2  | NO      |       |    |    |
| Load3  | NO      |       |    |    |
| Store1 | NO      |       |    |    |
| Store2 | NO      |       |    |    |
| Store3 | NO      |       |    |    |

### Estado de ER

Loop: L.D      F0,0(R1)  
 MUL.D    F4,F0,F2  
 S.D        F4,0(R1)  
 DADDIU   R1,R1,#-8  
 BNE       R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 | SI      | MULT      |          | R(F2)    | Load1        |              |
|        | Mul2 |         |           |          |          |              |              |

Estado de REG  
 R1=80

|    | F0    | F2 | F4   | F6 | F8 | F10 | F12 | ... |
|----|-------|----|------|----|----|-----|-----|-----|
| Qi | Load1 |    | Mul1 |    |    |     |     |     |
| Vi |       |    |      |    |    |     |     |     |

Ojo latencia del 1er load

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 3

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     |           |           |
| MUL.D | F4 | F0 | F2 | 2     |           |           |
| S.D   | F4 | 0  | R1 | 3     |           |           |
| L.D   | F0 | 0  | R1 |       |           |           |
| MUL.D | F4 | F0 | F2 |       |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |

|        | Ocupada | Direc | Vi | Qi   |
|--------|---------|-------|----|------|
| Load1  | SI      | 80    |    |      |
| Load2  | NO      |       |    |      |
| Load3  | NO      |       |    |      |
| Store1 | SI      | 80    |    | Mul1 |
| Store2 | NO      |       |    |      |
| Store3 | NO      |       |    |      |

### Estado de ER

Loop: L.D      F0,0(R1)  
       MUL.D    F4,F0,F2  
       S.D       F4,0(R1)  
       DADDIU   R1,R1,#-8  
       BNE      R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 | SI      | MULT      |          | R(F2)    | Load1        |              |
|        | Mul2 |         |           |          |          |              |              |

Estado de REG  
 R1=80

Ojo latencia del 1er load

|    | F0    | F2 | F4   | F6 | F8 | F10 | F12 | ... |
|----|-------|----|------|----|----|-----|-----|-----|
| Qi | Load1 |    | Mul1 |    |    |     |     |     |
| Vi |       |    |      |    |    |     |     |     |

# Planificación Dinámica:TOMASULO

❑ Ciclo 6: se lanza LD de 2ª iteración (IMPORTANTE: Dependencia EDE !! )

Cierre del bucle

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     |           |           |
| MUL.D | F4 | F0 | F2 | 2     |           |           |
| S.D   | F4 | 0  | R1 | 3     |           |           |
| L.D   | F0 | 0  | R1 | 6     |           |           |
| MUL.D | F4 | F0 | F2 |       |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |

|        | Ocupada | Direc | Vi | Qi   |
|--------|---------|-------|----|------|
| Load1  | SI      | 80    |    |      |
| Load2  | SI      | 72    |    |      |
| Load3  | NO      |       |    |      |
| Store1 | SI      | 80    |    | Mul1 |
| Store2 | NO      |       |    |      |
| Store3 | NO      |       |    |      |

Estado de ER

Loop: L.D F0,0(R1)  
 MUL.D F4,F0,F2  
 S.D F4,0(R1)  
 DADDIU R1,R1,#-8  
 BNE R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 | SI      | MULT      |          | R(F2)    | Load1        |              |
|        | Mul2 |         |           |          |          |              |              |

Estado de REG  
 R1=72

|    | F0    | F2 | F4   | F6 | F8 | F10 | F12 | ... |
|----|-------|----|------|----|----|-----|-----|-----|
| Qi | Load2 |    | Mul1 |    |    |     |     |     |
| Vi |       |    |      |    |    |     |     |     |

Renombrado

Ojo latencia del 1er load

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 7

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     |           |           |
| MUL.D | F4 | F0 | F2 | 2     |           |           |
| S.D   | F4 | 0  | R1 | 3     |           |           |
| L.D   | F0 | 0  | R1 | 6     |           |           |
| MUL.D | F4 | F0 | F2 | 7     |           |           |
| S.D   | F4 | 0  | R1 |       |           |           |

|        | Ocupada | Direc | Vi | Qi   |
|--------|---------|-------|----|------|
| Load1  | SI      | 80    |    |      |
| Load2  | SI      | 72    |    |      |
| Load3  | NO      |       |    |      |
| Store1 | SI      | 80    |    | Mul1 |
| Store2 | NO      |       |    |      |
| Store3 | NO      |       |    |      |

### Estado de ER

Loop: L.D      F0,0(R1)  
       MUL.D    F4,F0,F2  
       S.D       F4,0(R1)  
       DADDIU   R1,R1,#-8  
       BNE       R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 | SI      | MULT      |          | R(F2)    | Load1        |              |
|        | Mul2 | SI      | MULT      |          | R(F2)    | Load2        |              |

Estado de REG  
 R1=72

|    | F0    | F2 | F4   | F6 | F8 | F10 | F12 | ... |
|----|-------|----|------|----|----|-----|-----|-----|
| Qi | Load2 |    | Mul2 |    |    |     |     |     |
| Vi |       |    |      |    |    |     |     |     |

Renombrado

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 10: Finaliza 1er LD

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     | 9         | 10        |
| MUL.D | F4 | F0 | F2 | 2     |           |           |
| S.D   | F4 | 0  | R1 | 3     |           |           |
| L.D   | F0 | 0  | R1 | 6     | 10        |           |
| MUL.D | F4 | F0 | F2 | 7     |           |           |
| S.D   | F4 | 0  | R1 | 8     |           |           |

|        | Ocupada | Direc | Vi | Qi   |
|--------|---------|-------|----|------|
| Load1  | NO      |       |    |      |
| Load2  | SI      | 72    |    |      |
| Load3  | NO      |       |    |      |
| Store1 | SI      | 80    |    | Mul1 |
| Store2 | SI      | 72    |    | Mul2 |
| Store3 | NO      |       |    |      |

### Estado de ER

Loop: L.D      F0,0(R1)  
       MUL.D    F4,F0,F2  
       S.D       F4,0(R1)  
       DADDIU   R1,R1,#-8  
       BNE      R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
| 4      | Mul1 | SI      | MULT      | M(80)    | R(F2)    |              |              |
|        | Mul2 | SI      | MULT      |          | R(F2)    | Load2        |              |

Estado de REG  
 R1=64

|    | F0    | F2 | F4   | F6 | F8 | F10 | F12 | ... |
|----|-------|----|------|----|----|-----|-----|-----|
| Qi | Load2 |    | Mul2 |    |    |     |     |     |
| Vi |       |    |      |    |    |     |     |     |

Ojo latencia del 1er load

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 11: Finaliza 2º LD

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     | 9         | 10        |
| MUL.D | F4 | F0 | F2 | 2     |           |           |
| S.D   | F4 | 0  | R1 | 3     |           |           |
| L.D   | F0 | 0  | R1 | 6     | 10        | 11        |
| MUL.D | F4 | F0 | F2 | 7     |           |           |
| S.D   | F4 | 0  | R1 | 8     |           |           |

|        | Ocupada | Direc | Vi | Qi   |
|--------|---------|-------|----|------|
| Load1  | NO      |       |    |      |
| Load2  | NO      |       |    |      |
| Load3  | SI      | 64    |    |      |
| Store1 | SI      | 80    |    | Mul1 |
| Store2 | SI      | 72    |    | Mul2 |
| Store3 | NO      |       |    |      |

### Estado de ER

Loop: L.D      F0,0(R1)  
       MUL.D    F4,F0,F2  
       S.D        F4,0(R1)  
       DADDIU   R1,R1,#-8  
       BNE        R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
| 3      | Mul1 | SI      | MULT      | M(80)    | R(F2)    |              |              |
| 4      | Mul2 | SI      | MULT      | M(72)    | R(F2)    |              |              |

### Estado de REG R1=64

|    | F0    | F2 | F4   | F6 | F8 | F10 | F12 | ... |
|----|-------|----|------|----|----|-----|-----|-----|
| Qi |       |    | Mul2 |    |    |     |     |     |
| Vi | M(72) |    |      |    |    |     |     |     |



# Planificación Dinámica:TOMASULO

## ❑ Ciclo 15: Finaliza 1er MULT

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     | 9         | 10        |
| MUL.D | F4 | F0 | F2 | 2     | 11-14     | 15        |
| S.D   | F4 | 0  | R1 | 3     |           |           |
| L.D   | F0 | 0  | R1 | 6     | 10        | 11        |
| MUL.D | F4 | F0 | F2 | 7     | 12-15     |           |
| S.D   | F4 | 0  | R1 | 8     |           |           |

|        | Ocupada | Direc | Vi     | Qi   |
|--------|---------|-------|--------|------|
| Load1  | NO      |       |        |      |
| Load2  | NO      |       |        |      |
| Load3  | SI      | 64    |        |      |
| Store1 | SI      | 80    | M()*F2 |      |
| Store2 | SI      | 72    |        | Mul2 |
| Store3 | NO      |       |        |      |

### Estado de ER

Loop: L.D F0,0(R1)  
 MUL.D F4,F0,F2  
 S.D F4,0(R1)  
 DADDIU R1,R1,#-8  
 BNE R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 |         |           |          |          |              |              |
|        | Mul2 | SI      | MULT      | M(72)    | R(F2)    |              |              |

### Estado de REG R1=64

|    | F0    | F2 | F4   | F6 | F8 | F10 | F12 | ... |
|----|-------|----|------|----|----|-----|-----|-----|
| Qi |       |    | Mul2 |    |    |     |     |     |
| Vi | M(72) |    |      |    |    |     |     |     |

# Planificación Dinámica:TOMASULO

## ❑ Ciclo 16: Finaliza 2º MULT, se ejecuta 1er ST

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     | 9         | 10        |
| MUL.D | F4 | F0 | F2 | 2     | 11-14     | 15        |
| S.D   | F4 | 0  | R1 | 3     | 16        | --        |
| L.D   | F0 | 0  | R1 | 6     | 10        | 11        |
| MUL.D | F4 | F0 | F2 | 7     | 12-15     | 16        |
| S.D   | F4 | 0  | R1 | 8     |           |           |

|        | Ocupada | Direc | Vi     | Qi |
|--------|---------|-------|--------|----|
| Load1  | NO      |       |        |    |
| Load2  | NO      |       |        |    |
| Load3  | SI      | 64    |        |    |
| Store1 | NO      |       |        |    |
| Store2 | SI      | 72    | M()*F2 |    |
| Store3 | NO      |       |        |    |

### Estado de ER

Loop: L.D      F0,0(R1)  
 MUL.D    F4,F0,F2  
 S.D        F4,0(R1)  
 DADDIU   R1,R1,#-8  
 BNE       R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 |         |           |          |          |              |              |
|        | Mul2 |         |           |          |          |              |              |

### Estado de REG R1=64

|    | F0    | F2 | F4     | F6 | F8 | F10 | F12 | ... |
|----|-------|----|--------|----|----|-----|-----|-----|
| Qi |       |    |        |    |    |     |     |     |
| Vi | M(72) |    | M()*F2 |    |    |     |     |     |

# Planificación Dinámica:TOMASULO

❑ Ciclo 17: se ejecuta 2º store

| Instr |    | J  | K  | Issue | Ejecución | escritura |
|-------|----|----|----|-------|-----------|-----------|
| L.D   | F0 | 0  | R1 | 1     | 9         | 10        |
| MUL.D | F4 | F0 | F2 | 2     | 11-14     | 15        |
| S.D   | F4 | 0  | R1 | 3     | 16        | --        |
| L.D   | F0 | 0  | R1 | 6     | 10        | 11        |
| MUL.D | F4 | F0 | F2 | 7     | 12-15     | 16        |
| S.D   | F4 | 0  | R1 | 8     | 17        | --        |

|        | Ocupada | Direc | Vi | Qi |
|--------|---------|-------|----|----|
| Load1  | NO      |       |    |    |
| Load2  | NO      |       |    |    |
| Load3  | SI      | 64    |    |    |
| Store1 | NO      |       |    |    |
| Store2 | NO      |       |    |    |
| Store3 | NO      |       |    |    |

Estado de ER

Loop: L.D F0,0(R1)  
 MUL.D F4,F0,F2  
 S.D F4,0(R1)  
 DADDIU R1,R1,#-8  
 BNE R1,R2,Loop

| Tiempo | FU   | Ocupada | Operación | S1<br>Vj | S2<br>Vk | ER.P:J<br>Qj | ER.P:K<br>Qk |
|--------|------|---------|-----------|----------|----------|--------------|--------------|
|        | Add1 |         |           |          |          |              |              |
|        | Add2 |         |           |          |          |              |              |
|        | Add3 |         |           |          |          |              |              |
|        | Mul1 |         |           |          |          |              |              |
|        | Mul2 |         |           |          |          |              |              |

Estado de REG  
 R1=64

|    | F0    | F2 | F4     | F6 | F8 | F10 | F12 | ... |
|----|-------|----|--------|----|----|-----|-----|-----|
| Qi |       |    |        |    |    |     |     |     |
| Vi | M(72) |    | M()*F2 |    |    |     |     |     |

## □ Nomenclatura:

o Instrucción aritmética:  $D \leftarrow OP(S1, S2)$

- Instrucción aritmética que realiza la operación OP sobre el contenido de los registros S1 y S2, y deposita el resultado en el registro D

o Instrucción Load:  $D \leftarrow Mem[Dirección]$

- Carga en el registro D el contenido de la posición de memoria "Dirección"

o Instrucción Store:  $Mem[Dirección] \leftarrow S$

- Almacena el contenido del registro S en la posición de memoria "Dirección"

o Estación de reserva x:  $ER(x)(Busy, OP, Q_j, V_j, Q_k, V_k)$

o Registro x:  $Reg(x)(Q_i, V_i)$

o Load Buffer x:  $LB(x)(Busy, Dir)$

o Store Buffer x:  $SB(x)(Busy, Dir, Q_i, V_i)$

# Planificación Dinámica: TOMASULO (detalle de las fases)

## □ Fase Issue

| Tipo de instrucción                      | Esperar hasta que ...   | Hacer ...  |
|--|---|--|
| Aritmética:<br>$D \leftarrow OP(S1, S2)$ | La estación de reserva $ER(x)$ está libre y es capaz de ejecutar $OP$ | $ER(x).Busy \leftarrow Yes$<br>$ER(x).OP \leftarrow OP$<br>$ER(x).Q_j \leftarrow Reg(S1).Q_i$<br>$ER(x).V_j \leftarrow Reg(S1).V_i$<br>$ER(x).Q_k \leftarrow Reg(S2).Q_i$<br>$ER(x).V_k \leftarrow Reg(S2).V_i$<br><br>$Reg(D).Q_i \leftarrow x$ |
| Load:<br>$D \leftarrow Mem[Dirección]$   | El Load Buffer $LB(x)$ está libre                                     | $LB(x).Busy \leftarrow Yes$<br>$LB(x).Dir \leftarrow Dirección$<br><br>$Reg(D).Q_i \leftarrow x$   |
| Store:<br>$Mem[Dirección] \leftarrow S$  | El Store Buffer $SB(x)$ está libre                                    | $SB(x).Busy \leftarrow Yes$<br>$SB(x).Dir \leftarrow Dirección$<br>$SB(x).Q_i \leftarrow Reg(S).Q_i$<br>$SB(x).V_i \leftarrow Reg(S).V_i$  |

# Planificación Dinámica: TOMASULO (detalle de las fases)

## □ Fase Ejecución

| Tipo de instrucción                      | Esperar hasta que ...  | Hacer ...  |
|--|--|--|
| Aritmética:<br>$D \leftarrow OP(S1, S2)$ | $(ER(x).Q_j = 0) \text{ Y } (ER(x).Q_k = 0)$   | Ejecutar cálculo OP sobre la UF<br>usando operandos $V_j$ y $V_k$<br>Generar RESULTADO |
| Load:<br>$D \leftarrow Mem[Dirección]$   | (La dirección efectiva está<br>disponible) Y<br>(LB(x).Dir no tiene<br>dependencias respecto de<br>Stores lanzados antes)<br><br>(Uso de cola de Load / Store:<br>explicación posterior) | RESULTADO $\leftarrow Mem[LB(x).Dir]$  |
| Store:<br>$Mem[Dirección] \leftarrow S$  | (La dirección efectiva está<br>disponible) Y<br>(SB(x).Dir no tiene<br>dependencias con Load ni<br>Stores previos) Y<br>(SB(x).Q <sub>i</sub> ) = 0)                                     | $Mem[SB(x).Dir] \leftarrow SB(x).V_i$<br>SB(x).Busy $\leftarrow$ No                    |

# Planificación Dinámica: TOMASULO (detalle de las fases)

## □ Fase Write

| Tipo de instrucción                                   | Esperar hasta que ...                                      | Hacer ...  |
|---|--|--|
| Aritmética:<br>$D \leftarrow OP(S1, S2)$              | (Ejecución completa en $ER(x)$ ) Y (CDB disponible)        | <u>Escribir sobre CDB: (x, RESUL)</u><br>$\forall z (Si \text{ Reg}(z).Q_i = x) \Rightarrow (\text{Reg}(z).Q_i = 0) \text{ Y } (\text{Reg}(z).V_i = \text{RESUL})$<br>$\forall z (Si \text{ ER}(z).Q_j = x) \Rightarrow (\text{ER}(z).Q_j = 0) \text{ Y } (\text{ER}(z).V_j = \text{RESUL})$<br>$\forall z (Si \text{ ER}(z).Q_k = x) \Rightarrow (\text{ER}(z).Q_k = 0) \text{ Y } (\text{ER}(z).V_k = \text{RESUL})$<br>$\forall z (Si \text{ SB}(z).Q_i = x) \Rightarrow (\text{SB}(z).Q_i = 0) \text{ Y } (\text{SB}(z).V_i = \text{RESUL})$<br><br>$ER(x).Busy = \text{No}$ |
| Load:<br>$D \leftarrow \text{Mem}[\text{Dirección}]$  | (Acceso a memoria completo en $LB(x)$ ) Y (CDB disponible) | <u>Escribir sobre CDB: (x, RESUL)</u><br>$\forall z (Si \text{ Reg}(z).Q_i = x) \Rightarrow (\text{Reg}(z).Q_i = 0) \text{ Y } (\text{Reg}(z).V_i = \text{RESUL})$<br>$\forall z (Si \text{ ER}(z).Q_j = x) \Rightarrow (\text{ER}(z).Q_j = 0) \text{ Y } (\text{ER}(z).V_j = \text{RESUL})$<br>$\forall z (Si \text{ ER}(z).Q_k = x) \Rightarrow (\text{ER}(z).Q_k = 0) \text{ Y } (\text{ER}(z).V_k = \text{RESUL})$<br>$\forall z (Si \text{ SB}(z).Q_i = x) \Rightarrow (\text{SB}(z).Q_i = 0) \text{ Y } (\text{SB}(z).V_i = \text{RESUL})$<br><br>$LB(x).Busy = \text{No}$ |
| Store:<br>$\text{Mem}[\text{Dirección}] \leftarrow S$ | Nada   | Nada   |

## *Resumen de ventajas e inconvenientes*

- Elimina el cuello de botella de los registros
- Evita EDL y EDE
- Permite el unrolling en HW
- No esta limitado a bloques básicos *si existe predicción de saltos*
- Complejidad
- Muchas cargas de registros asociativas por ciclo
- CDB limita el rendimiento
- Excepciones imprecisas



# Reduciendo la penalización de los saltos

## ❑ Tipos de saltos: Estadísticas

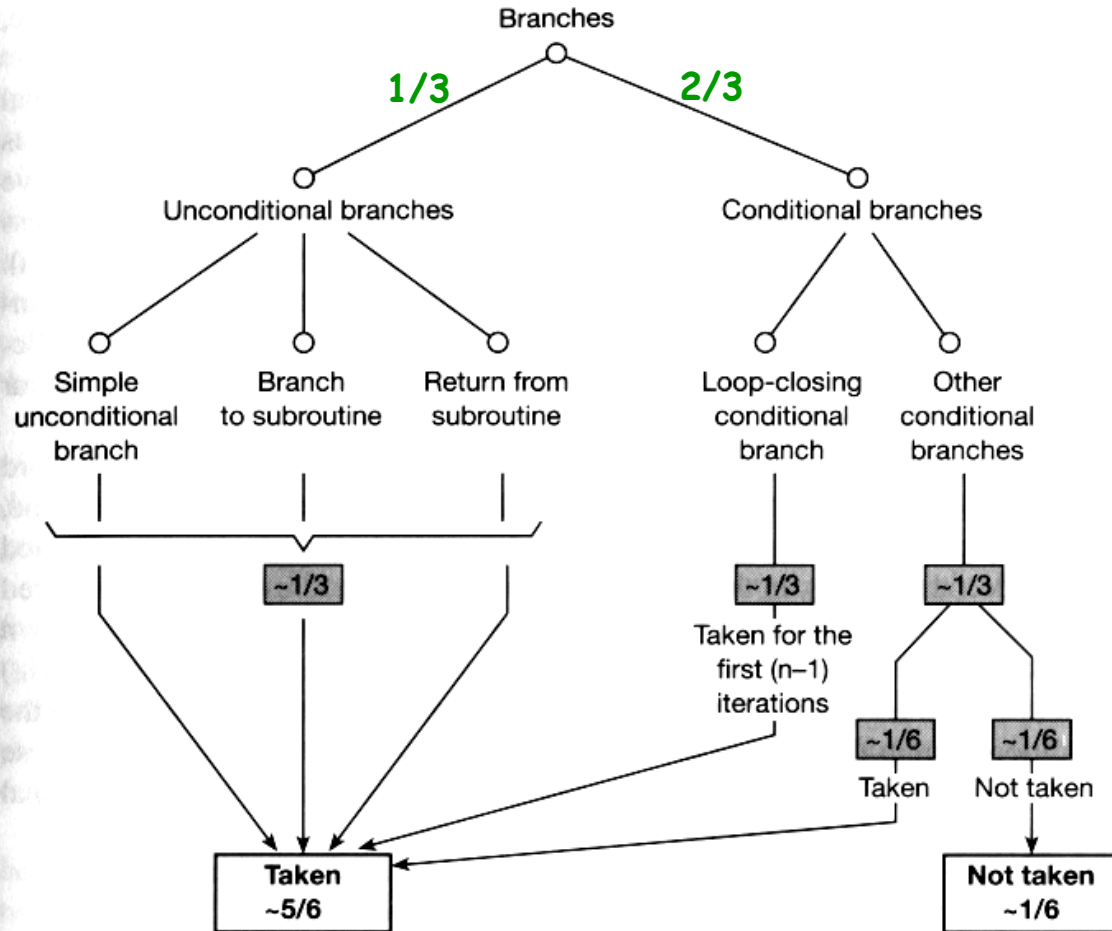
[Grohoski, G. (1990) IBM J. Res. Develop.; otros estudios dan resultados similares]

### En promedio

- Instrucciones de salto  
1 de cada 5 instrucc.
- Saltos condicionales  
2 de cada 3 saltos
- Saltos incondicionales  
1 de cada 3 saltos
- Saltos tomados  
5 de cada 6 saltos
- Saltos condicionales tomados  
3 de cada 4 saltos condic.
- Saltos incondicionales tomados  
Todos

### Conclusión (en promedio)

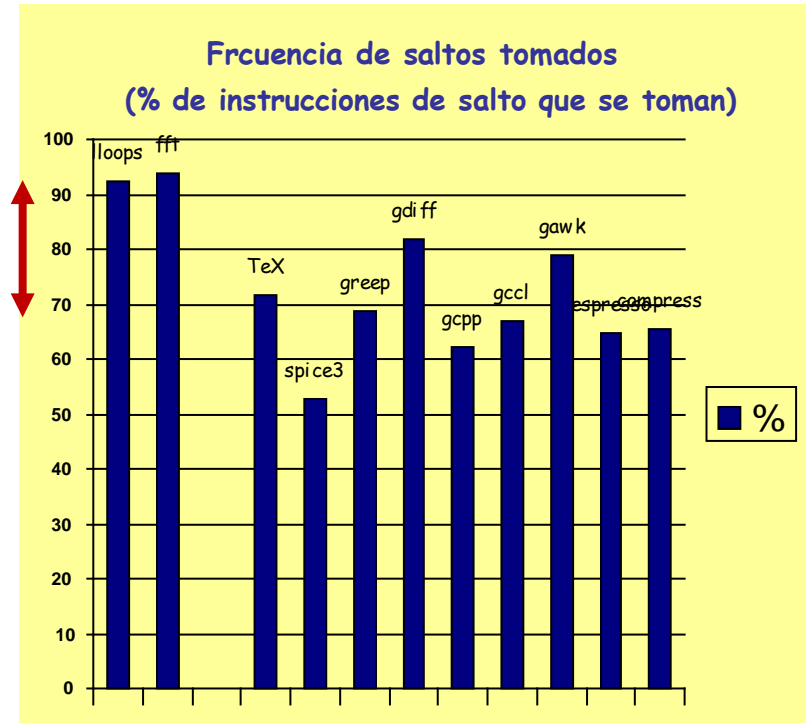
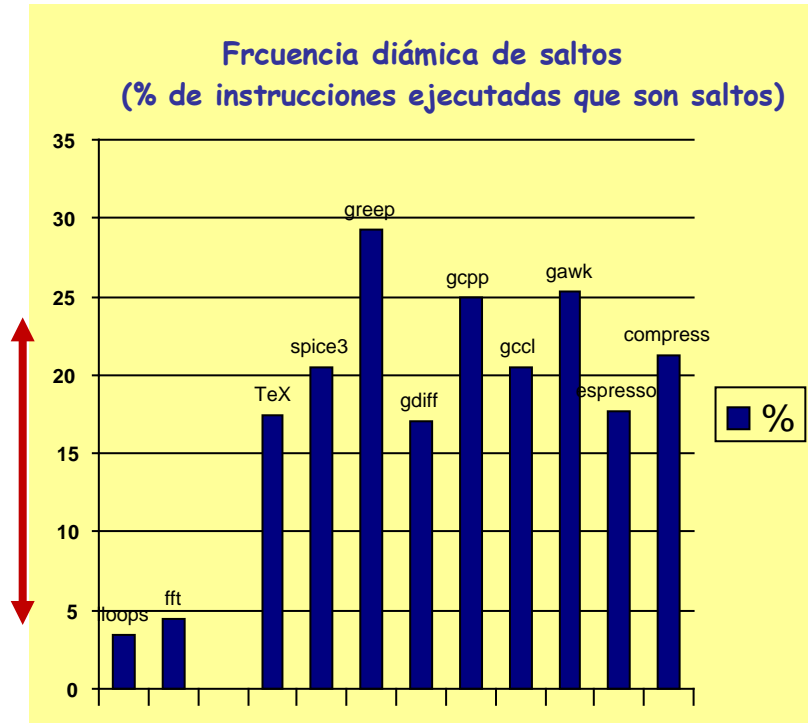
- 1 de cada 6 instrucciones  
es un salto tomado
- 1 de cada 8 instrucciones  
es un salto condicional
- 1 de cada 10 instrucciones  
es un salto condicional y tomado



Programas enteros una de cada 4-5 instrucciones. Flotantes 1 de cada 10-20 instrucciones

# Reduciendo la penalización de los saltos

## Tipos de saltos: Estadísticas



### Conclusión

- Frecuencia de los saltos depende del tipo de programa
- El comportamiento depende del tipo de programa

# Reduciendo la penalización de los saltos

## □ Predicción

### Idea Básica

Cuando se detecta una instrucción de salto condicional sin resolver

- Se supone o predice el camino del salto: **tomado o no tomado (Taken - Untaken)**
- Si el salto se predice como tomado se predice la dirección destino del salto
- La ejecución continúa de forma **especulativa** a lo largo del camino supuesto

Cuando se resuelve la condición

- Si la predicción fue correcta
  - ⇒ La ejecución se confirma y continúa normalmente
- Si la predicción fue incorrecta (fallo de predicción o "misprediction")
  - ⇒ Se descartan todas las instrucciones ejecutadas especulativamente
  - ⇒ Se reanuda la ejecución a lo largo del camino correcto

### Problemas a resolver en instrucciones de salto

- 1) Predecir la dirección de la instrucción destino del salto con un retardo mínimo (para saltos tomados)
- 2) Predecir el camino que tomará el salto
  - TAKEN (Tomado)
  - UNTAKEN (No Tomado)

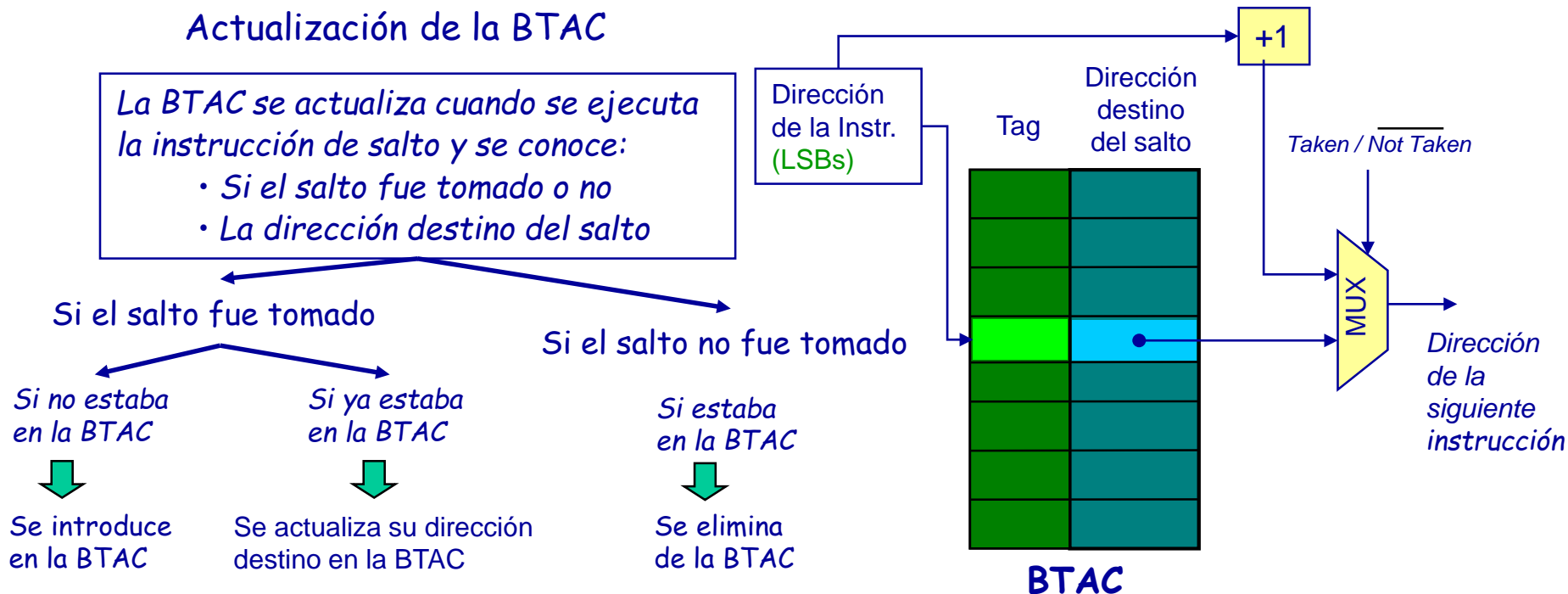
# Tratamiento de Saltos: Predicción

## ❑ Acceso rápido a la dirección destino del salto I

### Branch Target Address Cache (BTAC)

- Cache que almacena la dirección destino de los últimos saltos tomados
- Cuando se accede a una instrucción
  - Se accede simultáneamente a la BTAC utilizando la dirección de la instrucción
  - Si la dirección de la instrucción está en la BTAC, entonces es un salto. Si el salto se predice como tomado la dirección destino del salto se lee de la BTAC

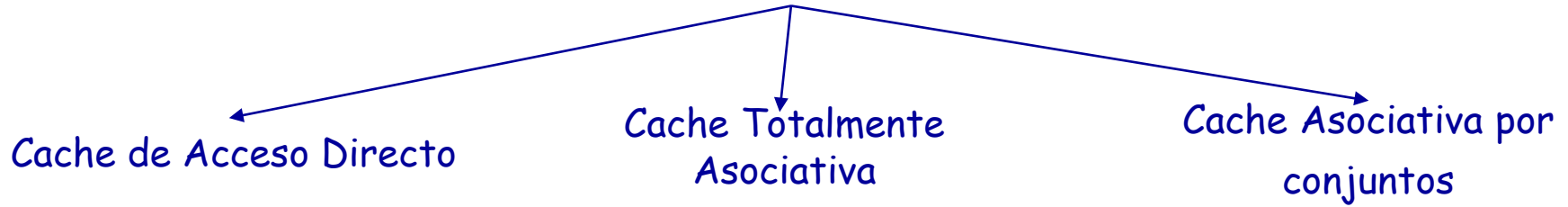
### Actualización de la BTAC



# Tratamiento de Saltos: Predicción

## ❑ Acceso a la dirección destino del salto II

### Alternativas de diseño de la BTAC



**Ventaja:** Menor coste  
**Desventaja:** "Aliasing"  
(destrucción de información si dos saltos compiten por la misma entrada)

**Ventaja:** menos Aliasing  
**Desventaja:** Mayor coste HW

Solución intermedia

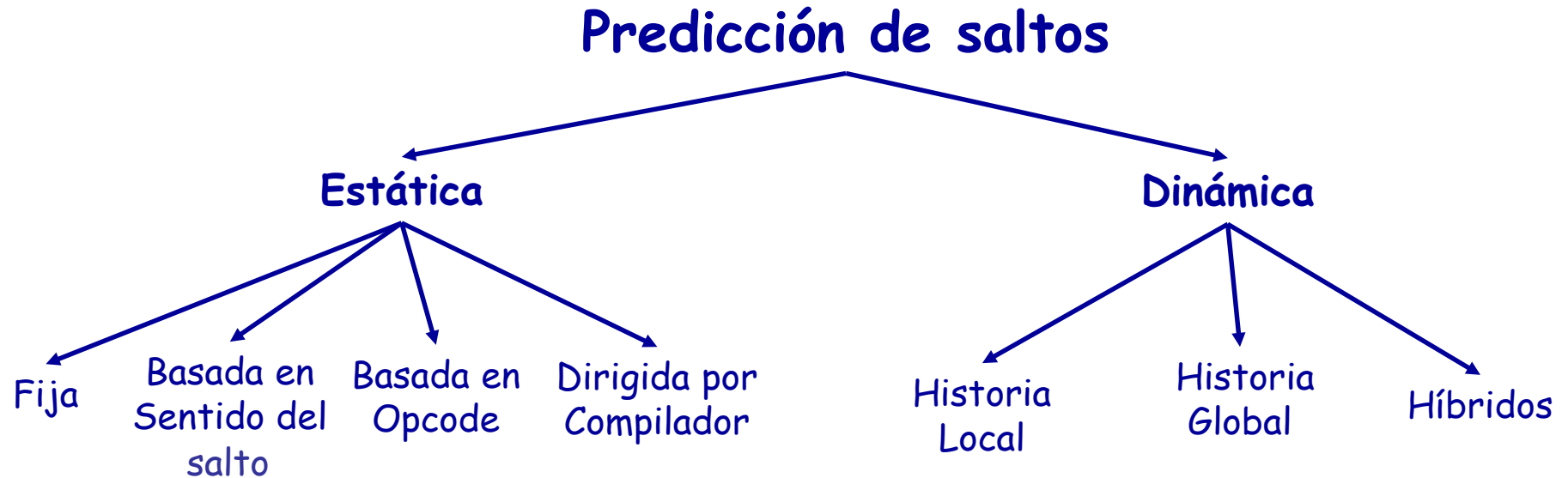
EJEMPLOS: Pentium (256) , Pentium II (512), Pentium 4 (4K) , AMD 64 (2K)

### Variación Branch Target Instruction Cache

- Almacenar la instrucción "más ventajas" si tempo de acceso a las instrucciones es alto
- Ejemplos: AMD K6,K7, NexGen Nx586

# Tratamiento de Saltos: Predicción

## □ Clasificación de técnicas de predicción de saltos



# Tratamiento de Saltos: Predicción

## ❑ Predicción estática

### Predicción Fija

**ALWAYS TAKEN**

- Predecir todos los saltos como **tomados**
- Mayor número de aciertos de predicción (3 de cada 4 saltos cond. son tomados)
- Mayor coste hardware (necesita almacenar la dirección destino del salto)

**ALWAYS NOT TAKEN**

- Predecir todos los saltos como **no tomados**
- Menor número de aciertos de predicción (sólo 1 de cada 4 saltos cond. es no tomado)
- Menor coste hardware

### Predicción basada en el SENTIDO del salto

**Saltos hacia atrás : TOMADOS**

La mayoría de saltos hacia atrás corresponden a bucles

**Saltos hacia delante: NO TOMADOS**

La mayoría de saltos hacia delante corresponden a IF-THEN-ELSE

*Mal comportamiento en programas con pocos bucles y muchos IF-THEN-ELSE*

# Tratamiento de Saltos: Predicción

## ❑ Predicción estática

### Predicción basada en el OPCODE de la instrucción de salto

**Fundamento:** La probabilidad de que un salto sea tomado depende del tipo de salto

*El salto es tomado para ciertos códigos de operación y no tomado para otros*

### Predicción dirigida por el COMPILADOR

#### Basada en el tipo de CONSTRUCCIÓN

El compilador predice si el salto será tomado o no dependiendo del tipo de construcción de control

#### Basada en PROFILING

El compilador predice en función del comportamiento de esa instrucción en ejecuciones previas del programa

#### Especificado por el PROGRAMADOR

El programador indica al compilador si el salto debe ser tomado o no (mediante directivas específicas)

- Se añade un **Bit de Predicción** al opcode de la instrucción
- El compilador activa o desactiva este bit para indicar su predicción



# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

### Idea básica

La predicción se realiza observando el comportamiento de las instrucciones de salto en las últimas ejecuciones (Historia)

*Necesario almacenar la historia de las ultimas ejecuciones del salto*

*Predictores de  
1 bit de historia*

*Predictores de  
2 bits de historia (bimodal)*

*Predictores de  
3 bits de historia*

#### EJEMPLOS

- Gmicro 100 (1991)
- Alpha 21064 (1992)
- R8000 (1994)

#### EJEMPLOS

- MC68060 (1993)
- Pentium (1994)
- Alpha 21064A (1994)
- Alpha 21164 (1995)

- PA 8500 (1999)
- UltraSparc (1995)
- PowerPC 604 (1995)
- PowerPC 620 (1996)
- R10000 (1996)

#### EJEMPLOS

- PA 8000 (1996)

### Evolución

Predictores correlacionados  
Predictores híbridos

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

### Predictor de un BIT

- Utilizan un bit de predicción por cada instrucción de salto
- El bit de predicción refleja el comportamiento de la última ejecución de la instrucción de salto  
⇒ Indica si en la anterior ejecución el salto fue tomado o no

#### Predicción

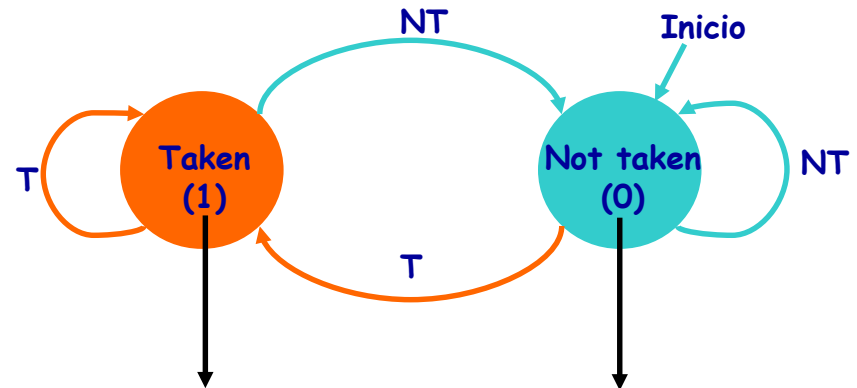
- El salto se predice como *Taken* si en la última ejecución fue tomado
- El salto se predice como *Not Taken* si en la última ejecución no fue tomado

#### FUNCIONAMIENTO

- Máquina de dos estados:
  - Not taken (0)
  - Taken (1)
- Registro de historia
  - Contador saturado de 1 bit
- Predicción
  - Valor del registro de historia

#### LIMITACIÓN

- Sólo se registra el comportamiento de la última ejecución del salto
- Dos malas predicciones en los cambios



Predicción: Taken

Predicción: Not taken

Más Bits

Cambios de estado:

T: el salto ha sido tomado

NT: el salto no ha sido tomado

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

### Predictor de dos bits (BIMODAL)

- Utilizan dos bits de predicción por cada instrucción de salto
- Estos bits reflejan el comportamiento de las últimas ejecuciones de ese salto

#### Predicción

- Un salto que se toma repetidamente se predice como *Taken*
- Un salto que no se toma repetidamente se predice como *Not taken*
- Si un salto toma una dirección inusual una sola vez, el predictor mantiene la predicción usual

#### Funcionamiento

Máquina de cuatro estados:

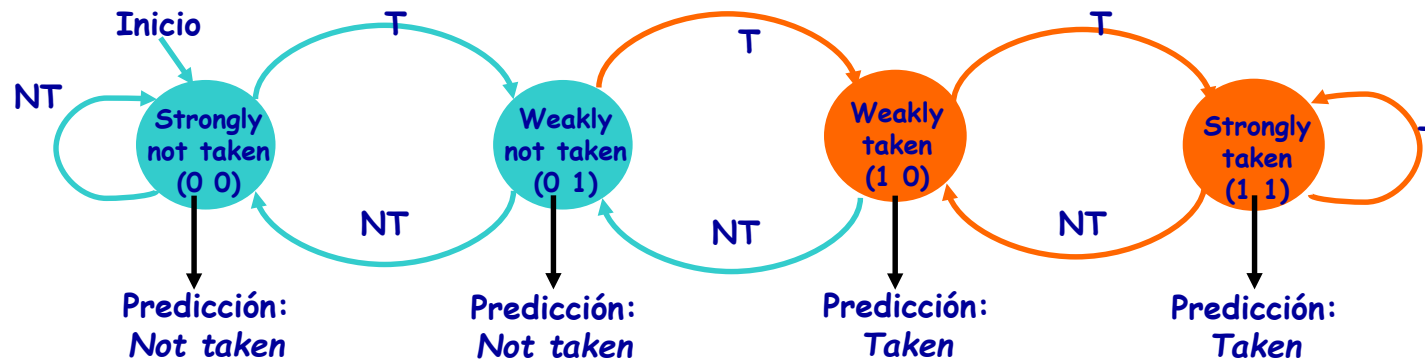
- Strongly not taken (00)
- Weakly not taken (01)
- Weakly taken (10)
- Strongly taken (11)

- Registro de historia

- Contador saturado de 2 bits

- Predicción

- bit más significativo del registro de historia



**Cambios de estado:**

T: el salto ha sido tomado

NT: el salto no ha sido tomado

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

### Implementación de los bits de predicción

#### 1) Branch Target Buffer (BTB)

Añade los bits de predicción a las entradas de la BTAC. La BTAC con bits de predicción se denomina BTB

##### EJEMPLOS

|              |             |
|--------------|-------------|
| • MC 68060   | 256 x 2 bit |
| • Pentium    | 256 x 2 bit |
| • R8000      | 1K x 1 bit  |
| • PM1        | 1K x 2 bit  |
| • Pentium II | 512x2 bit   |
| • Pentium 4  | 4Kx2bits    |

#### 2) Tabla de historia de saltos (BHT)

Utiliza una tabla especial, distinta de la BTAC para almacenar los bits de predicción

##### EJEMPLOS

|                |             |
|----------------|-------------|
| • Gmicro 100   | 256 x 1 bit |
| • PowerPC 604  | 512 x 2 bit |
| • R10000       | 512 x 2 bit |
| • PowerPC 620  | 2K x 2 bit  |
| • PA 8000      | 256 x 3 bit |
| • Alpha 21164A | 2K x 2 bit  |
| • AMD64        | 16Kx2bits   |

Actual: Samsung M3 ( 3 niveles; 128, 4k, 16K)

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos: Implementación

### 1) Branch Target Buffer (BTB): bits acoplados

#### La BTB almacena

- La dirección destino de los últimos saltos tomados
- Los bits de predicción de ese salto

#### Actualización de la BTB

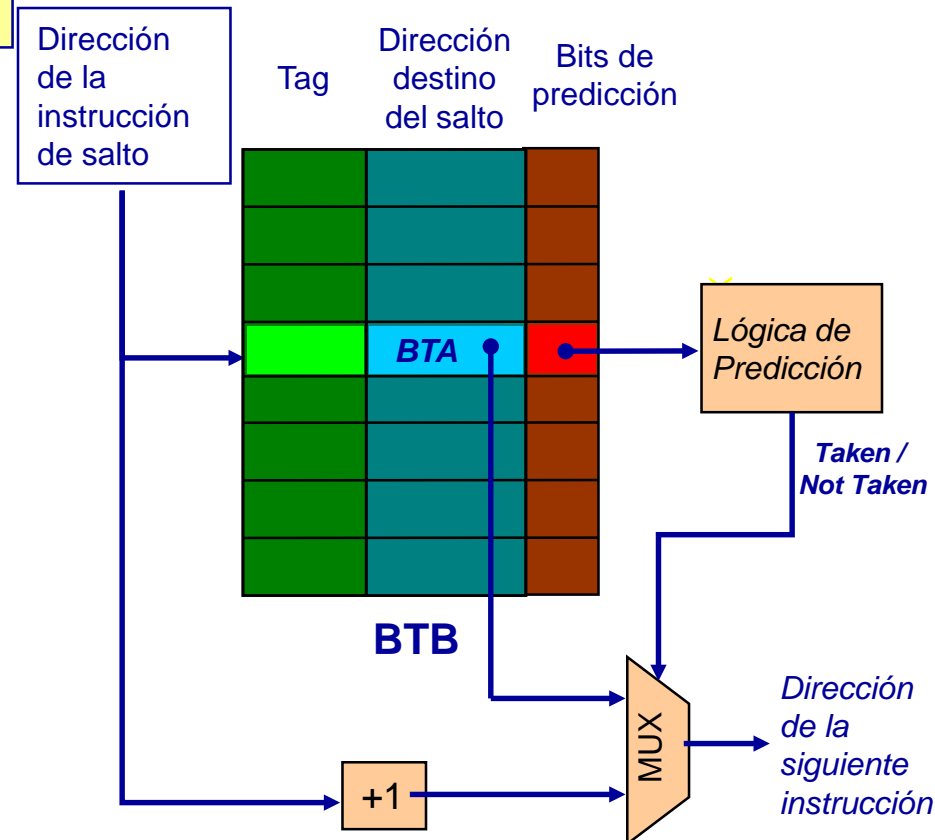
Los campos de la BTB se actualizan después de ejecutar el salto y se conoce:

- Si el salto fue tomado o no  
⇒ Actualizar bits de predicción
- La dirección destino del salto  
⇒ Actualizar BTA

#### Predicción Implícita (sin bits de predicción)

Imita un sólo bit de predicción

- Si la instrucción de salto está en la BTB  
⇒ El salto se predice como tomado
- Si la instrucción de salto no está en la BTB  
⇒ El salto se predice como no tomado



**DESVENTAJA:** Sólo se pueden predecir aquellas instrucciones de salto que están en la BTB

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos: Implementación

### 2) Tabla de historia de saltos (BHT): bits desacoplados

*Existen dos tablas distintas:*

- La BTAC, que almacena la dirección destino de los últimos saltos tomados
- La BHT, que almacena los bits de predicción de todas las instrucciones de salto condicional

#### Ventaja

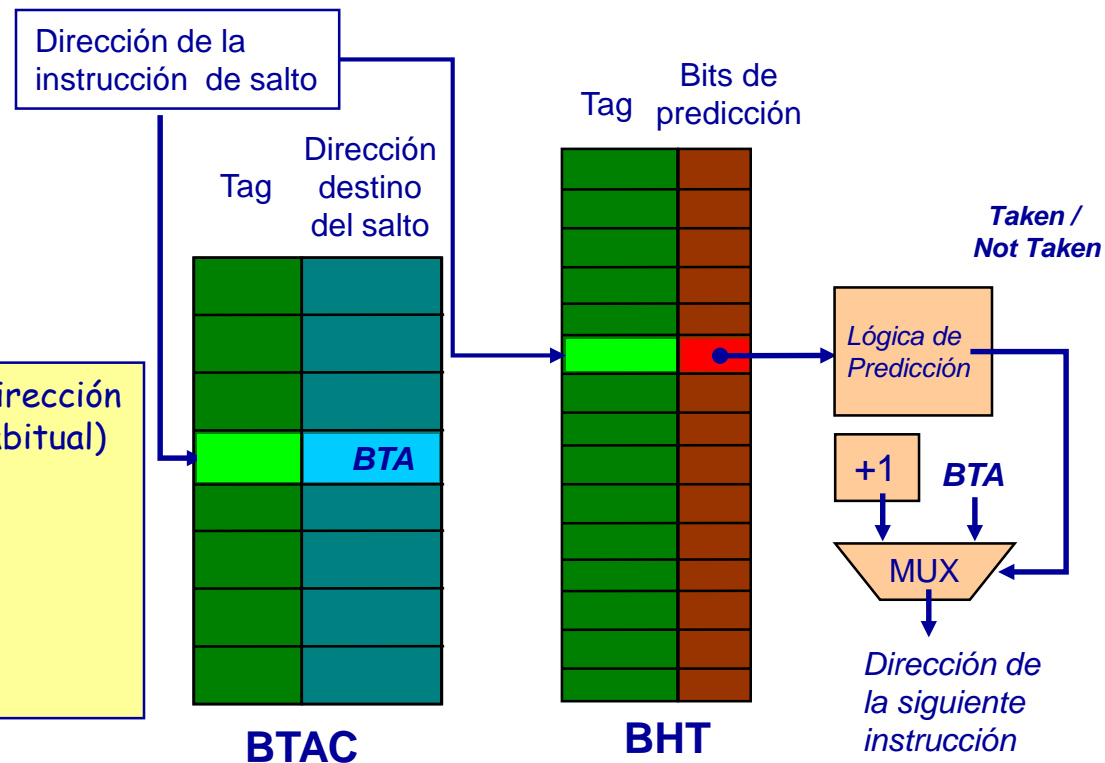
Puede predecir instruc. que no están en la BTAC (más entradas en BHT que en BTAC)

#### Desventaja

Aumenta el hardware necesario  
⇒ 2 tablas (mas bit de marca)

#### Acceso a la BHT

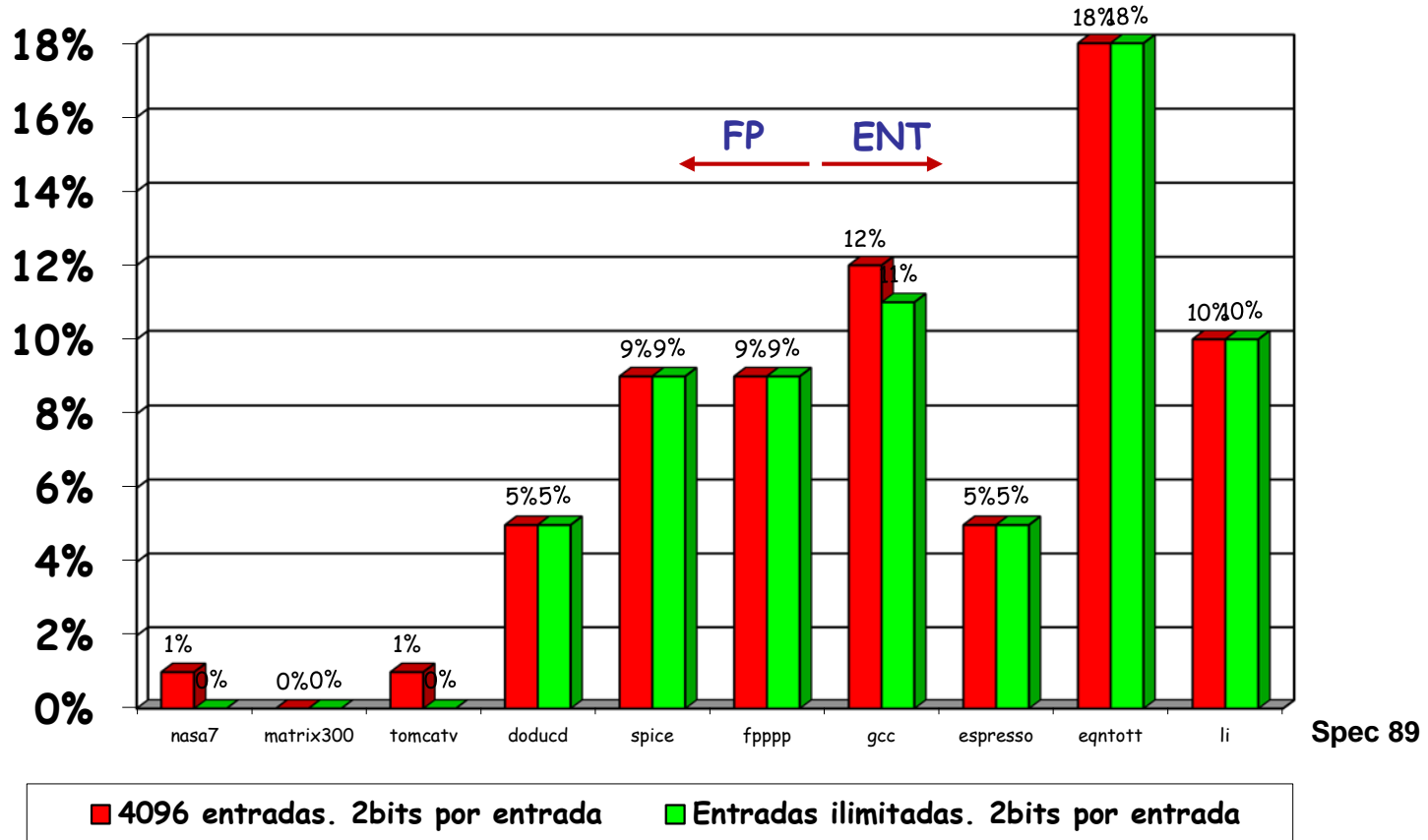
- Usando los bits menos significativos de la dirección
  - Sin TAGs ⇒ Menor coste (opción + habitual)
  - Compartición de entradas  
⇒ Se degrada el rendimiento
- Asociativa por conjuntos
  - Mayor coste ⇒ Tablas pequeñas
  - Para un mismo coste hardware  
⇒ Peor comportamiento



# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

Comportamiento: % de saltos mal predichos



- A partir de cierto valor, aumentar el tamaño del predictor no mejora el éxito de las predicciones
- Muchos fallos en algunos programas (enteros) ¿Por qué?

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

Otras formas de gestionar la historia

### 1. Muchas instrucciones de salto ejecutan patrones repetitivos



*Si conocemos el comportamiento del salto en las 3 últimas ejecuciones podemos predecir como se comportará en la siguiente ejecución*

⇒ Predicción basada en **historia LOCAL**

| <u>Historia</u> | <u>Predicción</u> |
|-----------------|-------------------|
| 111             | 0 (NT)            |
| 011             | 1 (T)             |
| 101             | 1 (T)             |
| 110             | 1 (T)             |

Material de apoyo: H&P 5<sup>th</sup> ed. Sección 3.3



# Tratamiento de Saltos: Predicción

## □ Predictores Dinámicos

### 2. Muchas instrucciones de salto dependen del comportamiento de otros saltos recientes (historia global)

**Observación:** Los saltos están relacionados; el comportamiento de los últimos saltos afecta a la predicción actual. **Idea:** Almacenar el comportamiento de los últimos  $n$  saltos y usarlo en la selección de la predicción.

#### Ejemplo:

**b1:** if (d = 0) then  
    d = 1

**b2:** if (d = 1) then



L1:

L2:

(Sup: d está almacenado en R1)

```
BNEZ  R1, L1 ; salto b1 (Salto si d ≠ 0)
ADDI   R1, R0, #1 ; Como d=0, hacer d=1
SUBI   R3, R1, #1 ; R3=d(R1)-1
BNEZ   R3, L2 ; salto b2 (Salto si d≠1)
.....
```

R3=0 ⇒ d=1  
R3≠0 ⇒ d≠1

*Si conocemos el comportamiento de la última ejecución de b1 podemos predecir el comportamiento de b2 en la siguiente ejecución ("si b1 no se toma, entonces b2 tampoco")*

Predicción basada en historia **GLOBAL**

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

### Ejemplo ( continua )

#### Relación entre los dos saltos

```
L1:  BNEZ  R1, L1 ; salto b1 (Salto si d ≠ 0)
      ADDI  R1, R0, #1 ; Como d=0, hacer d=1
      SUBI   R3, R1, #1 ; R3=d(R1)-1
      BNEZ   R3, L2 ; salto b2 (Salto si d≠ 1)
L2:  .....
```

Caso 1: d=0,1,2,...

| Valor de d | d ≠ 0? | salto b1 | d antes de b2 | d≠1? | salto b2 |
|------------|--------|----------|---------------|------|----------|
| 0          | no     | NT       | 1             | no   | NT       |
| 1          | si     | T        | 1             | no   | NT       |
| 2          | si     | T        | 2             | si   | T        |

Si b1 no se toma, entonces b2 tampoco: correlación entre saltos

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

### Ejemplo ( continua )

Comportamiento del predictor de un bit. ( estado inicial "not taken" NT)

L1: BNEZ R1, L1 ; salto b1 (Salto si d ≠ 0)  
ADDI R1, R0, #1 ; Como d=0, hacer d=1  
SUBI R3, R1, #1 ; R3=d(R1)-1  
BNEZ R3, L2 ; salto b2 (Salto si d≠ 1)  
.....  
L2:

Caso 2: d=2,0,2,0,...

| Valor de d | Predicción de b1 | b1 | Nueva predicción de b1 | Predicción de b2 | b2 | Nueva predicción de b2 |
|------------|------------------|----|------------------------|------------------|----|------------------------|
| 2          | NT               | T  | T                      | NT               | T  | T                      |
| 0          | T                | NT | NT                     | T                | NT | NT                     |
| 2          | NT               | T  | T                      | NT               | T  | T                      |
| 0          | T                | NT | NT                     | T                | NT | NT                     |

Muchos fallos de predicción

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

### ▪ Solución

- Predictor de dos niveles (**1**, **1**):

Para cada salto existen  $2^1$  predictores de **1** bit. El comportamiento del último salto (**1**) determina el predictor que se usa.

- Predictor de dos niveles (**m**, **n**)

Para cada salto existen  $2^m$  predictores de **n** bits. El comportamiento de los últimos **m** saltos determinan el predictor que se usa

### Significado de los bit de predicción en un predictor (1,1)

Dos predictores  
de un bit (**P0/P1**)

Casos  
posibles

| Bits de predicción: | Predicción si el último salto no tomado: <b>usar P0</b> | Predicción si el último salto tomado: <b>usar P1</b> |
|---------------------|---|--|
| <b>P0 / P1</b>      |   |  |
| NT / NT             | NT  | NT   |
| NT / T              | NT  | T  |
| T / NT              | T   | NT   |
| T / T               | T   | T  |

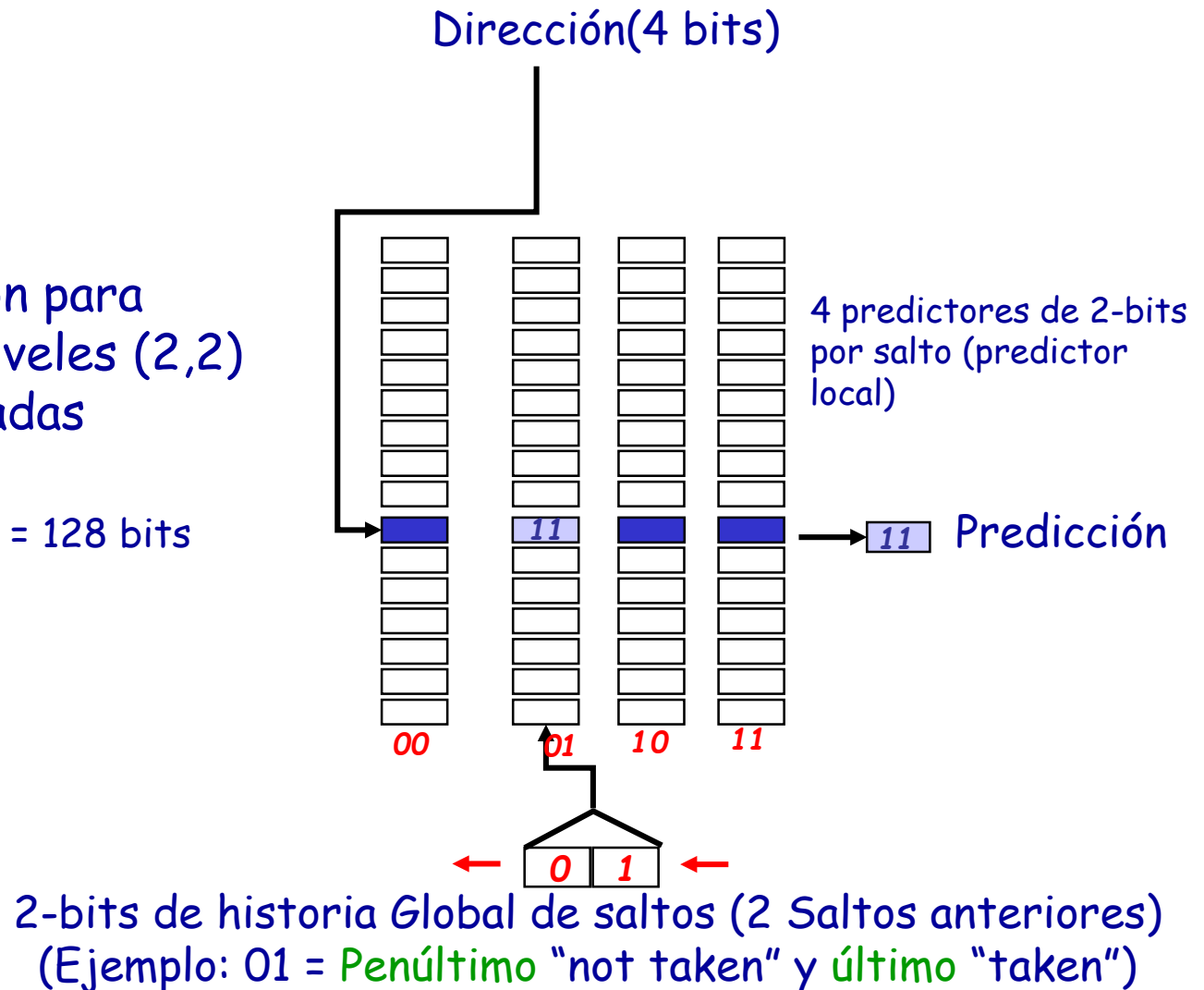
Ejemplo: Tamaño de un predictor (**2**, **2**) de 4k entradas:  $2^2 \times 2 \times 4K = 32Kb$

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

Implementación para  
Predictor de dos niveles (2,2)  
con 16 entradas

Tamaño =  $2^2 \times 2 \times 16 = 128$  bits



# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

### Ejemplo ( continua )

Comportamiento del predictor de dos niveles (1,1). ( estado inicial "not taken" NT)

```
L1:    BNEZ  R1, L1 ; salto b1 (Salto si d ≠ 0)
      ADDI  R1, R0, #1 ; Como d=0, hacer d=1
      SUBI   R3, R1, #1 ; R3=d(R1)-1
      BNEZ  R3, L2 ; salto b2 (Salto si d≠ 1)
L2:    .....
```

Caso 2: d=2,0,2,0,...

Sólo se predice mal la 1ª iteración (d=2)

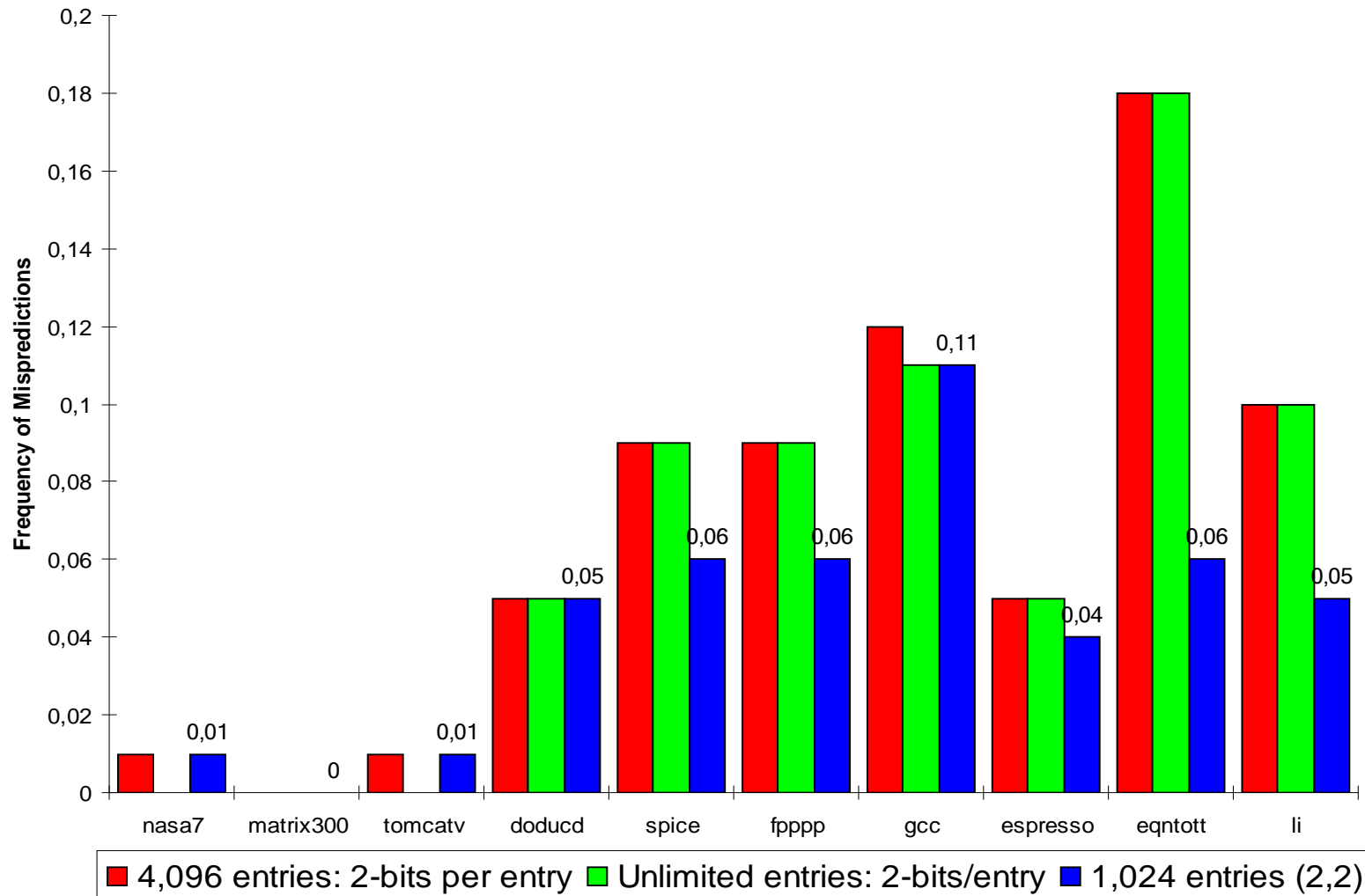
| d = ? | Predicción de b1 | b1       | Nueva predicción de b1 | Predicción de b2 | b2       | Nueva predicción de b2 |
|-------|------------------|----------|------------------------|------------------|----------|------------------------|
| 2     | NT/NT            | <u>T</u> | T/NT                   | NT/ <u>NT</u>    | <u>T</u> | NT/T                   |
| 0     | <u>T</u> /NT     | NT       | T/NT                   | <u>NT</u> /T     | NT       | NT/T                   |
| 2     | <u>T</u> /NT     | T        | T/NT                   | NT/ <u>T</u>     | T        | NT/T                   |
| 0     | <u>T</u> /NT     | NT       | T/NT                   | <u>NT</u> /T     | NT       | NT/T                   |

(Subrayado en rojo: Bit de predicción seleccionado en cada caso, en función del comportamiento del salto anterior)

# Tratamiento de Saltos: Predicción

## ❑ Predictores Dinámicos

Comportamiento: Frecuencia de saltos ejecutados que se predicen mal



# Tratamiento de Saltos: Predicción

## ❑ Predictores híbridos

### Idea básica

- Cada uno de los predictores estudiados tiene sus ventajas y sus inconvenientes
- Combinando el uso de distintos predictores y aplicando uno o otro según convenga, se pueden obtener predicciones mucho más correctas

### Predictor híbrido

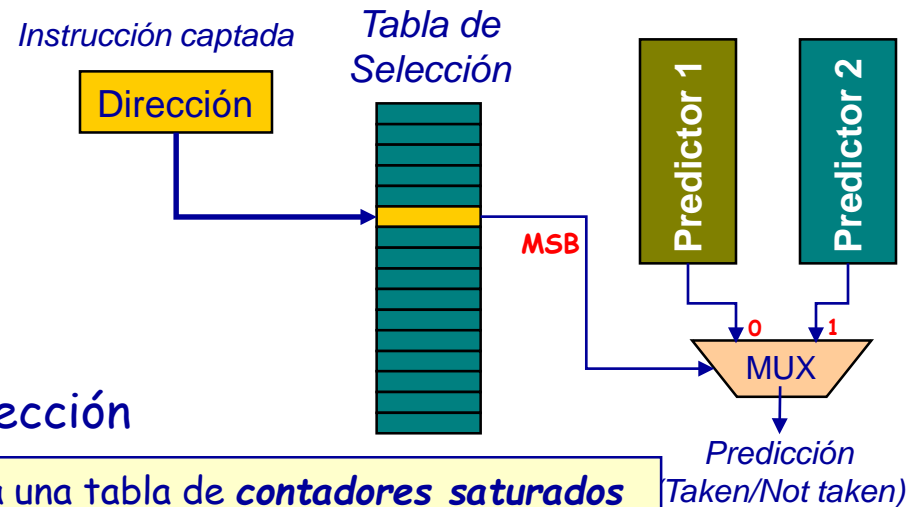
Mezcla varios predictores y añade un mecanismo de selección del predictor

### Mecanismo de selección

Elige, en cada caso, el predictor que haya dado mejores resultados hasta el momento

### Implementación del mecanismo de selección

Para combinar dos predictores, P1 y P2, se utiliza una tabla de **contadores saturados de dos bits** indexada por la dirección de la instrucción de salto



| P1      | P2      | Actualiz. del contador |
|---------|---------|------------------------|
| Fallo   | Fallo   | Cont no varía          |
| Fallo   | Acierto | Cont = Cont +1         |
| Acierto | Fallo   | Cont = Cont -1         |
| Acierto | Acierto | Cont no varía          |

- Si P2 acierta más que P1  
⇒ *Cont aumenta*
- Si P1 acierta más que P2  
⇒ *Cont disminuye*

| Bit más signif. del contador | Predictor seleccionado |
|------------------------------|------------------------|
| 0                            | P1                     |
| 1                            | P2                     |



# Ejemplo: Alpha 21264

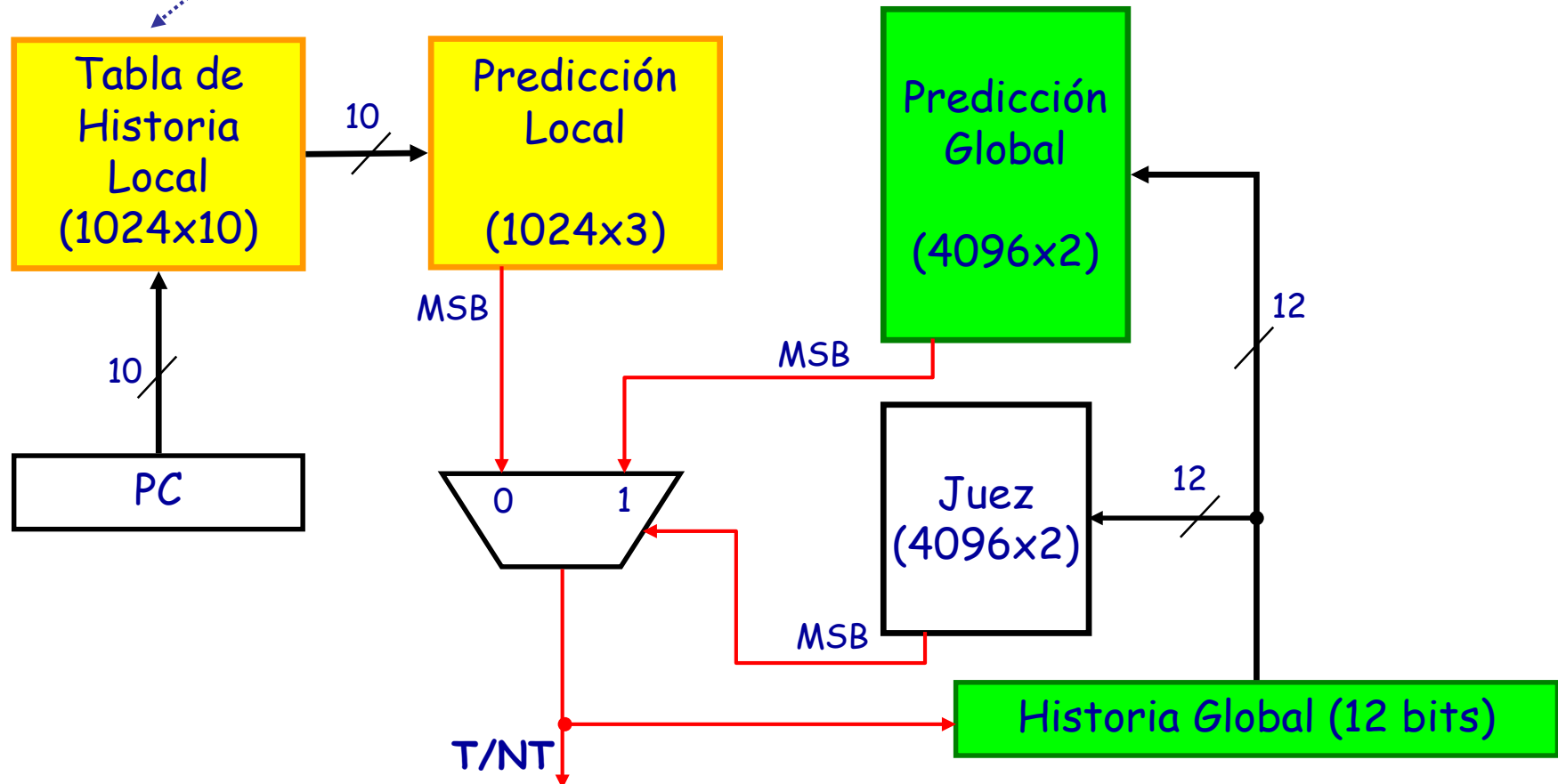
---

- ❑ Predictor competitivo (Tournament Predictor)
- ❑ Predictor Local: Predicción de un salto en función del comportamiento previo de ese mismo salto
  - o Considera las 10 últimas ejecuciones del salto
- ❑ Predictor global: Predicción de un salto en función del comportamiento de los últimos 12 saltos ejecutados
- ❑ Juez: Decide cuál de las dos predicciones se aplica
  - o Selecciona el predictor que esté manifestando el mejor comportamiento
- ❑ Actualización: al resolver cada salto
  - o Se actualizan los predictores en función de su acierto o fallo
  - o Si los dos predictores hicieron una predicción distinta, se actualiza el juez para que favorezca al que acertó
- ❑ Gran importancia para la ejecución especulativa en 21264 (hasta 80 instrucciones en la ventana)
- ❑ Tasas de predicción correcta (benchmarks): 90-100%

# Tournament predictor del Alpha 21264

(IEEE Micro, Marzo 1999. Art. en Campus Virtual)

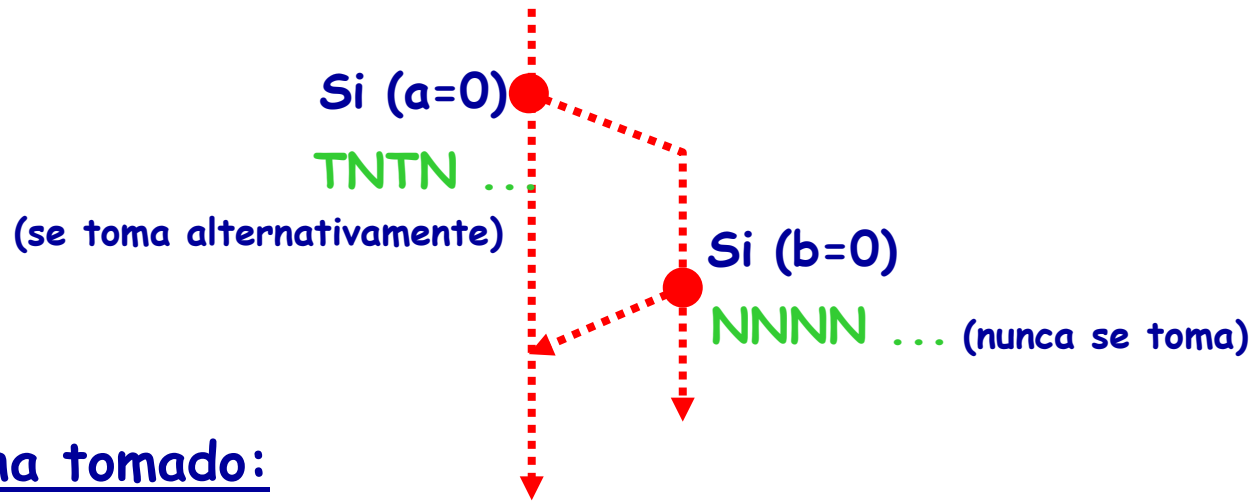
Comportamiento de las 10 últimas  
ejecuciones de 1024 saltos



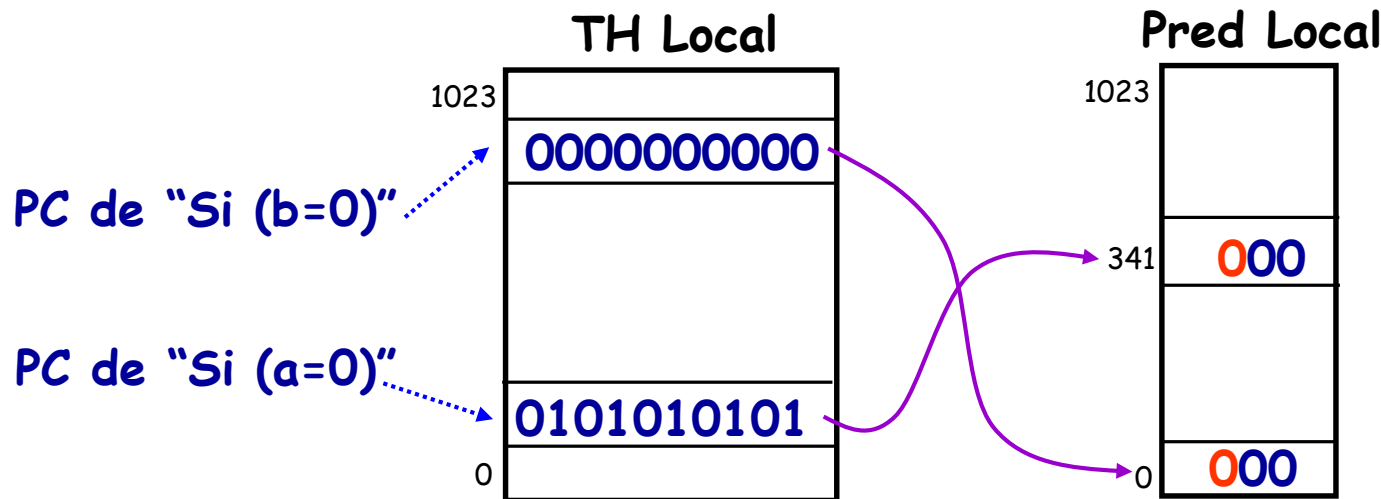
Juez: Acierto global y fallo local => incrementa  
Fallo global y acierto local => decrementa

# Ejemplos de funcionamiento (1)

Programa con dos saltos que tiene el comportamiento descrito

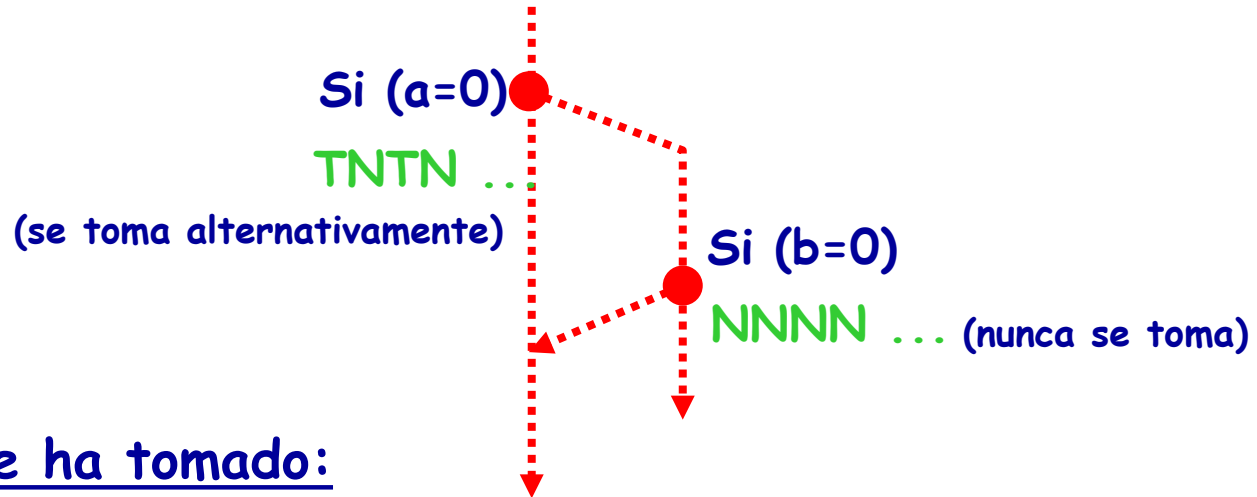


"Si(a=0)" se ha tomado:

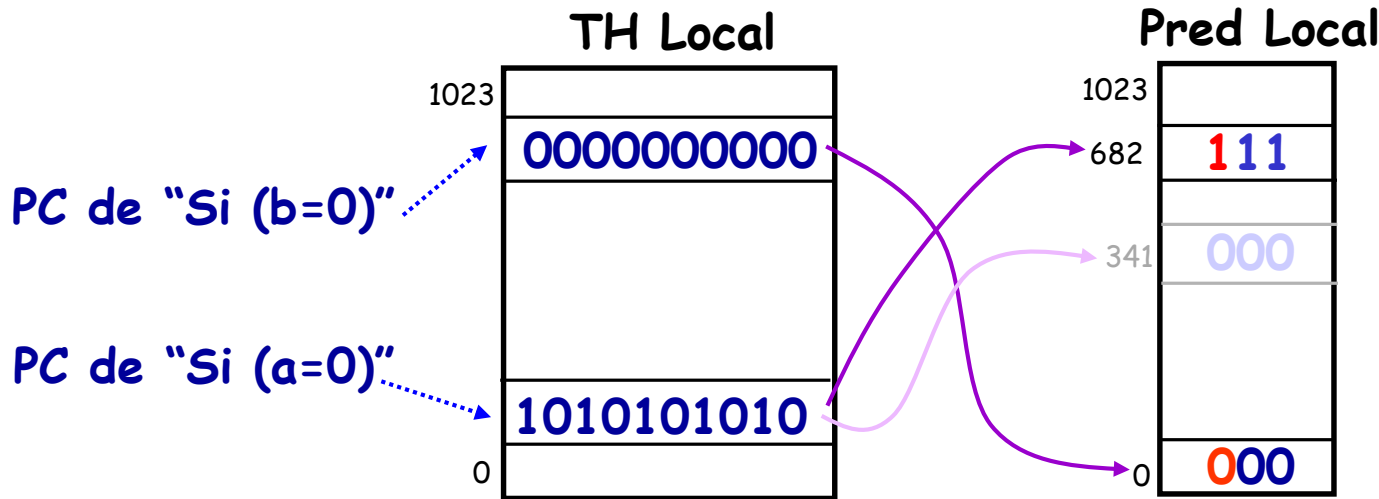


## Ejemplos de funcionamiento (2)

## Programa con dos saltos que tiene el comportamiento descrito

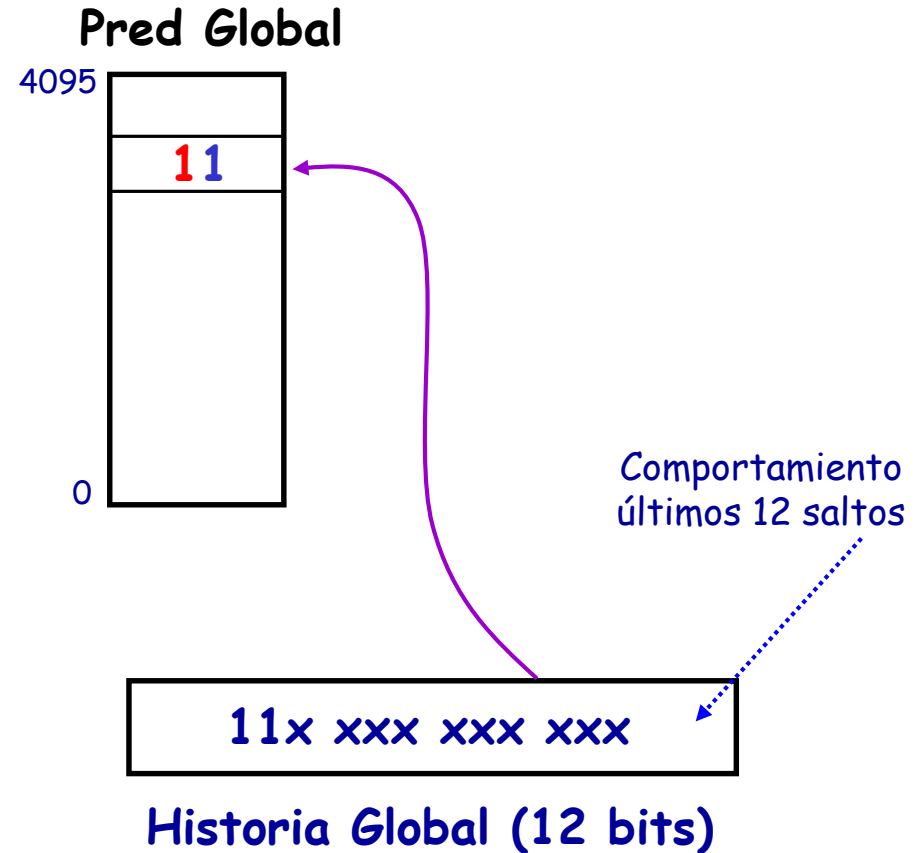
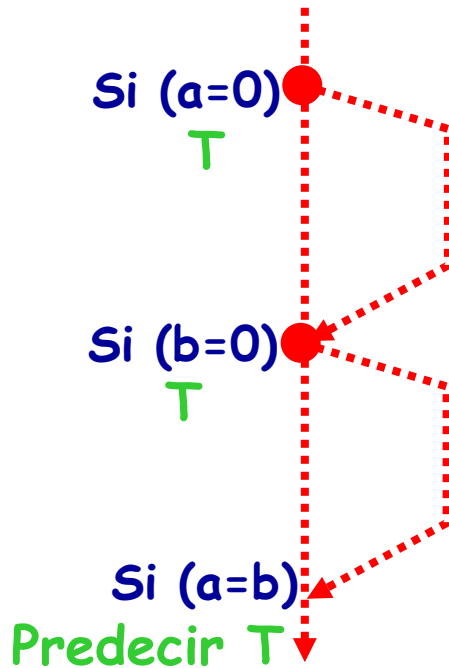


"Si(a=0)" no se ha tomado:



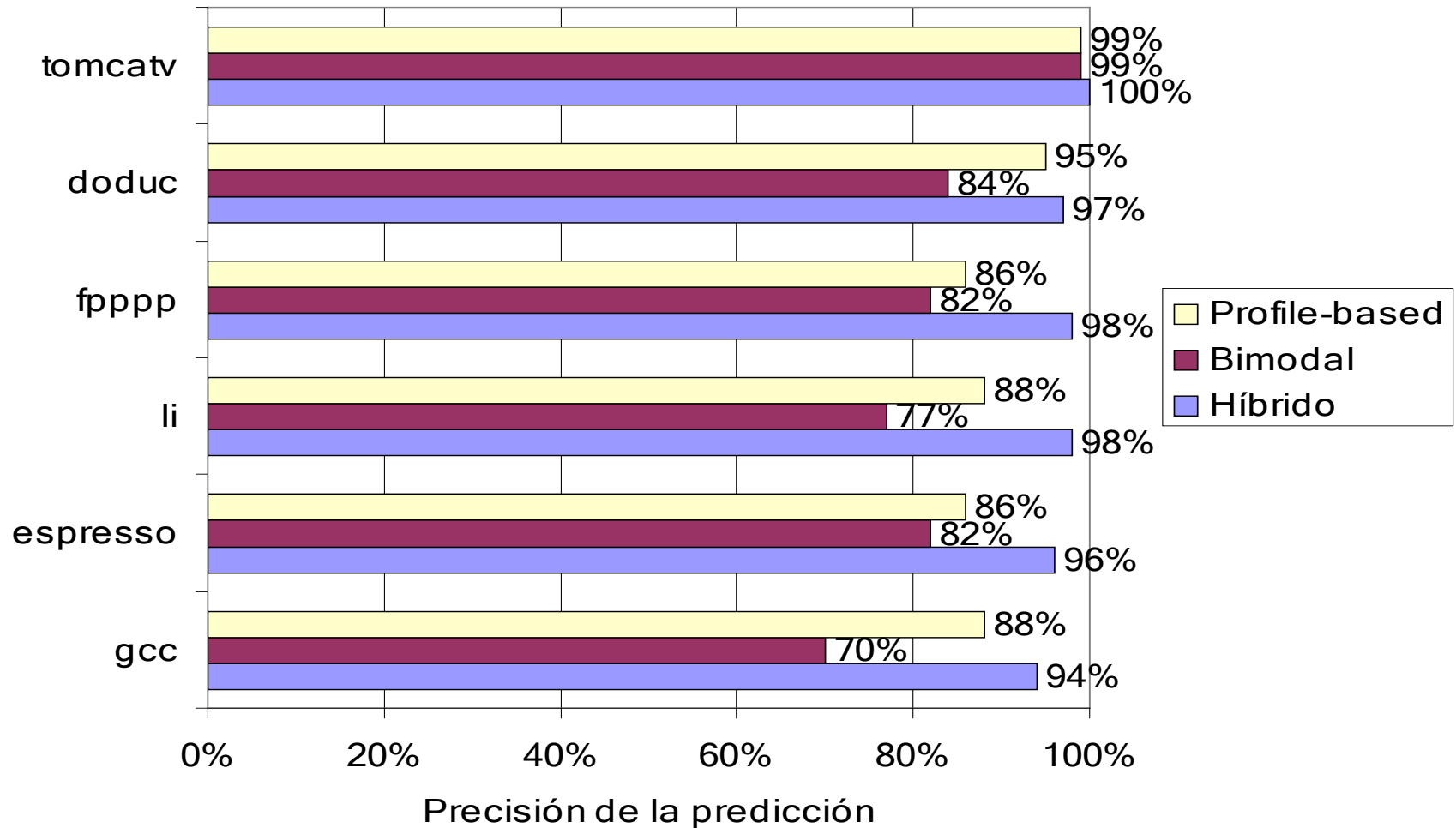
# Ejemplos de funcionamiento (3)

Programa con tres saltos que tiene el comportamiento descrito



# Tratamiento de Saltos: Predicción

## ❑ Predictores: Comportamiento

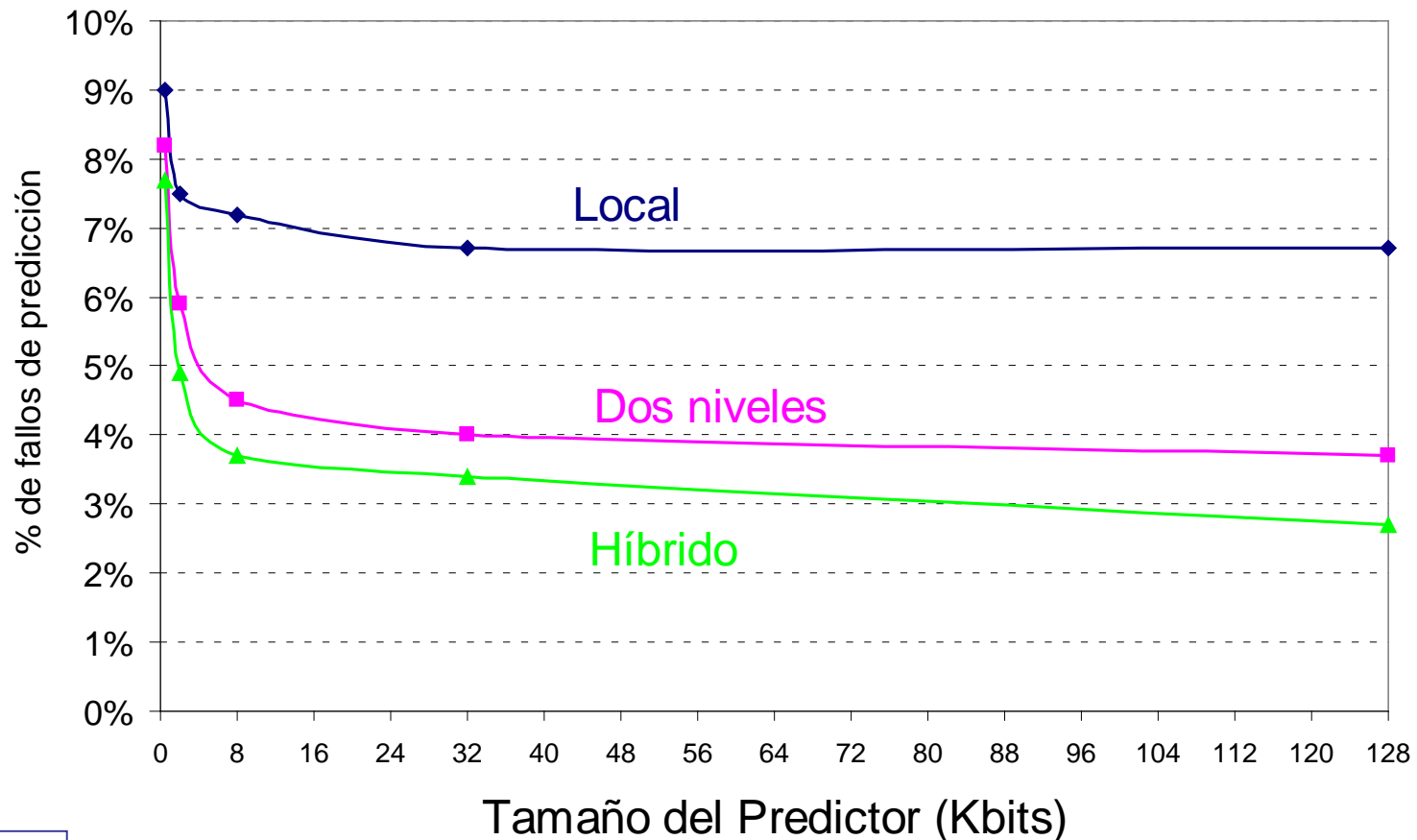


Profile\_based- Predictor estático

# Tratamiento de Saltos: Predicción

## ❑ Predictores: Comportamiento

- La ventaja del predictor híbrido es su capacidad de seleccionar el predictor correcto para un determinado salto
- Muy importante para programas enteros
  - Un predictor híbrido selecciona el global casi 40% de las veces para SPEC integer y menos del 15% de las veces para SPEC FP



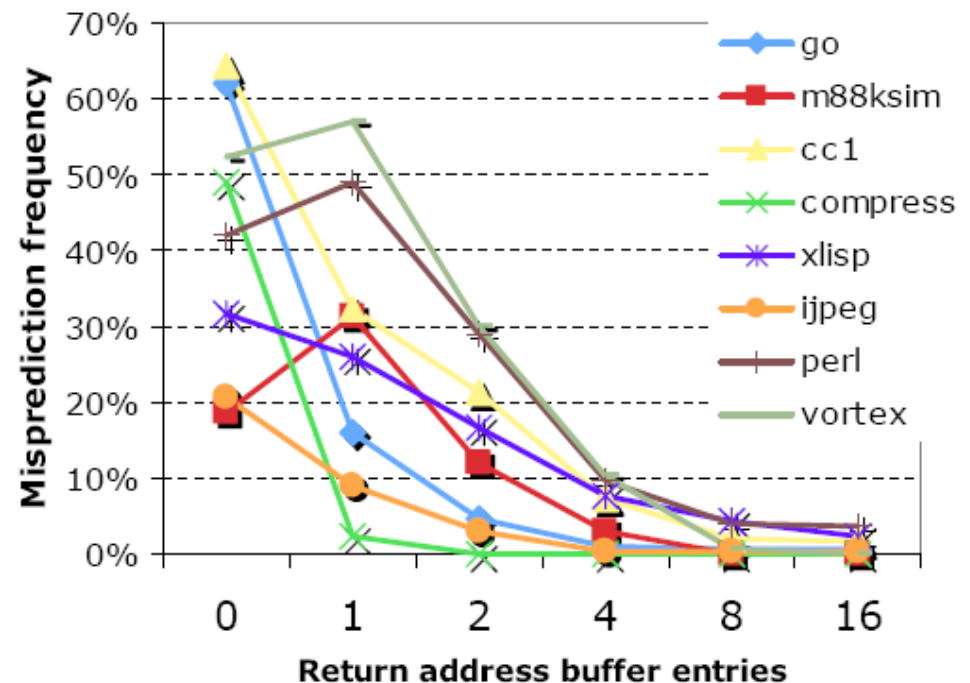
# Tratamiento de Saltos: Predicción

## □ Predicción de los retornos

- La precisión de los predictores con los retornos es muy baja: La dirección de retorno es diferente en función de la llamada
- Solución : Pila de direcciones de retorno( 8 a 16 entradas )

### EJEMPLOS

- |                    |             |
|--------------------|-------------|
| • UltraSparc I, II | 4 entradas  |
| • Pentium Pro      | 16 entradas |
| • R10000           | 1 entrada   |





# Tratamiento de Saltos: Predicción

## ❑ Recuperación de fallos de predicción (misprediction)

### Tareas básicas

- 1) Descartar los resultados de las instrucciones ejecutadas especulativamente
- 2) Reanudar la ejecución por el camino correcto con un retardo mínimo

### 1) Descarte de los resultados

- Los resultados de estas instrucciones especulativas se almacenan en **registros temporales** (registros de renombramiento o *Buffer* de reordenamiento)
- Estas instrucciones no modifican los contenidos de los registros de la arquitectura ni de la memoria

Si la ejecución fue correcta

Se **actualizan** los registros de la arquitectura y/o la memoria

Si la ejecución fue incorrecta

Se **descartan** los resultados de los registros temporales

# Tratamiento de Saltos: Predicción

## ❑ Recuperación de fallos de predicción (misprediction)

### 2) Reanudación de la ejecución por el camino correcto

El procesador debe guardar, al menos, la dirección de comienzo del camino alternativo

Si la predicción fue "Not taken"

El procesador debe calcular y almacenar la dirección destino del salto

Si la predicción fue "Taken"

El procesador debe almacenar la dirección de la instrucción siguiente al salto

Ejemplos: PowerPC 601 - 603 - 605

### Reducción de los retardos en la recuperación de fallos

El procesador puede guardar, no solo la dirección del camino alternativo, sino prebuscar y almacenar algunas instrucciones de este camino

Si la predicción fue "Taken"

- El procesador almacena la dirección del camino secuencial
- El procesador prebusca y almacena las primeras instrucciones secuenciales

Si la predicción fue "Not taken"

- El procesador calcula y almacena la dirección destino del salto
- El procesador prebusca y almacena las primeras instrucciones del destino del salto

Ejemplos: 2 buffer Power1, Power2, Pentium, UltraSparc( 16 ), R10000 (256 bits)  
3 buffer Nx586 ( 2 pendientes )

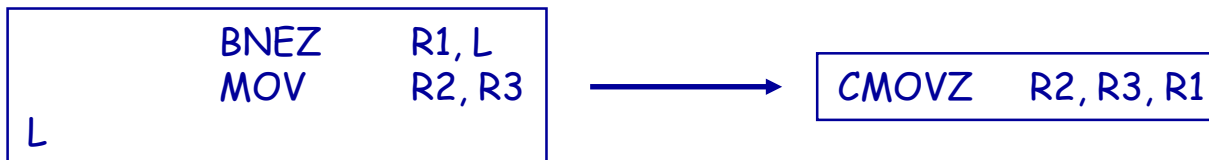
# Tratamiento de Saltos: Otras alternativas

## ❑ Ejecución condicional de instrucciones

### Idea básica

- Eliminar, parcialmente, los saltos condicionales mediante *instrucciones de ejecución condicional*
- Una instrucción de ejecución condicional está formada por:
  - Una condición
  - Una operación
- Ejecución condicional
  - Si la condición es **cierta**  $\Rightarrow$  La instrucción se ejecuta
  - Si la condición es **falsa**  $\Rightarrow$  La instrucción se comporta como NOP

### Ejemplo



**Ventaja:** Buena solución para implementar alternativas simples de control

**Desventaja:** Consumen tiempo en todos los casos. Más lentas que las incondicionales

**Ejemplos:** ARM, Alpha, Hp-Pa, MIPS, Sparc

## Resumen

- ✓ Predictor bimodal bueno para Loop (programas FP)
- ✓ Predictores de dos niveles buenos para IF then else
- ✓ Predicción de la dirección destino importante
- ✓ Ejecución condicional reduce el numero de saltos

- ❑ La predicción de saltos introduce ESPECULACION
  - Dos tipos de instrucciones en el procesador
    - Las independientes
    - Las que dependen de una predicción de salto. Su finalización depende del acierto o fallo en la predicción.

¿Cómo podemos implementar esta distinción con un modelo de ejecución con finalización Fuera de orden?

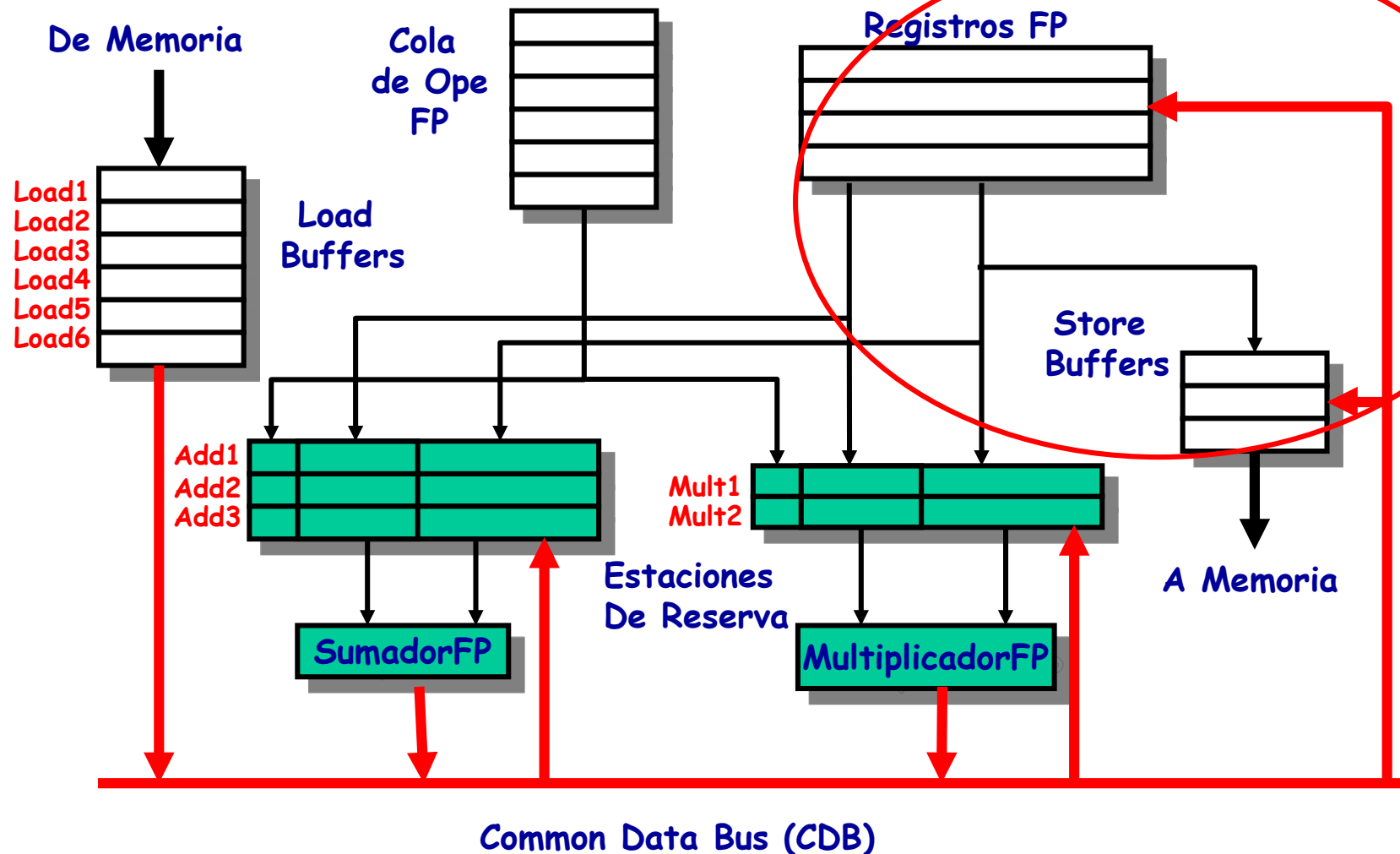
Modificando el Algoritmo de Tomasulo para **forzar** finalización en orden

## ALGORITMO DE TOMASULO CON ESPECULACION

- La etapa WB no almacena resultados en registros
  - Los resultados se guardan temporalmente en un buffer (ROB)
- Se añade una etapa de finalización (commit): se escriben resultados en reg o memoria
- Las instrucciones hacen la etapa de finalización en orden
- Una instr que depende de un salto sin resolver no puede hacer la etapa de finalización

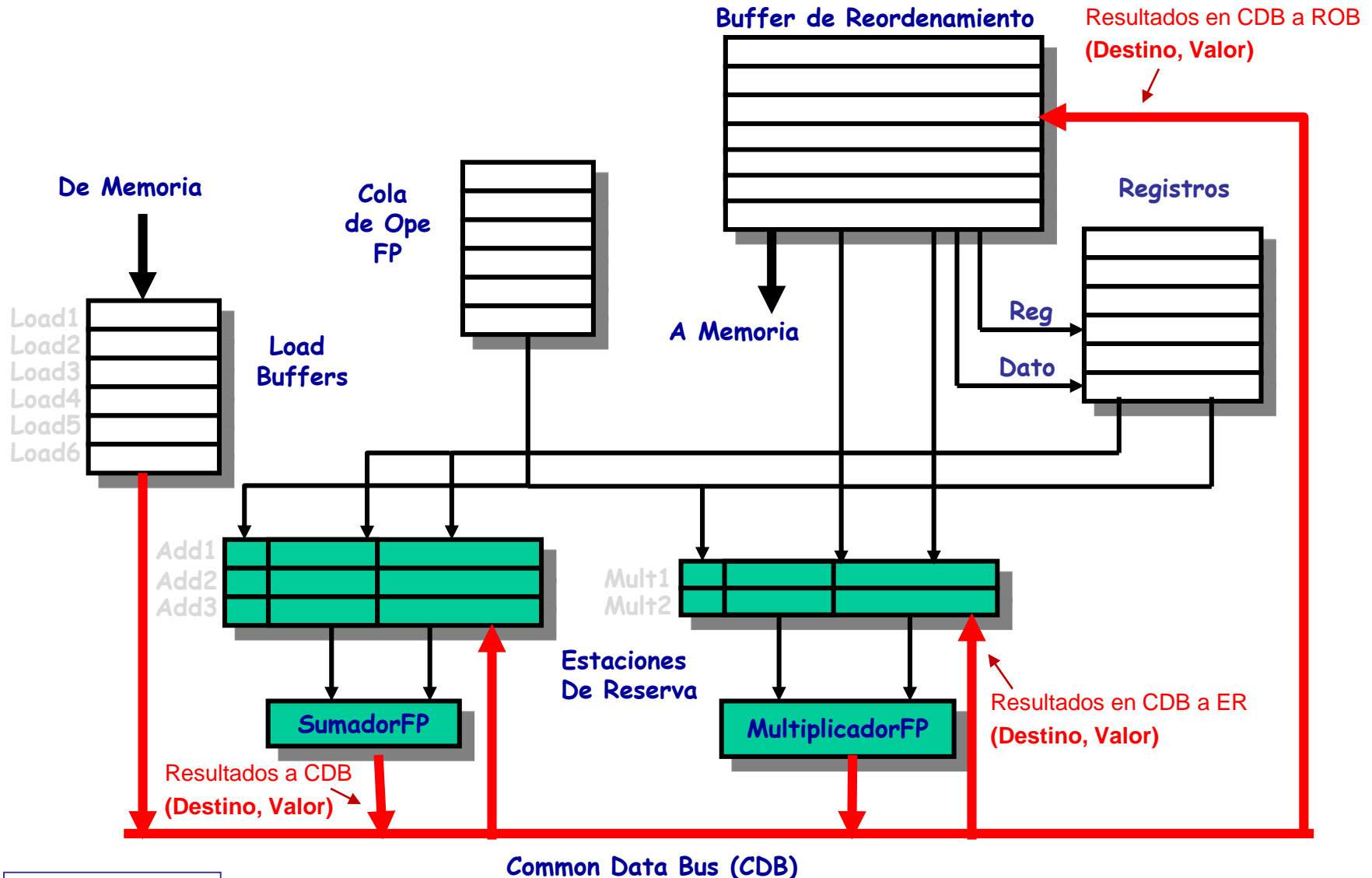
# Especulación

## ❑ Algoritmo de TOMASULO



# Especulación

## ❑ Algoritmo de TOMASULO con especulación



## ❑ El Buffer de Reordenamiento (ROB)

- o Almacena resultados de instrucciones cuya ejecución ha finalizado, pero...
  - están a la espera de actualizar registros o memoria (finalización en orden)
  - son dependientes de un salto (ejecución especulativa)
- o Permite el paso de operandos entre instrucciones especuladas con dependencia LDE.
- o Puede implementarse como cola circular con dos punteros
  - 1: instrucción más antigua; 2: primer hueco libre

## ❑ Los operandos de una instrucción pueden llegar hasta la ER desde:

- o CDB (la instrucción que genera el operando todavía no ha realizado la fase de escritura)
- o ROB (la instrucción que genera el operando se ha ejecutado, pero no ha actualizado el banco de registros)
- o Registros (la instrucción que genera el operando ha finalizado completamente)



- ❑ Estructura del ROB: cada entrada contiene 4 campos
  - o Tipo de instrucción
    - Salto (sin reg destino), Store (destino en memoria), Aritmética/Load (con destino en registro)
  - o Destino
    - Número de registro (Aritmética/Load)
    - Dirección de memoria (Store)
  - o Valor
    - Resultado de la ejecución de la instrucción. Guarda el valor hasta que se actualiza registro destino o memoria.
      - En Store: Mientras el valor no está disponible, guarda el nº de la entrada del ROB donde está la instrucción que lo producirá.
  - o Listo
    - La instrucción ha completado la fase de ejecución y el resultado está disponible en el campo "Valor"

# Especulación: fases

## ❑ Algoritmo de TOMASULO con especulación

### • Los 4 estados del algoritmo de Tomasulo especulativo

#### ✓ **Issue:** Toma la instrucción de la cola

Es necesario: ER con entrada libre y Buffer de Reordenamiento (ROB) con entrada libre.

Toma operandos de registros o de resultados almacenados en ROB por instrucciones previas (ver transparencia siguiente)

Marca R. Destino con n° de entrada del ROB asignada

Marca ER asignada con n° de entrada del ROB asignada

#### ✓ **Ejecución:** Opera sobre los operandos

Espera hasta que los operandos estén disponibles. Chequea CDB.

#### ✓ **Escribe resultados:** Finaliza ejecución

Escribe a través de CDB en todas las ER de Fus y entradas del ROB que estén a la espera del resultado. Libera ER. No escribe en registros, ni memoria.

Envía por CDB resultado + n° de entrada del ROB a la que se dirige

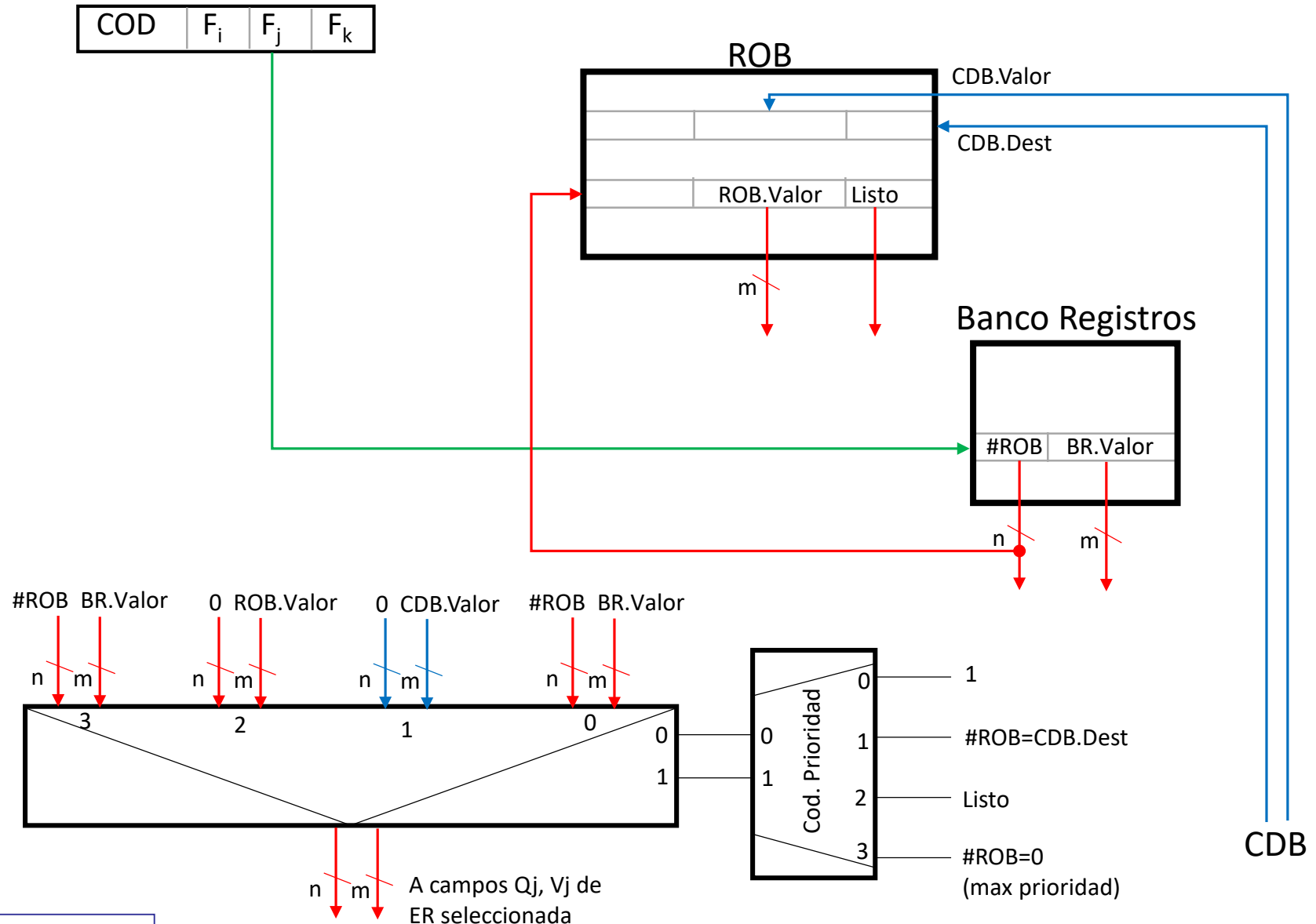
#### ✓ **Commit:** Actualiza registros desde el ROB

Cuando la Instrucción esta en la cabecera del ROB y resultado presente:

Actualiza Registro (o escribe en memoria) y elimina la instrucción del ROB.

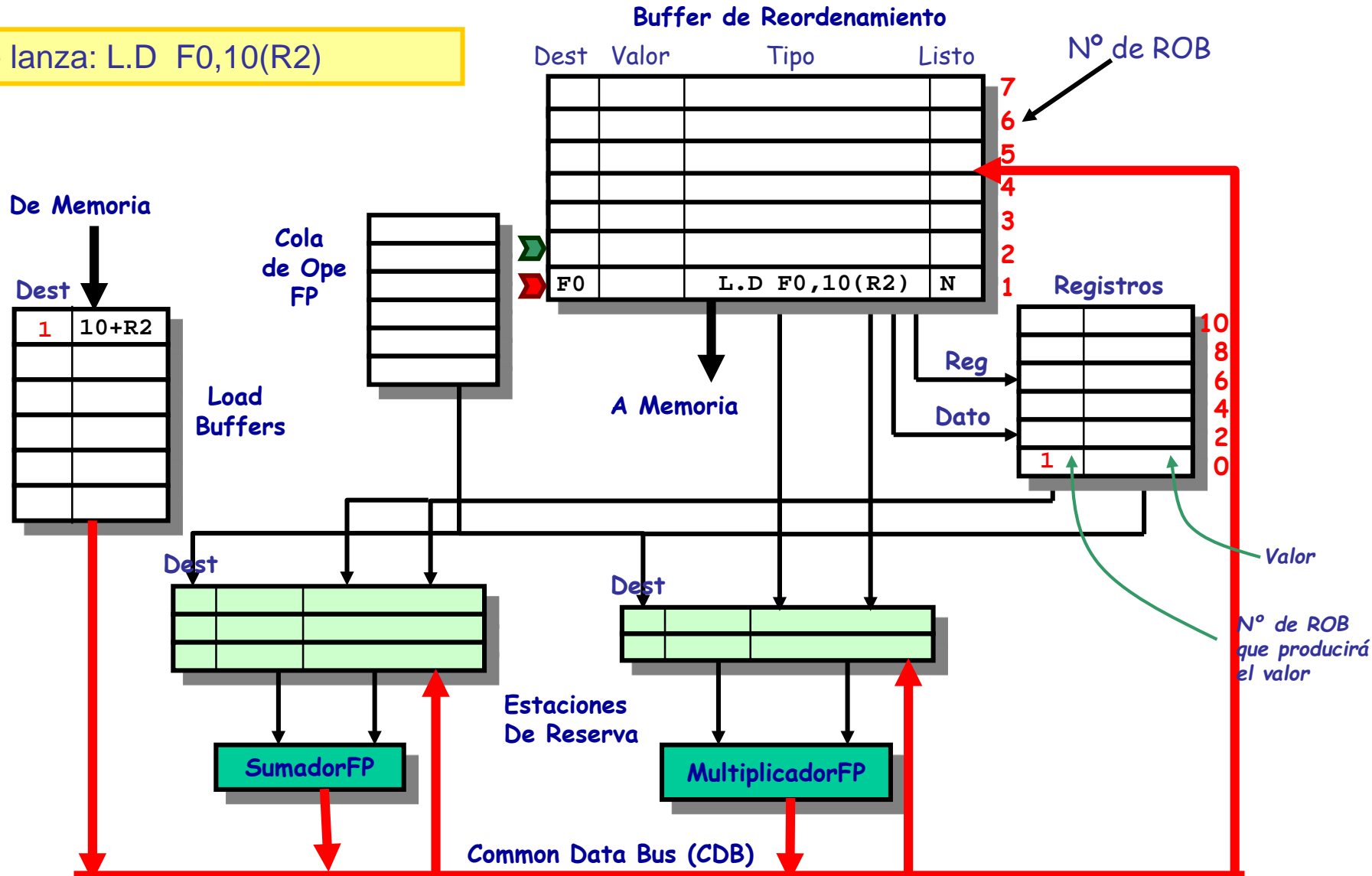
# Especulación: fases

Gestión de operandos en la fase “issue” de una instrucción: operando  $F_j$  (similar para  $F_k$ )



# Especulación: Ejemplo

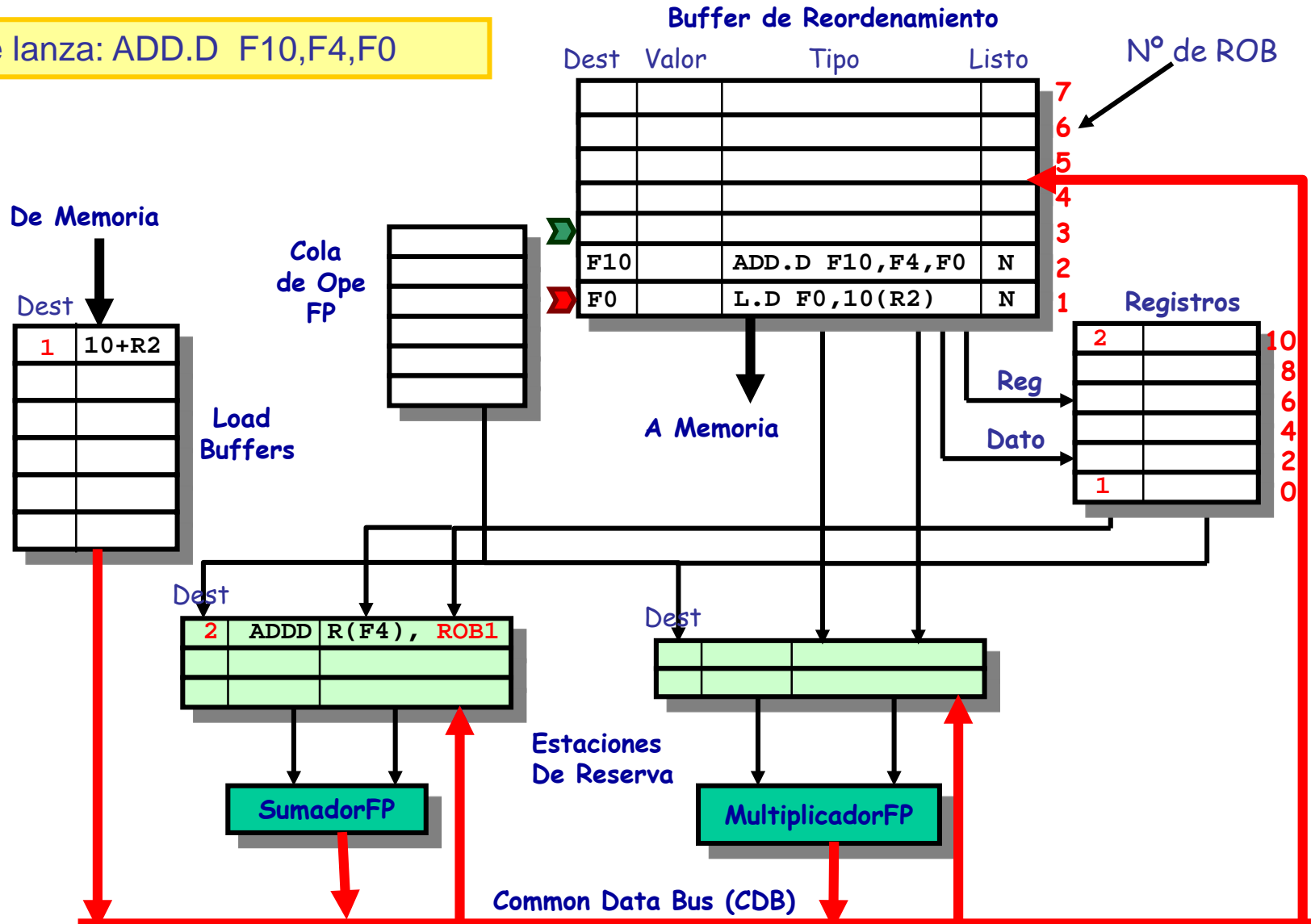
Se lanza: L.D F0,10(R2)



Los LB y ER no tienen un nº de TAG, pero tienen un campo "Destino" (nº de entrada del ROB donde se escribirá el resultado)

# Especulación: Ejemplo

Se lanza: ADD.D F10,F4,F0

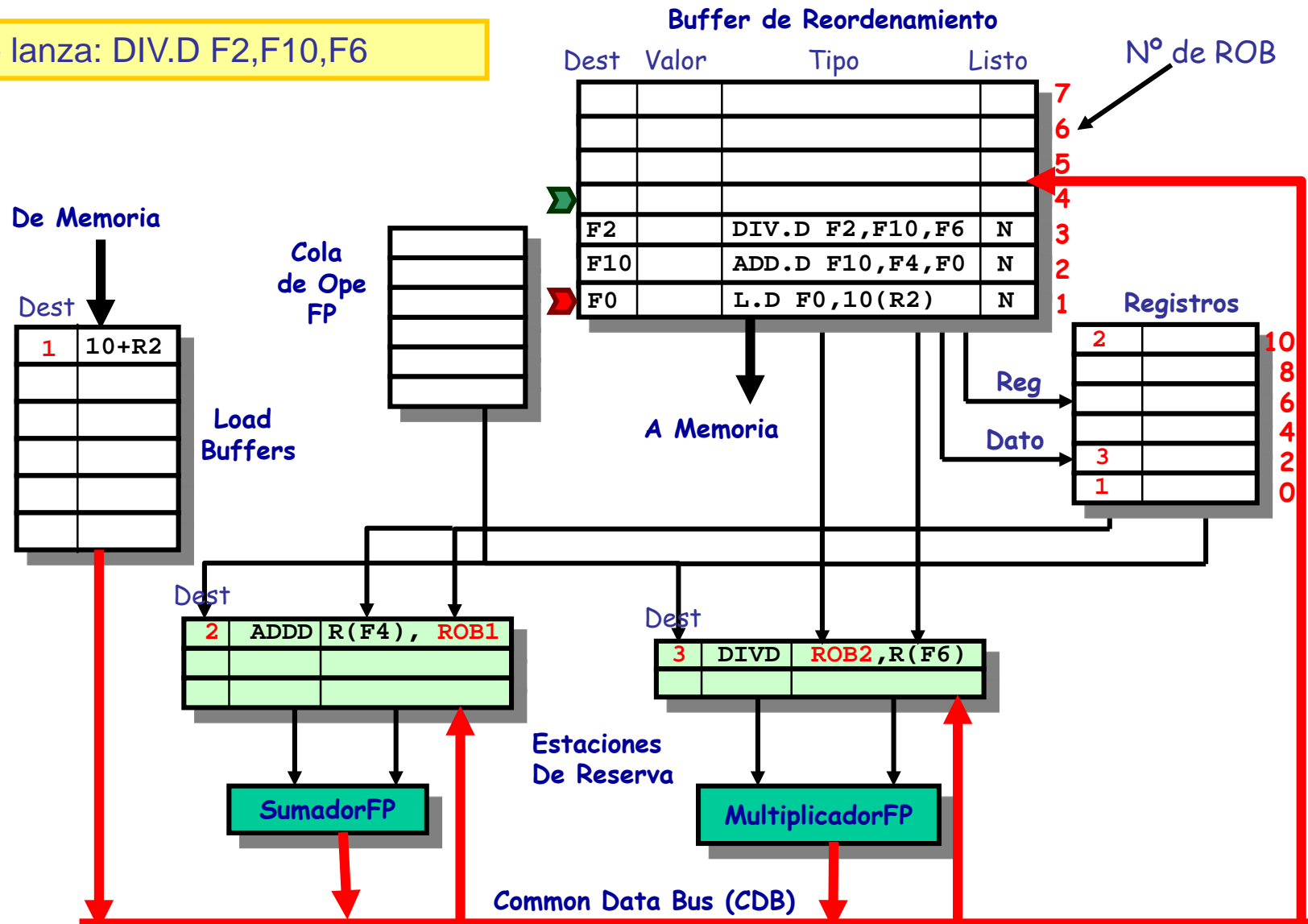


El reg F4 no tiene marca → copiar valor en ER

El reg F0 tiene marca (1) y campo "Listo" de ROB1=N → copiar marca en ER

## Especulación: Ejemplo

Se lanza: DIV.D F2,F10,F6



# Especulación: Ejemplo

Se lanza: BNE F2, ---  
L.D F4, 0(R3)  
ADD.D F0,F4,F6

Buffer de Reordenamiento

| Dest | Valor | Tipo            | Listo |
|------|-------|-----------------|-------|
| F0   |       | ADD.D F0,F4,F6  | N     |
| F4   |       | L.D F4,0(R3)    | N     |
| ---  |       | BNE F2,---      | N     |
| F2   |       | DIV.D F2,F10,F6 | N     |
| F10  |       | ADD.D F10,F4,F0 | N     |
| F0   |       | L.D F0,10(R2)   | N     |

Nº de ROB

De Memoria

| Dest |       |
|------|-------|
| 1    | 10+R2 |
| 5    | 0+R3  |
|      |       |
|      |       |
|      |       |
|      |       |

Cola de Ope FP

Load Buffers

A Memoria

Registros

Reg

Dato

|   |  |
|---|--|
| 2 |  |
|   |  |
|   |  |
| 5 |  |
| 3 |  |
| 6 |  |
|   |  |
|   |  |
|   |  |
|   |  |

Dest

|   |      |             |
|---|------|-------------|
| 2 | ADDD | R(F4), ROB1 |
| 6 | ADDD | ROB5, R(F6) |
|   |      |             |
|   |      |             |

Dest

|   |      |             |
|---|------|-------------|
| 3 | DIVD | ROB2, R(F6) |
|   |      |             |
|   |      |             |
|   |      |             |

Estaciones De Reserva

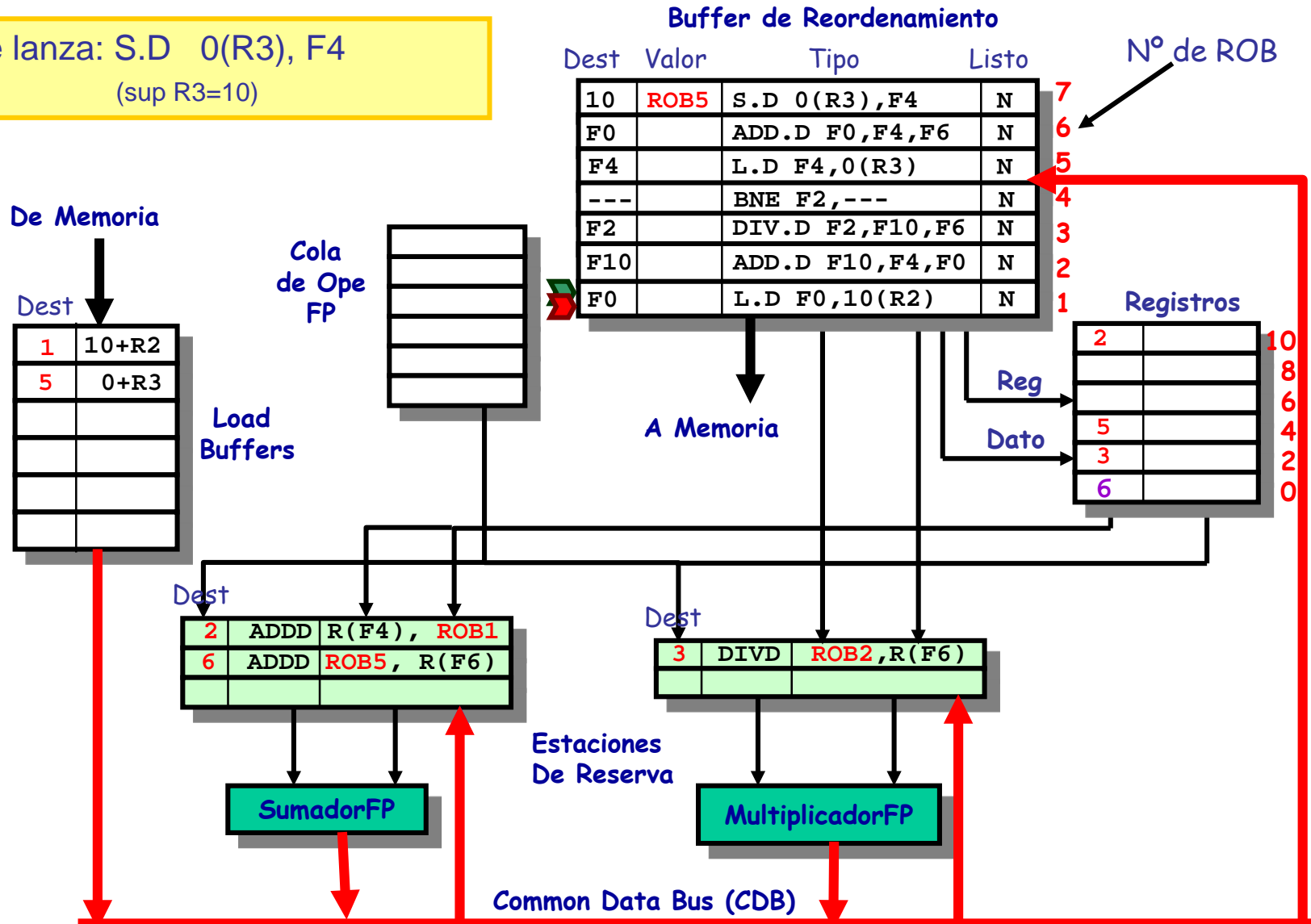
SumadorFP

MultiplicadorFP

Common Data Bus (CDB)

# Especulación: Ejemplo

Se lanza: S.D 0(R3), F4  
(sup R3=10)





# Especulación: Ejemplo

Se ejecuta: L.D F4,0(R3)

Buffer de Reordenamiento

| Dest | Valor | Tipo            | Listo | Nº de ROB |
|------|-------|-----------------|-------|-----------|
| 10   | M[10] | S.D 0(R3),F4    | Y     | 7         |
| F0   |       | ADD.D F0,F4,F6  | N     | 6         |
| F4   | M[10] | L.D F4,0(R3)    | Y     | 5         |
| ---  |       | BNE F2,---      | N     | 4         |
| F2   |       | DIV.D F2,F10,F6 | N     | 3         |
| F10  |       | ADD.D F10,F4,F0 | N     | 2         |
| F0   |       | L.D F0,10(R2)   | N     | 1         |

De Memoria

Dest

|   |       |
|---|-------|
| 1 | 10+R2 |
|   |       |
|   |       |
|   |       |
|   |       |
|   |       |

Load Buffers

Cola de Ope FP

A Memoria

Reg

Dato

Registros

|   |  |
|---|--|
| 2 |  |
|   |  |
|   |  |
| 5 |  |
| 3 |  |
| 6 |  |

Dest

|   |      |        |       |
|---|------|--------|-------|
| 2 | ADDD | R(F4), | ROB1  |
| 6 | ADDD | M[10], | R(F6) |
|   |      |        |       |
|   |      |        |       |

Dest

|   |      |       |       |
|---|------|-------|-------|
| 3 | DIVD | ROB2, | R(F6) |
|   |      |       |       |
|   |      |       |       |

Estaciones De Reserva

SumadorFP

MultiplicadorFP

Common Data Bus (CDB)

Resultado a ROB5 y todo ROB o ER con "marca" ROB5

# Especulación: Ejemplo

Se ejecuta: ADD.D F0,F4,F6

Buffer de Reordenamiento

| Dest | Valor | Tipo            | Listo |
|------|-------|-----------------|-------|
| 10   | M[10] | S.D 0(R3),F4    | Y     |
| F0   | F4+F6 | ADD.D F0,F4,F6  | Y     |
| F4   | M[10] | L.D F4,0(R3)    | Y     |
| ---  |       | BNE F2,---      | N     |
| F2   |       | DIV.D F2,F10,F6 | N     |
| F10  |       | ADD.D F10,F4,F0 | N     |
| F0   |       | L.D F0,10(R2)   | N     |

Nº de ROB

De Memoria

| Dest |       |
|------|-------|
| 1    | 10+R2 |
|      |       |
|      |       |
|      |       |
|      |       |
|      |       |

Cola de Ope FP

Load Buffers

A Memoria

Reg

Dato

Registros

|   |  |
|---|--|
| 2 |  |
|   |  |
|   |  |
| 5 |  |
| 3 |  |
| 6 |  |

Dest

|   |      |             |
|---|------|-------------|
| 2 | ADDD | R(F4), ROB1 |
|   |      |             |
|   |      |             |

Dest

|   |      |            |
|---|------|------------|
| 3 | DIVD | ROB2,R(F6) |
|   |      |            |
|   |      |            |

Estaciones De Reserva

SumadorFP

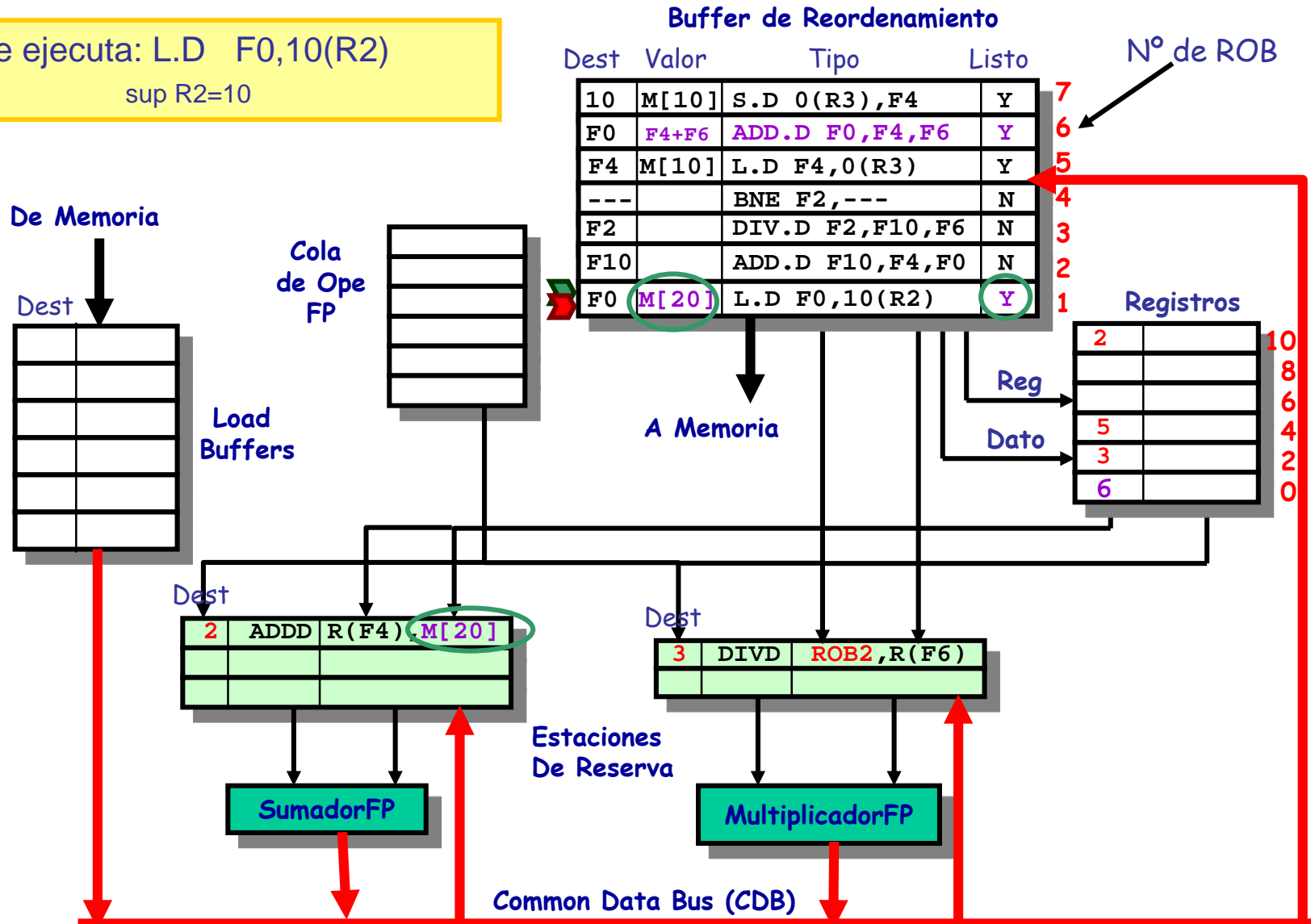
MultiplicadorFP

Common Data Bus (CDB)

Resultado a ROB6 y todo ROB o ER con "marca" ROB6

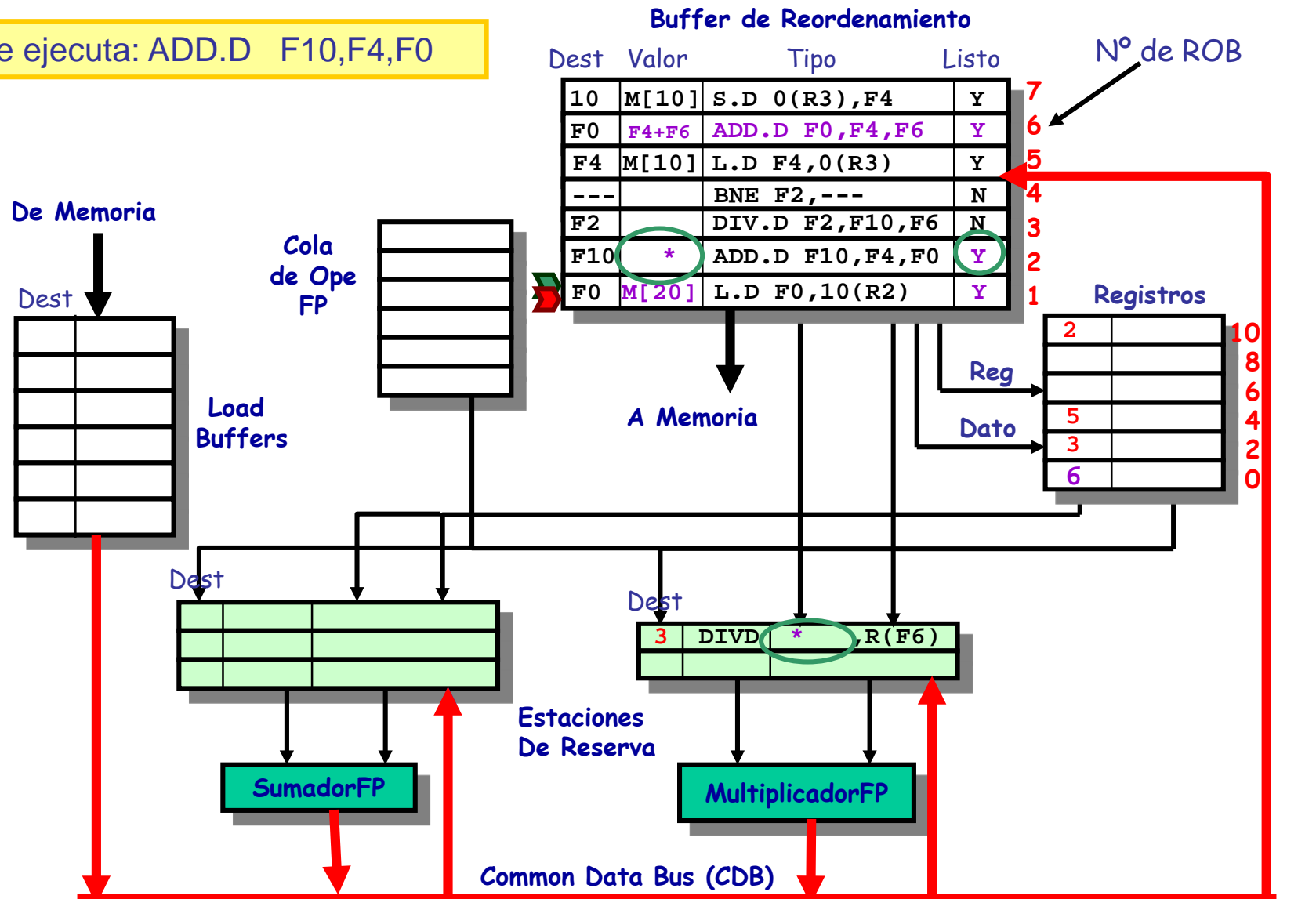
# Especulación: Ejemplo

Se ejecuta: L.D F0,10(R2)  
sup R2=10



## Especulación: Ejemplo

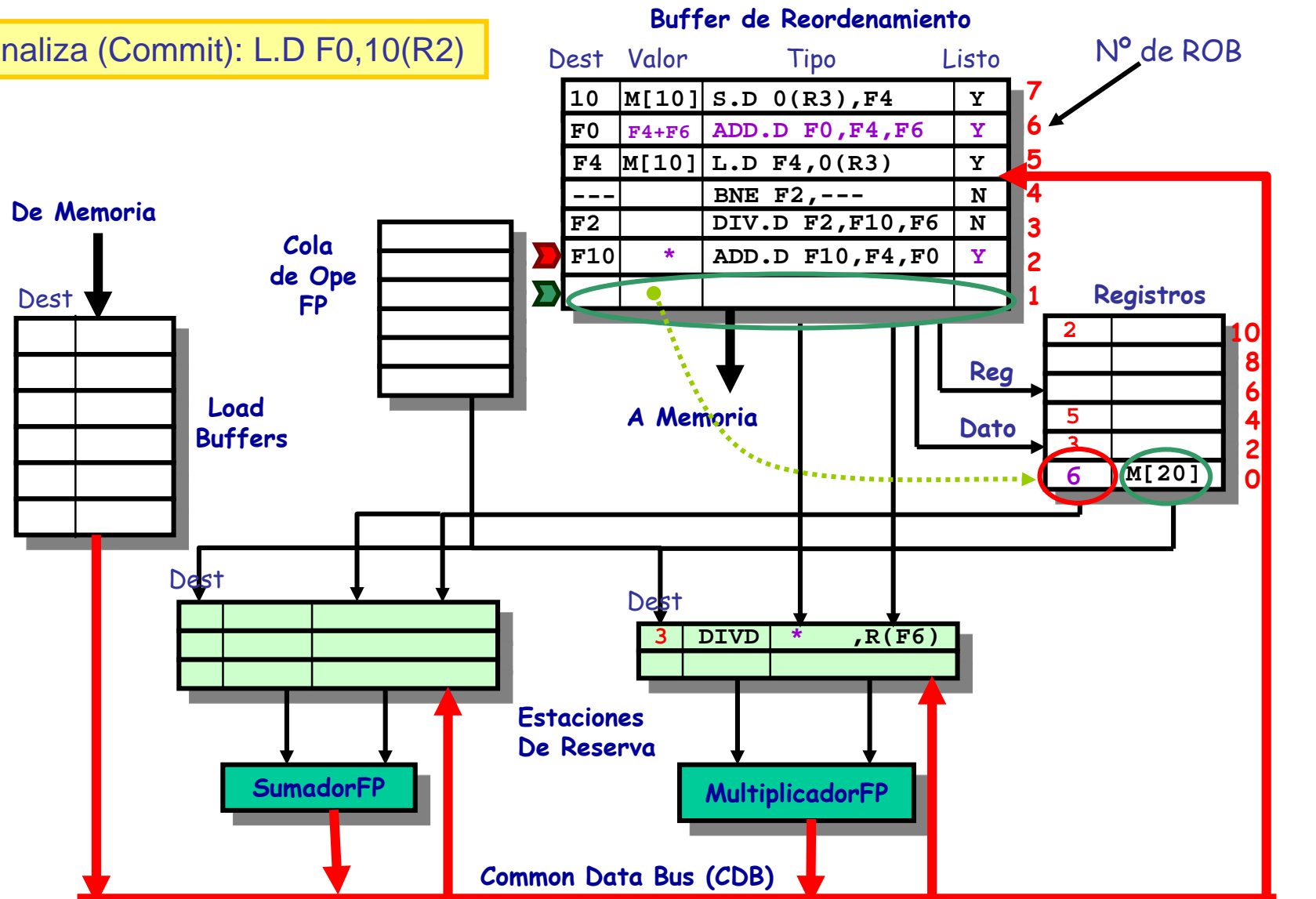
Se ejecuta: ADD.D F10,F4,F0



\* = R(F4)+M[20]

## Especulación: Ejemplo

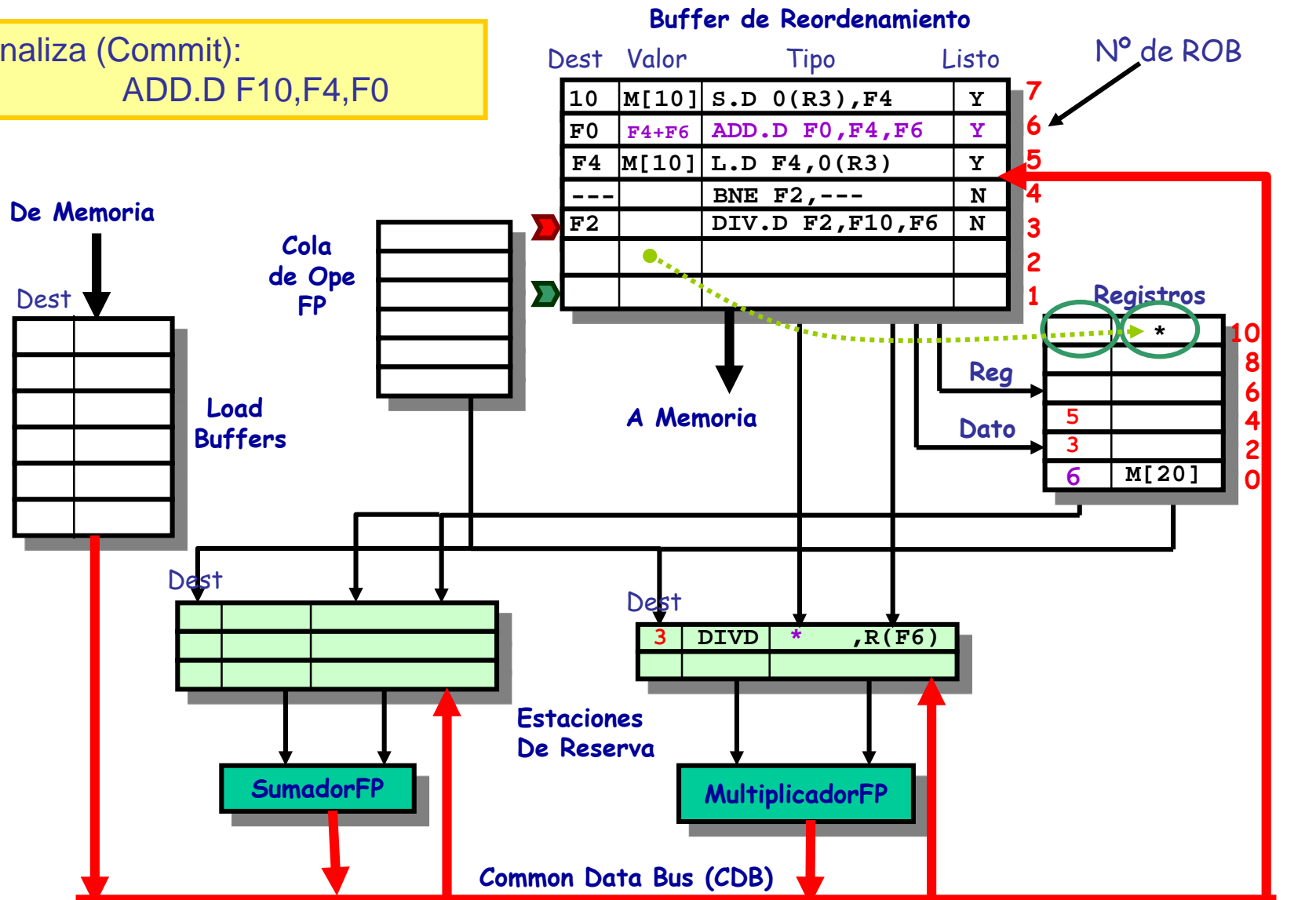
Finaliza (Commit): L.D F0,10(R2)



\* = R(F4)+M[20]

## Especulación: Ejemplo

Finaliza (Commit):  
ADD.D F10,F4,F0



\* = R(F4)+M[20]

# Especulación: Saltos e interrupciones

---

- ❑ El ROB permite recuperarse de saltos mal predichos e implementar un modelo de excepciones precisas
- ❑ Si una instrucción de salto bien predicha llega a cabecera de ROB =>
  - o Eliminarla de ROB
- ❑ Si una instrucción de salto mal predicha llega a cabecera de ROB =>
  - o Borrar contenido del ROB, ER y Buffer de Load
  - o Borrar marcas (campo "Nº de ROB)" de todos los registros.
  - o Buscar instrucción correcta.
- ❑ Si una instrucción genera una interrupción =>
  - o Registrar la petición en el ROB
  - o Si la instrucción llega a la cabecera del ROB (no especulada), entonces reconocer la interrupción.
  - o Cualquier instrucción anterior habrá finalizado. Por tanto ninguna instrucción anterior puede provocar una excepción.

- ❑ Riesgos EDE y EDL: no pueden aparecer dado que la actualización de memoria se hace en orden.
  - o Esperar hasta que la instrucción ST se halle en la cabecera de ROB => Todos los LD y ST anteriores se han completado.
  
- ❑ Riesgos LDE: Podrían producirse si un LD accede a la posición de memoria A, habiendo en el ROB un ST previo que almacena el resultado en A. Se evitan mediante el siguiente mecanismo:
  - o Un LD no ejecuta el acceso a memoria si hay un ST previo en el ROB con la misma dirección de memoria.
  - o Tampoco se ejecuta el LD si está pendiente el cálculo de la dirección efectiva de algún ST del ROB