Superescalares fuera de orden

- Los riesgos WAR y WAW se eliminan con:

  Renombramiento de registros

- El orden del programa se recupera con:

  Buffer de reordenamiento

- Riesgos RAW y estructurales se resuelven con:

  Estaciones de reservación

Renombramiento

- RAW => dependencia en el algoritmo.
  - Dependencias reales.
- WAR y WAW ocurren por escacez de recursos.
  - Dependencias de nombre.
  - Se resuelven con renombramiento.
- Ejemplo

Secuencia de instrucciones con dependencias reales y de nombre

```
i1:  R1 ← R2/R3   # la división toma varios ciclos en finalizar
i2:  R4 ← R1+R5   # riesgo RAW con i1
i3:  R5 ← R6+R7   # riesgo WAR con i2
i4:  R1 ← R8+R9   # riesgo WAW con i1
```

En orden: i2 se atasca. También i3 e i4. Al terminal la / avanzan las 3

Fuera de orden: la dependencia en i2 no impide que i3 e i4 se despachen

Renombramiento

- Registro lógicos: son los provistos por la ISA.
- Registros físicos: lugares que hay para almacenar.
- Los lógicos se renombran como físicos.
- Los operandos se corresponden con reg físicos.
- Al destino se le asigna un nombre físico nuevo.
- Próxima instrucción: $R_i \leftarrow R_j$ Op R_k
- Tabla de correspondencia ***Rename*(R_a)**
- **Freelist**: lista de nombres libres.
- **First** es el primero libre.

Etapa de renombramiento

$Inst_m : R_i \leftarrow R_j Op R_k$ se convierte en $R_a \leftarrow R_b Op R_c$ donde
$Rb = \text{Renombrar}(R_j);$
$Rc = \text{Renombrar}(R_k);$
$R_a = \text{listalibre(primero)};$
$\text{Renombrar}(R_i) = \text{listalibre(primero)};$
$\text{primero} \leftarrow \text{próximo(primero)}.$

Renombrar	
R_i	R_a
R_j	R_b
R_k	R_c

Listalibre		
	R_a	
	R_b	
	R_c	
primero ->	R_d	

Renombrado en el ejemplo

- Freelist: R32, R33, R34, ...
- En i1 R2 y R3 se mantienen. R1 pasa a R32
- En i2 R5 se mantiene, R1 es R32. R4 es R33
- En i3 R6 y R7 se mantienen. R5 pasa a R34

```
i1:  R32 ← R2/R3     # la división toma varios ciclos en finalizar
i2:  R33 ← R32+R5    # Se mantiene aún el riesgo RAW con i1
i3:  R34 ← R6+R7     # ya no hay riesgo WAR con i2
i4:  R1 ← R8+R9      # Aun sin cambiar
```

i1 e i2 comienzan a ejecutar, se renombra i4: R1 pasa a R35

```
i4:  R35 ← R8+R9   # Ya no hay riesgo con i1
```


Buffer de reordenamiento (ROB)

- Probablemente i3 e i4 terminen antes que i1 e i2
- R34 y R35 no se deben copiar a R5 y R1 antes de que R32 y R33 se copien a R1 y R2.
- Sino i1 pisaría el resultado de i4.
  - Si en i1 hay una excepción, la i3 no debe escribir.
- El orden de escrituras se consigue con el ROB.
- ROB es una cola FIFO.
- Cada entrada es:
  a)Flag de terminación. b)Valor calculado.
  c)Destino del resultado. d)Tipo de inst.

Buffer de reordenamiento

Terminó la Ejecución?	Valor	Registro del resultado	Tipo de instrucción	
********	********	********	********	<- Cabeza
********	********	********	********	
				<- Cola

La instrucción renombrada se agrega al ROB

Etapa de renombramiento (pasos adicionales)
$ROB(cola) = (falso,\ NA,\ R_i,\ op);$
$cola \leftarrow próximo(cola)$

Cuando la instrucción finaliza, el valor se carga al ROB y la bandera se pone en verdadero. El commit (consignación) se realiza cuando la instrucción llega al tope de la FIFO.

Etapa de finalización
$si\ ((ROB(cabeza) = Inst_m \wedge bandera(ROB(cabeza)))$
$then\ begin\ R_i = valor;\ cabeza \leftarrow próximo(cabeza)\ end$
$else\ repetir\ el\ mismo\ control\ el\ próximo\ ciclo$

Instancias del buffer de reordenamiento (ROB)

	Bandera	Valor	Nom.Reg	Tipo	
(a) Después de renombrar las primeras 3 instrucciones					
$i1$	no_lista	ninguno	R1	Aritm	←Cabeza
$i2$	no_lista	ninguno	R4	Aritm	
$i3$	no_lista	ninguno	R5	Aritm	
					←Cola
(b) Después de la ejecución de $i3$ e $i4$					
$i1$	no_lista	ninguno	R1	Aritm	←Cabeza
$i2$	no_lista	ninguno	R4	Aritm	
$i3$	lista	alguno	R5	Aritm	
$i4$	lista	alguno	R1	Aritm	
					←Cola
(c) Después de finalizar $i1$					
$i1$	lista	alguno	R1	Aritm	
$i2$	no_lista	ninguno	R4	Aritm	←Cabeza
$i3$	lista	alguno	R5	Aritm	
$i4$	lista	alguno	R1	Aritm	
					←Cola

Estaciones de reservación y Ventana de instrucciones

- Tareas del front-end:
  - Renombrado. Bit ready del registro destino.
  - Inserción de instrucciones en el ROB.
- Funciones del Commit:
  - El resultado se copia al registro lógico.
  - Se mantiene el orden del programa.

- ¿Dónde esperan las instrucciones antes de ejecutar?
- ¿Cómo se sabe que está lista para ejecutar?

Una Estación de Reservación contiene

- La operación que debe ser realizada.
- El valor de los operandos fuentes o sus nombres físicos (una bandera indica si hay un valor o un nombre).
- El nombre del registro físico donde queda el resultado.
- La entrada en el ROB donde se resguarda el resultado.

Estaciones de reservación y Ventana de Instrucciones

Etapa de Despacho

Para la $Inst_m$: $R_a \leftarrow R_b \, op \, R_c$ *(los registro son físicos)*
Si todas las estaciones de reservación para op están ocupadas
then generar atasco y repetir en el próximo ciclo
Llenar (próxima) estación de reservación para op con la tupla
{ *operador* = *op*,
Si Listo(R_b) *then (valor*[R_b]*,verdadero) else* (R_b, *falso*),
Si Listo(R_c) *then (valor*[R_c]*,verdadero) else* (R_c, *falso*),
R_a,
puntero a una entrada en ROB
}

Front-end

Etapa de Emisión

Si (bandera(R_b) $\wedge$ *bandera*(R_c))
then enviar (valor[R_b]*,valor*[R_c]) *a la unidad funcional*
else repetir el siguiente ciclo

Back-end

Ejecución

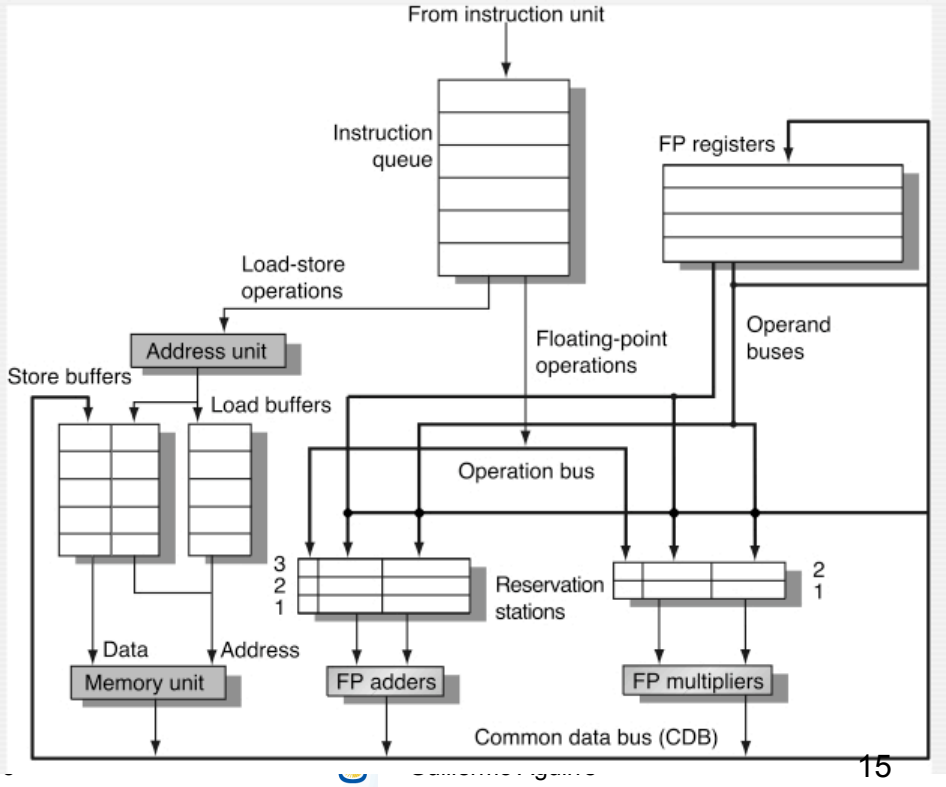
- Cuando se termina de calcular el resultado, se difunde el valor y el nombre del registro físico al que va destinado.
- Cada estación con un operando igual al registro destino, copiará el valor y pondrá la bandera en true.
- También se copia en el registro destino y su bit ready se pone en on.
- Almacena el resultado en el ROB, si actúa como registro físico.

Ejemplo

- Los registros lógicos han sido consignados (committed). Los físicos están libres.
- R1 se renombró como R32. Bit ready en off.
- Al despachar i2, se envía R32 y flag en false.
- i2 no está lista para ser emitida (issue).
- i1 difunde el resultado con el nombre R32.
- La estación de i2 graba el resultado y pone el flag en true.
- En el ciclo siguiente se emite i2.

	Etapa	Recurso leído	Recursos escritos o utilizados
Extremo inicial	Fetch	PC Predictor de salto Cache de Instrucciones	PC Buffer de instrucciones
	Decodific- Renombrado	buffer de instrucciones Mapa de registros	Buffer de decodificación Mapa de registros ROB
	Despacho	Buffer de decodificación Mapa de registros Banco registros (lóg y fís)	Estación de reservación ROB
Extremo Final	Emisión	Estaciones de reservación	Unidades Funcionales Cache de datos
	Ejecución	Unidades funcionales	Estaciones de reservación ROB Registros Físicos Predictor de saltos Buffer de almacenamiento
	Commit	ROB Registros físicos Buffer de almacenamiento	ROB Registros lógicos Mapa de registros Cache de datos

MIPS usando Tomasulo



Etapas en Tomasulo (1)

✓**<u>Etapa de issue</u>:** Obtiene la instrucción. Si hay ER despacha la instrucción. Los oprandos son: el valor del registro o bien el tag de la ER que lo produce. Así, se renombran registros, evitando WAR Y WAW. Si no hay ER se produce stall

Etapas en Tomasulo (2)

✓**<u>Etapa de ejecución</u>:** Se observa el CDB esperando los operandos. Cuando se cargan los valores de los operandos, la unidad ejecuta.

Esta demora evita RAW.

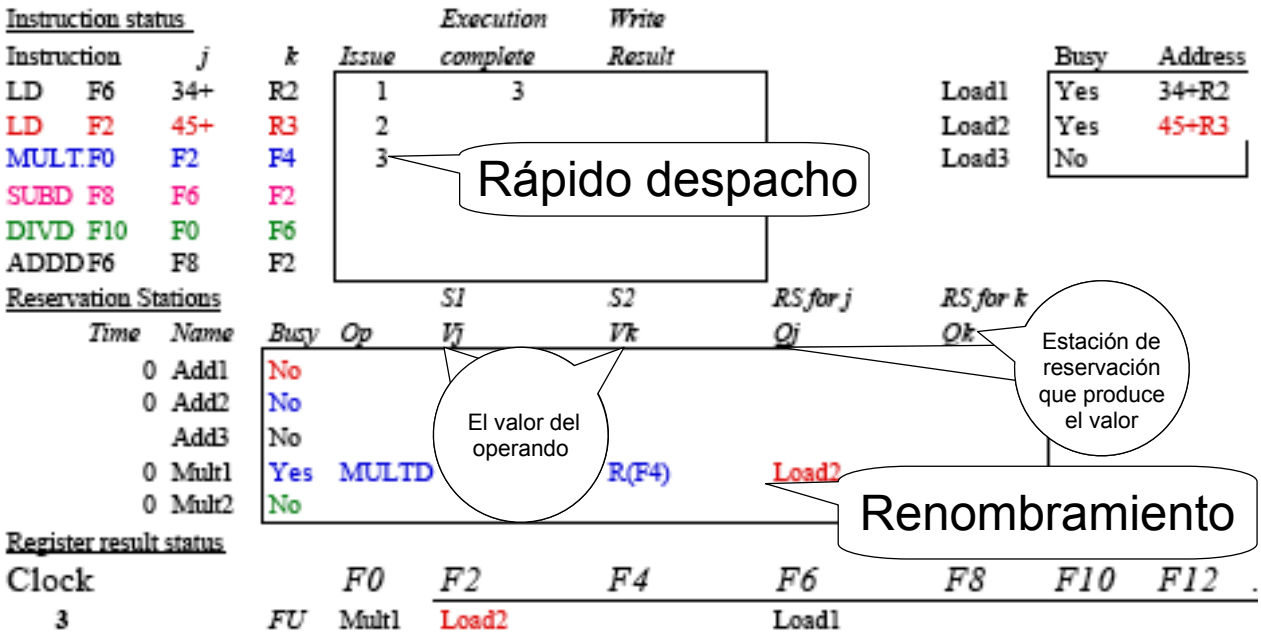
Etapas en Tomasulo (3)

✓**<u>Etapa de escritura de resultados</u> :** Cuando el resultado está disponible se copia en el CDB. Los buffers de salida aquí también escriben hacia la memoria.

Características del esquema de Tomasulo

- Las estaciones de reservación.
- Detección de riesgos y control de ejecución distribuidos.
- Los resultados son pasados directamente a las unidades que los usan.
- Bus común de resultado. CDB en 360/91

Ejemplo de Tomasulo - Ciclo 3



Load 1 ha finalizado. ¿Alguien espera ese valor?

Ejemplo de Tomasulo - Ciclo 9

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	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6						
Reservation Stations					S1	S2	RS for j	RS for k		
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
	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
9			FU	Mult1	M(45+R3)		Add2	M()–M()	Mult2	

Ejemplo de Tomasulo - Ciclo 12

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	8				
DIVD	F10	F0	F6	5						
ADDD	F6	F8	F2	6	10	11				
Reservation Stations					S1	S2	RS for j	RS for k		
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
	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				F0	F2	F4	F6	F8	F10	F12
12			FU	Mult1	M(45+R3)		(M-M)+M()	M()–M()	Mult2	

Todas las instrucciones cortas ya han terminado

Ejemplo de Tomasulo - Ciclo 16

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	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					S1	S2	RS for j	RS for k		
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
	0	Add1	No							
	0	Add2	No							
		Add3	No							
	0	Mult1	No							
	40	Mult2	Yes	DIVD	M*F4	M(34+R2)				
Register result status										
Clock				F0	F2	F4	F6	F8	F10	F12
16			FU	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2	

Ejemplo de Tomasulo - Ciclo 55

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	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					S1	S2	RS for j	RS for k		
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
	0	Add1	No							
	0	Add2	No							
		Add3	No							
	0	Mult1	No							
	1	Mult2	Yes	DIVD	M*F4	M(34+R2)				
Register result status										
Clock				F0	F2	F4	F6	F8	F10	F12
55			FU	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2	

Ejemplo de Tomasulo - Ciclo 56

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	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					S1	S2	RS for j	RS for k		
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
	0	Add1	No							
	0	Add2	No							
		Add3	No							
	0	Mult1	No							
	0	Mult2	Yes	DIVD	M*F4	M(34+R2)				
Register result status										
Clock				F0	F2	F4	F6	F8	F10	F12
56			FU	M*F4	M(45+R3)		(M–M)+M()	M()–M()	Mult2	

Ejemplo de Tomasulo - Ciclo 57

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	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 j	RS for k		
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk		
	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
57			FU	M*F4	M(45+R3)		(M–M)+M()	M()–M()	M*F4/M	

Tomasulo Vs ScoreBoard

Ventana	14	5
Hazard Estructural	Stall	stall
WAR	Evita renombrando	Stall
WAW	Evita renombrando	Stall
Resultados	Difunde	En registros
Control	Distribuido	Score