

PLANIFICACIÓN DINÁMICA

❑ También denominada *reordenación dinámica de instrucciones*

- La **emisión** puede hacerse **fuera de orden**
- Consideremos

DIV.D F0, F2, F4

ADD.D F10, F0, F8

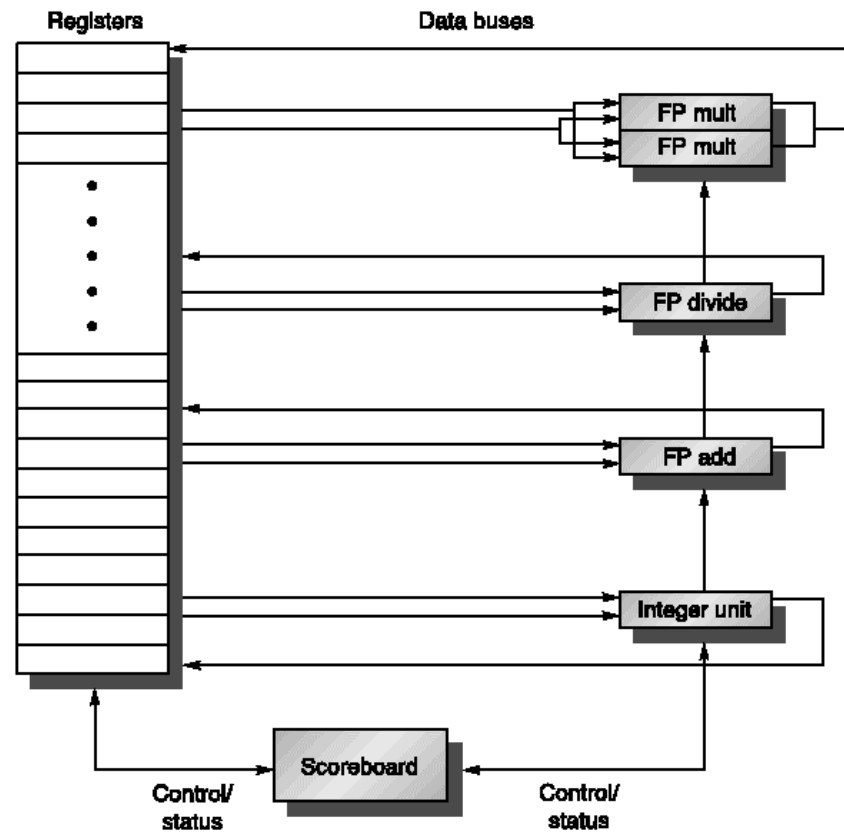
SUB.D F12, F8, F14

- DIV.D tiene una latencia grande y ADD.D una dependencia con DIV.D =>
 - Aconsejable **emitir antes SUB.D que ADD.D**

SCOREBOARD (marcador)

- ❑ **Hardware** que detecta los riesgos y **controla la emisión, ejecución y terminación de las instrucciones.**
 - Emisión en orden. Ejecución y terminación: fuera de orden.
 - Varias instrucciones en la etapa de ejecución (EX).
 - Varias unidades funcionales (posiblemente encauzadas) forman la etapa EX.
- ❑ En DLX (similar a MIPS)
 - Supondremos dos multiplicadores FP (10 ciclos), un divisor FP (40 ciclos), un sumador FP (2 ciclos) y una unidad funcional entera (1 ciclo).
 - Consideraremos sólo las etapas relevantes para la técnica de *scoreboard*: *Issue, read operands, EX y write result*.

MIPS CON SCOREBOARD: ESTRUCTURA BÁSICA



1/ Las líneas verticales son de control. 2/ Cada conjunto de buses (dos de entrada y uno de salida) es compartido por un grupo de unidades funcionales.

SCOREBOARD: ESTRUCTURA

- ❑ Tabla de **estados de instrucciones**: Indica en qué etapa está cada instrucción.
- ❑ Tabla de **estados de las unidades funcionales**. Por cada unidad funcional hay los siguientes campos:
 - *Busy*. Indica si la unidad funcional (U.F.) está ocupada o no.
 - *Op*. Operación que está siendo ejecutada, en su caso, en la U.F.
 - *Fi*. Registro destino del resultado de la U.F.
 - *Fj*, *Fk*. Registros fuente de los operandos que debe procesar la U.F.
 - *Qj*, *Qk*. Unidades funcionales que tienen como registros destino *Fj* y *Fk*.
 - *Rj*, *Rk*. Indicadores de que *Fj*, *Fk* están listos para ser procesados.
- ❑ Tabla **registros-resultados**. Por cada registro un campo.
 - Si una instrucción activa tiene el registro como su destino, el campo indicará qué **unidad funcional** escribirá en ese registro.
 - Si no, el campo queda en blanco.

SCOREBOARD: ESTADO INICIAL

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No								
Mult1	No								
Mult2	No								
Add	No								
Divide	No								

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional									

SCOREBOARD: COMPORTAMIENTO

❑ La etapa ID se divide en dos etapas:

- *Issue.*

- Descodifica la instrucción. No se emite si hay **riesgo estructural o WAW**.
 - Si hay una unidad funcional libre para la instrucción y ninguna instrucción activa tiene el mismo registro destino, el *scoreboard* emite la instrucción y actualiza su estructura de datos interna.
- Si una instrucción no se emite tampoco se emite ninguna instrucción posterior.
 - En consecuencia, el buffer de instrucciones FIFO que hay entre IF e ISSUE se llena con una instrucción más (la buscada por IF).

SCOREBOARD: COMPORTAMIENTO (cont.)

- *Read operands.*

- **Esperar** a que los **operandos fuente** estén **disponibles**.

- Un operando fuente está disponible si ninguna instrucción activa emitida antes está pendiente de escribirlo o si el registro que contiene el operando está siendo escrito por una unidad funcional.
 - Cuando los operandos fuente están disponibles el *scoreboard* ordena que la unidad funcional lea los operandos de los registros e inicie la ejecución.

- De esta forma **se resuelven las dependencias RAW dinámicamente**.

- Las instrucciones pueden enviarse a ejecución **fuera de orden**.

- Etapa EX.

- Cuando una unidad funcional termina una ejecución lo notifica al *scoreboard*.

SCOREBOARD: COMPORTAMIENTO (cont.)

- ❑ Etapa *write result* (reemplaza a WB).
 - Una vez que la unidad funcional ha terminado el *scoreboard* verifica las posibles **dependencias WAR**.
 - Si hay, detiene la instrucción.
 - Si no hay escribe el resultado.
 - Ejemplo:

```
DIVD F0,F2,F4  
ADDD F10,F0,F8  
SUBD F8,F8,F14
```
 - SUBD se ejecutará antes que ADDD (porque esta tiene una dependencia con DIVD) pero será detenida en la etapa WB hasta que ADDD (que se emitió antes) haya leído sus operandos para evitar la dependencia WAR.

SCOREBOARD: COMPORTAMIENTO DETALLADO

Notaciones:

- FU: Unidad funcional usada por la instrucción.
- C(F): Contenido de la columna C, fila F de la tabla de estados de las unidades funcionales.
- Result(X): Contenido del componente X del vector registros-resultados, es decir, unidad funcional que escribirá en X.
- op: Operación que corresponde a la instrucción.
- ‘D’, ‘S1’, ‘S2’: Nombres (identificadores) de los registros destino, fuente1 y fuente2, respectivamente.

Etapa	Esperar a	Registrar en tablas del <i>scoreboard</i>
Issue	Busy(FU) = No & Result(‘D’) = $\emptyset$	Result(‘D’) $\leftarrow$ FU; Busy(FU) $\leftarrow$ Yes; Op(FU) $\leftarrow$ op; Fi(FU) $\leftarrow$ ‘D’; Fj(FU) $\leftarrow$ ‘S1’; Fk(FU) $\leftarrow$ ‘S2’; Qj(FU) $\leftarrow$ Result(‘S1’); Rj(FU) $\leftarrow$ (if Qj(FU) = $\emptyset$ then Yes else No); Qk(FU) $\leftarrow$ Result(‘S2’); Rk(FU) $\leftarrow$ (if Qk(FU) = $\emptyset$ then Yes else No);
Read operands	Rj(FU) $\neq$ No $\neq$ Rk(FU)	Rj(FU) $\leftarrow$ No; Rk(FU) $\leftarrow$ No
Execution complete	Que la unidad funcional termine	
Write result	$\forall f$ (Fj(f) $\neq$ Fi(FU) or Rj(f) = No) & (Fk(f) $\neq$ Fi(FU) or Rk(f) = No)	$\forall f$ (if Qj(f) = FU then {Rj(f) $\leftarrow$ Yes; Qj(f) $\leftarrow$ $\emptyset$ }); $\forall f$ (if Qk(f) = FU then {Rk(f) $\leftarrow$ Yes; Qk(f) $\leftarrow$ $\emptyset$ }); Result(Fi(FU)) $\leftarrow$ $\emptyset$; Busy(FU) $\leftarrow$ No; $\forall c$ (if c $\neq$ Busy then c(FU) $\leftarrow$ $\emptyset$)

Se ha supuesto una instrucción con dos registros fuente y un registro destino, para contemplar el caso más general. Si la instrucción no tuviese, por ejemplo, registro fuente S2, habría que ignorar Fk(FU), Qk(FU), Rk(FU).

EJEMPLO SCOREBOARD: CICLO 1

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X			
LD F2, 45(R3)				
MULTD F0, F2, F4				
SUBD F8, F6, F2				
DIVD F10, F0, F6				
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F6	R2	Ø	Ø	Ø	Yes	Ø
Mult1	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Divide	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Ø	Ø	Ø	Integer	Ø	Ø	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 2

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X		
LD F2, 45(R3)				
MULTD F0, F2, F4				
SUBD F8, F6, F2				
DIVD F10, F0, F6				
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F6	R2	Ø	Ø	Ø	No	Ø
Mult1	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Divide	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Ø	Ø	Ø	Integer	Ø	Ø	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 4

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)				
MULTD F0, F2, F4				
SUBD F8, F6, F2				
DIVD F10, F0, F6				
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult1	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Divide	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 5

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X			
MULTD F0, F2, F4				
SUBD F8, F6, F2				
DIVD F10, F0, F6				
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2	R3	Ø	Ø	Ø	Yes	Ø
Mult1	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Divide	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Ø	Integer	Ø	Ø	Ø	Ø	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 6

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X		
MULTD F0, F2, F4	X			
SUBD F8, F6, F2				
DIVD F10, F0, F6				
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2	R3	Ø	Ø	Ø	No	Ø
Mult1	Yes	Mult	F0	F2	F4	Integer	Ø	No	Yes
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Divide	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Mult1	Integer	Ø	Ø	Ø	Ø	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 7

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X	X	
MULTD F0, F2, F4	X			
SUBD F8, F6, F2	X			
DIVD F10, F0, F6				
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2	R3	Ø	Ø	Ø	No	Ø
Mult1	Yes	Mult	F0	F2	F4	Integer	Ø	No	Yes
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	Yes	Sub	F8	F6	F2	Ø	Integer	Yes	No
Divide	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Mult1	Integer	Ø	Ø	Add	Ø	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 8

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X	X	X
MULTD F0, F2, F4	X			
SUBD F8, F6, F2	X			
DIVD F10, F0, F6	X			
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Mult1	Yes	Mult	F0	F2	F4	$\emptyset$	$\emptyset$	Yes	Yes
Mult2	No	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Add	Yes	Sub	F8	F6	F2	$\emptyset$	$\emptyset$	Yes	Yes
Divide	Yes	Div	F10	F0	F6	Mult1	$\emptyset$	No	Yes

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Mult1	$\emptyset$	$\emptyset$	$\emptyset$	Add	Divide	$\emptyset$	$\emptyset$	$\emptyset$

EJEMPLO SCOREBOARD: CICLO 9

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X	X	X
MULTD F0, F2, F4	X	X		
SUBD F8, F6, F2	X	X		
DIVD F10, F0, F6	X			
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult1	Yes	Mult	F0	F2	F4	Ø	Ø	No	No
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	Yes	Sub	F8	F6	F2	Ø	Ø	No	No
Divide	Yes	Div	F10	F0	F6	Mult1	Ø	No	Yes

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Mult1	Ø	Ø	Ø	Add	Divide	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 12

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X	X	X
MULTD F0, F2, F4	X	X		
SUBD F8, F6, F2	X	X	X	X
DIVD F10, F0, F6	X			
ADDD F6, F8, F2				

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult1	Yes	Mult	F0	F2	F4	Ø	Ø	No	No
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Divide	Yes	Div	F10	F0	F6	Mult1	Ø	No	Yes

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Mult1	Ø	Ø	Ø	Ø	Divide	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 16

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X	X	X
MULTD F0, F2, F4	X	X		
SUBD F8, F6, F2	X	X	X	X
DIVD F10, F0, F6	X			
ADDD F6, F8, F2	X	X	X	

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult1	Yes	Mult	F0	F2	F4	Ø	Ø	No	No
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	Yes	Add	F6	F8	F2	Ø	Ø	No	No
Divide	Yes	Div	F10	F0	F6	Mult1	Ø	No	Yes

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Mult1	Ø	Ø	Add	Ø	Divide	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 20

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X	X	X
MULTD F0, F2, F4	X	X	X	X
SUBD F8, F6, F2	X	X	X	X
DIVD F10, F0, F6	X			
ADDD F6, F8, F2	X	X	X	

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult1	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	Yes	Add	F6	F8	F2	Ø	Ø	No	No
Divide	Yes	Div	F10	F0	F6	Ø	Ø	Yes	Yes

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Ø	Ø	Ø	Add	Ø	Divide	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 22

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X	X	X
MULTD F0, F2, F4	X	X	X	X
SUBD F8, F6, F2	X	X	X	X
DIVD F10, F0, F6	X	X		
ADDD F6, F8, F2	X	X	X	X

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult1	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Divide	Yes	Div	F10	F0	F6	Ø	Ø	No	No

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Ø	Ø	Ø	Ø	Ø	Divide	Ø	Ø	Ø

EJEMPLO SCOREBOARD: CICLO 62

Tabla de estados de las instrucciones.

Instrucción	Issue	Read operands	Execution complete	Write result
LD F6, 34(R2)	X	X	X	X
LD F2, 45(R3)	X	X	X	X
MULTD F0, F2, F4	X	X	X	X
SUBD F8, F6, F2	X	X	X	X
DIVD F10, F0, F6	X	X	X	X
ADDD F6, F8, F2	X	X	X	X

Tabla de estados de las unidades funcionales.

Unidad funcional	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult1	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Mult2	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Add	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
Divide	No	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Tabla registros-resultados.

	F0	F2	F4	F6	F8	F10	F12	...	F30
Unidad funcional	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

SCOREBOARD: COSTES, BENEFICIOS Y LIMITACIONES

- ❑ Mejora del rendimiento en el CDC 6600:
 - 70% en los programas FORTRAN.
 - 150% en los programas escritos directamente en ensamblador.
- ❑ Coste:
 - En el CDC la lógica del *scoreboard* resultó de una complejidad similar a la de una unidad funcional. ¡Sorprendentemente bajo!
 - El principal componente del coste es el gran número de buses entre los registros y las unidades funcionales.
 - Hay un número limitado de buses lo que supone una limitación al número de unidades funcionales que pueden, a la vez, continuar las etapas *read operands* y *write result*.

SCOREBOARD: COSTES, BENEFICIOS Y LIMITACIONES

- ❑ **Interés actual** en técnicas de planificación dinámica como esta.
 - Motivado por el objetivo de emitir varias instrucciones por ciclo (el coste de tener más buses es, por tanto, inevitable) e ideas como la especulación.
- ❑ Limitaciones:
 - Número de entradas del *scoreboard*. Determina el tamaño de la ventana.
 - Ventana es el conjunto de instrucciones examinadas como candidatas para ser ejecutadas.
 - Número y tipos de unidades funcionales.
 - Esto determina que haya más o menos riesgos estructurales.
 - La aparición de dependencias WAR y WAW.
 - Consecuencia de la ejecución y terminación fuera de orden.

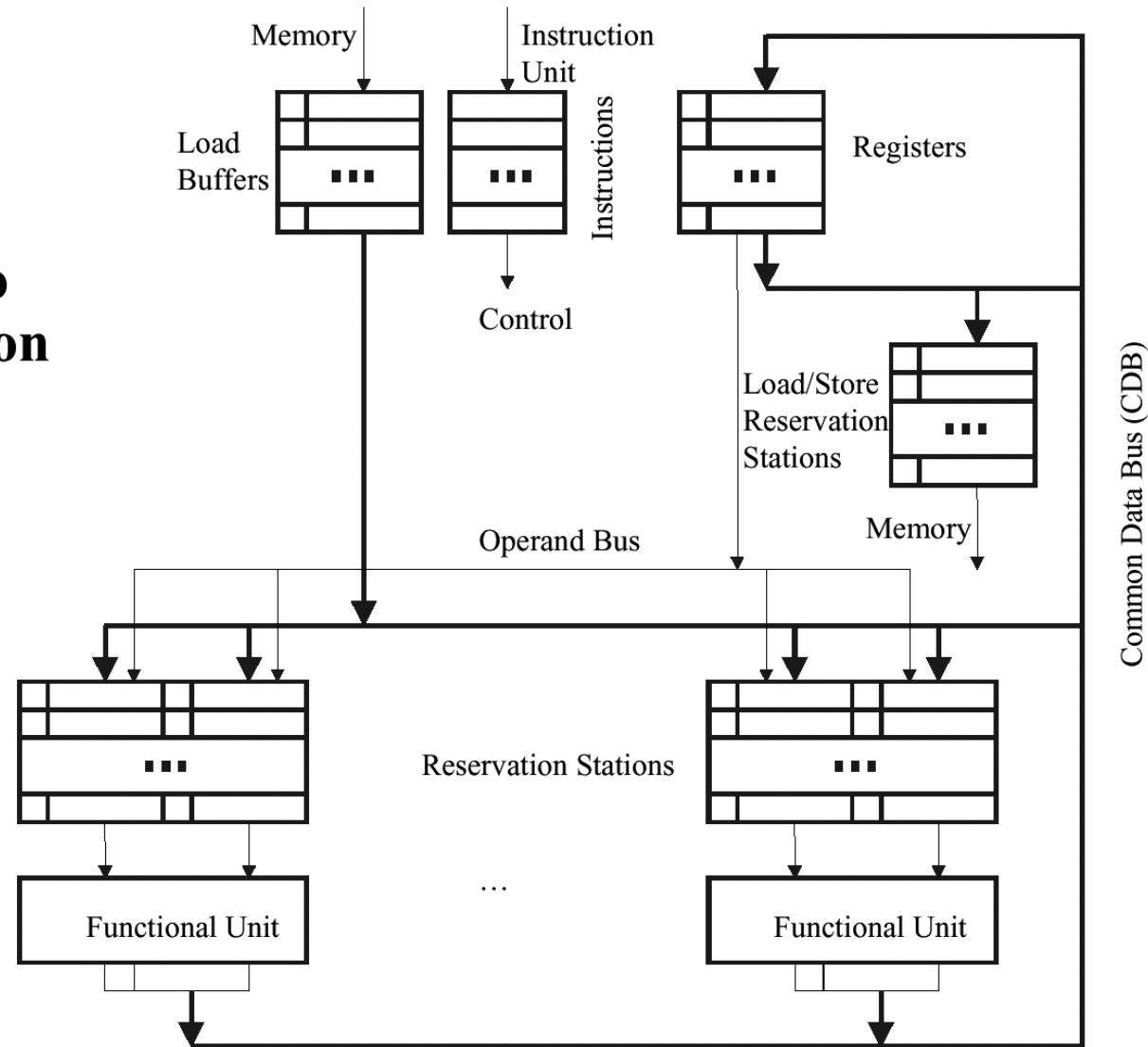
PLANIFICACIÓN DINÁMICA MEDIANTE EL ALGORITMO DE TOMASULO

- ❑ Esquema, inventado por Robert Tomasulo e implementado por primera vez en el IBM 360/91 (año 1967).
- ❑ Tras un largo periodo en que decayó su uso, desde los años 90 es de empleo generalizado, con múltiples variantes.
 - Procesadores recientes que usan variantes de este esquema: Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604, ...
- ❑ Combina la redenominación de registros (para eliminar dinámicamente las dependencias WAR y WAW) con elementos clave del *scoreboarding*.
- ❑ Emisión en orden. Ejecución y terminación: fuera de orden.
- ❑ Control distribuido (en el scoreboard es centralizado).

ALGORITMO DE TOMASULO

- ❑ Cuando una instrucción descodificada no tiene disponibles sus operandos, en lugar de esperar en la etapa *issue* impidiendo que avancen otras instrucciones...
 - Es enviada a una **estación de reserva**, o **RS** (*Reservation Station*), asociada con la unidad funcional que utilizará.
 - La instrucción **esperará** en la estación de reserva hasta que sus **operandos** estén **disponibles**.
 - Para saber si están disponibles observará el **bus** de resultados o **CDB** (*Common Data Bus*).
- ❑ Cuando todos los operandos de la instrucción estén disponibles, es enviada a la correspondiente unidad funcional para su ejecución.
- ❑ El resultado se envía al CDB y, desde el CDB, al registro destino y a todas las estaciones de reserva que estuvieran esperando ese resultado.

Tomasulo Organization



TOMASULO: ESTRUCTURA DETALLADA

- ❑ Cada **estación de reserva** es identificable mediante una marca única y se compone de los siguientes **campos**:
 - **Busy**. La estación de reserva está ocupada.
 - **Op**. La operación a ejecutar.
 - **Q_j (Q_k)** Identificador de la **estación de reserva que producirá el valor del operando** fuente S1 (S2).
 - Un valor cero indica que el operando fuente ya está disponible (o es innecesario).
 - **V_j (V_k)**. Valor del operando fuente S1 (S2), si Q_j (Q_k) vale cero.
- ❑ **Estaciones de reserva del buffer de carga**. Campos:
 - **Address**. Dirección de la que debe leer la instrucción *load*.
 - **Busy**. Ocupado o no. Si *busy=no*, se ignora *address*.

TOMASULO: ESTRUCTURA DETALLADA (cont.)

- ❑ **Estaciones de reserva del buffer de almacenamiento.** Campos:
 - **Address.** Dirección en la que debe escribir la instrucción *store*.
 - **Q_i .** Identificador de la **RS** cuya instrucción **producirá el resultado** que debe almacenarse en memoria.
 - **V.** Valor a escribir.
 - **Busy.** Ocupado o no. Si *busy=no*, los otros campos se ignoran.
- ❑ La **tabla registros-resultados** contiene, por cada registro, un **campo Q_i** que indica:
 - El identificador de la **RS** cuya instrucción **producirá el resultado** que debe almacenarse en ese registro.
 - Valor cero: ninguna instrucción activa tiene como destino el registro.

TOMASULO: COMPORTAMIENTO DETALLADO

□ Notaciones:

- **oper**: operación que corresponde a la instrucción.
- **'D', 'S1', 'S2'**: nombres (identificadores) de los registros destino, fuente1 y fuente2, respectivamente.
- **result**: valor producido por la instrucción de carga o aritmético-lógica.
- **trr**: tabla registros-resultados.
- **rsstore**: tabla de estaciones de reserva del buffer de almacenamiento.

TOMASULO: COMPORTAMIENTO DETALLADO (cont.)

- ❑ Etapa *issue* (supondremos que la instrucción es **aritmético-lógica**).
 - **Esperar** a que al menos una **estación de reserva**, **r**, asociada a la FU necesitada por la instrucción esté **desocupada**. Entonces...
 - **Enviar** la instrucción a la estación de reserva **r**, es decir:
 - Actualizar campos de la estación de reserva **r** (**RS[r]**):
 $RS[r].busy \leftarrow \text{yes}; RS[r].op \leftarrow \text{oper};$
 $\text{if } (trr['S1'].Q_i = 0) \text{ then } \{ RS[r].V_j \leftarrow S1; RS[r].Q_j \leftarrow 0 \}$
 $\text{else } \{ RS[r].Q_j \leftarrow trr['S1'].Q_i \} \quad /* \text{ campos } Q_j/V_j */$
 $\text{if } (trr['S2'].Q_i = 0) \text{ then } \{ RS[r].V_k \leftarrow S2; RS[r].Q_k \leftarrow 0 \}$
 $\text{else } \{ RS[r].Q_k \leftarrow trr['S2'].Q_i \} \quad /* \text{ campos } Q_k/V_k */$
 - Hacer la siguiente anotación: $trr['D'].Q_i \leftarrow r$.

TOMASULO: COMPORTAMIENTO DETALLADO (cont.)

□ Etapa *execute*.

- Esperar operandos y FU disponibles, es decir, esperar hasta que
 $(RS[r].Q_j = 0) \text{ and } (RS[r].Q_k = 0) \text{ and } (FU \text{ disponible})$
- Entonces ejecutar la instrucción (los operandos están en V_j y V_k).

□ Etapa *write result*.

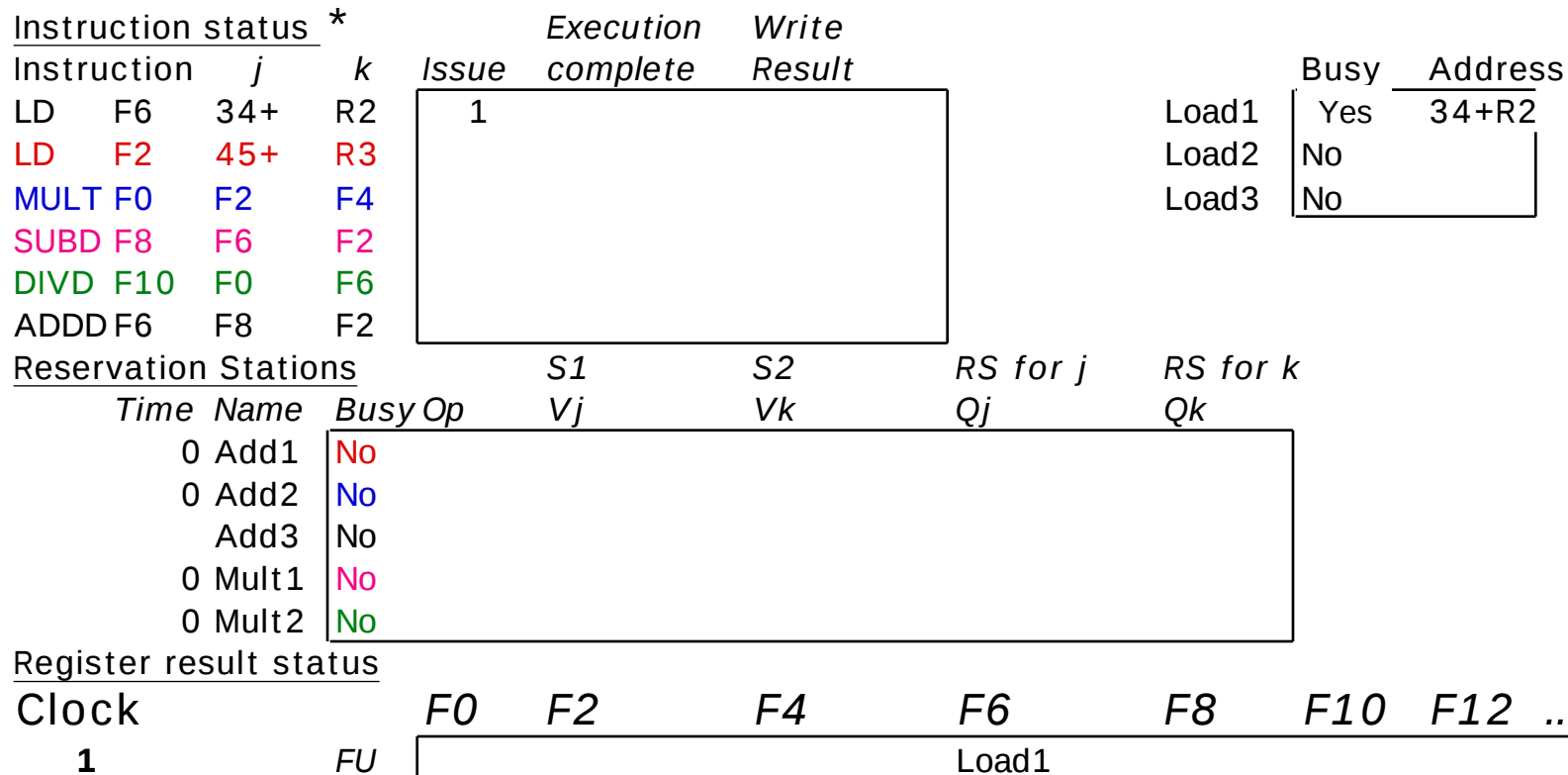
- Esperar a que la ejecución de la instrucción haya terminado y el CDB esté disponible. Entonces...

$\forall x \text{ (if } (RS[x].Q_j = r) \text{ then } \{RS[x].V_j \leftarrow \text{result}; RS[x].Q_j \leftarrow 0\};$
 $\forall x \text{ (if } (RS[x].Q_k = r) \text{ then } \{RS[x].V_k \leftarrow \text{result}; RS[x].Q_k \leftarrow 0\};$
 $\forall x \text{ (if } (rsstore[x].Q_i = r) \text{ then } \{rsstore[x].V \leftarrow \text{result}; rsstore[x].Q_i \leftarrow 0\};$
 $\forall x \text{ (if } (trr[x].Q_i = r) \text{ then } \{Reg[x] \leftarrow \text{result}; trr[x].Q_i \leftarrow 0\};$
 $RS[r].busy \leftarrow \text{no}$

Tomasulo Example Cycle 0

Instruction status					Execution	Write					
Instruction	<i>j</i>	<i>k</i>		Issue	complete	Result			Busy	Address	
LD	F6	34+	R2					Load1	No		
LD	F2	45+	R3					Load2	No		
MULT	F0	F2	F4					Load3	No		
SUBD	F8	F6	F2								
DIVD	F10	F0	F6								
ADDD	F6	F8	F2								
Reservation Stations					S1	S2	RS for <i>j</i>	RS for <i>k</i>			
	Time	Name	Busy Op		<i>V_j</i>	<i>V_k</i>	<i>Q_j</i>	<i>Q_k</i>			
	0	Add1	No								
	0	Add2	No								
	0	Add3	No								
	0	Mult1	No								
	0	Mult2	No								
Register result status											
Clock					F0	F2	F4	F6	F8	F10	F12 ... F30
0			FU								

Tomasulo Example Cycle 1



* No hay una tabla "instruction status" pero la información equivalente existe (distribuida por el hardware)

Tomasulo Example Cycle 2

Instruction status				Issue	Execution complete	Write Result		
Instruction	<i>j</i>	<i>k</i>					Busy	Address
LD F6	34+	R2		1	*		Load1	Yes 34+R2
LD F2	45+	R3		2			Load2	Yes 45+R3
MULT F0	F2	F4					Load3	No
SUBD F8	F6	F2						
DIVD F10	F0	F6						
ADDD F6	F8	F2						

Reservation Stations			S1	S2	RS for <i>j</i>	RS for <i>k</i>
Time	Name	Busy Op	<i>V_j</i>	<i>V_k</i>	<i>Q_j</i>	<i>Q_k</i>
0	Add1	No				
0	Add2	No				
	Add3	No				
0	Mult1	No				
0	Mult2	No				

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock										
2	FU		Load2		Load1					

Note: Unlike 6600, can have multiple loads outstanding

* Se supone que LOAD consume 2 ciclos en la etapa execution

Tomasulo Example Cycle 3

Instruction status				Issue	Execution complete	Write Result		
Instruction	<i>j</i>	<i>k</i>					Busy	Address
LD F6	34+	R2		1	3		Load1	Yes 34+R2
LD F2	45+	R3		2			Load2	Yes 45+R3
MULT F0	F2	F4		3			Load3	No
SUBD F8	F6	F2						
DIVD F10	F0	F6						
ADDD F6	F8	F2						

Reservation Stations			S1	S2	RS for <i>j</i>	RS for <i>k</i>
Time	Name	Busy Op	<i>V_j</i>	<i>V_k</i>	<i>Q_j</i>	<i>Q_k</i>
0	Add1	No				
0	Add2	No				
	Add3	No				
0	Mult1	Yes MULTD		R(F4)	Load2	
0	Mult2	No				

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock										
3	FU	Mult1	Load2		Load1					

- Note: registers names are removed (“renamed”) in Reservation Stations; MULT issued vs. scoreboard
- Load1 completing; what is waiting for Load1?

Tomasulo Example Cycle 4

Instruction status					Execution	Write						
Instruction		j	k	Issue	complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4			Load2	Yes	45+R3		
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4								
DIVD	F10	F0	F6									
ADDD	F6	F8	F2									
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	V_j	V_k	Q_j	Q_k				
	0	Add1	Yes	SUBD	M(34+R2)			Load2				
	0	Add2	No									
		Add3	No									
	0	Mult1	Yes	MULTD		R(F4)	Load2					
	0	Mult2	No									
Register result status												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
4		FU	Mult1	Load2			M(34+R2)	Add1				

- Load2 completing; what is waiting for it?

Tomasulo Example Cycle 5

Instruction status					Execution	Write						
Instruction		j	k	Issue	complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4								
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2									
Reservation Stations					$S1$	$S2$	$RS\ for\ j$	$RS\ for\ k$				
	Time	Name	Busy	Op	V_j	V_k	Q_j	Q_k				
	2	Add1	Yes	SUBD	$M(34+R2)$	$M(45+R3)$						
	0	Add2	No									
		Add3	No									
	10	Mult1	Yes	MULTD	$M(45+R3)$	$R(F4)$						
	0	Mult2	Yes	DIVD		$M(34+R2)$	Mult1					
Register result status												
Clock				$F0$	$F2$	$F4$	$F6$	$F8$	$F10$	$F12$	...	$F30$
5			FU	Mult1	$M(45+R3)$		$M(34+R2)$	Add1	Mult2			

Suponemos: N° ciclos consumidos en la etapa
 execution: suma, 2; multiplicación, 10; división, 40.

Tomasulo Example Cycle 6

Instruction status					Execution	Write						
Instruction	j	k	Issue	complete	Result			Busy	Address			
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4								
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6								
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	V_j	V_k	Q_j	Q_k				
	1	Add1	Yes	SUBD	M(34+R2)	M(45+R3)						
	0	Add2	Yes	ADDD		M(45+R3)	Add1					
		Add3	No									
	9	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result status												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
6			FU	Mult1	M(45+R3)		Add2	Add1	Mult2			

- Issue ADDD here vs. scoreboard?

Tomasulo Example Cycle 7

Instruction status					Execution	Write						
Instruction		j	k	Issue	complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7							
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6								
Reservation Stations					S1	S2	RS for j	RS for k				
	Time	Name	Busy	Op	V_j	V_k	Q_j	Q_k				
	0	Add1	Yes	SUBD	M(34+R2)	M(45+R3)						
	0	Add2	Yes	ADDD		M(45+R3)	Add1					
		Add3	No									
	8	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result status												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
7		FU	Mult1	M(45+R3)		Add2	Add1	Mult2				

- Add1 completing; what is waiting for it?

Tomasulo Example Cycle 8

Instruction status						<i>Execution</i>	<i>Write</i>						
Instruction		<i>j</i>	<i>k</i>	<i>Issue</i>		<i>complete</i>	<i>Result</i>			Busy	Address		
LD	F6	34+	R2	1		3	4			Load1	No		
LD	F2	45+	R3	2		4	5			Load2	No		
MULT	F0	F2	F4	3						Load3	No		
SUBD	F8	F6	F2	4		7	8						
DIVD	F10	F0	F6	5									
ADDD	F6	F8	F2	6									
Reservation Stations						<i>S1</i>	<i>S2</i>	<i>RS for j</i>	<i>RS for k</i>				
	<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>					
	0	Add1	No										
	2	Add2	Yes	ADDD	M()-M()	M(45+R3)							
	0	Add3	No										
	7	Mult1	Yes	MULTD	M(45+R3)	R(F4)							
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1						
Register result status													
Clock				<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	<i>...</i>	<i>F30</i>	
8			FU	Mult1	M(45+R3)		Add2	M()-M()	Mult2				

Tomasulo Example Cycle 9

Instruction status					Execution	Write						
Instruction		<i>j</i>	<i>k</i>	Issue	complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6								
Reservation Stations					S1	S2	RS for <i>j</i>	RS for <i>k</i>				
	Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>				
	0	Add1	No									
	1	Add2	Yes	ADDD	M()–M()	M(45+R3)						
	0	Add3	No									
	6	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result status												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
9			FU	Mult1	M(45+R3)		Add2	M()–M()	Mult2			

Tomasulo Example Cycle 10

Instruction status					Execution	Write						
Instruction	<i>j</i>	<i>k</i>	Issue		complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10							
Reservation Stations					S1	S2	RS for <i>j</i>	RS for <i>k</i>				
	Time	Name	Busy	Op	<i>V_j</i>	<i>V_k</i>	<i>Q_j</i>	<i>Q_k</i>				
	0	Add1	No									
	0	Add2	Yes	ADDD	M()–M()	M(45+R3)						
	0	Add3	No									
	5	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result status												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
10			FU	Mult1	M(45+R3)		Add2	M()–M()	Mult2			

- Add2 completing; what is waiting for it?

Tomasulo Example Cycle 11

Instruction status						<i>Execution</i>	<i>Write</i>						
Instruction		<i>j</i>	<i>k</i>	<i>Issue</i>		<i>complete</i>	<i>Result</i>			Busy	Address		
LD	F6	34+	R2	1		3	4			Load1	No		
LD	F2	45+	R3	2		4	5			Load2	No		
MULT	F0	F2	F4	3						Load3	No		
SUBD	F8	F6	F2	4		7	8						
DIVD	F10	F0	F6	5									
ADDD	F6	F8	F2	6		10	11						
Reservation Stations						<i>S1</i>	<i>S2</i>	<i>RS for j</i>	<i>RS for k</i>				
	<i>Time</i>	<i>Name</i>	<i>Busy</i>	<i>Op</i>		<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>				
	0	Add1	No										
	0	Add2	No										
	0	Add3	No										
	4	Mult1	Yes	MULTD	M(45+R3)		R(F4)						
	0	Mult2	Yes	DIVD			M(34+R2)	Mult1					
Register result status													
Clock				<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	<i>...</i>	<i>F30</i>	
11			FU	Mult1	M(45+R3)		(M-M)+M()	M()-M()	Mult2				

- Write result of ADDD here vs. scoreboard?

Tomasulo Example Cycle 12

Instruction status					Execution	Write						
Instruction	<i>j</i>	<i>k</i>	Issue		complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					<i>S1</i>	<i>S2</i>	<i>RS for j</i>	<i>RS for k</i>				
	Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>				
	0	Add1	No									
	0	Add2	No									
	0	Add3	No									
	3	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result status												
Clock				<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
12			FU	Mult1	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

- Note: all quick instructions complete already

Tomasulo Example Cycle 13

Instruction status					Execution	Write						
Instruction	<i>j</i>	<i>k</i>	Issue		complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					<i>S1</i>	<i>S2</i>	<i>RS for j</i>	<i>RS for k</i>				
	Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	2	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result status												
Clock				<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
13			FU	Mult1	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

Tomasulo Example Cycle 14

Instruction status					Execution	Write						
Instruction	<i>j</i>	<i>k</i>	Issue		complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3				Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					<i>S1</i>	<i>S2</i>	<i>RS for j</i>	<i>RS for k</i>				
	Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>				
	0	Add1	No									
	0	Add2	No									
	0	Add3	No									
	1	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result status												
Clock				<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
14			FU	Mult1	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

Tomasulo Example Cycle 15

Instruction status					Execution	Write						
Instruction		<i>j</i>	<i>k</i>	Issue	complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3	15			Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					S1	S2	RS for <i>j</i>	RS for <i>k</i>				
	Time	Name	Busy	Op	V _j	V _k	Q _j	Q _k				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	Mult1	Yes	MULTD	M(45+R3)	R(F4)						
	0	Mult2	Yes	DIVD		M(34+R2)	Mult1					
Register result status												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
15		FU	Mult1	M(45+R3)			(M-M)+M()	M()-M()	Mult2			

- Mult1 completing; what is waiting for it?

Tomasulo Example Cycle 16

Instruction status					Execution	Write						
Instruction	<i>j</i>	<i>k</i>	Issue	complete	Result			Busy	Address			
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3	15	16		Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					<i>S1</i>	<i>S2</i>	<i>RS for j</i>	<i>RS for k</i>				
	Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	Mult1	No									
	40	Mult2	Yes	DIVD	M*F4	M(34+R2)						
Register result status												
Clock				<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
16			FU	M*F4	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

- Note: Just waiting for divide

Tomasulo Example Cycle 55

Instruction status					Execution	Write						
Instruction	<i>j</i>	<i>k</i>	Issue		complete	Result			Busy	Address		
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3	15	16		Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5								
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					<i>S1</i>	<i>S2</i>	<i>RS for j</i>	<i>RS for k</i>				
	Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	Mult1	No									
	1	Mult2	Yes	DIVD	M*F4	M(34+R2)						
Register result status												
Clock				<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
55			FU	M*F4	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

Tomasulo Example Cycle 56

Instruction status					Execution	Write						
Instruction	<i>j</i>	<i>k</i>	Issue	complete	Result			Busy	Address			
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3	15	16		Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5	56							
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					<i>S1</i>	<i>S2</i>	<i>RS for j</i>	<i>RS for k</i>				
	Time	Name	Busy	Op	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	Mult1	No									
	0	Mult2	Yes	DIVD	M*F4	M(34+R2)						
Register result status												
Clock				<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>F12</i>	...	<i>F30</i>
56			FU	M*F4	M(45+R3)		(M-M)+M()	M()-M()	Mult2			

- Mult 2 completing; what is waiting for it?

Tomasulo Example Cycle 57

Instruction status					Execution	Write						
Instruction	<i>j</i>	<i>k</i>	Issue	complete	Result			Busy	Address			
LD	F6	34+	R2	1	3	4		Load1	No			
LD	F2	45+	R3	2	4	5		Load2	No			
MULT	F0	F2	F4	3	15	16		Load3	No			
SUBD	F8	F6	F2	4	7	8						
DIVD	F10	F0	F6	5	56	57						
ADDD	F6	F8	F2	6	10	11						
Reservation Stations					S1	S2	RS for <i>j</i>	RS for <i>k</i>				
	Time	Name	Busy	Op	V _j	V _k	Q _j	Q _k				
	0	Add1	No									
	0	Add2	No									
		Add3	No									
	0	Mult1	No									
	0	Mult2	No									
Register result status												
Clock				F0	F2	F4	F6	F8	F10	F12	...	F30
57		FU		M*F4	M(45+R3)		(M-M)+M()	M()-M()	M*F4 / M			

- Again, in-order issue, out-of-order execution, completion

Compare to Scoreboard Cycle 62

Instruction status				Read	Execution	Write	
Instruction	<i>j</i>	<i>k</i>		Issue	operand complete	Result	
LD	F6	34+	R2	1	2	3	4
LD	F2	45+	R3	5	6	7	8
MULT	F0	F2	F4	6	9	19	20
SUBD	F8	F6	F2	7	9	11	12
DIVD	F10	F0	F6	8	21	61	62
ADDD	F6	F8	F2	13	14	16	22

Functional unit status		dest	S1	S2	FU for	FU for	<i>kFj?</i>	<i>Fk?</i>		
Time	Name	Busy	Op	<i>Fi</i>	<i>Fj</i>	<i>Fk</i>	<i>Qj</i>	<i>Qk</i>	<i>Rj</i>	<i>Rk</i>
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	0 Divide	No								

Register result status		F0	F2	F4	F6	F8	F10	F12	...	F30
Clock										
62	FU									

- Why takes longer on Scoreboard/6600?

TOMASULO vs. SCOREBOARD

- ❑ Principal diferencia: tratamiento de las **dependencias WAR y WAW**.
 - Con *scoreboard* las dependencias WAW y WAR producen paradas. Con el algoritmo de Tomasulo, no, gracias a su **redenominación de registros**.
 - Al enviar la instrucción a la estación de reserva **se cambian los identificadores de los registros por los de las estaciones de reserva** de las instrucciones que calcularán sus valores =>
 - **WAR(x,J,K)** (x: operando; J,K: instrucciones). Si al emitir J, x está disponible, se guardará en la RS de J, antes de que K lo pueda modificar. Si x no está disponible se redenomina a la RS de la instrucción I que lo producirá. No importará que K lo escriba antes, ya que **J no tomará el valor de x sino del CDB** cuando lo produzca I.
 - **WAW(D,J,K)**. Al emitir J se anota $\text{trr}['D'] \leftarrow r_J$ y al emitir K, $\text{trr}['D'] \leftarrow r_K$. Puesto que la emisión se hace en orden, **sólo la última instrucción actualizará realmente el destino D**.

TOMASULO: OTRAS CARACTERÍSTICAS

- ❑ Eficiente en la ejecución solapada de bucles.
 - Ignorando el tiempo de control del bucle se consigue una eficiencia **equivalente** a la obtenida mediante desenrollamiento y planificación **estáticos**.
 - Las estaciones de reserva permiten la ejecución solapada de varias copias del bucle **sin** necesidad de **tantos registros** como en el desenrollamiento y planificación estáticos.
- ❑ Inconvenientes:
 - **Complejidad** => gran cantidad de hardware.
 - Rendimiento limitado por la existencia de un **único CDB**.
 - **Varios CDB** => **más hardware** (en particular, el hardware asociativo de comparación de marcas en las RS crece con cada CDB adicional).