Arquitectura de Computadores

SEGMENTACIÓN DE CAUCE
RIESGOS DE CONTROL

Riesgos de Control

- Cambios en el flujo de instrucciones perturban el cauce
  - lo más frecuente es un flujo de instrucciones **consecutivas**
  - un salto lo cambia y hay que desechar las instrucciones posteriores captadas hasta el momento del salto
- Estos son los Riesgos de Control

- Veremos algunas técnicas para lidiar con ellos
  - Estáticas: compilador
  - Dinámicas: tiempo de ejecución

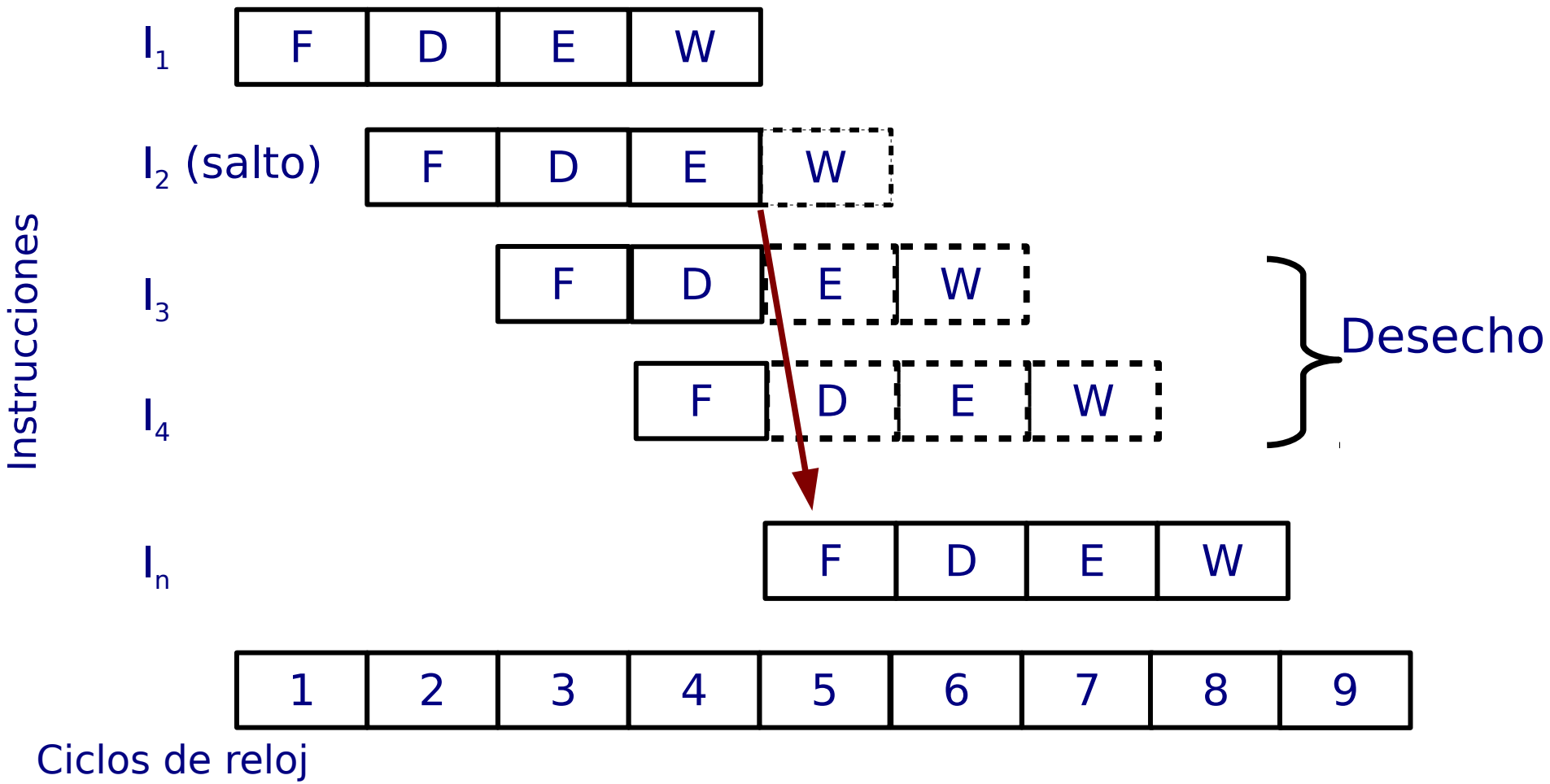
Resolución del Salto

- La instrucción de salto requiere hacer una suma para obtener la dirección del destino del salto
  - PC + desplazamiento en modo inmediato
- Para un salto condicional tb evaluar la condición
  - Lectura de un GPR (o lectura del PSW)
- EL PC se carga (o no) con la nueva dirección
- Todo esto es lo que llamamos “Resolución del salto”

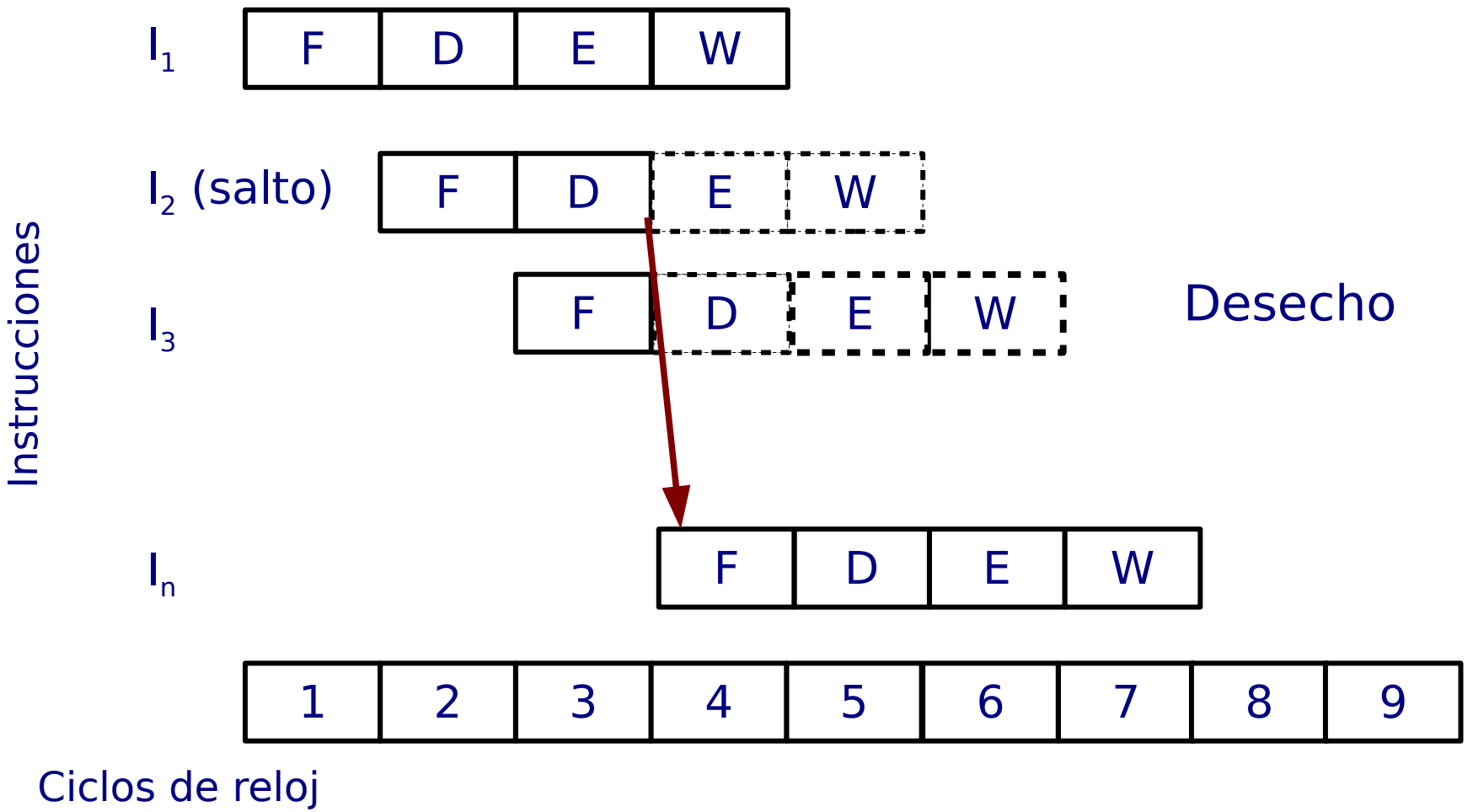
Magnitud de la perturbación

- Depende de la etapa en que se resuelve el salto.
- Contra más tarde, más instrucciones habrán entrado en el cauce
  - Se le llaman "Huecos de Retardo" (HR)
  - EX: 2 HR
  - DE: 1 HR (DLX)
  - Recordar que resolver en DE requiere HW extra:
    - detector precoz de saltos en el propio fetch
    - Sumador extra en DE
- Las instrucciones en los HR deben descartarse cuando el salto se toma

Etapa de Ejecución: 2 huecos de retardo



Etapa de Decodificación: Un hueco



Técnicas para reducir estos efectos

- Salto retardado
  - Confía en que el compilador sitúe una instrucción después de la del salto que siempre ha de ejecutarse y completarse, se salte o no.
- Técnicas especulativas
  - Predicción:
    - de que el salto se va a tomar y buscar instrucciones del destino del salto, O
    - de que NO se va a tomar y buscar las inmediatas al salto
  - Si se falla la predicción no se permite completar la(s) instrucciones en progreso en el cauce y se pierden ciclos

Salto retardado

- Dejar que se completen las instrucciones captadas situadas tras el salto que normalmente habría que desechar
- Reordernar instrucciones: poner instrucciones de "antes del salto" justo después para que sean éstas las que se completen
  - Se aprovecha el/los huecos de retardo con instrucciones "útiles"
- Esta reordenación la hace el compilador (estática)
- No siempre se encuentran
  - A veces no hay instrucciones que puedan sacarse de la secuencia normal debido a dependencias de datos
  - En ese caso hay que poner NOP y se desaprovecha el hueco

Ejemplo de salto retardado (DLX-1HR)

```
...
bucle:
lw      r1, a(r4);
lw      r2, b(r4);
subi    r3,r3,#1
add     r5, r1,r2
add     r6, r6,r5
sw      c(r4), r5
addi    r4,r4,#4
bnez    r3, bucle
nop
...
```

HR despediciado con NOP
Para 100 iteraciones:
9·100=900 ciclos sin paradas

```
...
bucle:
lw      r1, a(r4);
lw      r2, b(r4);
subi    r3,r3,#1
add     r5, r1,r2
add     r6, r6,r5
addi    r4,r4,#4
bnez    r3, bucle
sw      c-4(r4), r5
...
```

Tras reordenar:
8·100=800 ciclos sin paradas

A = 900/800 =1,125 --> 12,5%

Técnicas especulativas: Pred. estática

- Fija: siempre se predice lo mismo para todo salto
- Indicada por el compilador
  - Mediante un bit añadido al código de la instrucción
  - Repertorio permite indicar si ha de ser 1 o 0 para predicción de salto tomado (1) o no tomado (0)
  - Dinámicamente, cuando se ejecute el programa, el procesador buscará instrucciones para el/los HR:
    - (1) del destino del salto
    - (0) las inmediatas a la instrucción de salto
- Si la predicción
  - Se falla: no se permite completar la(s) instrucciones en progreso en el cauce y se pierden ciclos
  - Se acierta: se completan las instrucciones del HR

Predicción Estática (DLX-1HR)

```
...
bucle:
lw      r1, a(r4);
lw      r2, b(r4);
subi    r3,r3,#1
add     r5, r1,r2
add     r6, r6,r5
sw      c(r4), r5
addi    r4,r4,#4
bnez    r3, bucle
...
```

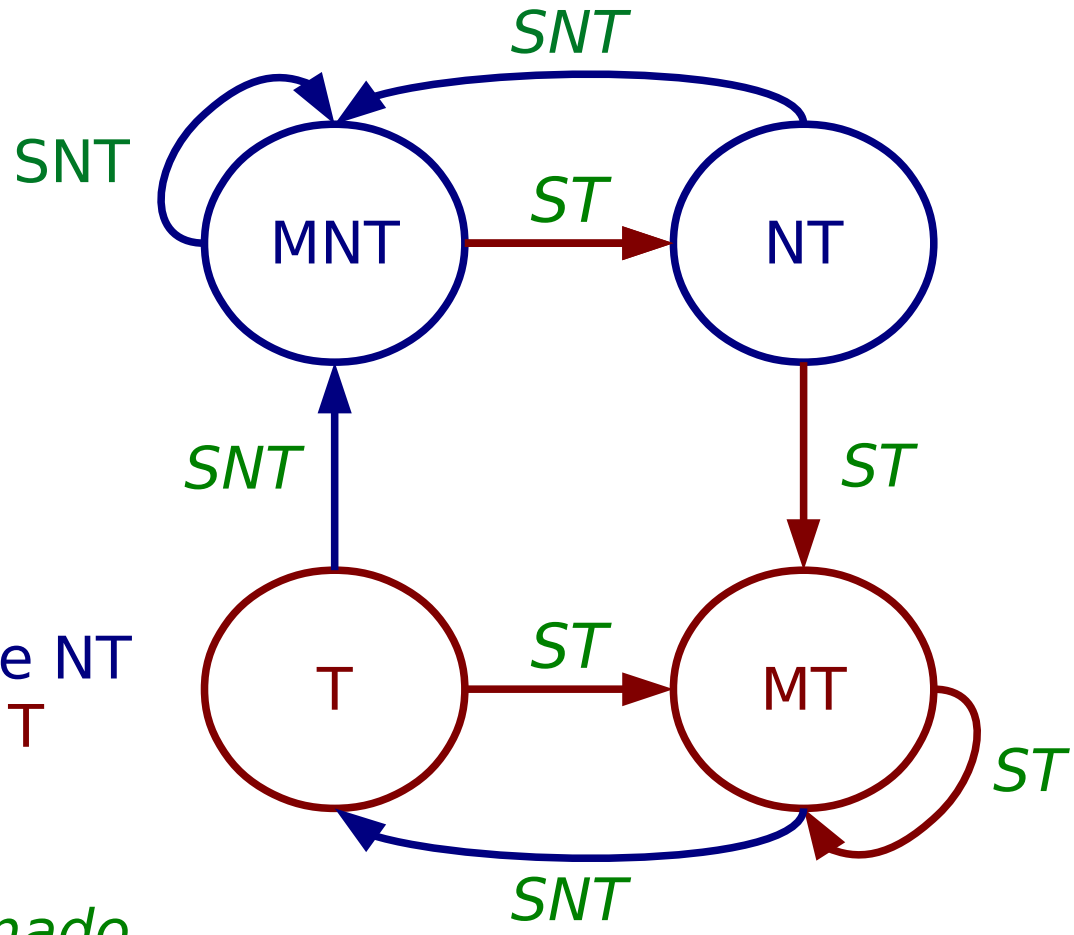
Predicción de Salto NO-tomado:
- Se fallan todas las iteraciones menos la última
- Tf = n-1/n (n iteraciones)
- 99 fallos: 8+1 ciclos/it = 891 ciclos;
- 1 acierto: 8 ciclos. Total = 899 ciclos

Predicción de Salto Tomado:
- Solo se falla la última iteración
- Tf = 1/n (n iteraciones)
- 99 aciertos: 8*99 = 792 ciclos
- 1 fallo: 9 ciclos. Total = 801 ciclos

Técnicas especulativas: Pred. Dinámica

- El procesador incorpora un hardware que evalúa la probabilidad de que el salto se tome o no cada vez que se ejecuta
- Se utiliza un Buffer de destino de saltos BTB (Branch Target Buffer) que guarda información de saltos recientes
- Asocia a **cada instrucción de salto** una historia de las ejecuciones recientes de dicha instrucción
  - ello le sitúa en un ESTADO en un autómata
  - el estado inicial se asigna por predicción estática
- En bucles visitados varias veces se puede lograr una única predicción errónea --> solo 1 o 2 ciclos perdidos

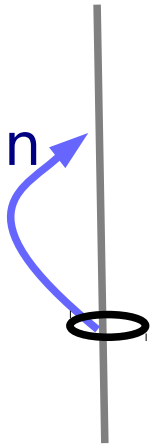
Ejemplo de predicción dinámica -2 bits



Predicciones:
NT= No Tomado
T= Tomado
MNT= Muy posible NT
MT= Muy posible T

ST: Salto Tomado
SNT: Salto No Tomado

Bucle simple n iteraciones



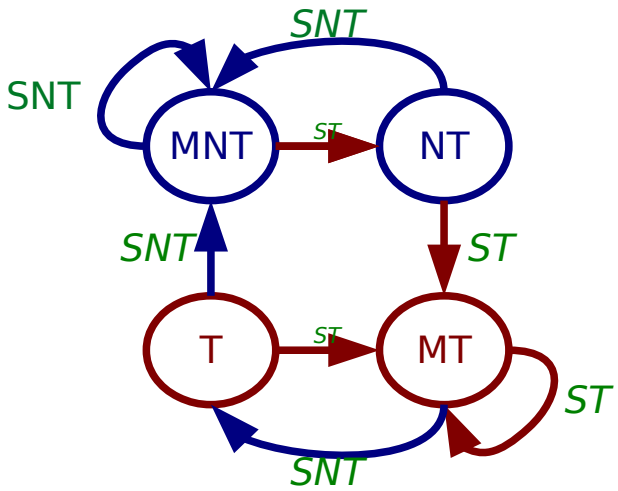
- Pred. inicial: NT.
  - 1º: ST Fallo (NT→ MT)
  - 2º →n-1: ST Acierto (MT→ MT)
  - n: SNT Fallo (MT→ T)
  - Total : 2 fallos

---->Tasa de Fallos Tf = 2/n

---> Ciclos de penalización
2 para 1 HR, 4 para 2 HR

- Supuesto de Pred. inicial: T.

--→ Deberías obtener Tf= 1/n



Contraste Dinámica - Estática

```
...
bucle:
lw      r1, a(r4);
lw      r2, b(r4);
subi    r3,r3,#1
add     r5, r1,r2
add     r6, r6,r5
sw      c(r4), r5
addi    r4,r4,#4
bnez    r3, bucle
sw      suma(r0), r6
```

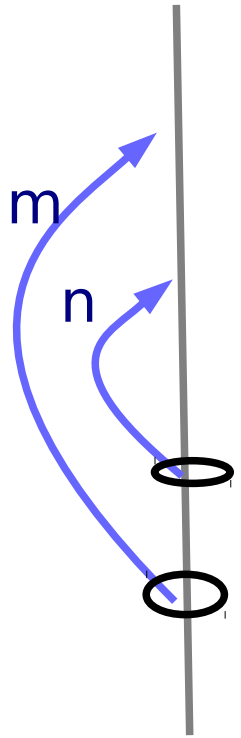
Ej: Pred. inicial NT:

Total : 2 fallos (Tasa fallos = 2%)
Penal = 2 cicles DLX (DS)

- Total = 8*100 + 2 = 802 cicles

Comparar con la misma pred. estática.

Bucles anidados



Bucle interior: PEI (Predicción Estática Inicial) NT
Bucle exterior: PEI MNT

Tasa de fallo global? $T_f = \Sigma \text{fallos} / \text{ total saltos}$

Total saltos = mn + m = m(n+1)
Bucle interior:
1ª visita: n iter. 2 fallos. Salida: T
2ª a m: c/u: n iters 1 fallo, salida T
= 2 + (m-1)*1 fallos
Bucle exterior:
m iters. MNT → NT → MT → T 3 fallos

$T_f = (m+4) / m(n+1)$