Arquitectura e Ingeniería de Computadores

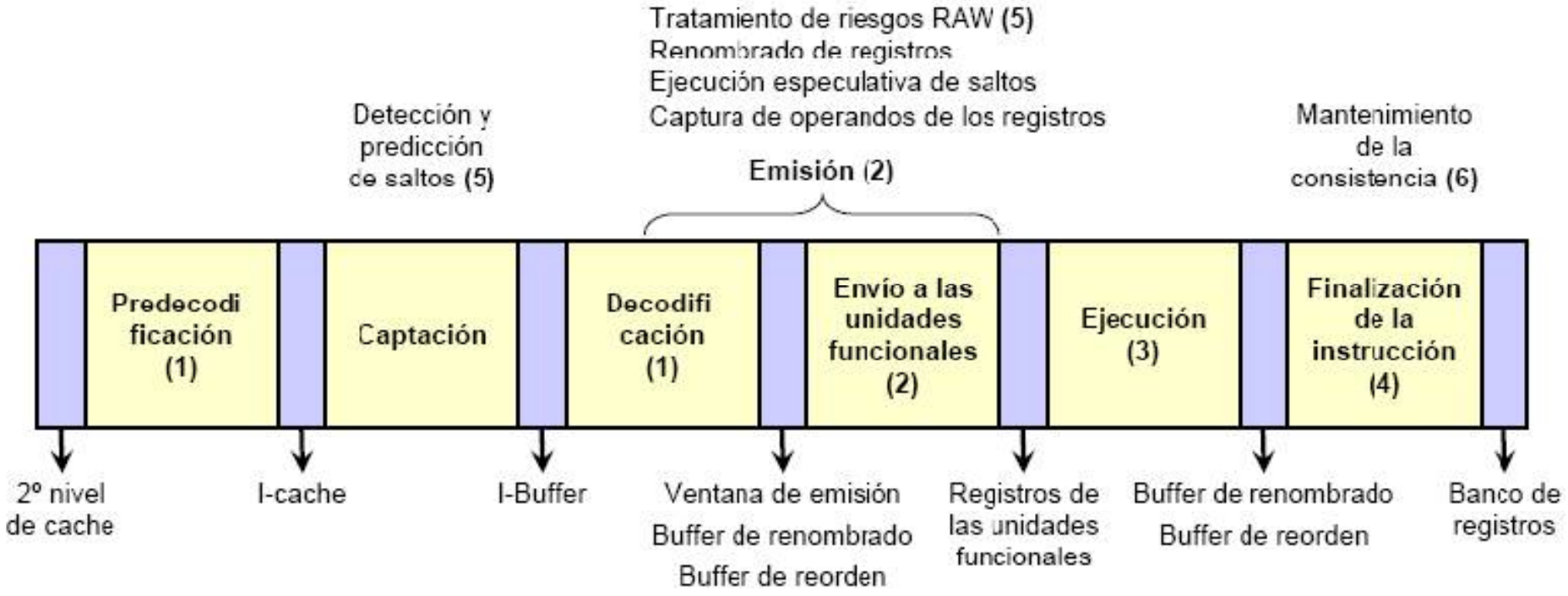
Tema 2 – Segmentación y superescalares clase 4

Ingeniería en Informática

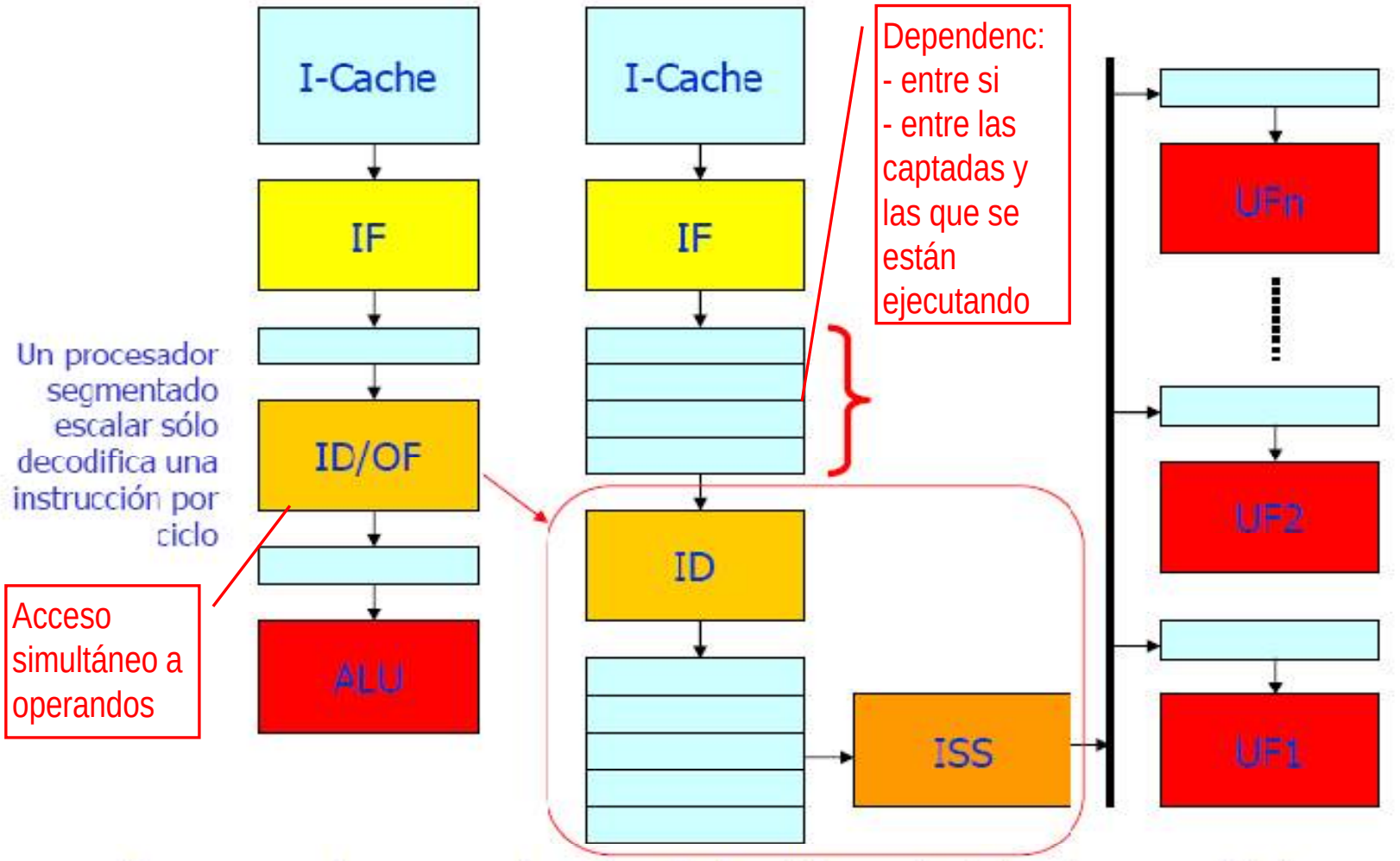
Departamento de Tecnología Informática y Computación

- Aspectos del procesamiento Superescalar y etapas del cauce donde se resuelven

1. **Decodificación Paralela** (Decodificación a mayor velocidad → **Predecodificación**)
2. **Emisión Paralela** de Instrucciones a las Unidades Funcionales (**Dependencias**)
3. **Ejecución Paralela** en los distintos Unidades Funcionales, UF, (segmentadas).
4. **Finaliza** (o se **Completa**) **el Procesamiento** de la Instrucción
5. **Detección y Resolución de Dependencias**
6. **Mantener la consistencia** secuencial (desacoplar la ejecución y la escritura de resultados)

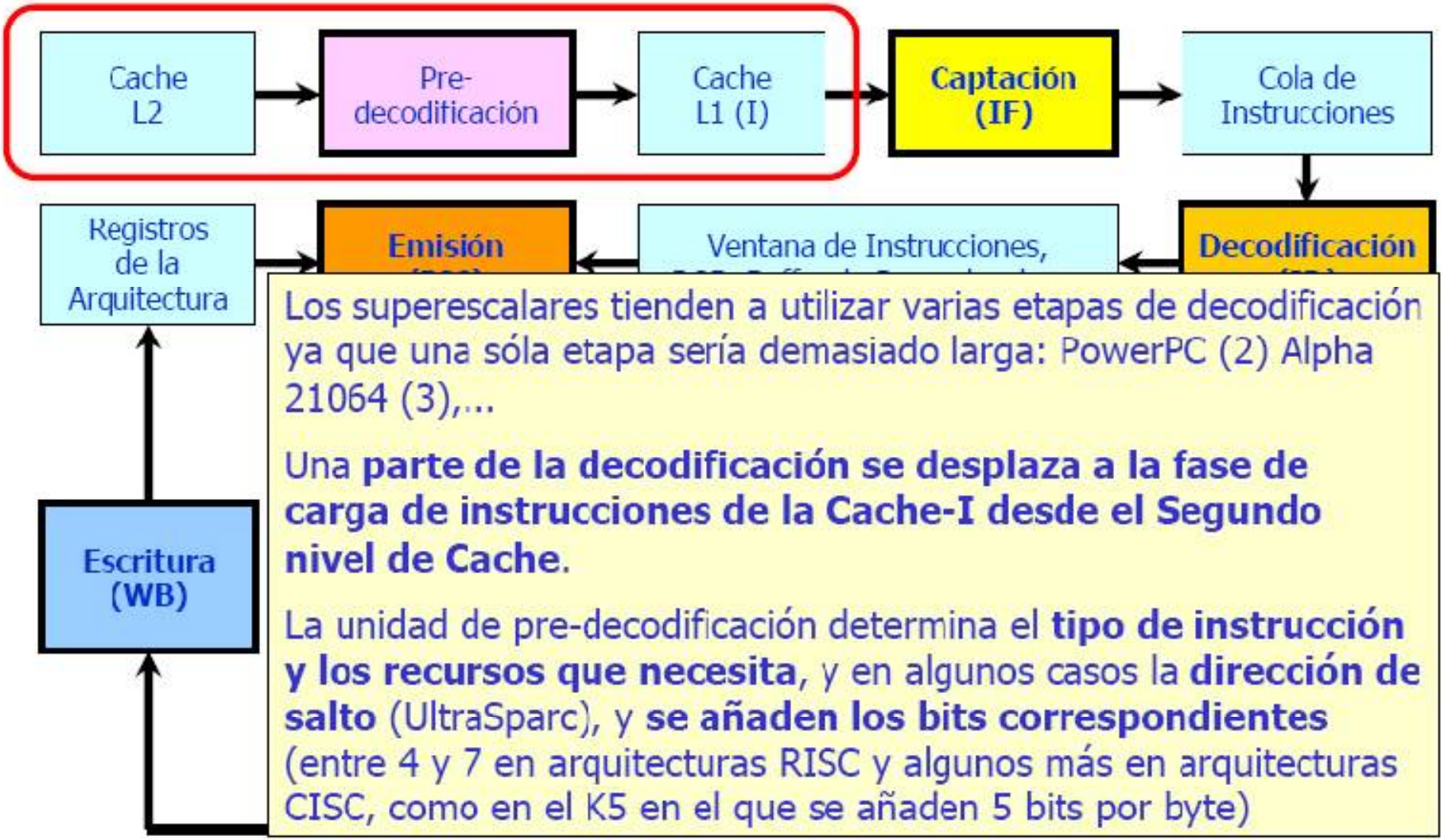


1. Decodificación Paralela



En un procesador superescalar se han de decodificar varias instrucciones por ciclo (y comprobar las dependencias con las instrucciones que se están ejecutando)

1.1 Predecodificación



Los superescalares tienden a utilizar varias etapas de decodificación ya que una sóla etapa sería demasiado larga: PowerPC (2) Alpha 21064 (3),...

Una **parte de la decodificación se desplaza a la fase de carga de instrucciones de la Cache-I desde el Segundo nivel de Cache.**

La unidad de pre-decodificación determina el **tipo de instrucción y los recursos que necesita**, y en algunos casos la **dirección de salto** (UltraSparc), y **se añaden los bits correspondientes** (entre 4 y 7 en arquitecturas RISC y algunos más en arquitecturas CISC, como en el K5 en el que se añaden 5 bits por byte)

- **Predecodificación: incremento en el ancho de banda**

Cuando las instrucciones se pasan de la cache L2 a la L1, la unidad de predecodificación añade bits (4 – 7 en instrucciones RISC) para facilitar la decodificación.

Predecodificación: bits añadidos

Los bits de predecodificación suelen indicar:

– Si es una instrucción de salto o no (se puede empezar su procesamiento antes)

– El tipo de unidad funcional que va a utilizar (se puede emitir más rápidamente si hay cauces para enteros o coma flotante…)

– Si hace referencia a memoria o no

• Ejemplos de uso de los bits de predecodificación:

– HP-PA 7200: Se añaden bits por cada par de instrucciones para facilitar la comprobación de si las instrucciones se pueden ejecutar en paralelo en el cauce o hay algún tipo de dependencias (10% de SRAM).

– AMD K6: La lógica de decodificación indica la longitud de la instrucción x86 en bytes permitiendo la localización del final de cada instrucción y se almacena en la cache L1 junto con las instrucciones. En una cache de 64 Kbytes se añaden 20 Kbytes para los bits de decodificación.

Ejemplos

Número de bits de Predecodificación utilizados

PA 7200 (1995)	5
PA 8000 (1996)	5
PowerPC 620 (1995)	7
UltraSparc (1995)	4
HAL PM1 (1995)	4
AMD K5 (1995)	5*
R10000 (1996)	4

* En el K5, los 5 bits de predecodificación se añaden a cada byte

Sin Predecodificación

Tipo/Año de lanzamiento	Instr./ciclo
PA 7100 (1992)	2
PowerPC 601 (1993)	4
PowerPC 604 (1994)	3
R8000 (1994)	4
SuperSparc (1992)	3
AM 29000 sup. (1995)	3
α21164	4
PentiumPro (1995)	3

Con Predecodificación

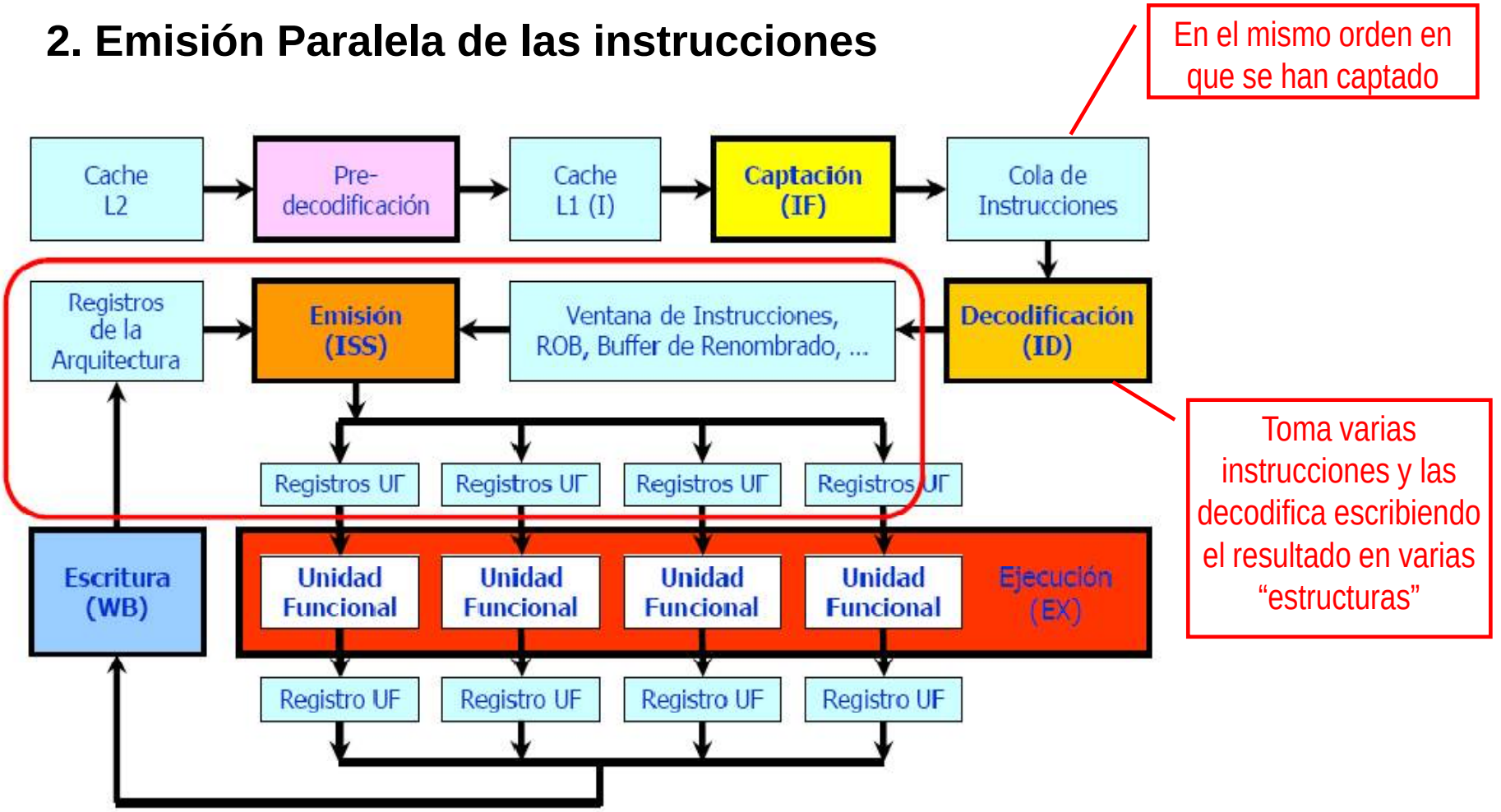
Tipo/Año de lanzamiento	Instr./ciclo	MHz[1]	Etapas cauce[2]
PA7200 (1995)	2	100	n. d.
PA8000 (1996)	4	150	n. d.
PowerPC 620 (1995)	4	133	F, D/I, E
R10000 (1995)	4	200	F, D, I, E
UltraSparc (1995)	4	167	F, D, I, E
Hal PM1 (1995)	4	154	F, D, I, E
K5 (1995)	~2[3]	n. d.	n. d.

[1] Frecuencias de reloj iniciales

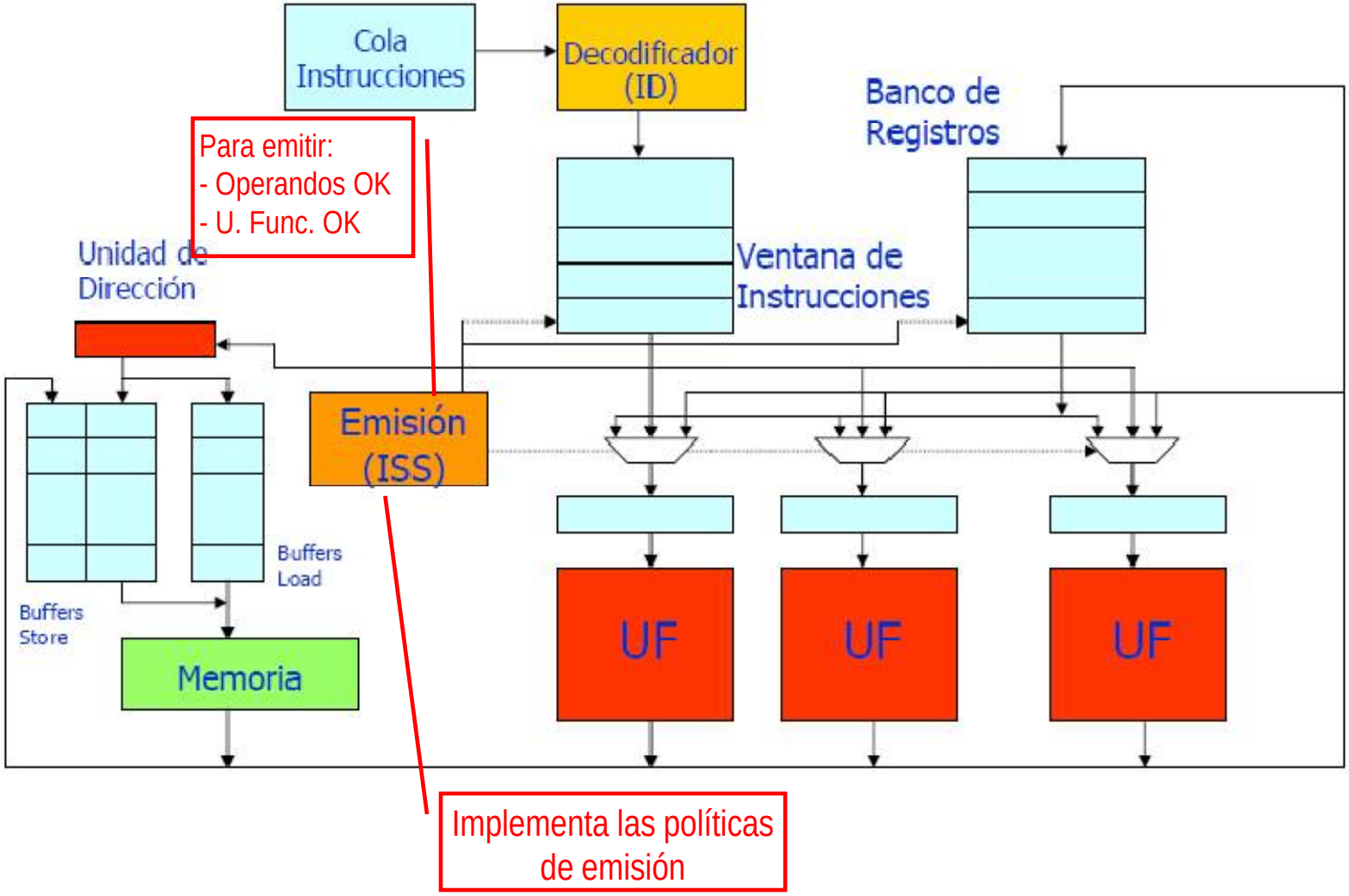
[2] Abreviaturas: F: *Fetch*, D: *Decode*, I: *Issue*, E: *Execute*. Las siguientes etapas se han omitido

[3] La tasa de emisión es de 4 microinstrucciones por ciclo, lo que hace unas 2 instrucciones x86 por ciclo

2. Emisión Paralela de las instrucciones

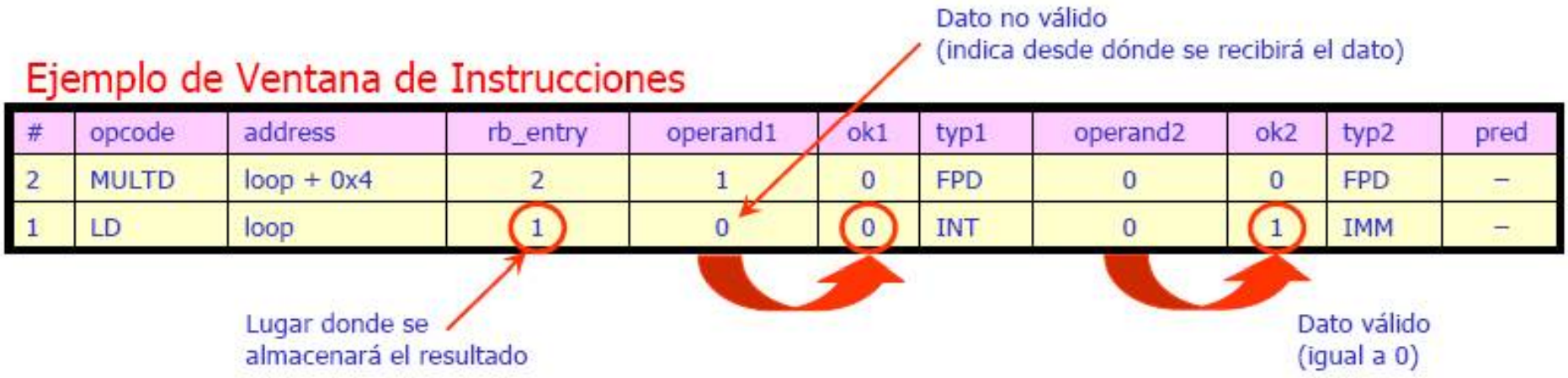


Emisión Paralela: ejemplo de estructura del cauce



Emisión Paralela: Ventana de instrucciones

- La ventana de instrucciones almacena las instrucciones pendientes (todas si la ventana es centralizada o las de un tipo determinado, si es distribuida
- Las instrucciones se cargan en la ventana una vez decodificadas y se utiliza un bit para indicar si un operando está disponible (se almacena el valor o se indica el registro desde donde se lee) o no (se almacena la unidad funcional desde donde llegará el operando)
- Una instrucción puede ser emitida cuando tiene todos sus operandos disponibles y la unidad funcional donde se procesará. Hay diversas posibilidades para el caso en el que varias instrucciones estén disponibles (características de los buses, etc.)



#	opcode	address	rb_entry	operand1	ok1	typ1	operand2	ok2	typ2	pred
2	MULTD	loop + 0x4	2	1	0	FPD	0	0	FPD	–
1	LD	loop	1	0	0	INT	0	1	IMM	–

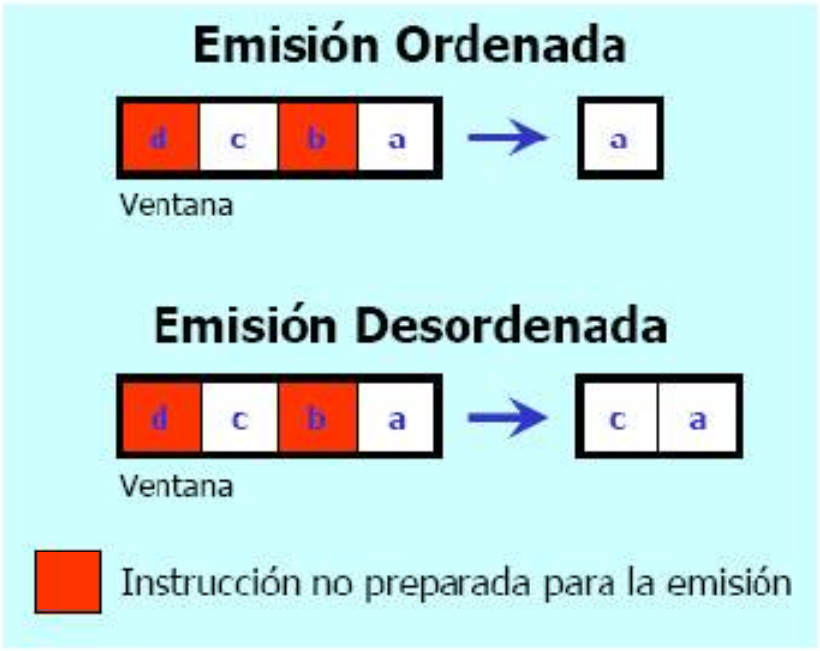
Emisión Paralela: Gestión de los bloqueos

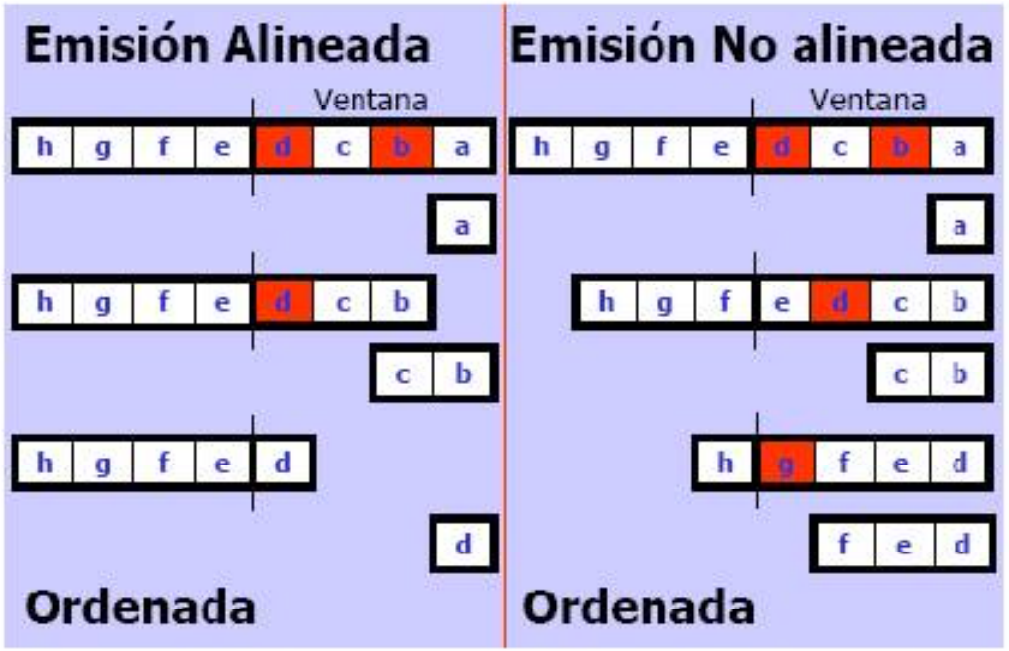
- **Orden:** Emisión Ordenada o Desordenada
- **Alineamiento:** Emisión Alineada o No alineada

Ejemplos:

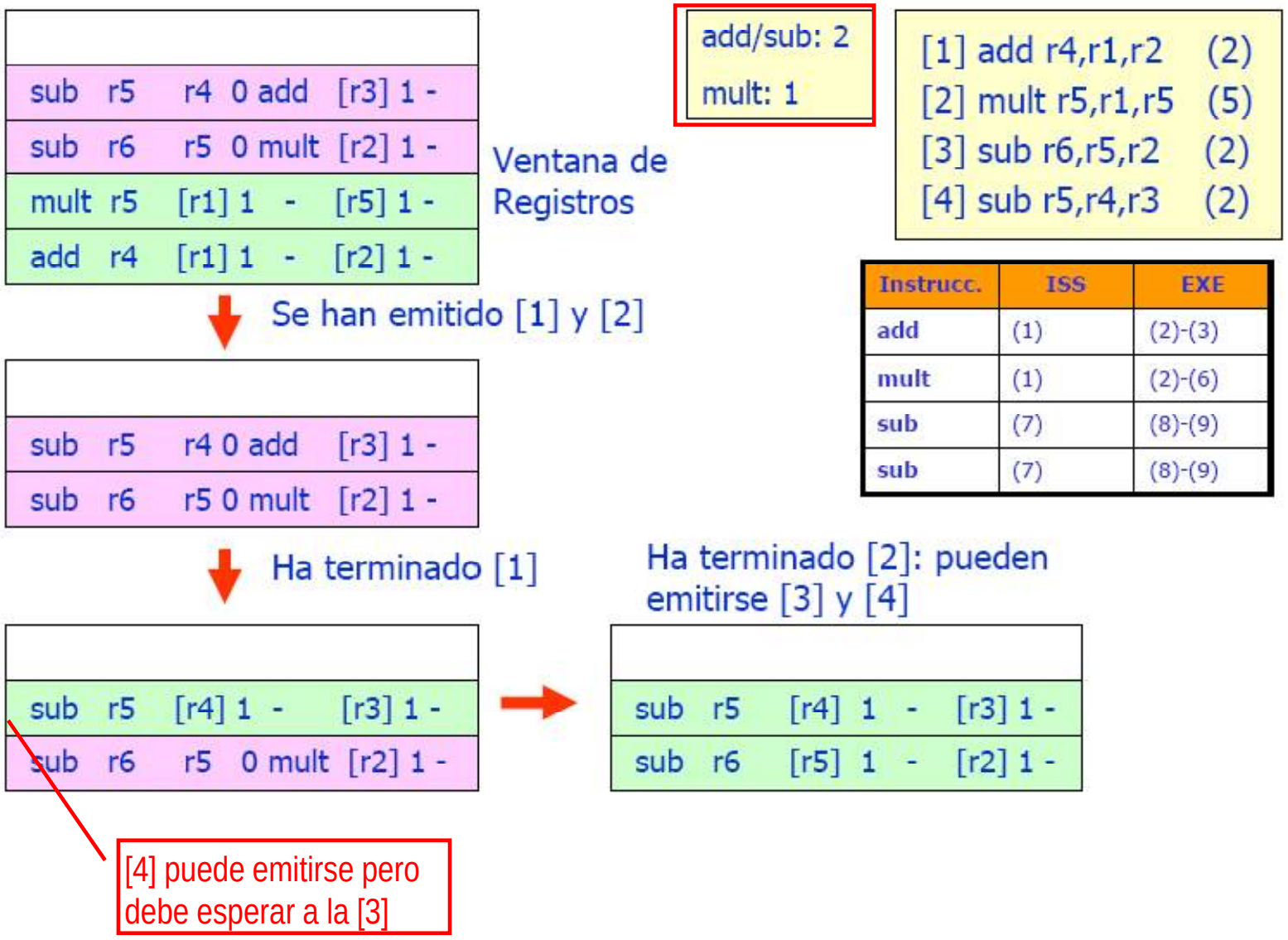
Alineada: SuperSparc (92), PowerPC (93, 95, 96), PA8000 (96), Alpha (92, 94, 95), R10000 (96)

No Alineada: MC88110 (93), PA7100LC (93), R8000 (94), UltraSparc (95)

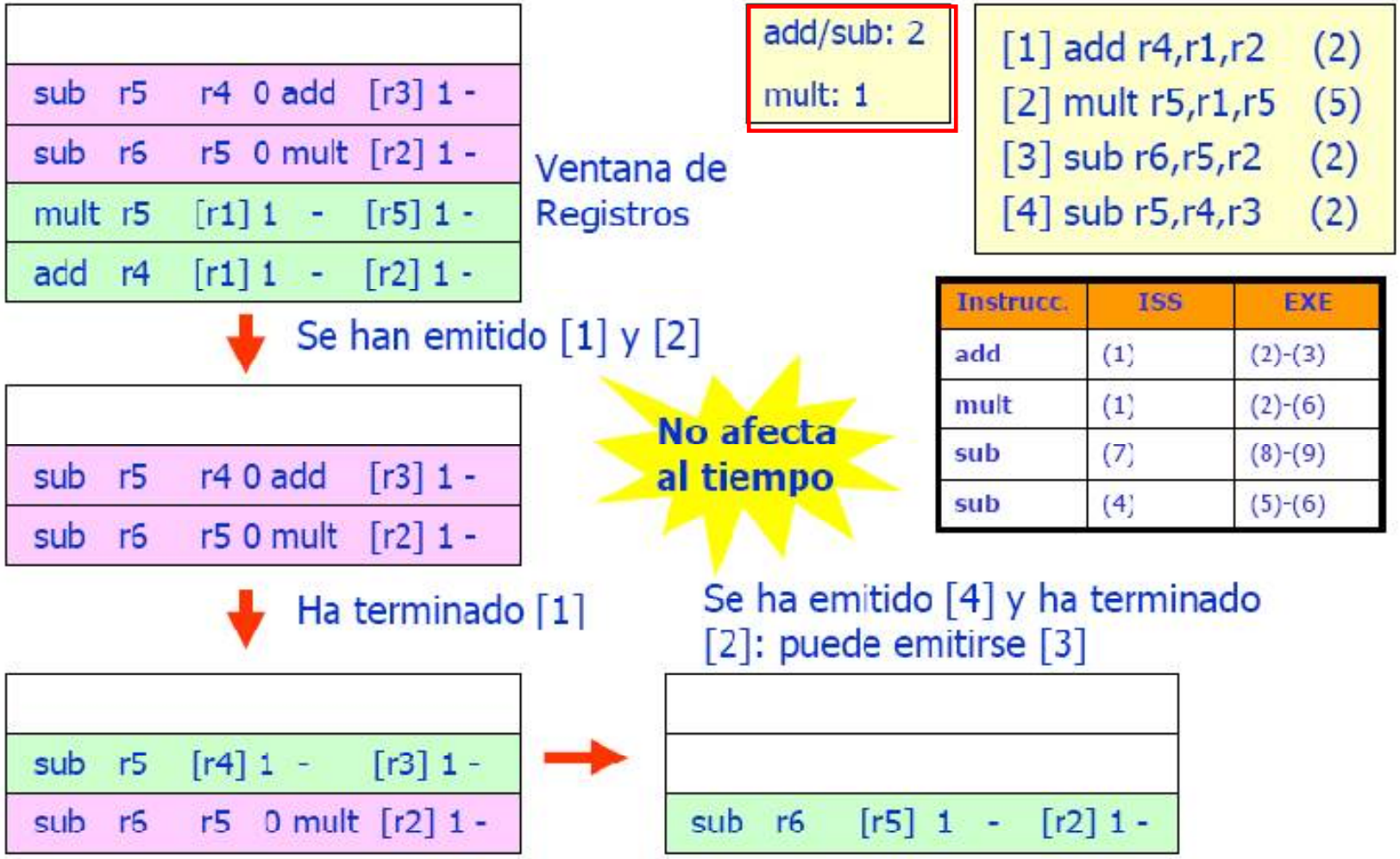




Emisión Paralela: Ejemplo1 de Emisión Ordenada

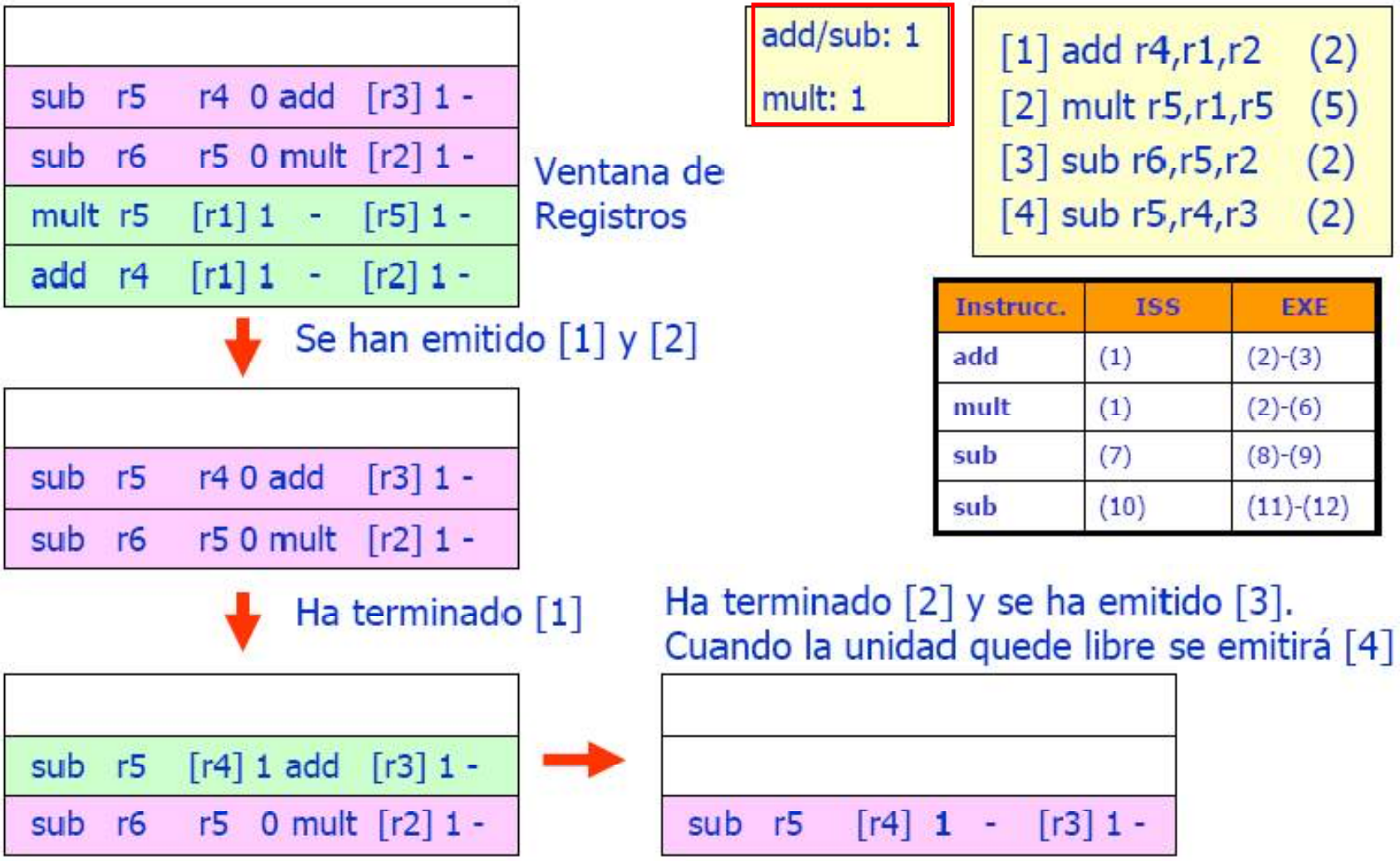


Emisión Paralela: Ejemplo1 de Emisión Desordenada

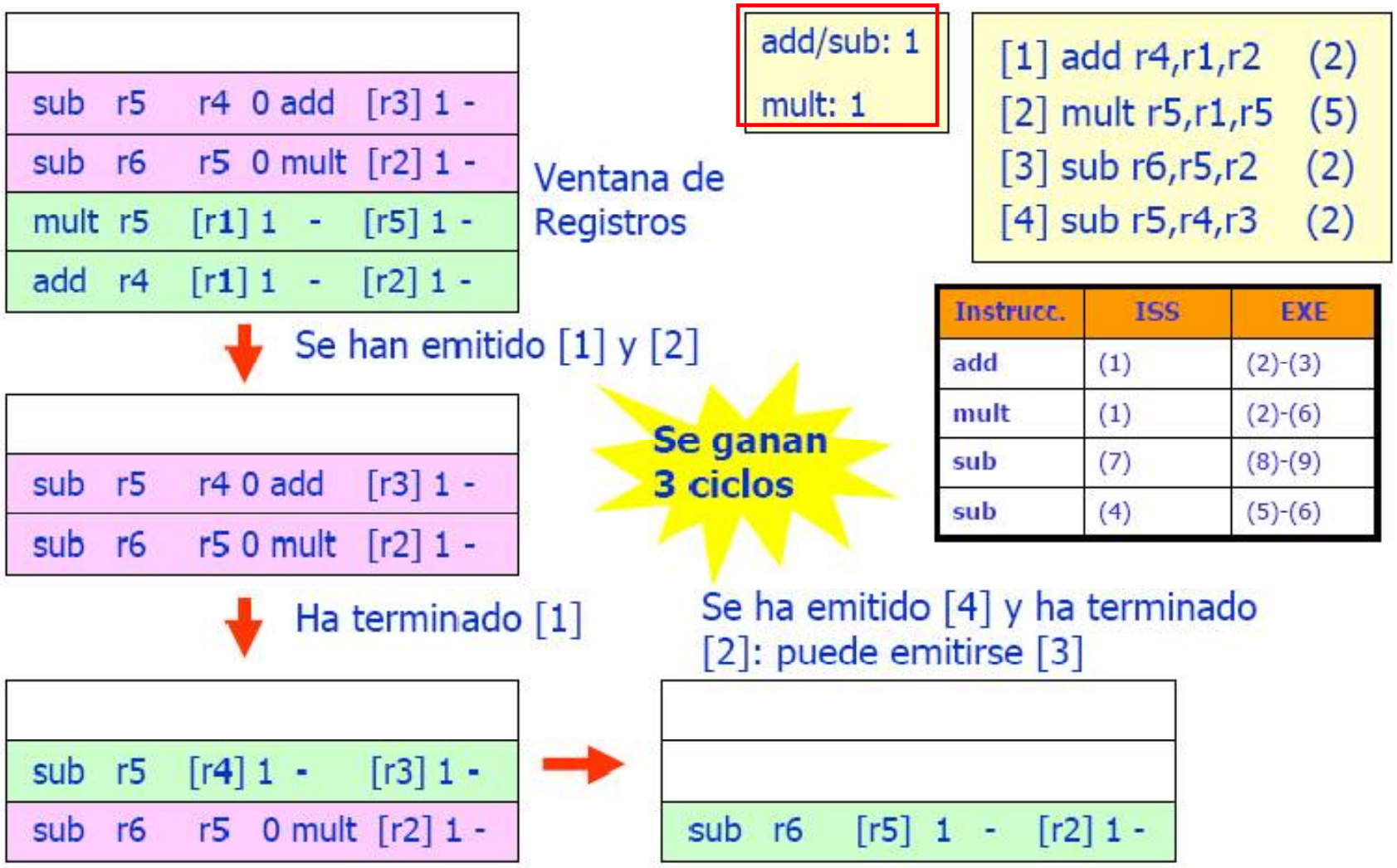


Instrucc.	ISS	EXE
add	(1)	(2)-(3)
mult	(1)	(2)-(6)
sub	(7)	(8)-(9)
sub	(4)	(5)-(6)

Emisión Paralela: Ejemplo2 de Emisión Ordenada

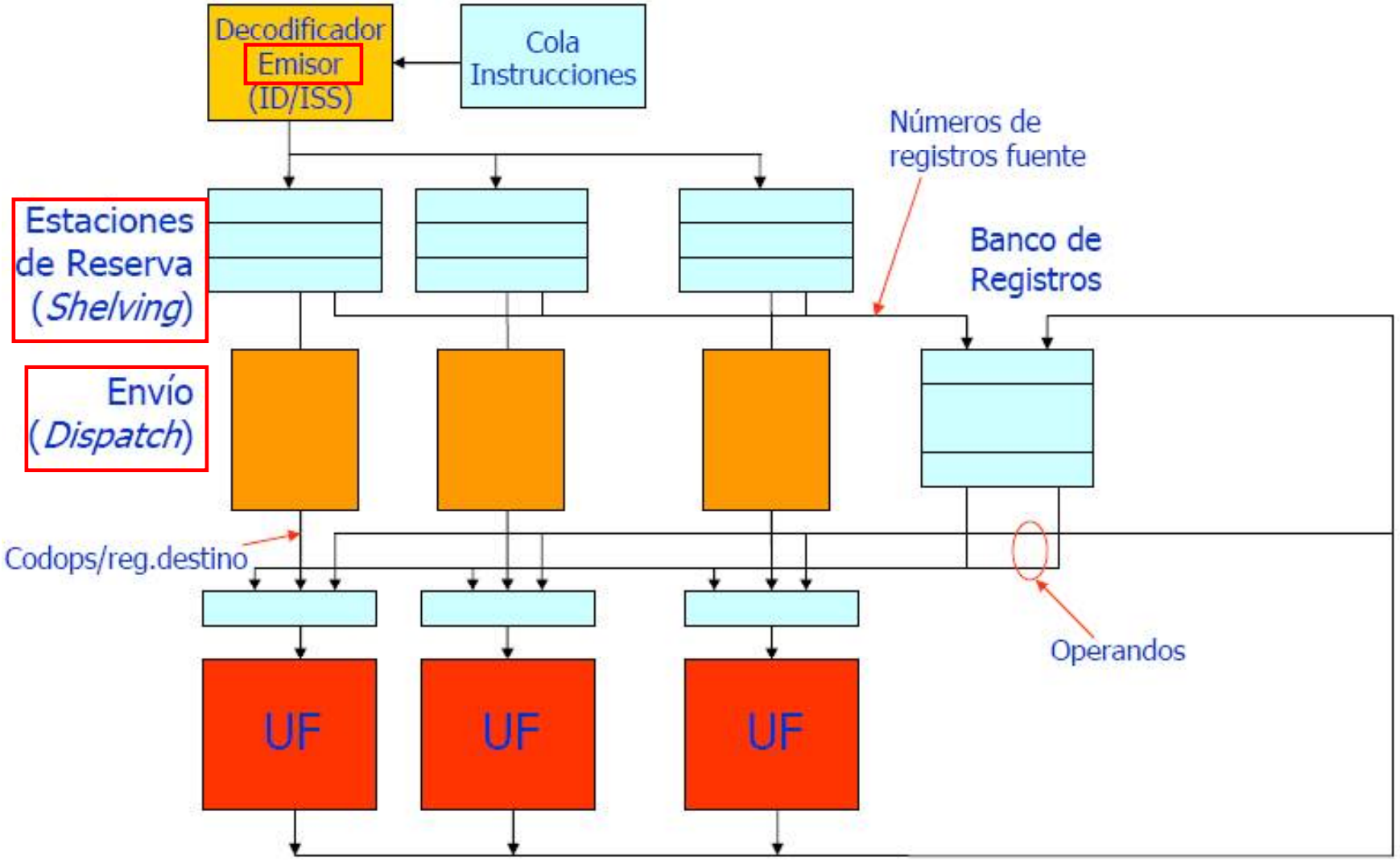


Emisión Paralela: Ejemplo2 de Emisión Desordenada

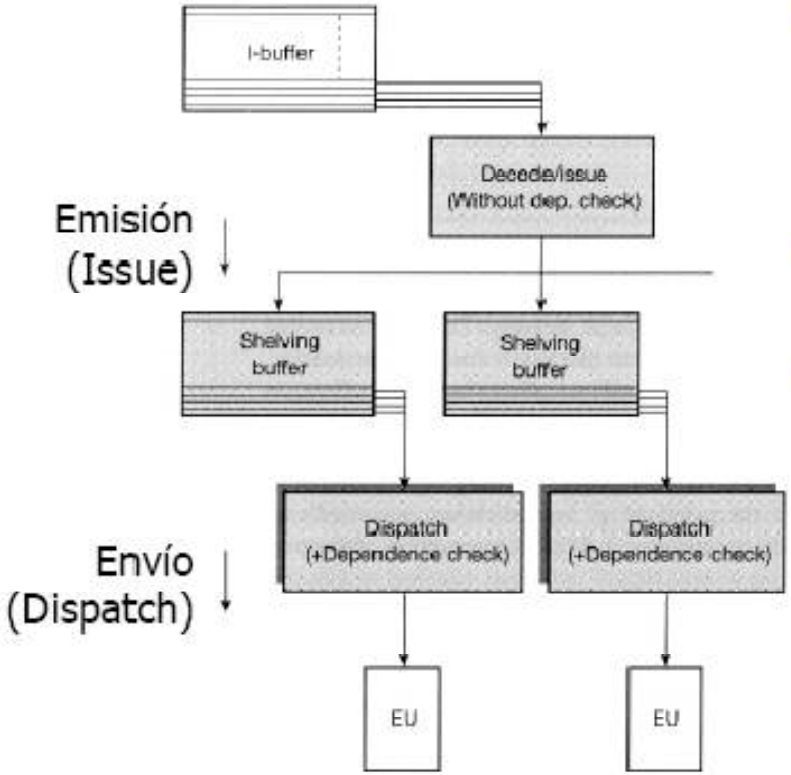


Instrucc.	ISS	EXE
add	(1)	(2)-(3)
mult	(1)	(2)-(6)
sub	(7)	(8)-(9)
sub	(4)	(5)-(6)

Emisión Paralela: Estaciones de Reserva (ventana de inst. distribuida)

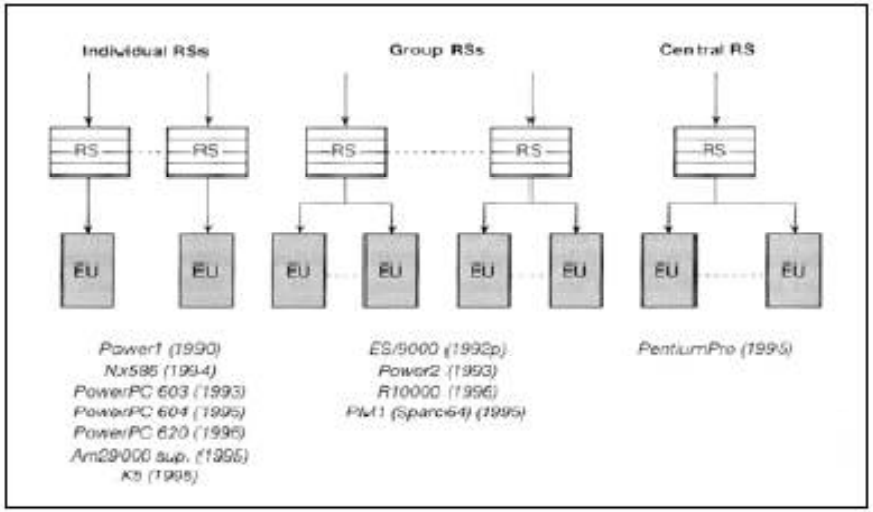


Emisión Paralela: Funcionamiento de las Estaciones de Reserva

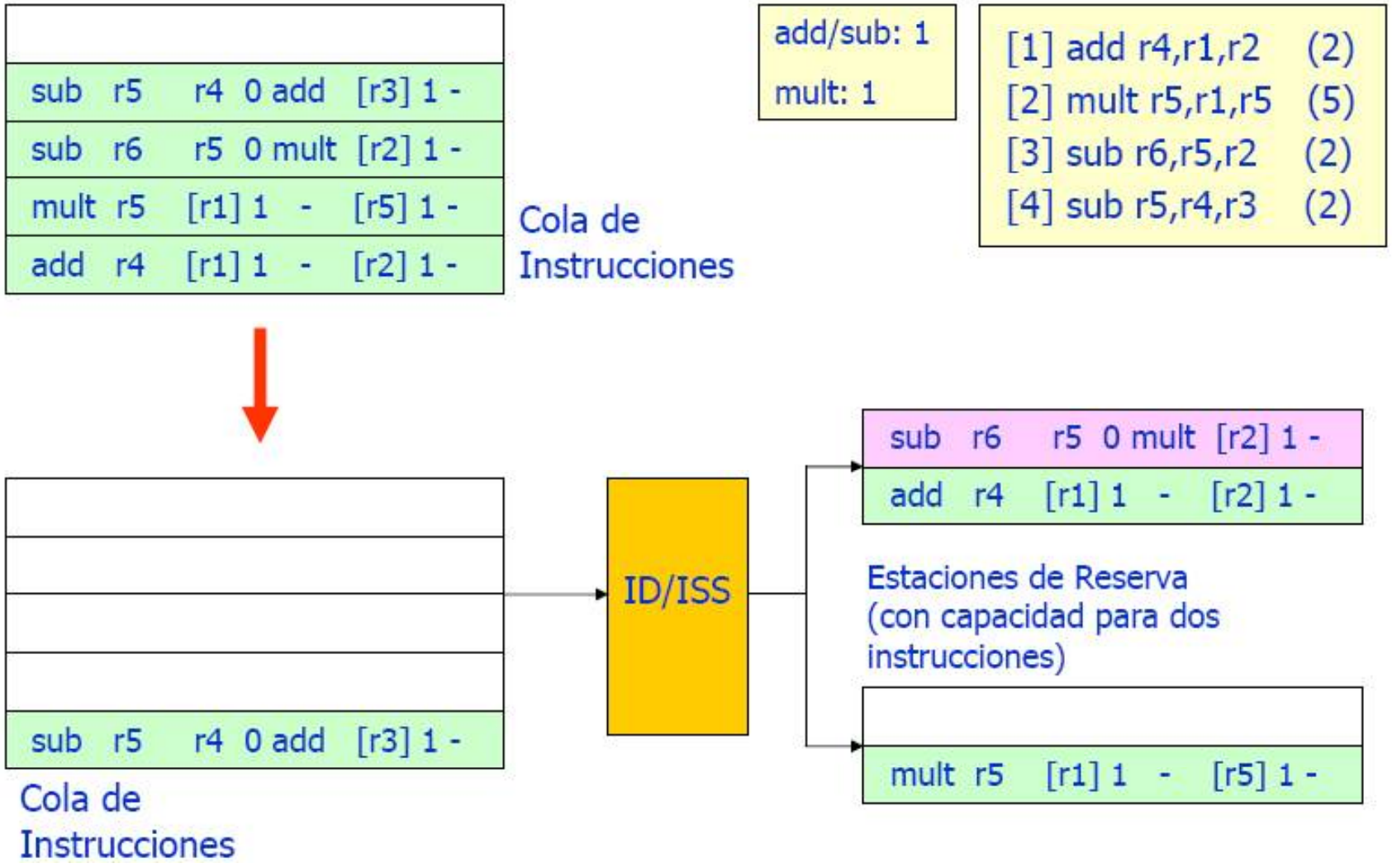


- Si no existen limitaciones en el hardware las instrucciones se emiten a los buffers de '*shelving*' (estaciones de reserva o consignas) independientemente de las dependencias
- Las instrucciones esperan hasta que se resuelvan las dependencias
- Las instrucciones se envían a las unidades funcionales una vez comprobada su independencia

Tipos de Estaciones de Reserva

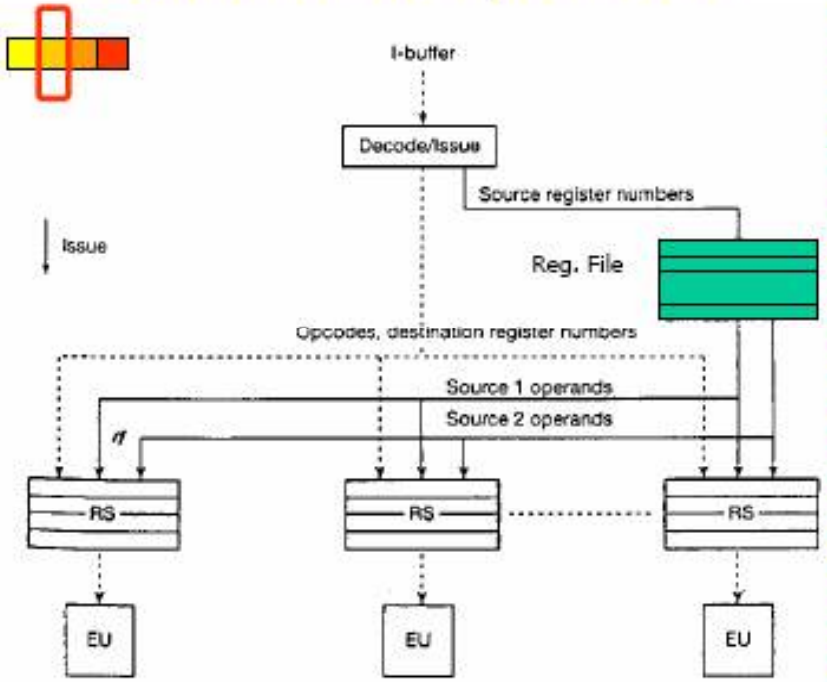


Emisión Paralela: Ejemplo de Funcionamiento de las Estaciones de Reserva



Estaciones de Reserva: Políticas de captación de operandos

Captación en la Decodificación/Emisión



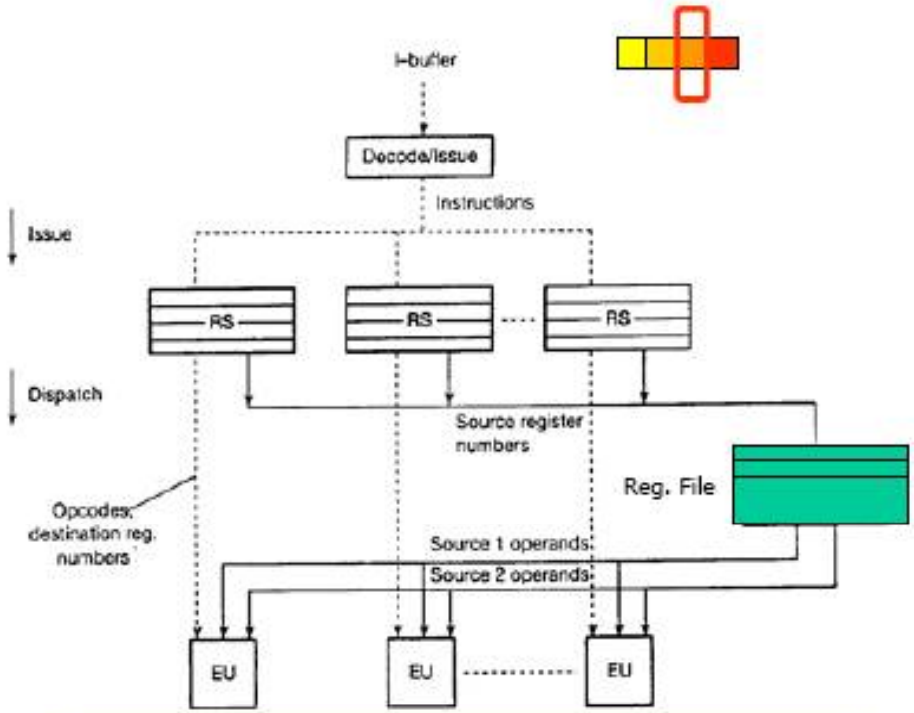
IBM 360/91 (1967)

PowerPC 603, 604, 620 (1996)

AMD K5 (1995)

Pentium Pro (1995)

Captación en el Envío



Power2 (1993)

PA 8000 (1996)

R10000 (1996)

Estaciones de Reserva: Esquemas de Envío (Dispatch)

Reglas de Selección: Se determina las instrucciones que pueden enviarse

Las instrucciones ejecutables

Reglas de Arbitraje: Instrucción que se envía si hay varias ejecutables

La más antigua entre las ejecutables

Orden de Envío: Ordenadas, Desordenadas, o Parcialmente ordenadas (ciertas instrucciones no ejecutables bloquean instrucciones de un tipo, pero no de otros)

Prest

Tend.

Ordenada: Power 1 (90), PowerPC 603 (93), Am29000 (95)

Parcialmente ordenada: Power 2 (93), PowerPC 604 (95), PowerPC 620(96)

Desordenada: ES/9000(92), PentiumPro (95), R10000(96), PA8000 (96)

Velocidad de Envío: Número de instrucciones que se envían por ciclo

Una por ciclo: PowerPC 603 (93), PowerPC 604 (94), PowerPC 620 (95)

Varias por Ciclo: Sparc64(95), R10000 (96), PA8000 (96), PentiumPro (95)

Estaciones de Reserva: Comprobación disponibilidad de operandos

(resolución de dependencias)

1. Cuando los operandos se cargan desde el fichero de registros hay que comprobar si los operandos están disponibles en los registros
2. Cuando se envía ('dispatch') una instrucción a las unidades funcionales los operandos de la instrucción almacenada deben estar disponibles.

Uso de Marcas ('*Scoreboarding*')

CDC6600 (1964)

- Cada registro del fichero de registros se amplía con un bit que indica si el dato almacenado en el registro es válido (está disponible) o no.
- Cuando se emite una instrucción el bit de validez del registro destino se pone a '0' indicando la no disponibilidad del dato hasta que, al término de la operación correspondiente se escribe el resultado (se pone a 1).

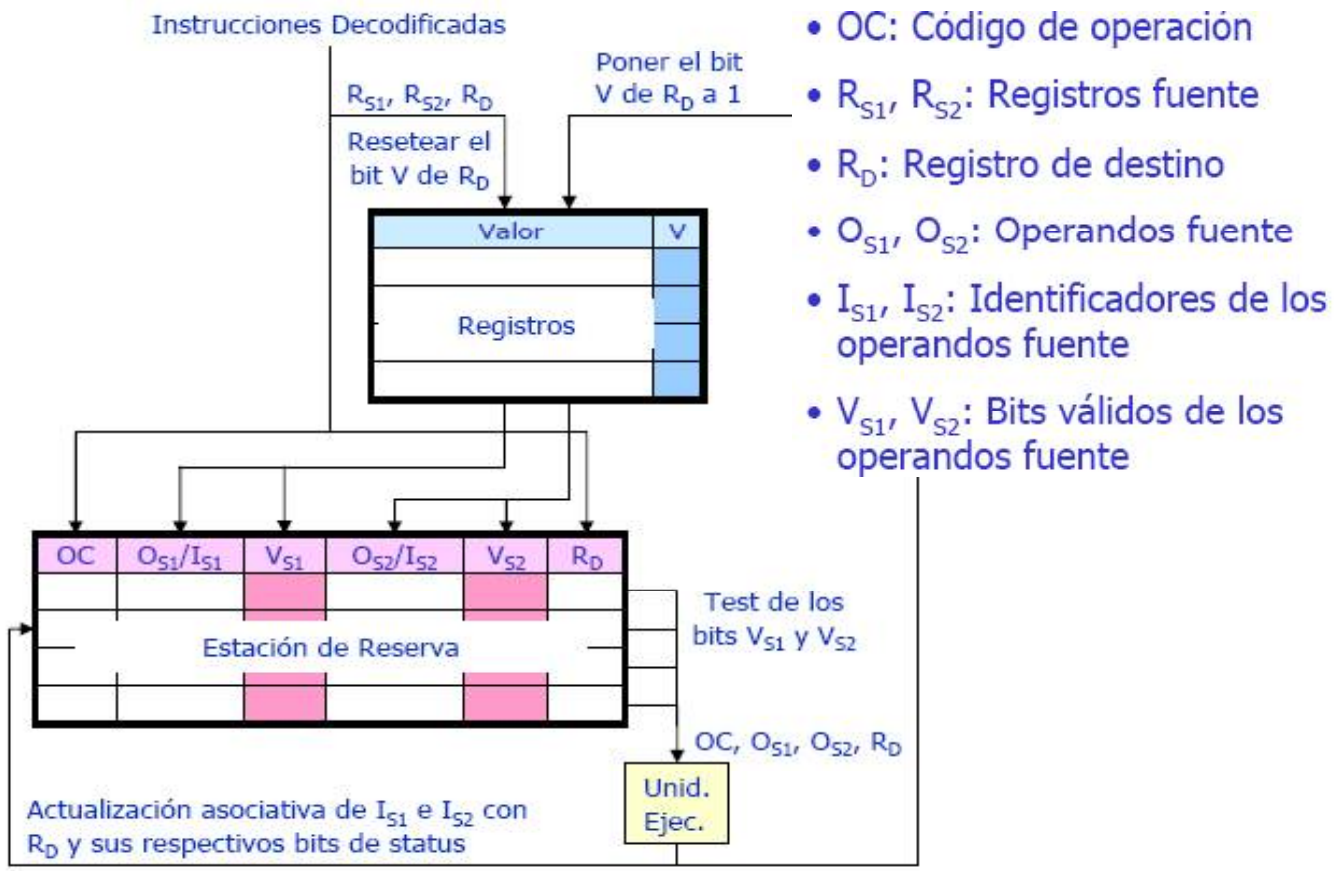
Esquemas para comprobar la disponibilidad de los operandos

- Comprobación directa de los bits de validez en el fichero de registros
- Comprobación de bits de estado explícitos en la ventana o estación de reserva.

Estaciones de Reserva: Esquemas de Comprobación de operandos

– Comprobación en la estación de reserva

– Comprobación en el banco de registros

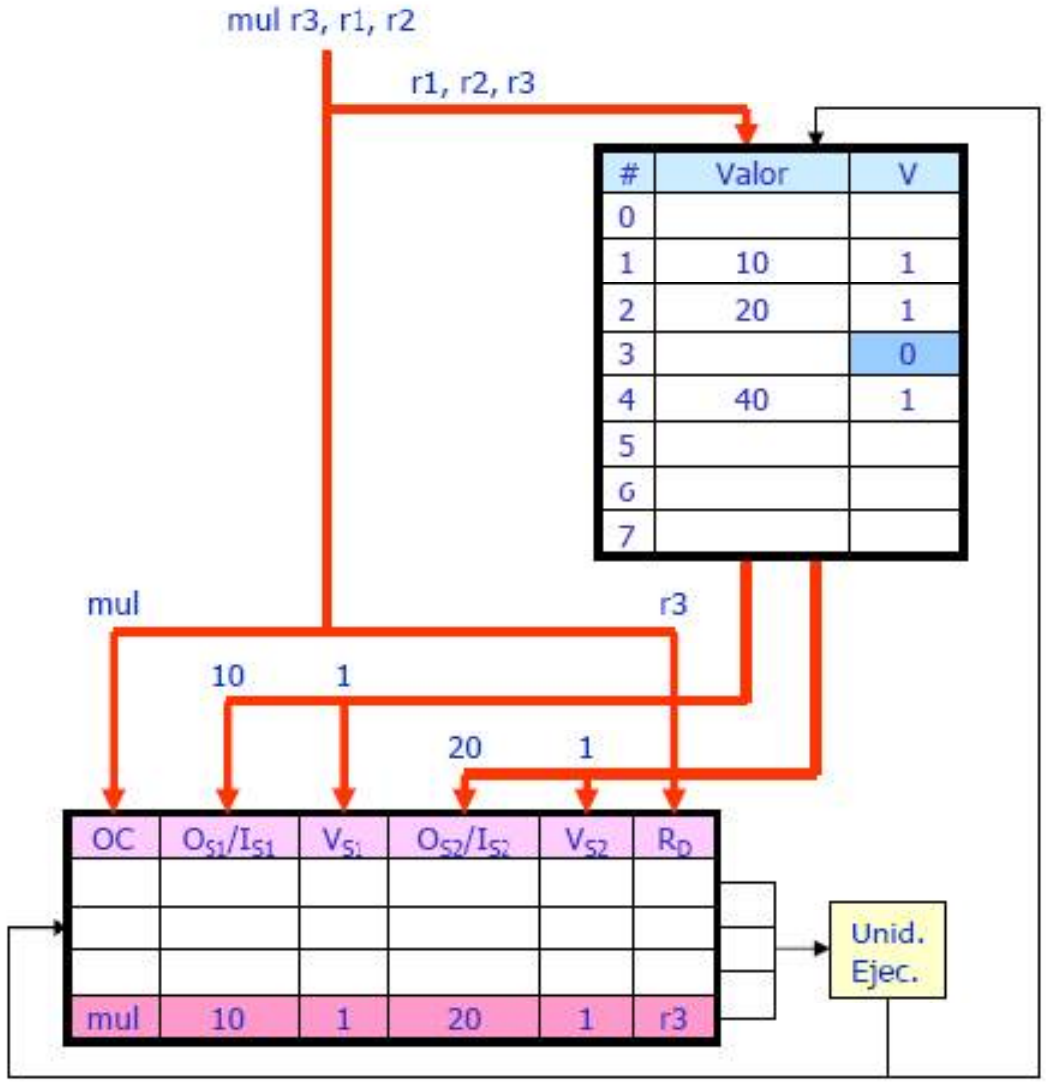


Estaciones de Reserva: Ejemplo de uso (i)

```
Ciclo i:      mul r3, r1, r2
Ciclo i+1:    add r5, r2, r3
              add r6, r3, r4
```

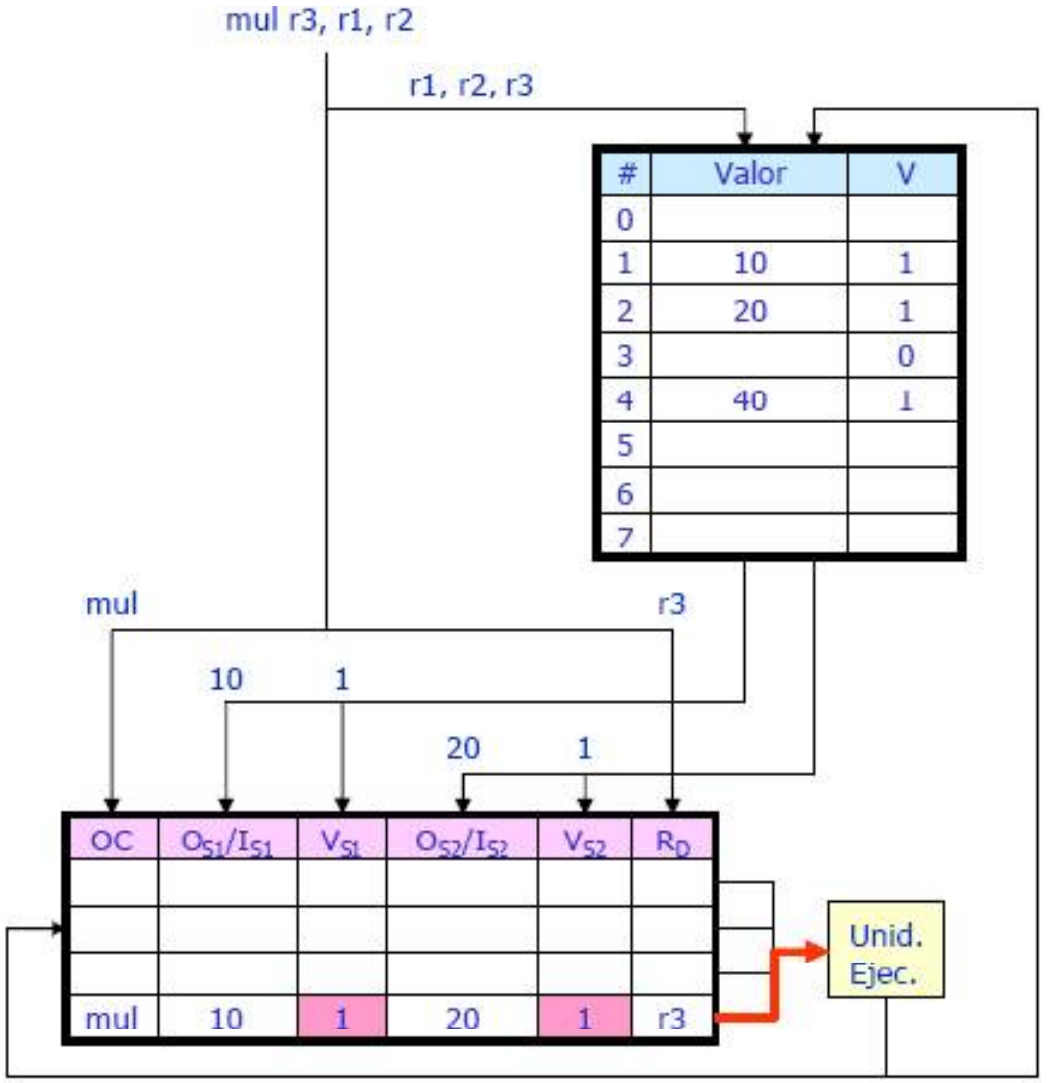
Ciclo i:

- Se emite la instrucción de multiplicación, ya decodificada, a la estación de reserva
- Se anula el valor de r3 en el banco de registros
- Se copian los valores de r1 y r2 (disponibles) en la estación de reserva



Estaciones de Reserva: Ejemplo de uso (ii)

Ciclo i:	mul r3, r1, r2
Ciclo i+1:	add r5, r2, r3
	add r6, r3, r4



Ciclo i + 1:

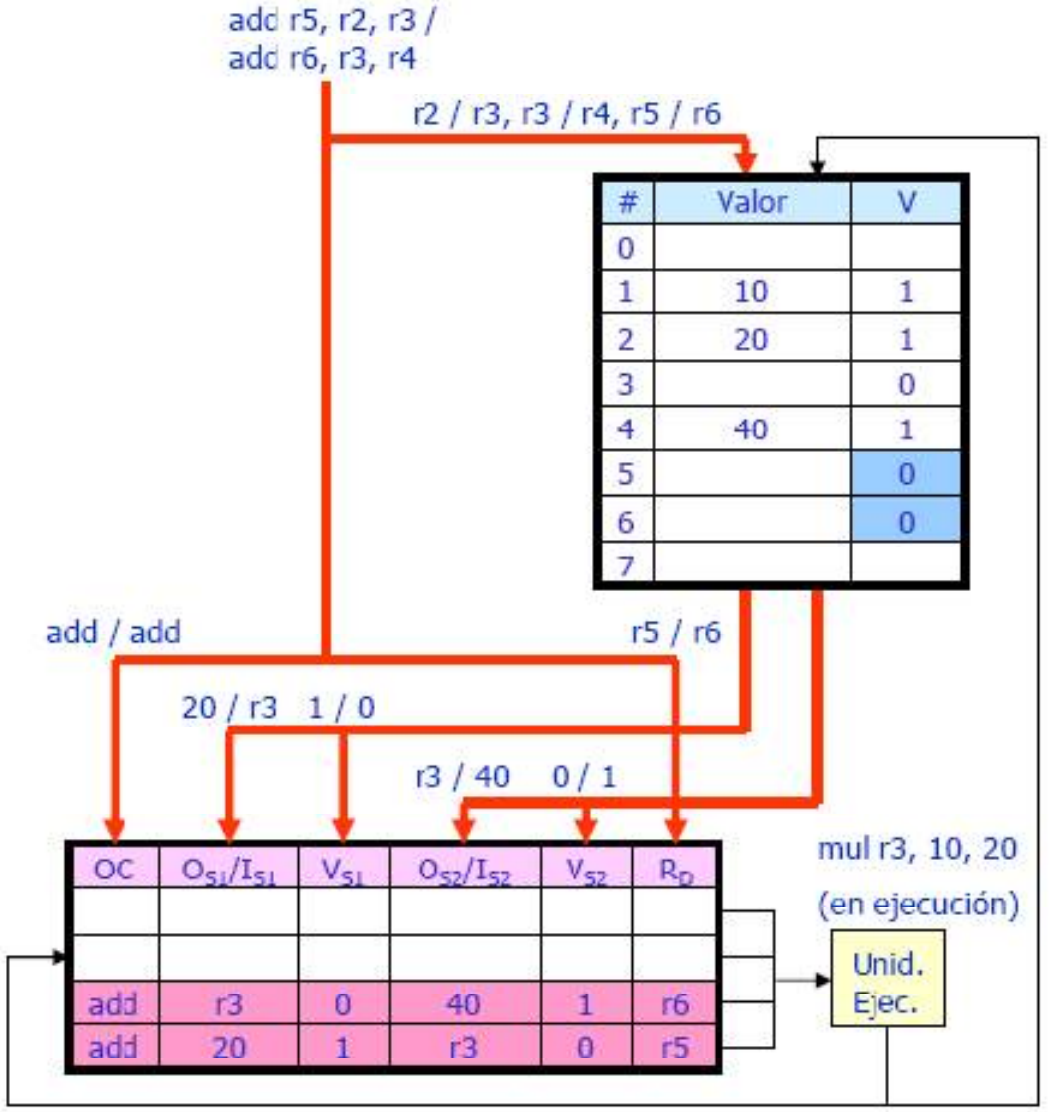
- La operación de multiplicación tiene sus operadores preparados (V_{S1} = 1 y V_{S2} = 1)
- Así que puede enviarse a la unidad de ejecución

Estaciones de Reserva: Ejemplo de uso (iii)

```
Ciclo i:     mul r3, r1, r2
Ciclo i+1:   add r5, r2, r3
             add r6, r3, r4
```

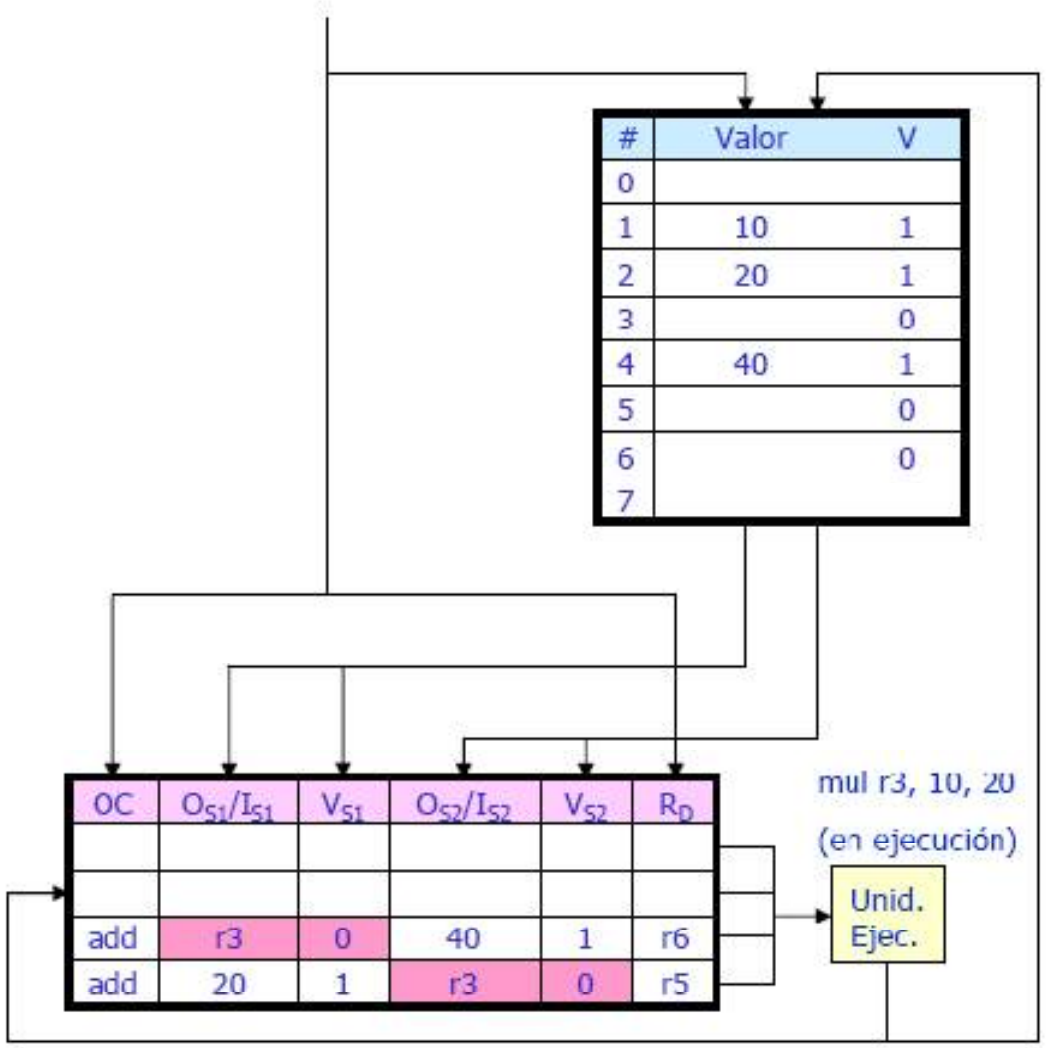
Ciclo i + 1 (*cont.*):

- Se emiten las dos instrucciones de suma a la estación de reserva
- Se anulan los valores de r5 y r6 en el banco de registros
- Se copian los valores de los operandos disponibles y los identificadores de los operandos no preparados



Estaciones de Reserva: Ejemplo de uso (iv)

```
Ciclo i:     mul r3, r1, r2
Ciclo i+1:   add r5, r2, r3
             add r6, r3, r4
```



Ciclos i + 2 .. i + 5:

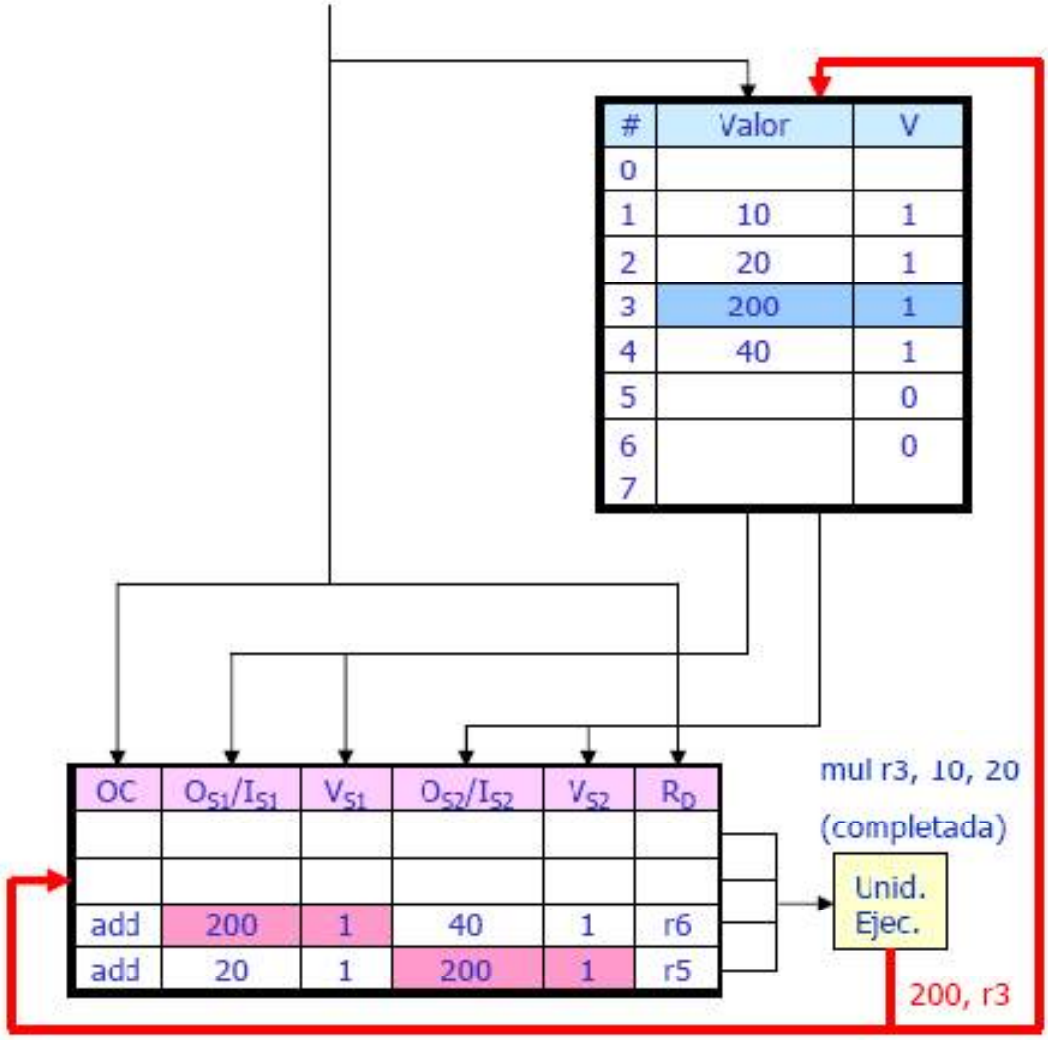
- La multiplicación sigue ejecutándose
- No se puede ejecutar ninguna suma hasta que esté disponible el resultado de la multiplicación (r3)

Estaciones de Reserva: Ejemplo de uso (v)

```
Ciclo i:     mul r3, r1, r2
Ciclo i+1:   add r5, r2, r3
             add r6, r3, r4
```

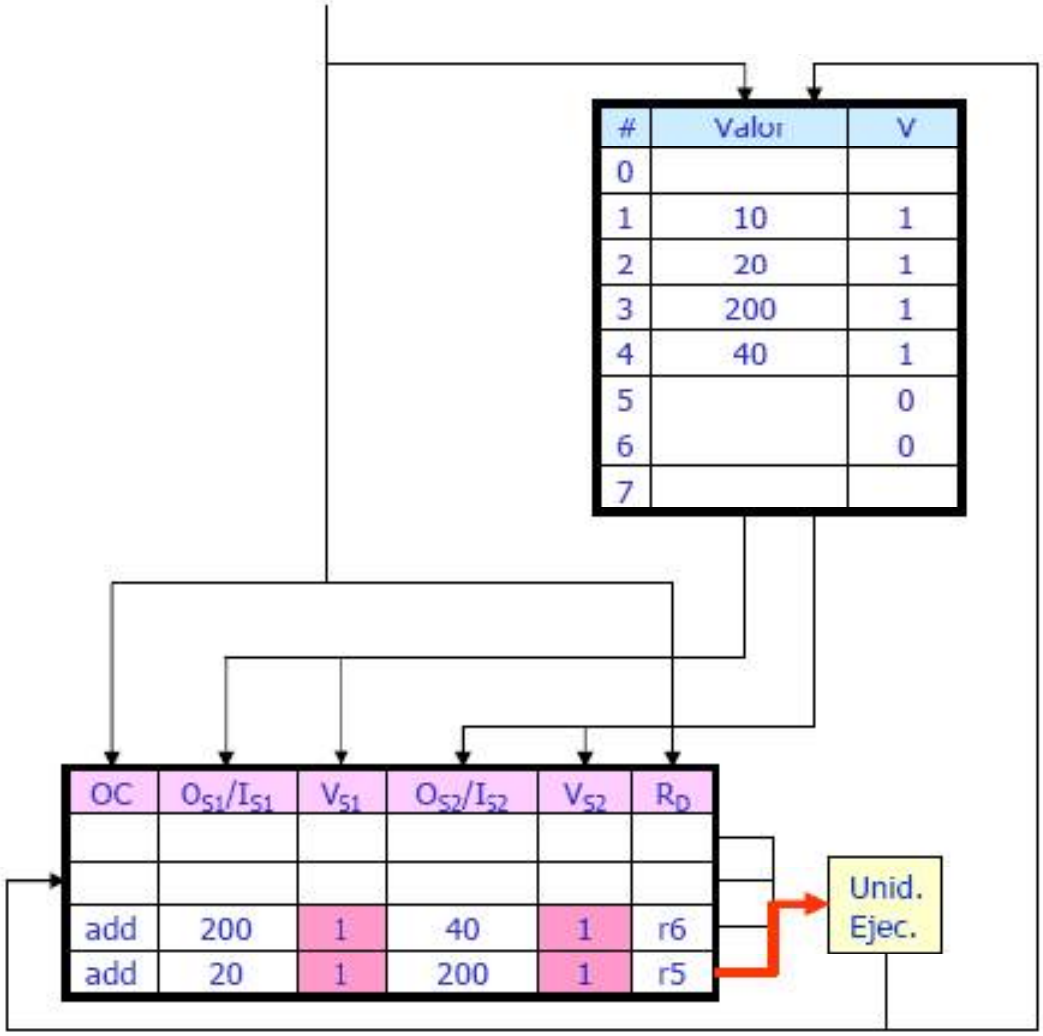
Ciclo i + 6:

- Se escribe el resultado de la multiplicación en el banco de registros y en las entradas de la estación de reserva
- Se actualizan los bits de disponibilidad de r3 en el banco de registros y en la estación de reserva



Estaciones de Reserva: Ejemplo de uso (vi)

```
Ciclo i:      mul r3, r1, r2
Ciclo i+1:    add r5, r2, r3
              add r6, r3, r4
```

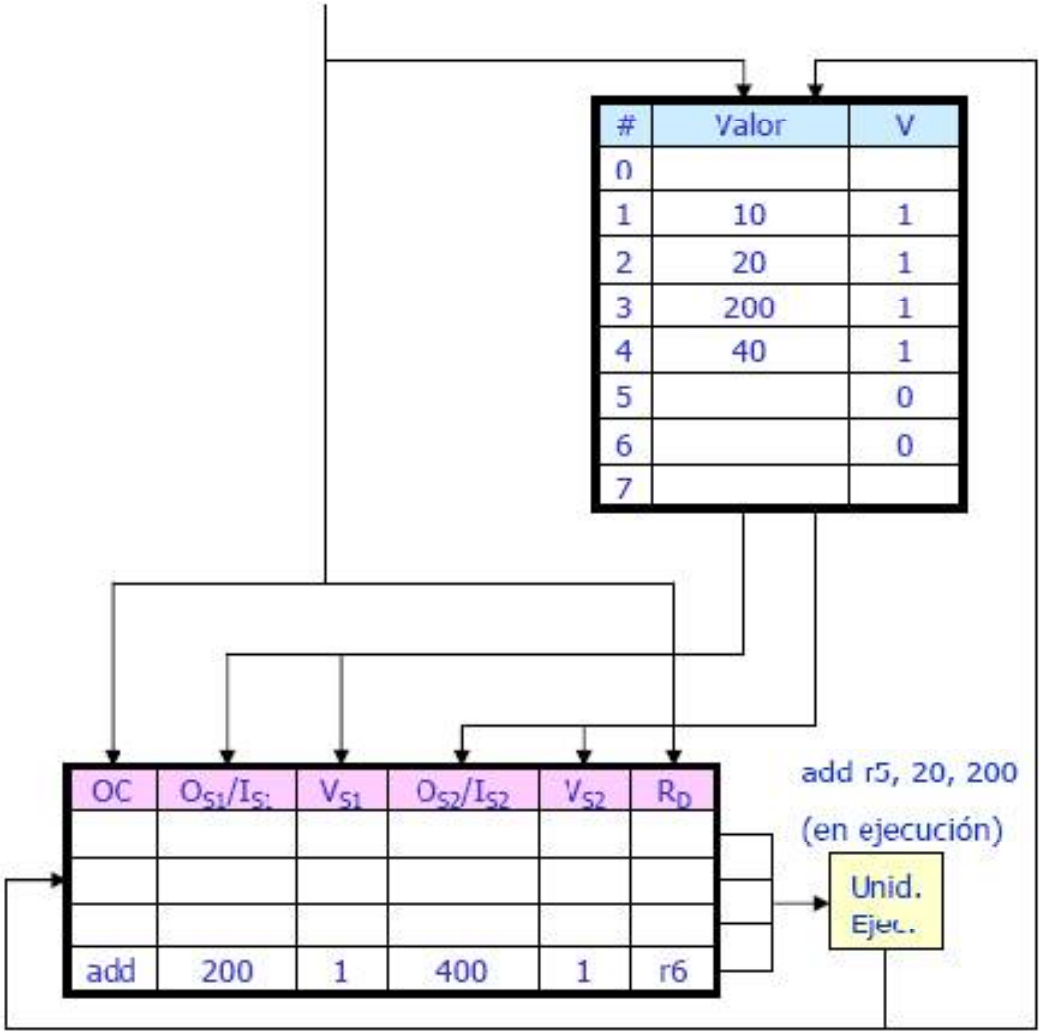


Ciclo i + 6 (*cont.*):

- Las sumas tienen sus operadores preparados (VS1 = 1 y VS2 = 1)
- Así que pueden enviarse a la unidad de ejecución

Estaciones de Reserva: Ejemplo de uso (vii)

```
Ciclo i:     mul r3, r1, r2
Ciclo i+1:   add r5, r2, r3
             add r6, r3, r4
```



Ciclo i + 6 (*cont.*):

- Como sólo hay una unidad de ejecución, se envía la instrucción más antigua de la estación de reserva, la primera suma