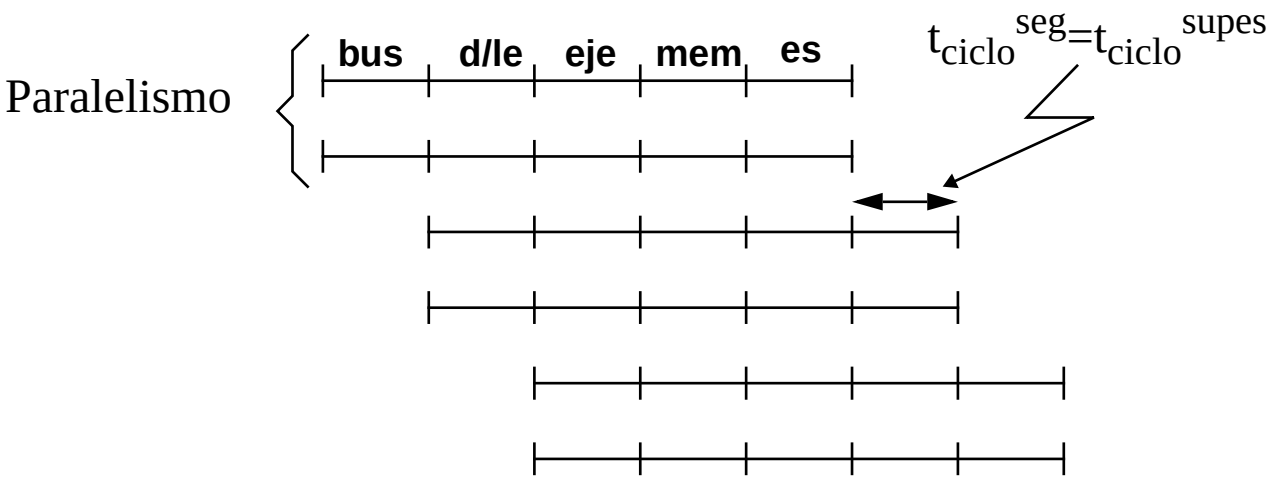
Introducción

PROCESADOR SUPERESCALAR



✓ Grado m: en cada ciclo se buscan/decodifican m instrucciones

Introducción

CICLOS POR INSTRUCCION

- ✓ tciclo identico al del procesador escalar basico
- ✓ Se ejecutan m instrucciones por ciclo (ideal)

$$CPI_{super} = CPI_{seg} / m$$

$$T_{super} = N \times CPI_{super} \times t_{ciclo} = T_{seg} / m$$

- ✓ Se espera un factor m de mejora del rendimiento

Introducción

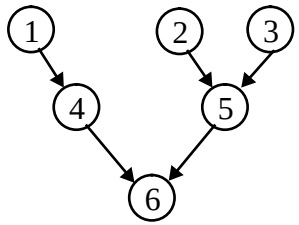
PROBLEMAS

- ✓ Dependencias de datos y de control
  - ✧ *Igual que en segmentados: RD, DR, RR y saltos.*
  - ✧ *Mayor incidencia en pérdida del rendimiento.*
    - ∗ Load retardado: Segmentado: un hueco de retardo (cortocircuito) -> insertar una instruccióon. Superescalar: insertar m-1+m instrucciones, (k-1)m si no hay cortocircuitos (k=nº etapas)
    - ∗ Operaciones multiciclo: ciclos-0 (el procesador no ejecuta ninguna instrucción nueva) y ciclos-1 (una)
    - ∗ Dependencias de control: detención afecta a mayor número de instrucciones. Predicción: mayor incidencia en rendimiento
- ✓ Riesgos estructurales: replicar U.Fs
  - ✧ *Control Complejo: m pipelines operando concurrentemente*
  - ✧ *Tecnología alta capacidad de integración*

Introducción

PROBLEMAS

- Número de instrucciones independientes:



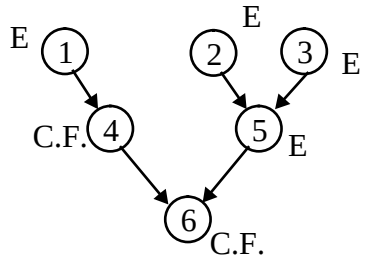
- Máquina sin limitación de recursos:

° **3 ciclos** { 1, 2, 3,
4, 5
6 }

- Máquina de grado 2

4 ciclos { 1, 2
3, 4
5
6 }

- Tipos de instrucciones:



- Máquina sin restricción de tipos de instrucciones:

° **3 ciclos** { 1, 2, 3,
4, 5
6 }

- Máquina de grado 2: 1 instrucción entera, 1 ins. C.F.

5 ciclos { 1
2, 4
3
5
6 }

Paralelismo a nivel de instrucción

- ✓ ipc: media del número de instrucciones de un programa que pueden ejecutarse por ciclo suponiendo recursos hardware ilimitados (= 1/CPI)
- ✓ Factores que determinan ipc
  - ✧ *% dependencias verdaderas*
  - ✧ *Latencias operaciones PF*
  - ✧ *% saltos*

DETERMINACIÓN DEL IPC

✓ Latencias de instrucciones y programas usados para bechmark.

Functional Unit	Issue Latency (cycles)		Result Latency (cycles)	
	single	double	single	double
Integer ALU (2)	1	n/a	1	n/a
Barrel Shifter	1	n/a	1	n/a
Branch Unit	1	n/a	1	n/a
Load Unit	1	2	2	3
Store Unit	1	2	n/a	n/a
Memory (cache reload)	13	n/a	12	n/a
Float Add	1	2	2	3
Float Multiply	1	2	4	6
Float Divide	12	27	12	27
Float Convert	1	2	2	4

Program	Description
5diff	text file comparison
awk	pattern scanning and processing
ccom	optimizing C compiler
compress	file compression using Lempel–Ziv encoding
doduc	Monte-Carlo simulation, double-precision floating-point
espresso	logic minimization
gnuchess	computer chess program
grep	reports occurrences of a string in one or more text files
irsim	delay simulator for VLSI layouts
latex	document preparation system
linpack	linear-equation solver, double-precision floating-point
nroff	text formatter for a typewriter-like device
simple	hydrodynamics code
spice2g6	circuit simulator
troff	text formatter for typesetting device
wolf	standard-cell placement using simulated annealing
whetstone	standard floating-point benchmark, double-precision floating-point
yacc	compiles a context-free grammar into LR(1) parser tables

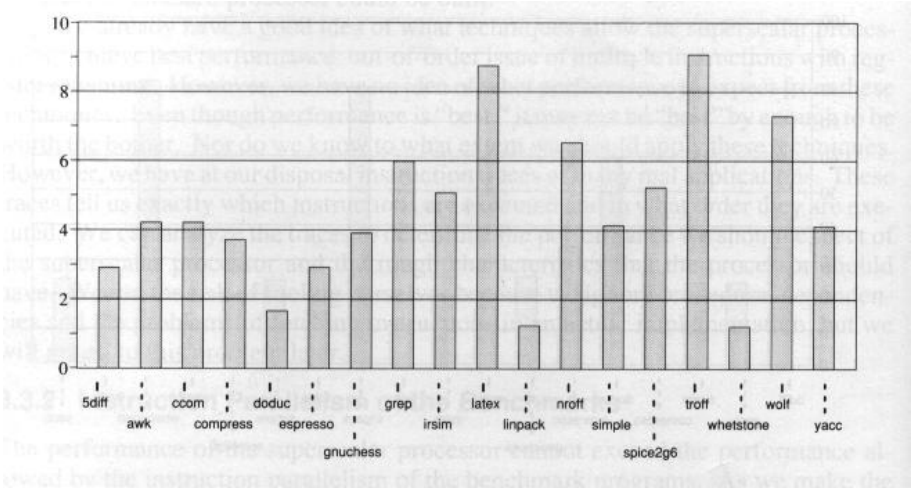
Paralelismo a nivel de instrucción

DETERMINACIÓN DEL IPC

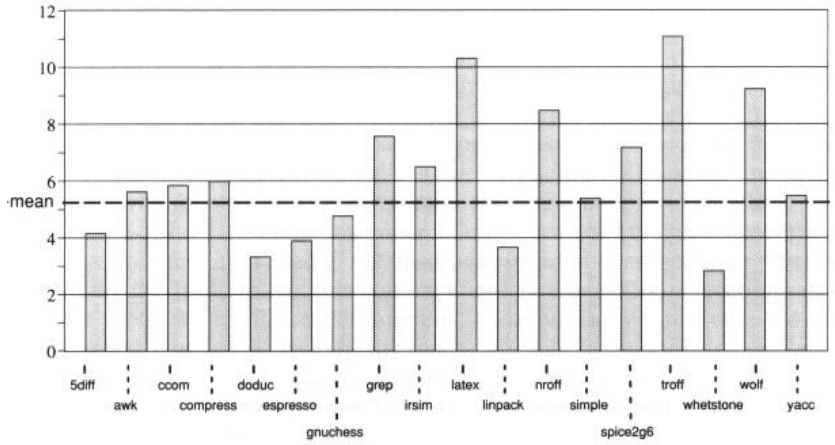
✓ Paralelismo limitado sólo por dependencias verdaderas

✧ *No hay otras dependencias de datos (renombre registros)*

✓ No hay instrucciones de salto (trazas con resultado de los saltos conocido)



Media armónica del ipc 3’3



A

Acelaración media con respecto a escalar: 5.4
Límite de rendimiento óptimo

Paralelismo a nivel de máquina

- ✓ ipcm: media del número de instrucciones que puede ejecutar por ciclo en un procesador concreto.
- ✓ Factores que determinan el paralelismo de máquina
  - ✧ *Número de instrucciones buscadas/decodificadas por ciclo*
  - ✧ *Número de UFs*
  - ✧ *Mecanismos para encontrar instrucciones independientes.*
    - ∗ Emisión-finalización fuera de orden
    - ∗ Renombre dinámico de registros (buffer de reordenacion o tabla de correspondencia)
    - ∗ Ventana de instrucciones (buffer entre decodificadores y UFs para permitir emisión fuera de orden)

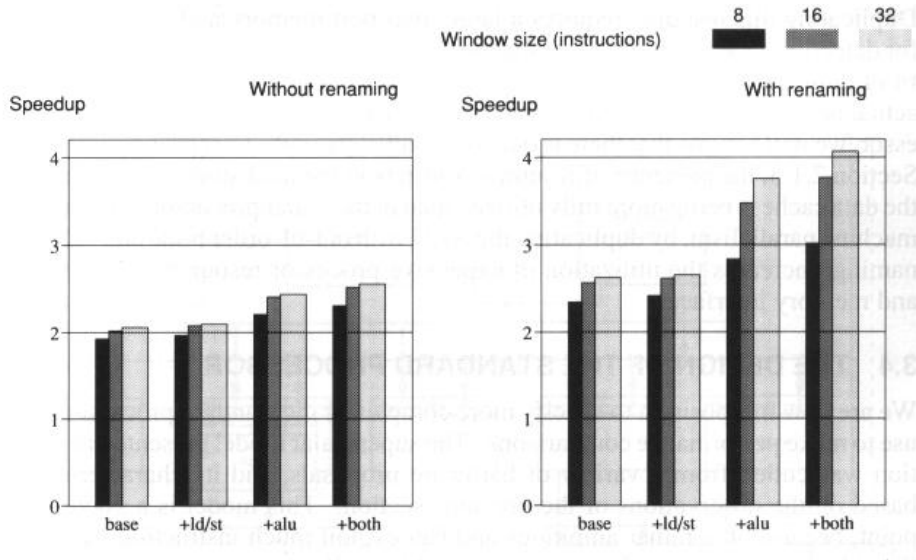
Paralelismo a nivel de máquina

DETERMINACIÓN DEL IPCM

✓ Procesador base : unidades func. del escalar y emisión fuera de orden.

✓ Benchmark: Muy pocas instrucciones PF

✓ No hay fallos en cache



Duplicar ALU:mejora el rendimiento(más si ventanas de instrucciones grandes).
Duplicar Ld/St: más costoso y menos justificable

Políticas de emisión de instrucciones

- ✓ Emisión: iniciar la etapa de ejecución de una instrucción
- ✓ Características del procesador base
  - ✧ *Busca y decodifica dos instrucciones por ciclo*
  - ✧ *Tres UFs de ejecución*
  - ✧ *Escribe dos resultados por ciclo*
  - ✧ *Secuencia de instrucciones:*
    - ∗ I1 dos ciclos.
    - ∗ (I3,I4) se ejecutan en la misma UF.
    - ∗ I6 depende de I5.

Emisión-Finalización en orden

- Se detiene la emisión de instrucciones cuando:
  - *Riesgo estructural*
  - *Instrucción con más de un ciclo de ejecución*
  - *Riesgo RD*

- Características
  - *Rendimiento pobre*
  - *Control simple:*
    - Comprobar riesgos estructurales entre decodificadas.
    - RD con las que están en ejecución.
    - Uso de cortocircuitos.

Emisión-Finalización en orden

ciclo	Búsqueda		Decodifica		Ejecución			Escritura	
1	1	2							
2	3	4	1	2					
3				2	1				
4	5	6	3	4	1	2			
5				4			3	1	2
6	7	8	5	6			4	3	
7				6		5		4	
8	9	10	7	8	6			5	
9								6	

Emisión en orden-finalización fuera de orden

- ✓ Se detiene la emisión de instr ucciones cuando
  - ✧ *Riesgo estructural*
  - ✧ *Riesgo RD*
  - ✧ *Riesgo RR*

- ✓ Características
  - ✧ *Mejor rendimiento: ejecución simultánea de instrucciones multiciclo*
  - ✧ *Control más complejo:escritura de resultados en orden correcto implica conocer estado de todas las instrucciones en ejecución*
  - ✧ *Dificultad para manejar interrupciones precisas.*

Emisión en orden-finalización fuera de orden

ciclo	Búsqueda		Decodifica		Ejecución			Escritura	
1	1	2							
2	3	4	1	2					
3	5	6	3	4	1	2			
4				4	1		3		2
5			5	6			4	1	3
6				6		5			4
7					6			5	
8								6	

Emisión fuera de orden-finalización fuera de orden

- ✓ Ventana de instrucciones: Buffer con instrucciones decodificadas. Examina riesgos y determina grupos de instrucciones a ejecutar
- ✓ Restricciones: riesgos y comportamiento lógico correcto
- ✓ Se detiene la emisión de instrucciones cuando
  - ✧ *Riesgo estructural*
  - ✧ *Riesgos RD,RR y DR*
- ✓ La decodificación no se detiene a menos que se llene la ventana de instrucciones.
- ✓ Características
  - ✧ *Rendimiento óptimo*
  - ✧ *Control complejo: pueden ocurrir todos los tipos de riesgos*

Emisión fuera de orden-finalización fuera de orden

ciclo	Búsqueda		Decodif		Ventana	Ejecución			Escritura	
1	1	2								
2	3	4	1	2	1, 2					
3	5	6	3	4	3, 4	1	2			
4			5	6	4, 5, 6	1		3	2	
5					5	6		4	1	3
6							5		4	6
7									5	

Emisión en orden- Finalización en orden

```
1 nivel:   Ld f3,[r12]
2          Ld f1,[r10]
3          Ld f2,[r11]
4          f4=f1*f3
5          f2=f2*f3
6          f1=1-f3
7          f4=f4+f1
8          St f2,[r10]
9          St f4,[r11]
10         r10=r10+1
11         r11=r11+1
12         r12=r12+1
13         r0=r0-1
14         bnz nivel
```

Ciclo	Bus.	Dec.	E	S1	S2	M_1	M_2	l/S	E1	E2
1	1, 2									
2	3, 4	1, 2								
3		2						1		
4	5, 6	3, 4						2		1
5	7, 8	5, 6				4		3	2	
6		6				4	5		3	
7		6				4	5			
8	9, 10	7, 8			6	4	5			
9		7, 8			6		5			4
10		8		7					6	5
11		8		7						
12	11, 12	9, 10						8	7	
13	13, 14	11, 12	10					9		
14		11	11							10
15		13, 14	12						11	
15			13							12

Emisión en orden - Finalización en desorden

```
1 nivel:   Ld f3,[r12]
2          Ld f1,[r10]
3          Ld f2,[r11]
4          f4=f1*f3
5          f2=f2*f3
6          f1=1-f3
7          f4=f4+f1
8          St f2,[r10]
9          St f4,[r11]
10         r10=r10+1
11         r11=r11+1
12         r12=r12+1
13         r0=r0-1
14         bnz nivel
```

Ciclo	Bus.	Dec.	E	S1	S2	M_1	M_2	l/S	E1	E2
1	1, 2									
2	3, 4	1, 2								
3		2						1		
4	5, 6	3, 4						2		1
5	7, 8	5, 6				4		3	2	
6	9, 10	7, 8			6	4	5		3	
7		7, 8			6	4	5			
8		7, 8				4	5		6	
9		8		7			5			4
10	11, 12	9, 10		7				8		5
11	13, 14	11, 12	10					9	7	
12		12	11							10
13		13, 14	12						11	
14		14	13						12	
15										13

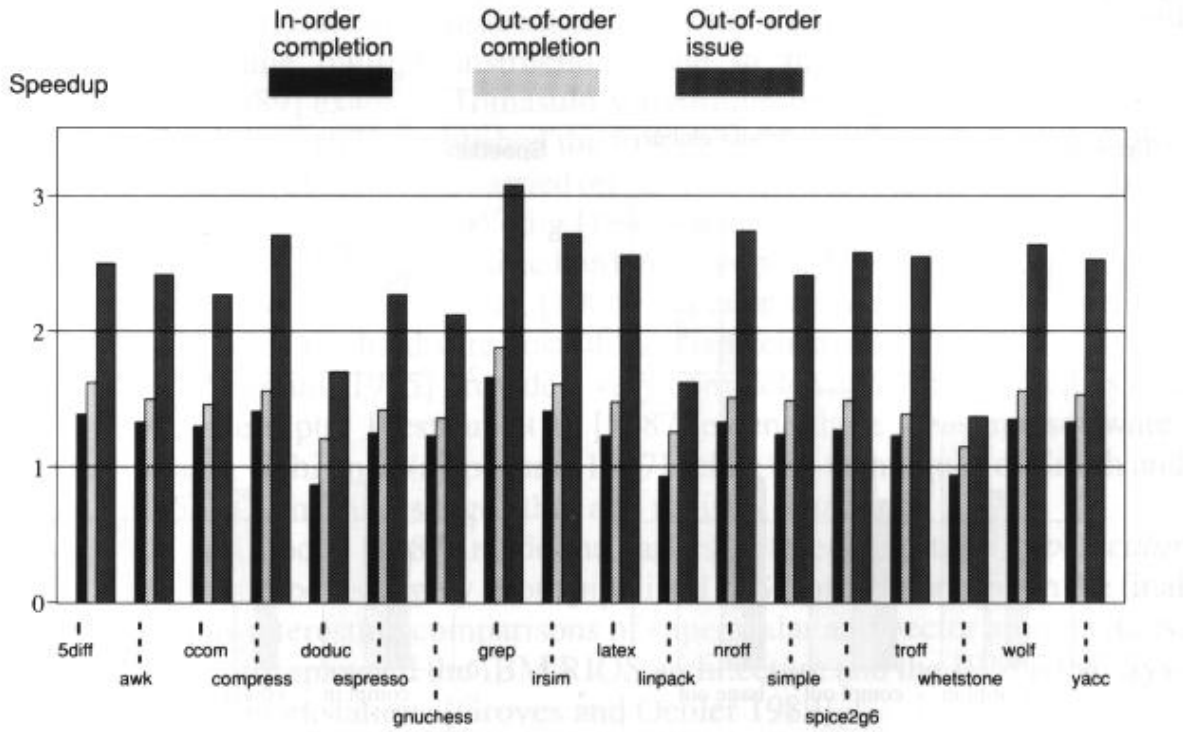
Emisión en desorden - Finalización en desorden

```
1 nivel:   Ld f3,[r12]
2          Ld f1,[r10]
3          Ld f2,[r11]
4          f4=f1*f3
5          f2=f2*f3
6          f1=1-f3
7          f4=f4+f1
8          St f2,[r10]
9          St f4,[r11]
10         r10=r10+1
11         r11=r11+1
12         r12=r12+1
13         r0=r0-1
14         bnz nivel
```

Ciclo	Bus.	Dec.	Ventana	E	S1	S2	M_1	M_2	l/S	E1	E2
1	1, 2										
2	3, 4	1, 2	1, 2								
3	5, 6	3, 4	2, 3, 4						1		
4	7, 8	5, 6	3, 4, 5, 6						2		1
5	9, 10	7, 8	5, 7, 8			6	4		3	2	
6	11, 12	9, 10	7, 8, 9, 10			6	4	5		3	
7	13, 14	11, 12	7,8,9,10,11,12				4	5			6
8			7,8,9,10,11,13,14	12			4	5			
9			8,9,10,11,14	13	7			5		12	4
10			9,11,14	10	7				8	13	5
11			14	11					9	10	7
12				14						11	

Emisión de instrucciones

ACELERACIÓN PARA DISTINTAS POLÍTICAS DE EMISIÓN



✧ *Comportamiento ideal de la unidad de fecth (búsqueda de instrucciones)..*

Emisión Fuera de orden

- ✓ Técnica de emisión de instr ucciones más adecuada para conseguir IPC alto.
- ✓ Problemas.
  - ✧ *Control complejo.*
  - ✧ *Aparecen todo los posibles riesgos de datos.*
    - ∗ Dependencias verdaderas.
    - ∗ Antidependencias.
    - ∗ Dependencias de salida.
- ✓ Nuevos elementos hardware:
- ✓ Emisión fuera de or den: Ventana de instrucciones.
- ✓ Evitar antidependencias y dependencias de salida: Buffer de reordenación.
  - ✧ *Realiza renombre dinámico de registros.*

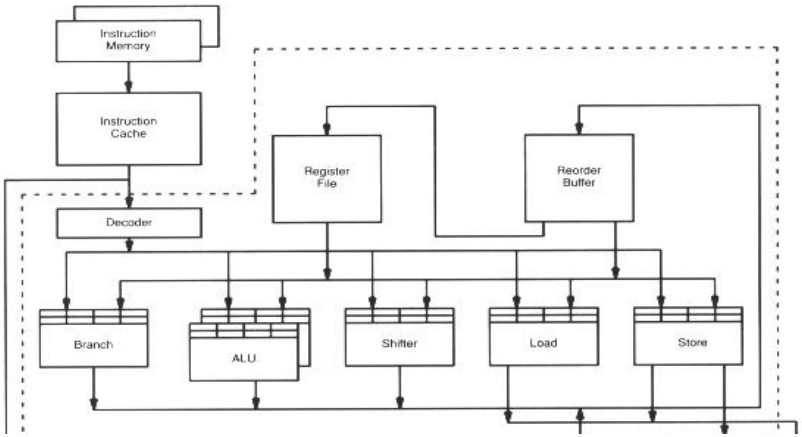
Ventana de instrucciones

- ✓ Buffer entre decodificador es y UFs de ejecución (etapa de D/L)
- ✓ Objetivo:permite continuar la decodificación de instr ucciones aunque otras anteriores no se emitan debido a riesgos
- ✓ Tarea:
  - ✧ *Lectura de operandos*
  - ✧ *emisión de instrucciones*
- ✓ Cada registro tiene los campos:
  - ✧ *CO: Código de operación*
  - ✧ *tag: de la entrada del buffer de reordenación*
  - ✧ *op1/tag: del operando fuente 1*
  - ✧ *op2/tag: del operando fuente 2*

Implementación de Ventanas de Instrucciones

VENTANA DISTRIBUIDA

✓ Distribuye la ventana de instrucciones entre las unidades funcionales (estaciones de reserva)



✓ Estaciones de Reserva (ER)

✧ *Mantiene inst decodificadas hasta que sus operandos sean válidos (instrucción preparada o ready)*

✧ *Emisión de instrucciones*

∗ Seleccionar una instrucción entre las preparadas

∗ Desactivar el registro ocupado por instrucción emitida

Implementación de Ventanas de Instrucciones

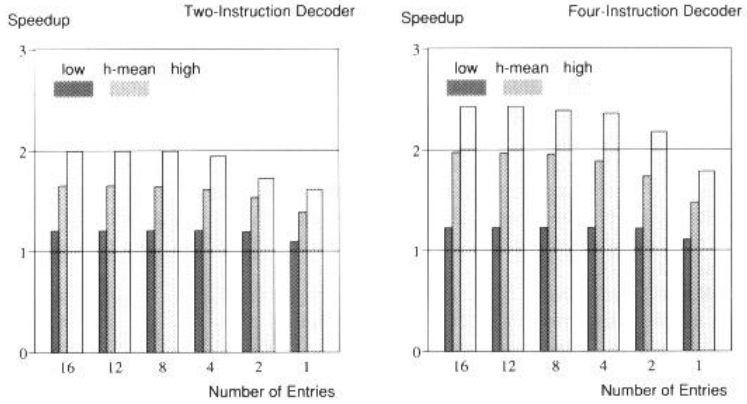
VENTANA DISTRIBUIDA

✓ Relación del tamaño de las Estaciones de Reserva con el r endimiento

✧ *ER llena (riesgos RD o UF ocupada). Opciones:*

∗ Detención de los decodificadores o aumentar el tamaño de las ER.

✓ Rendimiento de benchmarks



Tamaño de las estaciones de reserva: 1 a 16 registros.

✧ *Rendimiento se reduce si el tamaño de las ER es menor que 4.*

✧ *Las UFs de Load y Store deben tener tamaño 8 o mayor.*

Implementación de Ventanas de Instrucciones

VENTANA DISTRIBUIDA

✓ Ventajas

- ✧ *Control sencillo (aunque se repite en cada unidad funcional).*
- ✧ *Cada estación de reserva se adecua al tipo de instrucción.*
- ✧ *Facilita la emisión del máximo de instrucciones por ciclo.*

✓ Inconvenientes

- ✧ *Complejidad del tráfico de buses entre decodificadores y estaciones de reserva.*
- ✧ *Trozos de código concentrado en una unidad funcional detienen la decodificación.*

Implementación de Ventanas de Instrucciones

VENTANA CENTRALIZADA

- ✓ Ventaja
  - ✧ *Menor tamaño*
  - ✧ *Más eficiente*
- ✓ Inconvenientes
  - ✧ *Emisión de instrucciones control complejo*
  - ✧ *Seleccionar entre muchas instrucciones preparadas*
  - ✧ *Tener en cuenta UFs disponibles*
  - ✧ *Liberar varios registros de la Ventana en un ciclo*
  - ✧ *Entradas a la Ventana para instrucciones de cualquier tipo*
- ✓ Implementaciones
  - ✧ *Dispatch Stack*
  - ✧ *Register Update Unit (RUU*

Implementación de Ventanas de Instrucciones

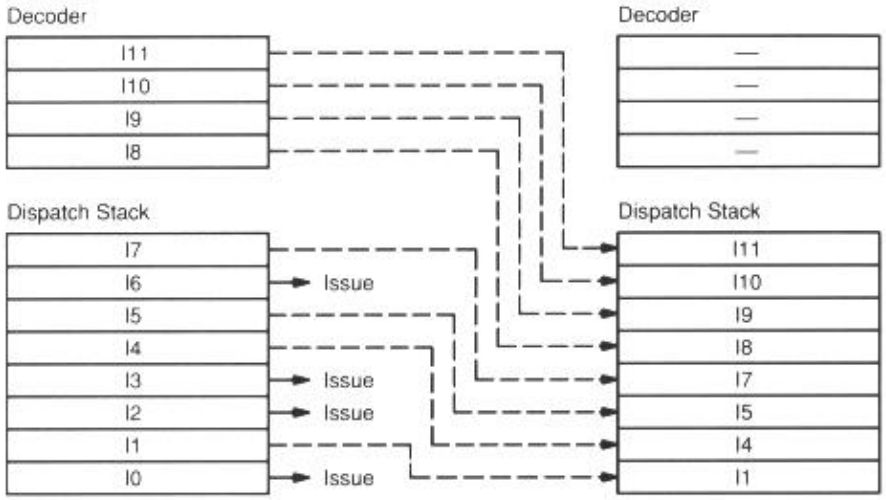
PILA DE DISTRIBUCIÓN (DISPATCH STACK)

✓ Funciones:

✧ *En cada ciclo emite instruccciones preparadas si la UF está libre.*

✧ *Si hay varias para la misma UF se elige la más antigua (está más cerca del fondo de la pila)*

✧ *Se comprime la pila para ocupar posiciones de las emitidas*

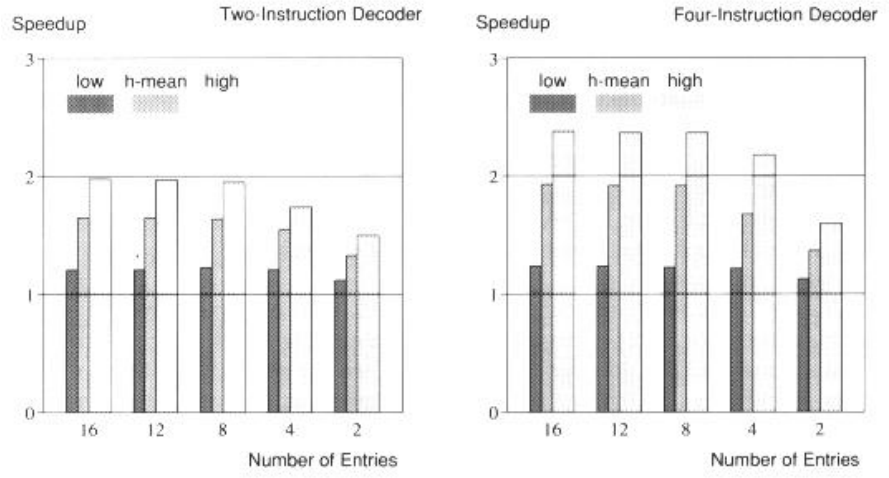


Implementación de Ventanas de Instrucciones

PILA DE DISTRIBUCIÓN (DISPATCH STACK)

✓ Rendimiento-tamaño de la Pila de distribución

✧ *8 entradas da el rendimiento óptimo (25% ER)*



✓ Comparación con estaciones de r eserva

✧ *Lógica de emisión es 1/10 de ER.*

✧ *Almacenamiento y lógica de cortocircuitos es 1/4 de ER.*

✧ *Compresión y lógica de ubicación equivale a un RISC.(150.000 transistores).*

Implementación de Ventanas de Instrucciones

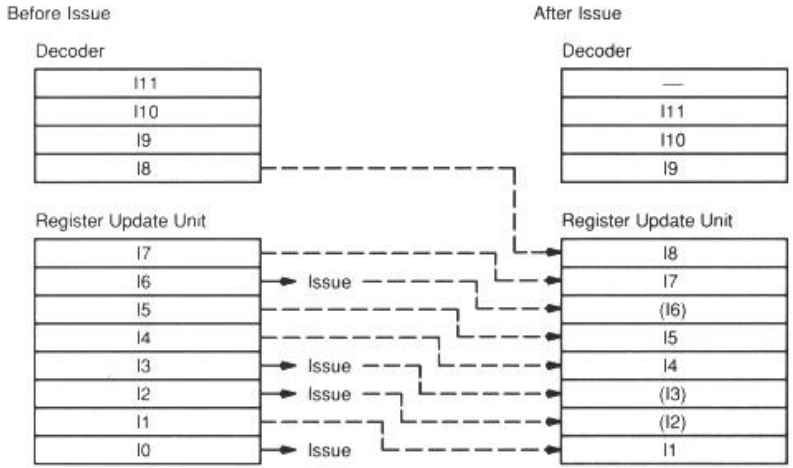
UNIDAD DE ACTUALIZACIÓN DE REGISTRO (REGISTER UPDATE UNIT)

✓ Características

✧ *Más simple que Dispatch Stack. Evita compresión.*

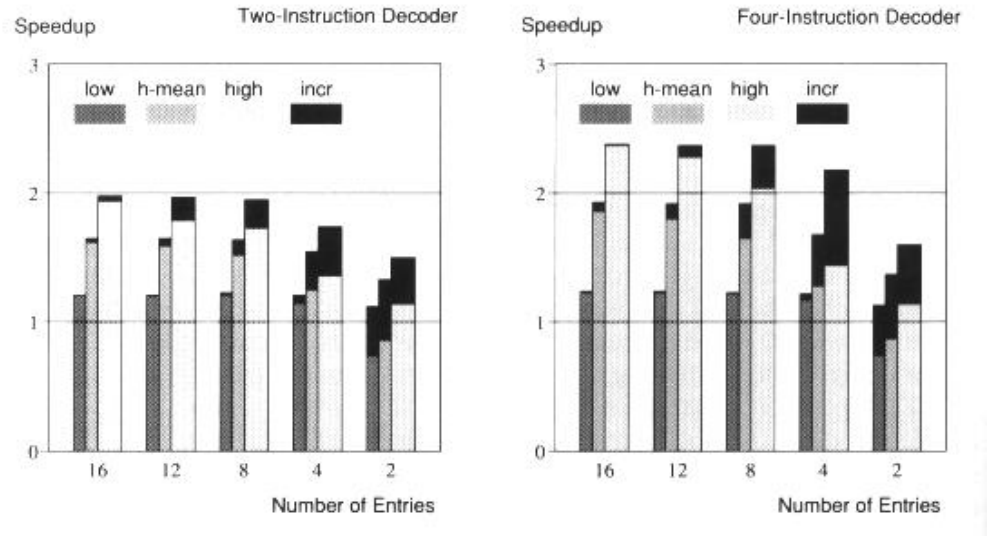
✧ *Entradas se asignan y liberan sólo en orden FIFO.*

∗ Ventana cumple con las funciones del BR.



Implementación de Ventanas de Instrucciones

COMPARACION DISPATCH STACK - RUU



Renombre dinámico de registros

EVITAR RIESGOS RR Y DR (CONFLICTOS DE ESCRITURA)

✓ Idea: Utilizar un nuevo registro para el resultado de cada instrucción

	Original	**Renombre**
(1)	r3 = r3 op r5	$r3_b = r3_a$ op $r5_a$
(2)	r4 = r3 + 1	$r4_a = r3_b + 1$
(3)	r3 = r5 + 1	$r3_c = r5_a + 1$
(4)	r7 = r3 op r4	$r7_a = r3_c$ op $r4_a$

✧ *Riesgo RR: instrucciones (1) y (3);*

✧ *Riesgo DR: instrucciones (2) y (3). La instrucción (3) no puede emitirse hasta que (2) pase etapa D/L*

✓ Renombre: cualquier instrucción con operando r3 que sea posterior a (3) lee r3c hasta nueva asignación. Cada nueva asignación r eemplaza (supersede) la anterior.

✧ *Se puede emitir (3), ahora independiente de (1) y (2)*

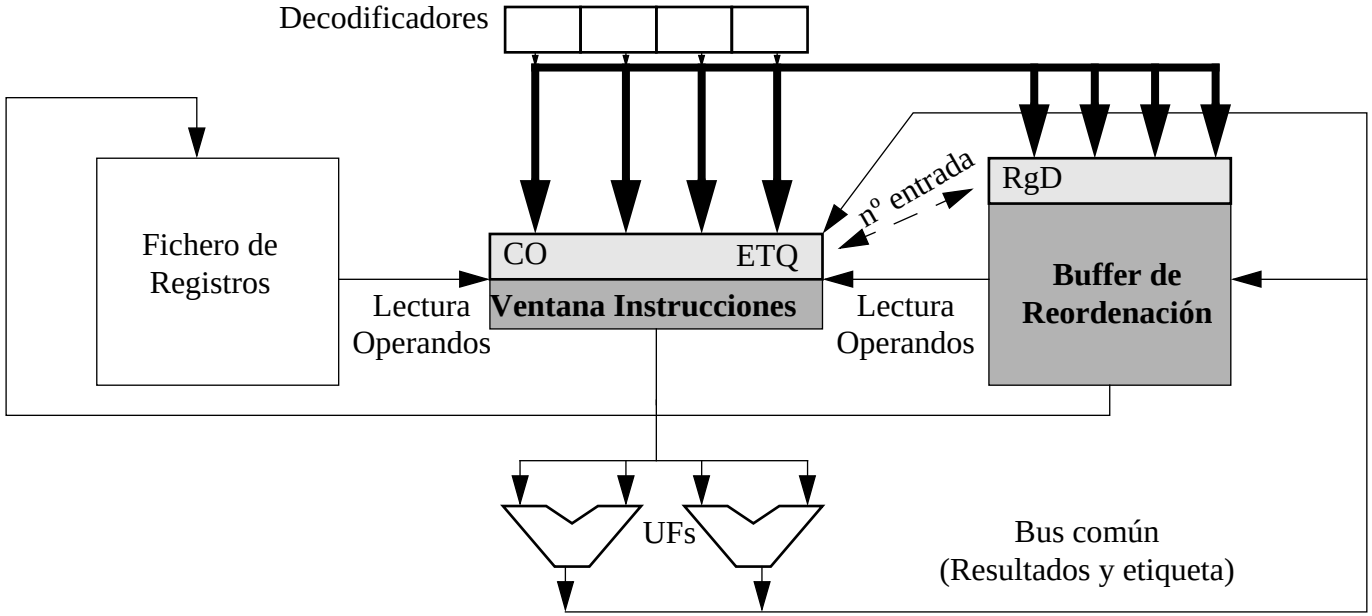
Buffer de reordenación

- ✓ Ejecución especulativa de instrucciones.
- ✓ Memoria FIFO asociativa.
- ✓ Campos de cada entrada:
  - ✧ *reg. destino.*
  - ✧ *resultado.*
  - ✧ *tag (número de la entrada del buffer de reordenación). Se asocia a la entrada correspondiente en la ventana de instrucciones.*

Emisión fuera de orden

VENTANA DE INSTRUCCIONES Y BUFFER DE REORDENACIÓN

- ✓ Etapa de Decodificación/Lectura de Operandos.
  - ✧ *Decodificción: asigna entradas en Ventana y Buffer de Reordenación.*



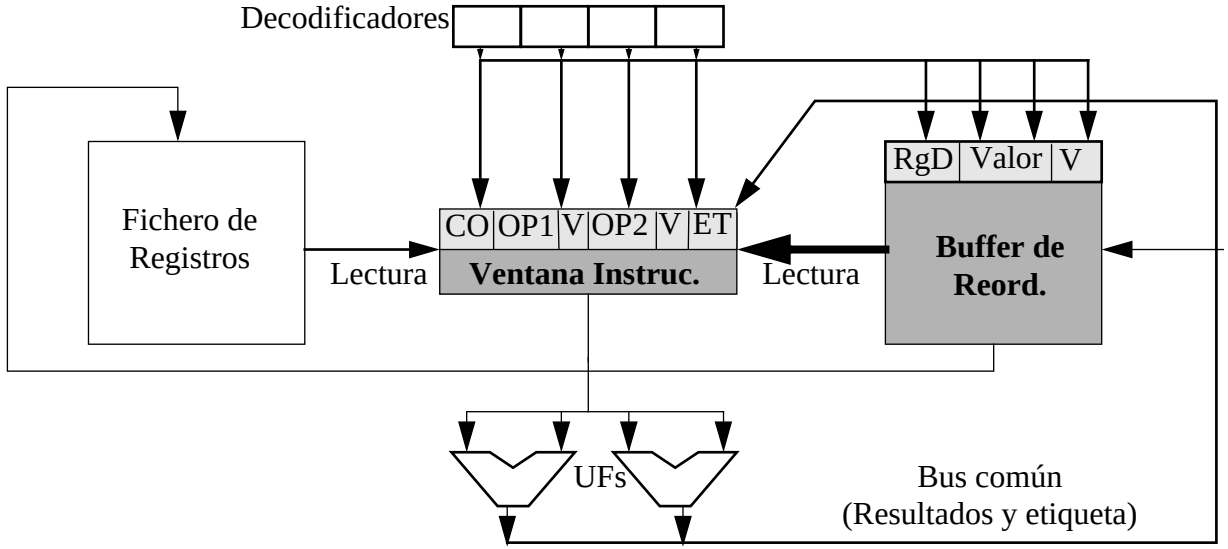
Emisión fuera de orden

VENTANA DE INSTRUCCIONES Y BUFFER DE REORDENACIÓN

✓ Etapa de Decodificación/Lectura de Operandos.

✧ *Lectura de Operandos: pueden estar en el buffer, en el banco de registros o computándose en alguna UF.*

✧ *Necesario comparar los RgOp. con todos los RgDest. presentes en el buffer.*



∗ Acierto y resultado válido en el buffer: Copiar el resultado en la ventana de instrucciones.

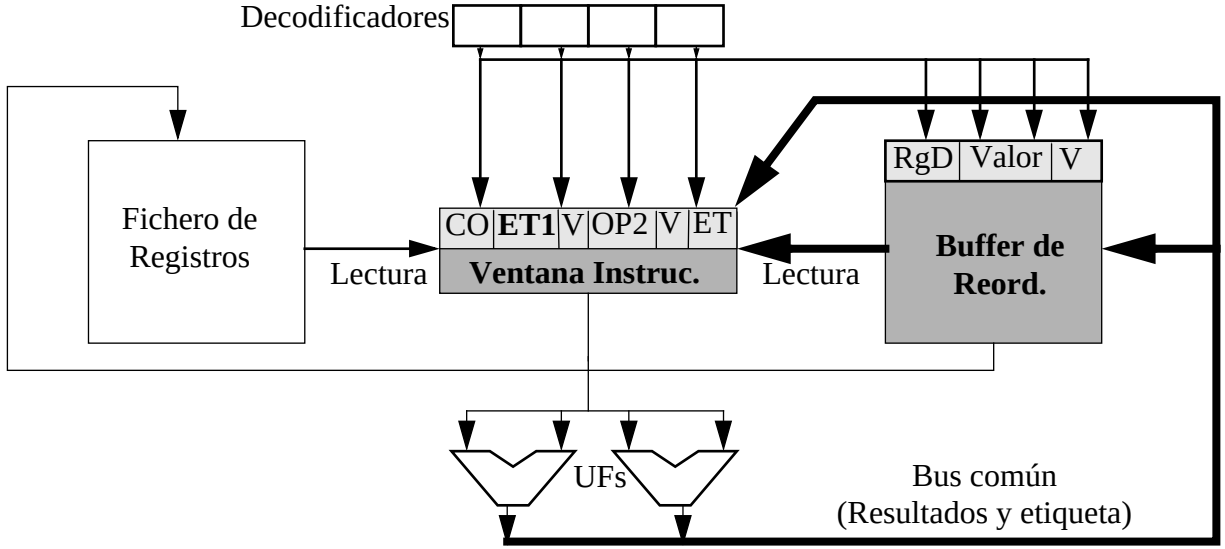
Emisión fuera de orden

VENTANA DE INSTRUCCIONES Y BUFFER DE REORDENACIÓN

✓ Etapa de Decodificación/Lectura de Operandos.

✧ *Lectura de Operandos: pueden estar en el buffer, en el fichero de registros o computándose en alguna UF.*

✧ *Necesario comparar los RgOp. con todos los RgDest. presentes en el buffer de reordenación.*



∗ Acierto y resultado no válido en el buffer: copiar la etiqueta en la Ventana. Cuando el resultado de la operación que generará el operando requerido se ponga en el bus común, es leido por la Ventana.

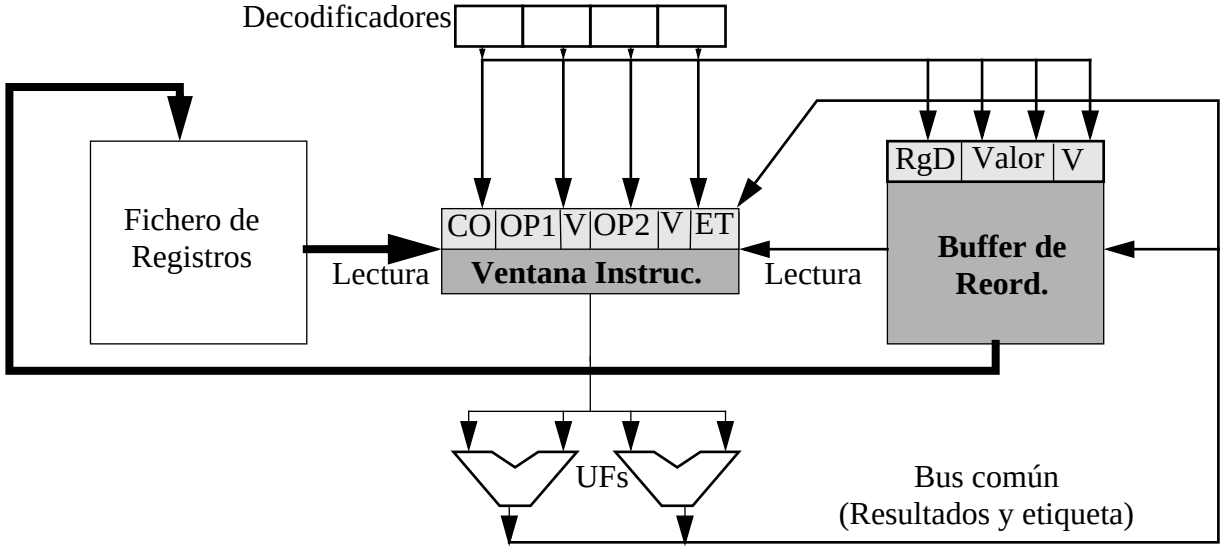
Emisión fuera de orden

VENTANA DE INSTRUCCIONES Y BUFFER DE REORDENACIÓN

✓ Etapa de Decodificación/Lectura de Operandos.

✧ *Lectura de Operandos: pueden estar en el buffer, en el fichero de registros o computándose en alguna UF.*

✧ *Necesario comparar los RgOp. con todos los RgDest. presentes en el buffer de reordenación.*



∗ Fallo en el buffer. Resultado ya ha sido escrito al fichero de registros. Leer de allí.

∗ Los resultados son escritos en el fichero de registros en orden desde el buffer de reordenación.

Emisión fuera de orden

```
1 nivel:    Ld f3,[r12]
2           Ld f1,[r10]
3           Ld f2,[r11]
4           f4=f1*f3
5           f2=f2*f3
6           f1=1-f3
7           f4=f4+f1
8           St f2,[r10]
9           St f4,[r11]
10          r10=r10+1
11          r11=r11+1
12          r12=r12+1
13          r0=r0-1
14          bnz nivel
```

C	Bus	Dec	V	E	S1	S2	M_1	M_2	M_e
1	1,2								
2	3,4	1,2	1,2						

Cod.	Op1	Op2	Tag
Ld	(r10)	-	#2
Ld	(r12)	-	#1

Ventana de instrucciones

Tag	Valor	R. Dest
#2	No disp.	f1
#1	No disp.	f3

Buffer de reordenación

Emisión fuera de orden

```
1 nivel:    Ld f3,[r12]
2           Ld f1,[r10]
3           Ld f2,[r11]
4           f4=f1*f3
5           f2=f2*f3
6           f1=1-f3
7           f4=f4+f1
8           St f2,[r10]
9           St f4,[r11]
10          r10=r10+1
11          r11=r11+1
12          r12=r12+1
13          r0=r0-1
14          bnz nivel
```

C	Bus	Dec	V	E	S1	S2	M_1	M_2	M_e
1	1,2								
2	3,4	1,2	1,2						
3	5,6	3,4	2,3,4						1

Cod.	Op1	Op2	Tag
*Flo.	#2	#1	#4
Ld	(r11)	-	#3
Ld	(r10)	-	#2

Ventana de instrucciones

Tag	Valor	R. Dest
#4	No disp.	f4
#3	No disp.	f2
#2	No disp.	f1
#1	No disp.	f3

Buffer de reordenación

Emisión fuera de orden

```
1 nivel:    Ld f3,[r12]
2           Ld f1,[r10]
3           Ld f2,[r11]
4           f4=f1*f3
5           f2=f2*f3
6           f1=1-f3
7           f4=f4+f1
8           St f2,[r10]
9           St f4,[r11]
10          r10=r10+1
11          r11=r11+1
12          r12=r12+1
13          r0=r0-1
14          bnz nivel
```

C	Bus	Dec	V	E	S1	S2	M_1	M_2	M_e	EB
1	1,2									
2	3,4	1,2	1,2							
3	5,6	3,4	2,3,4						1	
4	7,8	5,6	3,4,5,6						2	1

Cod.	Op1	Op2	Tag
- Flo.	[f3]	1	#6
* Flo.	#3	[f3]	#5
* Flo.	#2	[f3]	#4
Ld	(r11)	-	#3

Ventana de instrucciones

Tag	Valor	R. Dest
#6	No disp.	f3
#5	No disp.	f2
#4	No disp.	f4
#3	No disp.	f2
#2	No disp.	f1
#1	[f3]	f3

Buffer de reordenación

Emisión fuera de orden

```
2     Ld f1,[r10]
3     Ld f2,[r11]
4     f4=f1*f3
5     f2=f2*f3
6     f1=1-f3
7     f4=f4+f1
8     St f2,[r10]
9     St f4,[r11]
10    r10=r10+1
11    r11=r11+1
12    r12=r12+1
13    r0=r0-1
14    bnz nivel
```

C	Bus	Dec	V	E	S1	S2	M_1	M_2	M_e	EB	Co
1	1,2										
2	3,4	1,2	1,2								
3	5,6	3,4	2,3,4						1		
4	7,8	5,6	3,4,5,6						2	1	
5	9,10	7,8	4,5,7,8		6				3	2	1

Cod.	Op1	Op2	Tag
Store	#5	(r10)	#8
+ Flo.	#4	#6	#7
* Flo.	#3	[f3]	#5
* Flo.	[f1]	[f3]	#4

Ventana de instrucciones

Tag	Valor	R. Dest
#8	No disp.	-
#7	No disp.	f4
#6	No disp.	f1
#5	No disp.	f2
#4	No disp.	f4
#3	No disp.	f2
#2	[f1]	f1

Buffer de reordenación

Emisión fuera de orden

1 nivel:	Ld f3,[r12]
2	Ld f1,[r10]
3	Ld f2,[r11]
4	f4=f1*f3
5	f2=f2*f3
6	f1=1-f3
7	f4=f4+f1
8	St f2,[r10]
9	St f4,[r11]
10	r10=r10+1
11	r11=r11+1
12	r12=r12+1
13	r0=r0-1
14	bnz nivel

C	Bus	Dec	V	E	S1	S2	M_1	M_2	M_e	EB	Co
1	1,2										
2	3,4	1,2	1,2								
3	5,6	3,4	2,3,4						1		
4	7,8	5,6	3,4,5,6						2	1	
5	9,10	7,8	4,5,7,8		6				3	2	1
6	11,12	9,10	5,7,8,9,10		6		4			3	2

Cod.	Op1	Op2	Tag
+ E	(r10)	1	#10
Store	#7	(r11)	#9
Store	#5	(r10)	#8
+ Flo.	#4	#6	#7
* Flo.	[f2]	[f3]	#5

Ventana de instrucciones

Tag	Valor	R. Dest
#10	No disp.	r10
#9	No disp.	-
#8	No disp.	-
#7	No disp.	f4
#6	No disp.	f1
#5	No disp.	f2
#4	No disp.	f4
#3	[f2]	f2

Emisión fuera de orden

```
1 nivel:    Ld f3,[r12]
2           Ld f1,[r10]
3           Ld f2,[r11]
4           f4=f1*f3
5           f2=f2*f3
6           f1=1-f3
7           f4=f4+f1
8           St f2,[r10]
9           St f4,[r11]
10          r10=r10+1
11          r11=r11+1
12          r12=r12+1
13          r0=r0-1
14          bnz nivel
```

C	Bus	Dec	V	E	S1	S2	M_1	M_2	M_e	EB	Co
3	5,6	3,4	2,3,4						1		
4	7,8	5,6	3,4,5,6						2	1	
5	9,10	7,8	4,5,7,8		6				3	2	1
6	11,12	9,10	5,7,8,9,10		6		4			3	2
7	13,14	11,12	7,8,9,11,12	10			4	5		6	3

Cod.	Op1	Op2	Tag
+ E	(r12)	1	#12
+ E	(r11)	1	#11
Store	#7	(r11)	#9
Store	#5	(r10)	#8
+ Flo.	#4	[f1]	#7

Ventana de instrucciones

Tag	Valor	R. Dest
#12	No disp.	r12
#11	No disp.	r11
#10	No disp.	r10
#9	No disp.	-
#8	No disp.	-
#7	No disp.	f4
#6	[f1]	f1
#5	No disp.	f2
#4	No disp.	f4

Emisión fuera de orden

```
1 nivel:    Ld f3,[r12]
2           Ld f1,[r10]
3           Ld f2,[r11]
4           f4=f1*f3
5           f2=f2*f3
6           f1=1-f3
7           f4=f4+f1
8           St f2,[r10]
9           St f4,[r11]
10          r10=r10+1
11          r11=r11+1
12          r12=r12+1
13          r0=r0-1
14          bnz nivel
```

C	Bus	Dec	V	E	S1	S2	M_1	M_2	M_e	EB	Co
4	7,8	5,6	3,4,5,6						2	1	
5	9,10	7,8	4,5,7,8		6				3	2	1
6	11,12	9,10	5,7,8,9,10		6		4			3	2
7	13,14	11,12	7,8,9,11,12	10			4	5		6	3
8	? ?	13,14	7,8,9,12,13,14	11			4	5		10	

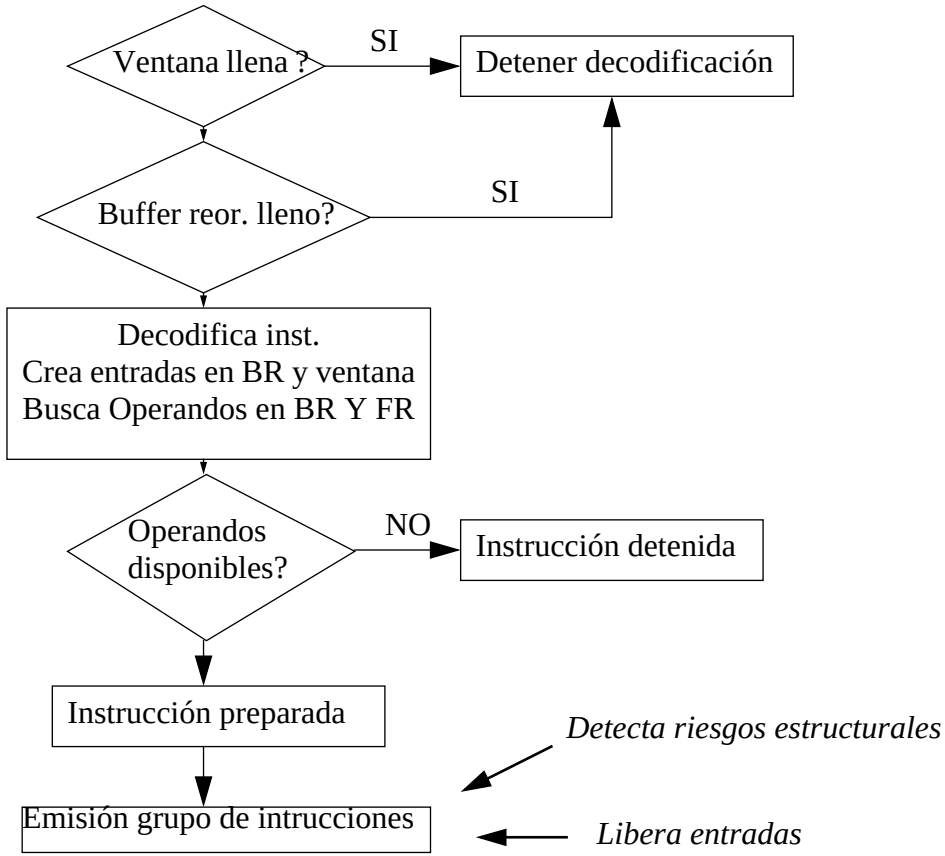
Cod.	Op1	Op2	Tag
bnz	nivel	-	#14
- E	(r0)	1	#13
+ E	(r12)	1	#12
Store	#7	(r11)	#9
Store	#5	(r10)	#8
+ Flo.	#4	[f1]	#7

Ventana de instrucciones

Tag	Valor	R. Dest
#14	No disp.	-
#13	No disp.	r0
#12	No disp.	r12
#11	No disp.	r11
#10	[r10]]	r10
#9	No disp.	-
#8	No disp.	-
#7	No disp.	f4
#6	[f1]	f1
#5	No disp.	f2
#4	No disp.	f4

Emisión fuera de orden

✓ Flujo de la emisión fuera de or den



Buffer de reordenación

ETAPAS RELACIONADAS CON EL BUFFER DE REORDENACIÓN

- ✓ Etapa de Decodificación/Lectura
  - ✧ *Se asigna una entrada en BR con campo reg dest y un valor al campo tag.*
  - ✧ *Se asigna una entrada en la ventana de intrucciones con campo tag igual al campo tag del buffer de reordenación.*
  - ✧ *Lectura de operandos en paralelo en BR, FR.*
    - ∗ Registro operando está en BR: el resultado o el tag más reciente se envía a la Ventana de Instrucciones.
    - ∗ Registro operando no está en BR: se lee en FR
- ✓ Etapa de ejecución
  - ✧ *A las unidades funcionales se les envían los operandos y el tag de la entrada del BR donde deben ser almacenado el resultados.*

Buffer de reordenación

ETAPAS RELACIONADAS CON EL BUFFER DE REORDENACIÓN

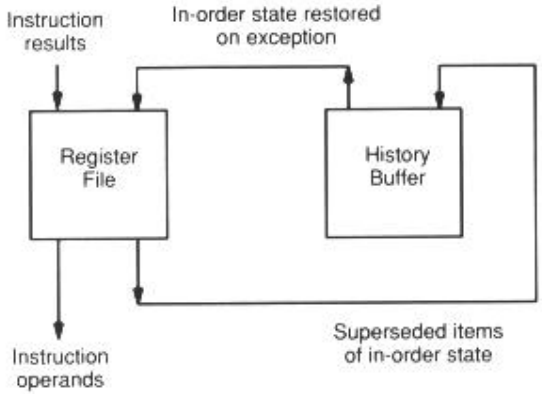
- ✓ Etapa de Escritura
  - ✧ *Las UFs realizan una escritura asociativa en BR y en la Ventana de Instrucciones direccionando con el campo tag.*

- ✓ Etapa de Commit (actualización de FR)
  - ✧ *Las entradas de BR que llegan completas al fondo de la FIFO se escriben en FR (orden de decodificación).*
  - ✧ *Las no completas esperan resultado de una UF*
  - ✧ *Instrucciones que producen excepciones: borrar BR*

Implementaciones alternativas de la ejecución especulativa

BUFFER DE HISTORIA

- ✓ Contiene los valores antiguos de los registros destinos que van a ser modificados por las instrucciones en curso.
- ✓ Funciona como una cola lifo.
  - ✧ *Al decodificar una instrucciones el contenido del registro destino se coloca en la cabecera de la cola.*
  - ✧ *Entrada del final del buffer de historia:*
    - ∗ Si la instrucción correspondiente se completó -> eliminar esta entrada.
    - ∗ En caso contrario, parar la decodificación (riesgo estructural en el BH).



Implementaciones alternativas de la ejecución especulativa

BUFFER DE HISTORIA

✓ Excepción o fallo de pr edicción pr oducidos por una instrucción:

- ✧ *Detener la decodificación.*
- ✧ *Completar ejecución instrucciones precedentes.*
- ✧ *Copiar desde BH a FR los valores de los registros destino de las instrucciones posteriores.*

✓ Desventajas

- ✧ *No hace renombre. Necesidad de mecanismos adicionales.*
- ✧ *FR necesita incrementar los puertos de lectura para escribir los contenidos de los resgistros detinos en el BH.*
- ✧ *Restaurarición del FR en excepciones y saltos costosa (en ciclos).*

Implementaciones alternativas de la ejecución especulativa

FICHERO FUTURO

- ✓ Presupone la existencia de un buffer de reordenación, per o
- ✓ Evita la lectura asociativa de operandos en el BR.
  - ✧ *En decodificacón se asocia un tag en la entrada correspondiente del FF.*
    - ✻ Entrada del registro destino.
    - ✻ Tag permite solucionar riesgos RR y DR (bus común como en Tomasulo).
  - ✧ *Lectura de los resultados más recientes o tag asociados se hace en FF.*
  - ✧ *Las instrucciones escriben en FF (si no hay supersede) y BR en paralelo.*