Técnicas hardware para extraer más paralelismo de instrucciones

- El paralelismo extraído por el compilador está limitado por las bifurcaciones que no son fácilmente predecibles
- Diversas técnicas ayudan a superar esa limitación
  - Instrucciones condicionales
  - Ejecución **especulativa** de instrucciones
    - ✓ Se ejecuta una instrucción sin que haya seguridad de que deba ejecutarse

Instrucciones condicionales..

- Son instrucciones cuyo significado es de la forma

  if *Condición* then *Operación* else *NOP*

- Los procesadores Alpha, MIPS, PowerPC y SPARC tienen una instrucción *move* condicional
- Principal ventaja:
  - Estas instrucciones permiten **eliminar bifurcaciones**. Ejemplo:

```
  BNEZ    R1,L
  MOV     R2,R3          ⇒        CMOVZ R2, R3, R1
L: ...
```

    - ✓ La eliminación de las bifurcaciones **facilita el movimiento de código y**, con ello, **la planificación global**

Instrucciones condicionales..

- Inconvenientes
  - La instrucción consume un ciclo de reloj aunque no haga ninguna operación
  - Cuando para llegar a una instrucción se tienen que cumplir **varias condiciones**, suprimir las bifurcaciones e introducir instrucciones condicionales requiere **instrucciones adicionales para calcular las condiciones**
  - Mayor complejidad de estas instrucciones =>
    - ✓ Mayor CPI que para la instrucción equivalente no condicional
    - ✓ O disminuir la frecuencia del reloj de la CPU

Instrucciones condicionales

- La mayoría de los procesadores sólo implementan unas **pocas** instrucciones condicionales **sencillas**
- Suelen funcionar bien
  - Cuando la **secuencia de instrucciones** dependiente de una bifurcación es **pequeña**
  - En los procesadores **superescalares**
    - ✓ Las instrucciones de bifurcación se emiten solas, lo que provoca burbujas que podrían ser ocupadas por instrucciones condicionales

Ejecución especulativa

- Los resultados no se consideran definitivos o visibles al resto del sistema hasta que se ha determinado la validez de la especulación
- La probabilidad de que las especulaciones sean acertadas ha de ser alta
  - De no ser así, el rendimiento podría empeorar por el tiempo gastado en:
    - ✓Ejecutar instrucciones sin deber
    - ✓Tareas para recuperar el sistema de los efectos de esas ejecuciones indebidas

Especulación soportada por el compilador con ayuda hardware..

- El **compilador** puede realizar movimientos especulativos de código
  - Colocar antes de una bifurcación una instrucción que depende de dicha bifurcación pero que previsiblemente se ejecutará
- Principales **limitaciones**
  - Hay que respetar las dependencias de datos
    - ✓ **Ni las instrucciones especulativas ni las interrupciones** generadas por ellas pueden hacer **modificaciones irreversibles** del estado del procesador
- Diversas **técnicas hardware** pueden **ayudar** a hacer efectivo el código especulativo generado por el compilador (bits para marcar las instrucciones especulativas, registros para almacenar resultados no definitivos, etc.)

Ejecución especulativa basada en hardware..

- Combina tres ideas clave:
  - Predicción dinámica de bifurcaciones
  - Especulación (ejecutar instrucciones antes de resolver las bifurcaciones de las que dependen)
  - Planificación dinámica (tipo Tomasulo, p. ej.)
- Ventajas sobre la especulación basada en el compilador
  - Más flexibilidad en el orden de las instrucciones (tienden a ejecutarse en cuanto están disponibles sus operandos) [sigue]

Ejecución especulativa basada en hardware (ventajas)..

- La **dirección** de muchas **referencias a memoria y**, por tanto, **la existencia o no de dependencias**, **no se conoce en tiempo de compilación** (pero **sí** en **tiempo de ejecución**) =>
  - ✓ el **hardware** puede hacer **cambios de orden,** en instrucciones con referencia a memoria, **imposibles para el compilador**
- La predicción dinámica de bifurcaciones es más eficiente que la estática
  - ✓ La especulación requiere un gran porcentaje de aciertos en las predicciones de las bifurcaciones

Ejecución especulativa basada en hardware (ventajas)

- Capacidad para mantener precisas las interrupciones
- No se necesita código compensatorio
- Especulación basada en
  - ✓ Hardware: no hay que recompilar el código compilado para otra máquina compatible
    - El mismo código se ejecutará eficientemente
  - ✓ Software: el mismo código no se ejecutará eficientemente (hay que recompilar)

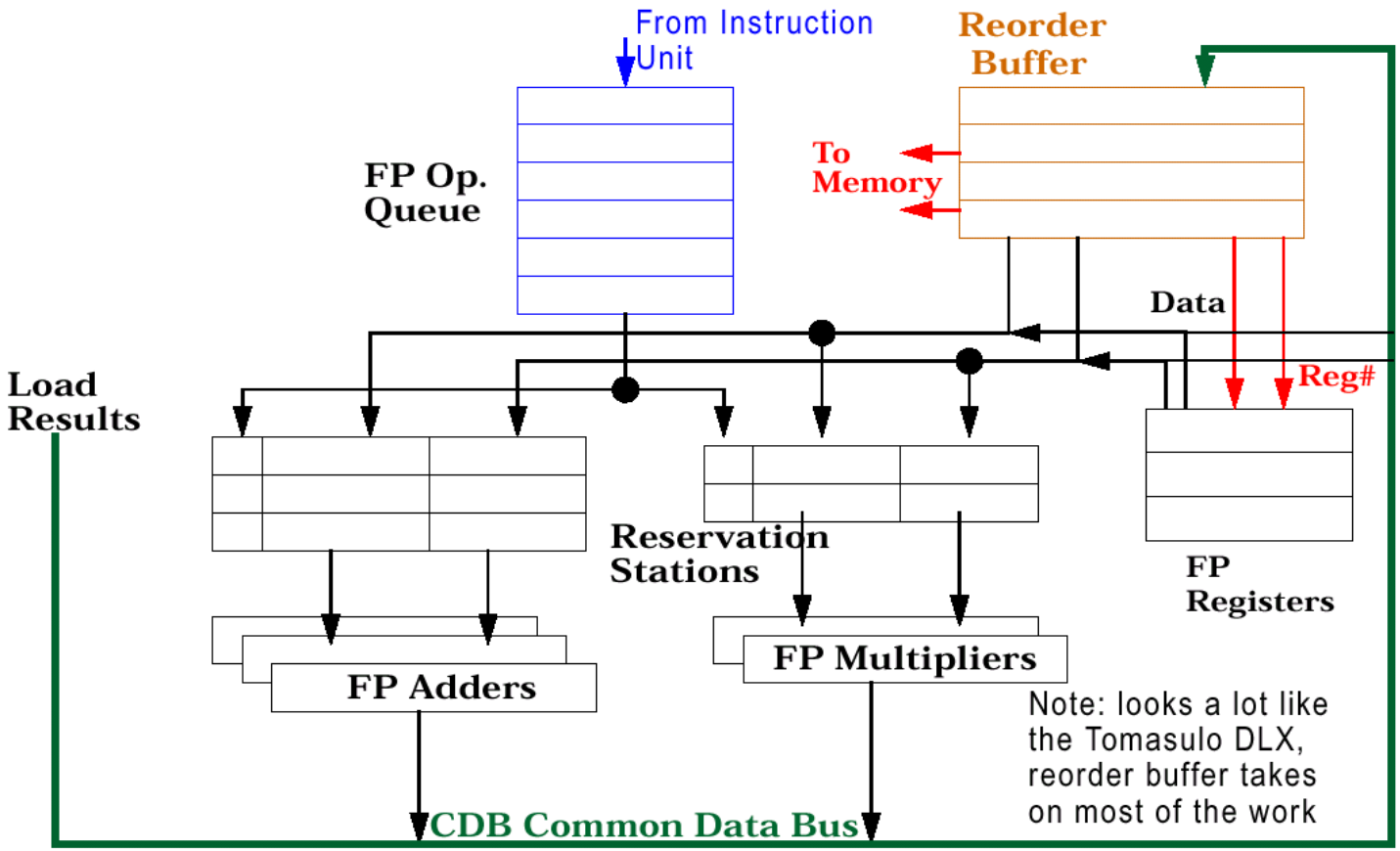
Ejecución especulativa basada en hardware..

- Principal inconveniente: complejidad del hardware
- Aproximación a la especulación basada en hardware
  - Especulación => **Ejecución fuera de orden**
  - Pero **las instrucciones dejan de ser especulativas en orden**
  - Una instrucción puede terminar su ejecución antes de dejar de ser especulativa
    - ✓ Hay que **separar** el proceso de **terminar la ejecución** del proceso por el que **la instrucción deja de ser especulativa**
  - Las instrucciones especulativas no deben hacer modificaciones irreversibles del estado

Implementación hardware de la ejecución especulativa: DLX FP especulativo

- Extensión del hardware del método de Tomasulo
- Hay que añadir:
  - Una etapa ***COMMIT*** en que las instrucciones dejan de ser especulativas (pasan a ser *comprometidas*)
  - El llamado ***REORDER BUFFER*** (**ROB**) para guardar las instrucciones y los **resultados** de las **instrucciones ejecutadas pero no comprometidas**
    - ✓ Similar a las estaciones de reserva
    - ✓ Integra las funciones de los buffers de carga y almacenamiento del algoritmo de Tomasulo

DLX FP especulativo



DLX FP especulativo: *reorder buffer*

- En cada entrada del ROB son destacables tres campos:
  - Tipo de instrucción:
    - ✓ bifurcación (no hay resultado destino)
    - ✓ almacenamiento (resultado en memoria)
    - ✓ carga y aritmético-lógica (resultado en registro)
  - Destino: identificación de registro o posición de memoria destino del resultado
    - ✓ Ello permite que el ROB sustituya a las estaciones de reserva en la redenominación de registros
  - Valor: Almacena el valor del resultado hasta que la instrucción deja de ser especulativa

DLX FP especulativo: etapas..

- ISSUE (a menudo llamada DISPATCH)
  - Tomar instrucción de la cola de operaciones de FP (en este estudio nos limitamos a instrucciones de punto flotante)
  - Esperar a que haya una **estación de reserva libre** para la operación y un **hueco en el ROB**. Entonces
    - ✓ **Emitir** la instrucción: enviarla a la estación de reserva y al ROB
      - El identificador de la entrada del ROB se anota en un campo adicional de la estación de reserva
    - ✓ **Enviar** los **operandos** (si están en los registros, en el ROB o en el CDB) **a la estación de reserva**

DLX FP especulativo: etapas..

- EJECUCIÓN (a veces llamada ISSUE)
  - Si los operandos están disponibles en la estación de reserva, ejecutar la operación
  - Si no, mirar el bus común de datos (CDB) para tomar el valor de los operandos cuando las unidades funcionales los coloquen en el bus
    - ✓ Esto soluciona las dependencias RAW

DLX FP especulativo: etapas..

- WRITE RESULT
  - Esperar a que el CDB quede libre. Entonces:
    - ✓ Escribir el resultado en el CDB
      - Adjuntar al resultado el identificador de la entrada de la instrucción en el *reorder buffer*
    - ✓ Llevar el resultado del CDB al *reorder buffer* y a las estaciones de reserva que lo estaban esperando
    - ✓ Marcar la estación de reserva como disponible

DLX FP especulativo: etapas

- COMMIT (también llamada COMPLETION o GRADUATION)
  - Si instrucción que alcanza la cabecera del ROB *es*
    - ✓una bifurcación predicha incorrectamente:
      - La **especulación** fue **equivocada**
      - Hay que **vaciar** el *reorder buffer* y continuar la ejecución en la instrucción correcta
    - ✓ otra instrucción y el resultado está en el ROB
      - **Actualizar** el registro o memoria con el resultado
      - **Retirar** la instrucción del ROB

DLX FP especulativo: estructura detallada..

- Estaciones de reserva (RS): como en Tomasulo salvo que:
  - **Qj (Qk)** no identifica la estación de reserva sino la **entrada del ROB** de la instrucción que producirá el valor del operando fuente S1 (S2)
  - Se añade un campo **Dest** para el identificador de la entrada del ROB correspondiente a la instrucción
- Tabla registros FP-resultados (trr)

  stro se tiene el **identificador** de la **entrada del**

  entrada del ROB de la **instrucción** cuyo

  tiene como **destino** el registro. Un **cero** indica que

  indica que **ninguna** instrucción en el ROB tiene

DLX FP especulativo: estructura detallada

- CDB. Dos partes:
  - Valor
  - Reorder: identificador de la entrada en el ROB de la instrucción que ha escrito el valor
- *Reorder buffer* (ROB). Campos por entrada:
  - Busy
  - Instrucción
  - Estado (etapa en la que se encuentra la instrucción)
  - Destino
    - ✓ Identificador del registro o posición de memoria destino del resultado
  - Valor
    - ✓ Almacena el valor del resultado hasta que la instrucción deja de ser especulativa

DLX FP especulativo: comportamiento detallado..

- Etapa ISSUE
  - Esperar a que una entrada del ROB, **b**, y una estación de reserva, **r**, estén libres. Entonces:
    - ✓ ROB[b].Busy ← yes; ROB[b].Destino ← 'D'; trr['D'] ← b;
      RS[r].Busy ← yes; RS[r].Dest ← b; RS[r].Op ← oper;
      if trr['S1'] = 0 **then** RS[r].V_j ← S1 **else**
        if ROB[trr['S1']].Valor ≠ Ø **then** RS[r].V_j ← ROB[trr['S1']].Valor
        else if CDB.Reorder = trr['S1'] **then** RS[r].V_j ← CDB.Valor;
        else RS[r].Q_j ← trr['S1'];
      if trr['S2'] = 0 **then** RS[r].V_k ← S2 **else**
        if ROB[trr['S2']].Valor ≠ Ø **then** RS[r].V_k ← ROB[trr['S2']].Valor
        else if CDB.Reorder = trr['S2'] **then** RS[r].V_k ← CDB.Valor;
        else RS[r].Q_k ← trr['S2'];

DLX FP especulativo: comportamiento detallado..

- Etapa EJECUCIÓN
  - Ejecutar la operación cuando los operandos y la FU estén disponibles, es decir, cuando
    - ✓ (RS[r].Qj = 0) and (RS[r].Qk = 0) and (FU disponible)
- Etapa WRITE RESULT
  - Esperar a que la ejecución de la instrucción haya terminado y el CDB esté disponible. Entonces:
    - ✓ CDB.Valor ← result; CDB.Reorder ← b;
      ∀x **if** RS[x].Q_j=b **then** {RS[x].V_j ← result; RS[x].Q_j ← 0};
      ∀x **if** RS[x].Q_k=b **then** {RS[x].V_k ← result; RS[x].Q_k ← 0};
      RS[r].busy ← no;
      ROB[b].Valor ← result;

DLX FP especulativo: comportamiento detallado..

- Etapa COMMIT
  - Esperar a que la instrucción esté en la cabecera (entrada que llamaremos h) del ROB y haya terminado su ejecución. Entonces:
    - ✓ **if** (ROB[h].Instrucción = Branch) and (bifurcación mal predicha) **then** {vaciar el ROB y la trr;
      actualizar el PC con la dirección del sucesor correcto} **else**
      if (ROB[h].Instrucción = Store) **then**
      Mem[ROB[h].Destino] ← ROB[h].Valor **else**
      Reg[ROB[h].Destino] ← ROB[h].Valor;
      trr[ROB[h].Destino] ← 0; ROB[h].Busy ← no

DLX FP especulativo: ejemplo
Ciclo 1

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1							
Add2							
Mult1							
Mult2							

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	Yes	LD F6, 34(R2)	Issue	F6		
2						
3						
4						
5						
6						

	F0	F2	F4	F6	F8	F10	...
trr				1			
Reg							

	CDB.Reorder =	CDB.Valor =	

DLX FP especulativo: ejemplo
Ciclo 2

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1							
Add2							
Mult1							
Mult2							

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	Yes	LD F6, 34(R2)	Ejec (1 d 2)	F6		
2	Yes	LD F2, 45(R3)	Issue	F2		
3						
4						
5						
6						

	F0	F2	F4	F6	F8	F10	...
trr		2		1			
Reg							

	CDB.Reorder =	CDB.Valor =	

DLX FP especulativo: ejemplo
Ciclo 3

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1							
Add2							
Mult1	Yes	MULTD		<F4>	#2		#3
Mult2							

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	Yes	LD F6, 34(R2)	Ejec (2 d 2)	F6		
2	Yes	LD F2, 45(R3)	Ejec (1 d 2)	F2		
3	Yes	MULTD F0,F2,F4	Issue	F0		
4						
5						
6						

	F0	F2	F4	F6	F8	F10	...
trr	3	2		1			
Reg							

	CDB.Reorder =	CDB.Valor =	

DLX FP especulativo: ejemplo
Ciclo 4

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	Yes	SUBD	<34+R2>			#2	#4
Add2							
Mult1	Yes	MULTD		<F4>	#2		#3
Mult2							

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	Yes	LD F6, 34(R2)	WR	F6	<34+R2>	
2	Yes	LD F2, 45(R3)	Ejec (2 d 2)	F2		
3	Yes	MULTD F0,F2,F4	Issue	F0		
4	Yes	SUBD F8,F6,F2	Issue	F8		
5						
6						

	F0	F2	F4	F6	F8	F10	...
trr	3	2		1	4		
Reg							

	CDB.Reorder = 1	CDB.Valor = <34+R2>	

DLX FP especulativo: ejemplo
Ciclo 5

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	Yes	SUBD	<34+R2>	<45+R3>		0	#4
Add2							
Mult1	Yes	MULTD	<45+R3>	<F4>	0		#3
Mult2	Yes	DIVD		<34+R2>	#3		#5

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	Yes	LD F2, 45(R3)	WR	F2	<45+R3>	
3	Yes	MULTD F0, F2, F4	Issue	F0		
4	Yes	SUBD F8, F6, F2	Issue	F8		
5	Yes	DIVD F10, F0, F6	Issue	F10		
6						

	F0	F2	F4	F6	F8	F10	...
trr	3	2		0	4	5	
Reg				<34+R2>			
	CDB.Reorder = 2			CDB.Valor = <45+R3>			

DLX FP especulativo: ejemplo
Ciclo 6

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	Yes	SUBD	<34+R2>	<45+R3>		0	#4
Add2	Yes	ADDD		<45+R3>	#4		#6
Mult1	Yes	MULTD	<45+R3>	<F4>	0		#3
Mult2	Yes	DIVD		<34+R2>	#3		#5

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	Yes	MULTD F0, F2, F4	Ejec (1 d 10)	F0		
4	Yes	SUBD F8, F6, F2	Ejec (1 d 2)	F8		
5	Yes	DIVD F10, F0, F6	Issue	F10		
6	Yes	ADDD F6, F8, F2	Issue	F6		

	F0	F2	F4	F6	F8	F10	...
trr	3	0		6	4	5	
Reg		<45+R3>		<34+R2>			
	CDB.Reorder =			CDB.Valor =			

DLX FP especulativo: ejemplo
Ciclo 8

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	Yes	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	Yes	MULTD	<45+R3>	<F4>	0		#3
Mult2	Yes	DIVD		<34+R2>	#3		#5

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	Yes	MULTD F0, F2, F4	Ejec (3 d 10)	F0		
4	Yes	SUBD F8, F6, F2	WR	F8	#1 - #2	
5	Yes	DIVD F10, F0, F6	Issue	F10		
6	Yes	ADDD F6, F8, F2	Issue	F6		

	F0	F2	F4	F6	F8	F10	...
trr	3	0		6	4	5	
Reg		<45+R3>		<34+R2>			
	CDB.Reorder = 4			CDB.Valor = #1 - #2			

DLX FP especulativo: ejemplo
Ciclo 10

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	Yes	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	Yes	MULTD	<45+R3>	<F4>	0		#3
Mult2	Yes	DIVD		<34+R2>	#3		#5

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	Yes	MULTD F0, F2, F4	Ejec (5 d 10)	F0		
4	Yes	SUBD F8, F6, F2	WR	F8	#1 - #2	
5	Yes	DIVD F10, F0, F6	Issue	F10		
6	Yes	ADDD F6, F8, F2	Ejec (2 d 2)	F6		

	F0	F2	F4	F6	F8	F10	...
trr	3	0		6	4	5	
Reg		<45+R3>		<34+R2>			
	CDB.Reorder =			CDB.Valor =			

DLX FP especulativo: ejemplo
Ciclo 11

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	**No**	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	**Yes**	**MULTD**	**<45+R3>**	**<F4>**	**0**		**#3**
Mult2	**Yes**	**DIVD**		**<34+R2>**	**#3**		**#5**

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	**Yes**	**MULTD F0, F2, F4**	**Ejec (6 d 10)**	**F0**		
4	**Yes**	**SUBD F8, F6, F2**	**WR**	**F8**	**#1 - #2**	
5	**Yes**	**DIVD F10, F0, F6**	**Issue**	**F10**		
6	**Yes**	**ADDD F6, F8, F2**	**WR**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	3	0		6	4	5	
Reg		<45+R3>		<34+R2>			

CDB.Reorder = 6 **CDB.Valor = #4 + #2**

DLX FP especulativo: ejemplo
Ciclo 15

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	No	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	**Yes**	**MULTD**	**<45+R3>**	**<F4>**	**0**		**#3**
Mult2	**Yes**	**DIVD**		**<34+R2>**	**#3**		**#5**

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	**Yes**	**MULTD F0, F2, F4**	**Ejec (10 d 10)**	**F0**		
4	**Yes**	**SUBD F8, F6, F2**	**WR**	**F8**	**#1 - #2**	
5	**Yes**	**DIVD F10, F0, F6**	**Issue**	**F10**		
6	**Yes**	**ADDD F6, F8, F2**	**WR**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	**3**	**0**		**6**	**4**	**5**	
Reg		**<45+R3>**		**<34+R2>**			

	CDB.Reorder =	CDB.Valor =	

DLX FP especulativo: ejemplo
Ciclo 16

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	No	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	**No**	MULTD	<45+R3>	<F4>	0		#3
Mult2	**Yes**	**DIVD**	**#2 * <F4>**	**<34+R2>**	**0**		**#5**

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	**Yes**	**MULTD F0, F2, F4**	**WR**	**F0**	**#2 * <F4>**	
4	**Yes**	**SUBD F8, F6, F2**	**WR**	**F8**	**#1 - #2**	
5	**Yes**	**DIVD F10, F0, F6**	**Issue**	**F10**		
6	**Yes**	**ADDD F6, F8, F2**	**WR**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	3	0		6	4	5	
Reg		<45+R3>		<34+R2>			

CDB.Reorder = 3 **CDB.Valor = #2 * <F4>**

DLX FP especulativo: ejemplo
Ciclo 17

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	No	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	No	MULTD	<45+R3>	<F4>	0		#3
Mult2	**Yes**	**DIVD**	**#2 * <F4>**	**<34+R2>**	**0**		**#5**

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	**No**	**MULTD F0, F2, F4**	**Commit**	**F0**	**#2 * <F4>**	
4	**Yes**	**SUBD F8, F6, F2**	**WR**	**F8**	**#1 - #2**	
5	**Yes**	**DIVD F10, F0, F6**	**Ejec (1 d 40)**	**F10**		
6	**Yes**	**ADDD F6, F8, F2**	**WR**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	**0**	**0**		**6**	**4**	**5**	
Reg	**#2 * <F4>**	**<45+R3>**		**<34+R2>**			

	CDB.Reorder =	CDB.Valor =	

DLX FP especulativo: ejemplo
Ciclo 18

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	No	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	No	MULTD	<45+R3>	<F4>	0		#3
Mult2	**Yes**	**DIVD**	**#2 * <F4>**	**<34+R2>**	**0**		**#5**

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	No	MULTD F0, F2, F4	Commit	F0	#2 * <F4>	
4	**No**	**SUBD F8, F6, F2**	**Commit**	**F8**	**#1 - #2**	
5	**Yes**	**DIVD F10, F0, F6**	**Ejec (2 d 40)**	**F10**		
6	**Yes**	**ADDD F6, F8, F2**	**WR**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	**0**	**0**		**6**	**0**	**5**	
Reg	**#2 * <F4>**	**<45+R3>**		**<34+R2>**	**#1 - #2**		
	CDB.Reorder =			**CDB.Valor =**			

DLX FP especulativo: ejemplo
Ciclo 56

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	No	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	No	MULTD	<45+R3>	<F4>	0		#3
Mult2	**Yes**	**DIVD**	**#2 * <F4>**	**<34+R2>**	**0**		**#5**

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	No	MULTD F0, F2, F4	Commit	F0	#2 * <F4>	
4	No	SUBD F8, F6, F2	Commit	F8	#1 - #2	
5	**Yes**	**DIVD F10, F0, F6**	**Ejec (40 d 40)**	**F10**		
6	**Yes**	**ADDD F6, F8, F2**	**WR**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	**0**	**0**		**6**	**0**	**5**	
Reg	**#2 * <F4>**	**<45+R3>**		**<34+R2>**	**#1 - #2**		
	CDB.Reorder =			**CDB.Valor =**			

DLX FP especulativo: ejemplo
Ciclo 57

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	No	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	No	MULTD	<45+R3>	<F4>	0		#3
Mult2	**No**	DIVD	#2 * <F4>	<34+R2>	0		#5

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	No	MULTD F0, F2, F4	Commit	F0	#2 * <F4>	
4	No	SUBD F8, F6, F2	Commit	F8	#1 - #2	
5	**Yes**	**DIVD F10, F0, F6**	**WR**	**F10**	**#3 - #1**	
6	**Yes**	**ADDD F6, F8, F2**	**WR**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	0	0		6	0	5	
Reg	#2 * <F4>	<45+R3>		<34+R2>	#1 - #2		

CDB.Reorder = 5 | **CDB.Valor = #3 - #1**

DLX FP especulativo: ejemplo
Ciclo 58

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	No	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	No	MULTD	<45+R3>	<F4>	0		#3
Mult2	No	DIVD	#2 * <F4>	<34+R2>	0		#5

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	No	MULTD F0, F2, F4	Commit	F0	#2 * <F4>	
4	No	SUBD F8, F6, F2	Commit	F8	#1 - #2	
5	**No**	**DIVD F10, F0, F6**	**Commit**	**F10**	**#3 - #1**	
6	**Yes**	**ADDD F6, F8, F2**	**WR**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	0	0		6	0	0	
Reg	#2 * <F4>	<45+R3>		<34+R2>	#1 - #2	#3 - #1	

	CDB.Reorder =	CDB.Valor =	

DLX FP especulativo: ejemplo
Ciclo 59

RS

r	Busy	Op	Vj	Vk	Qj	Qk	Dest
Add1	No	SUBD	<34+R2>	<45+R3>		0	#4
Add2	No	ADDD	#1 - #2	<45+R3>	0		#6
Mult1	No	MULTD	<45+R3>	<F4>	0		#3
Mult2	No	DIVD	#2 * <F4>	<34+R2>	0		#5

ROB

b	Busy	Instrucción	Estado	Destino	Valor	
1	No	LD F6, 34(R2)	Commit	F6	<34+R2>	
2	No	LD F2, 45(R3)	Commit	F2	<45+R3>	
3	No	MULTD F0, F2, F4	Commit	F0	#2 * <F4>	
4	No	SUBD F8, F6, F2	Commit	F8	#1 - #2	
5	No	DIVD F10, F0, F6	Commit	F10	#3 - #1	
6	**No**	**ADDD F6, F8, F2**	**Commit**	**F6**	**#4 + #2**	

	F0	F2	F4	F6	F8	F10	...
trr	**0**	**0**		**0**	**0**	**0**	
Reg	**#2 * <F4>**	**<45+R3>**		**#4 + #2**	**#1 - #2**	**#3 - #1**	

	CDB.Reorder =	CDB.Valor =	

ROB y precisión de las interrupciones

- El ROB permite conseguir interrupciones precisas
  - Los cambios del estado del modelo secuencial no son definitivos hasta que las instrucciones terminan la etapa *commit* (lo cual ocurre en orden)
  - Las interrupciones provocadas por una instrucción
    - ✓ Se anotan en el ROB
    - ✓ No se atienden hasta que la instrucción está en la cabecera del *reorder buffer*

Especulación en procesadores de emisión múltiple

- Dificultades por superar:
  - Decidir qué instrucciones pueden ser emitidas en el mismo ciclo
  - Renombrar los registros de varias instrucciones en un mismo ciclo
  - Es preciso aumentar la anchura del CDB
    - ✓ Mayor dificultad para observar desde las estaciones de reserva lo que se escribe en el CDB