

4.3 Cauces Aritméticos

Tema4. Segmentación

Arquitectura de los Computadores

Cauces aritméticos

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

4.3. CAUCES ARITMÉTICOS

- 4.3.1. Sumadores con salvaguarda de acarreo
- 4.3.2. Multiplicación encauzada
- 4.2.4. Ejemplo

Sumadores con salvaguarda de acarreo (CSA)

Introducción

Segmentación
del repertorio

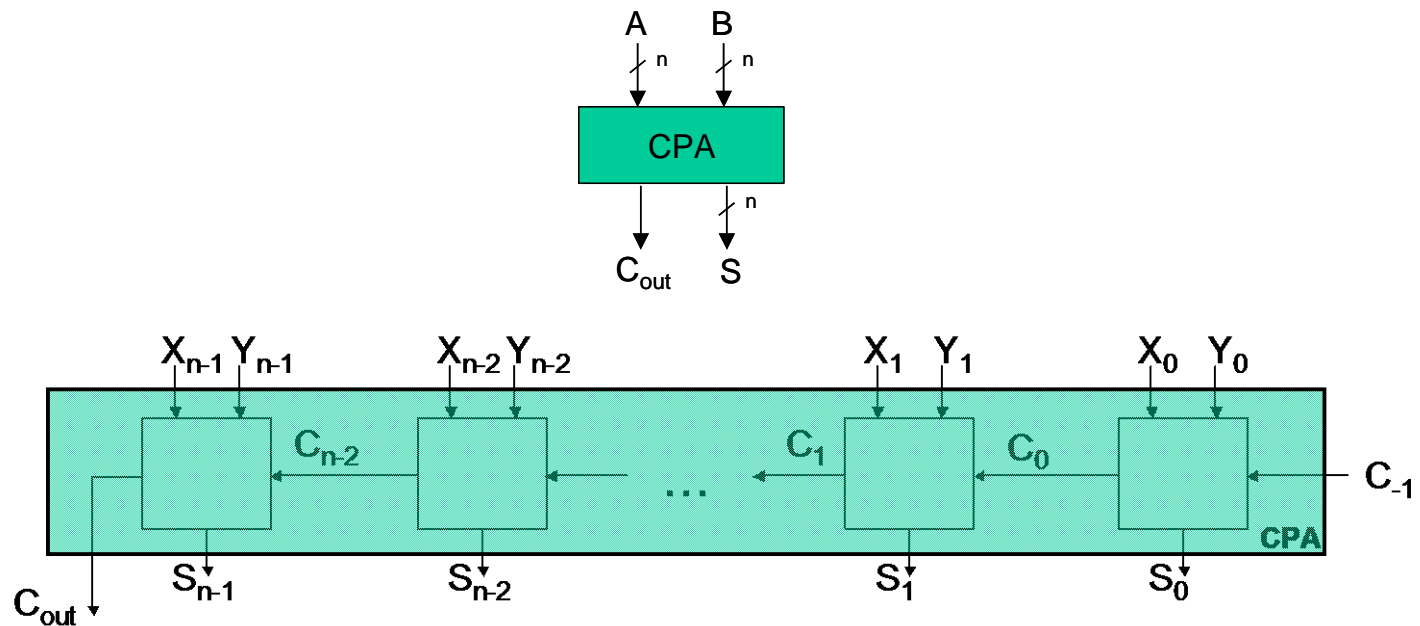
Cauces
aritméticos

Optimización

Superescalares

Segmentación

- ◆ **Sumador con propagación de acarreo (CPA).** Realiza la suma de dos números de N bits propagando los acarreos de un bit al siguiente.



Sumadores con salvaguarda de acarreo (CSA)

Introducción

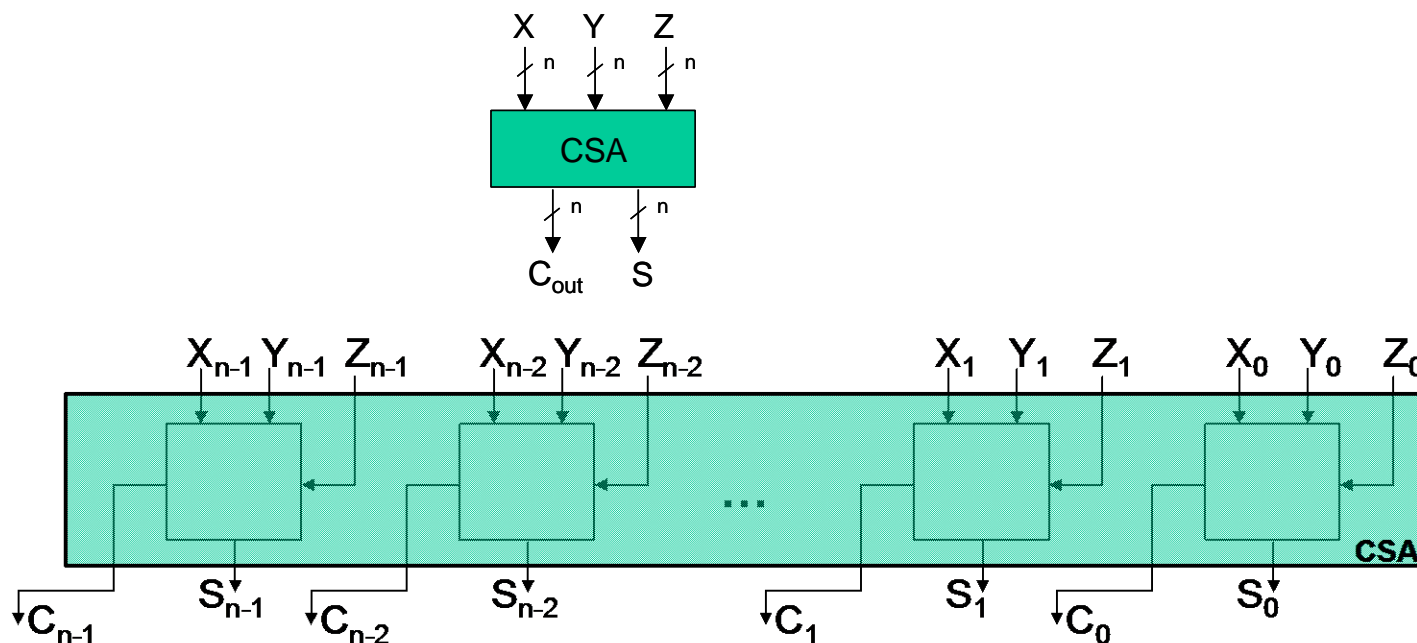
Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

- Sumador con salvaguarda de acarreos (CSA). Realiza la suma de tres números de N bits y da como resultado la suma de los bits sin considerar acarreos y un vector de acarreos.



Segmentación

$$S_i = x_i \oplus y_i \oplus z_i$$
$$C_{i+1} = x_i y_i \text{ OR } y_i z_i \text{ OR } z_i x_i$$

Sumadores con salvaguarda de acarreo (CSA)

Introducción

Segmentación
del repertorio

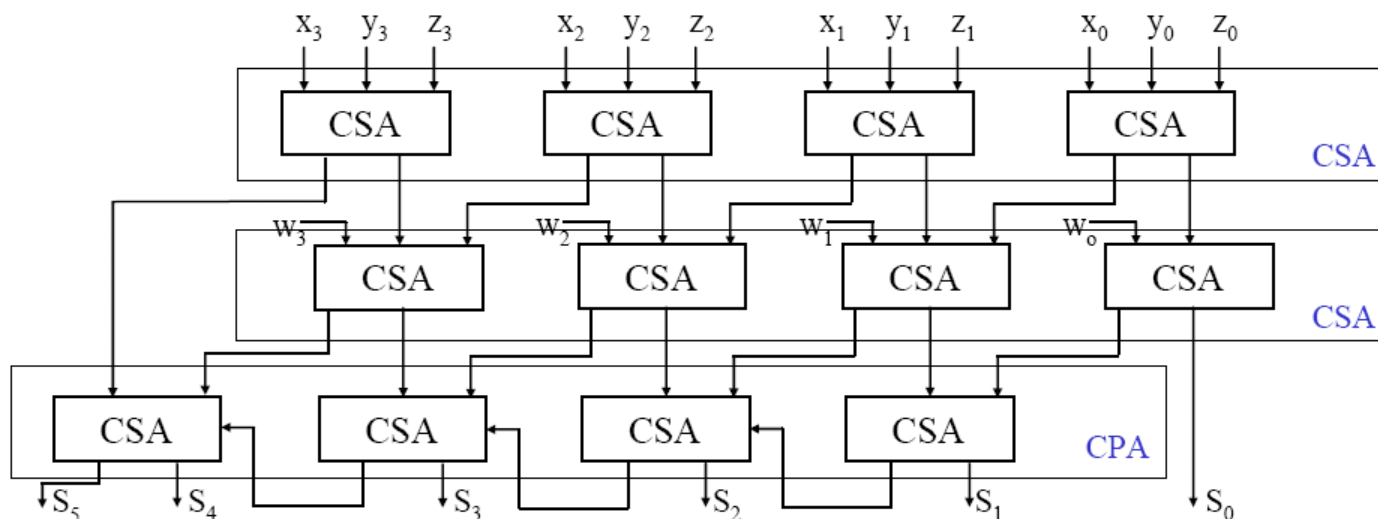
Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Objetivo: Acelerar la suma cuando se tienen que sumar mas de dos operandos
- Sólo en la última suma habrá que propagar los acarreos.
- Ej: diseño de un sumador de 4 operandos de 4 bits.



CSA: Carry Save Adder
CPA: Carry Propagate Adder

Sumadores con salvaguarda de acarreo (CSA)

Introducción

Segmentación
del repertorio

Cauces
aritméticos

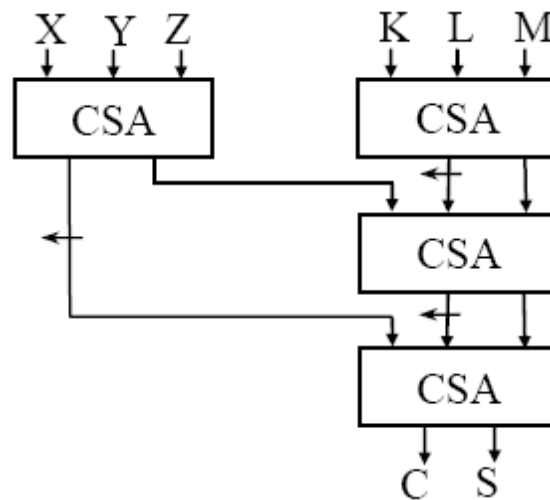
Optimización

Superescalares

Segmentación

- Árboles de Wallace: Es otra forma de organizar los CSA para tratar de mejorar el rendimiento.

Arbol de Wallace con 6 operandos



Número de niveles en un árbol de Wallace para k operandos

Nº de operandos	Nº de niveles
3	1
4	2
$5 \leq k \leq 6$	3
$7 \leq k \leq 9$	4
$10 \leq k \leq 13$	5
$14 \leq k \leq 19$	6
$20 \leq k \leq 28$	7
$29 \leq k \leq 42$	8
$43 \leq k \leq 63$	9

- Introducción
- Segmentación del repertorio
 - Cauces aritméticos
- Optimización
- Superescalares

Segmentación

- 🌲 Por medio de un árbol de Wallace podemos implementar un multiplicador

						a₅	a₄	a₃	a₂	a₁	a₀	=A
					x	b₅	b₄	b₃	b₂	b₁	b₀	=B
						a₅b₀	a₄b₀	a₃b₀	a₂b₀	a₁b₀	a₀b₀	=P₁
						a₅b₁	a₄b₁	a₃b₁	a₂b₁	a₁b₁	a₀b₁	=P₂
				a₅b₂	a₄b₂	a₃b₂	a₂b₂	a₁b₂	a₀b₂			=P₃
		a₅b₃	a₄b₃	a₃b₃	a₂b₃	a₁b₃	a₀b₃					=P₄
	a₅b₄	a₄b₄	a₃b₄	a₂b₄	a₁b₄	a₀b₄						=P₅
+	a₅b₅	a₄b₅	a₃b₅	a₂b₅	a₁b₅	a₀b₅						=P₆
P₁₁	P₁₀	P₉	P₈	P₇	P₆	P₅	P₄	P₃	P₂	P₁	P₀	=P

Multiplicador de números de 8 bits

Introducción

Segmentación
del repertorio

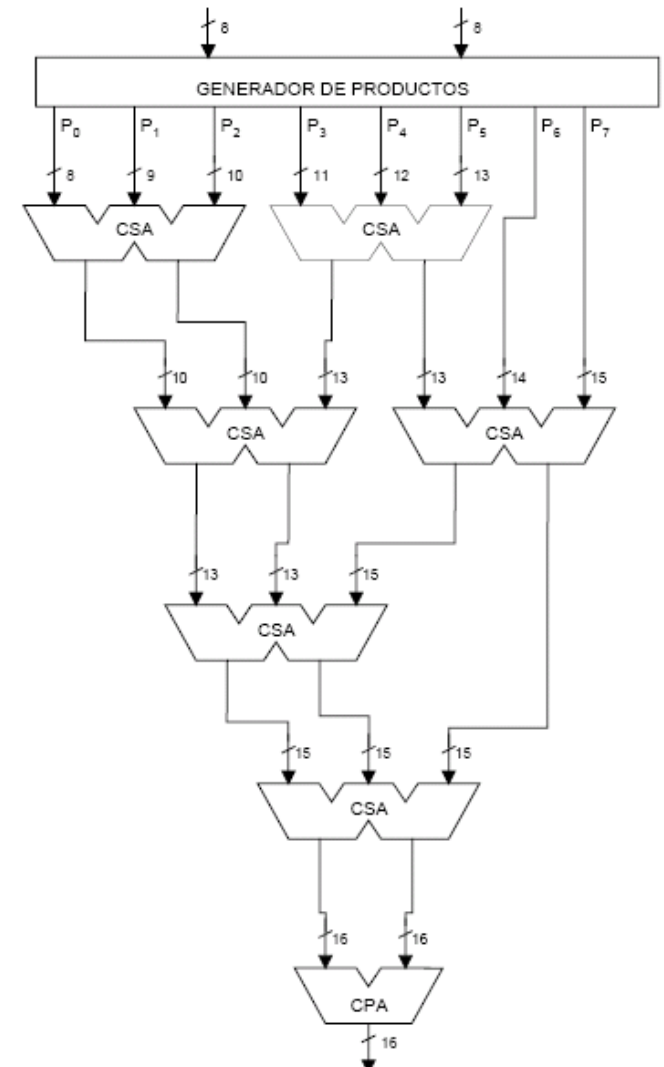
Cauces
aritméticos

Optimización

Superescalares

Segmentación

- El generador de productos genera todos los productos ya desplazados
- Se suman convenientemente a través de etapas CSA
- La última etapa es un CPA o CLA



Multiplicación encauzada

Introducción

Segmentación
del repertorio

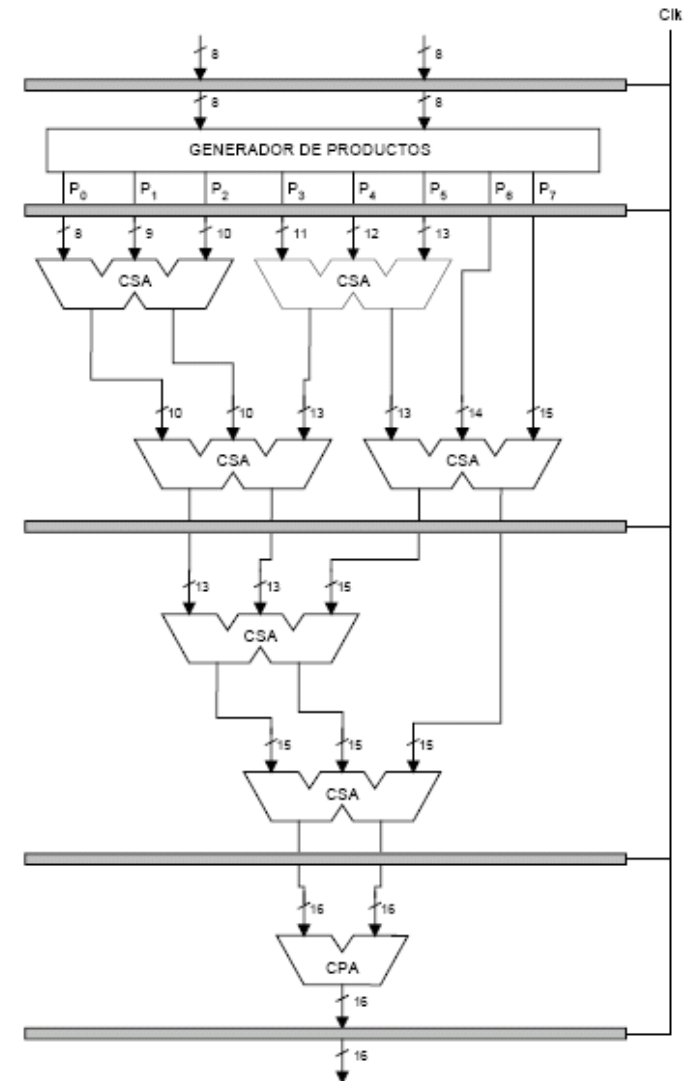
Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Para la segmentación del multiplicador basta definir las etapas e intercalar los registros que guarden los resultados intermedios.



Algunos ejemplos: IBM 360/91

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- El IBM sistema 360 modelo 91 de mediados de 1960
- Utiliza dos unidades de ejecución de coma flotante que pueden operar concurrentemente:
 - La unidad de suma de dos etapas segmentadas.
 - La unidad de multiplicación/división con un árbol segmentado de 6 etapas
 - La división se realiza por multiplicaciones repetidas (división por convergencia)

Algunos ejemplos: IBM 360/91

Introducción

Segmentación
del repertorio

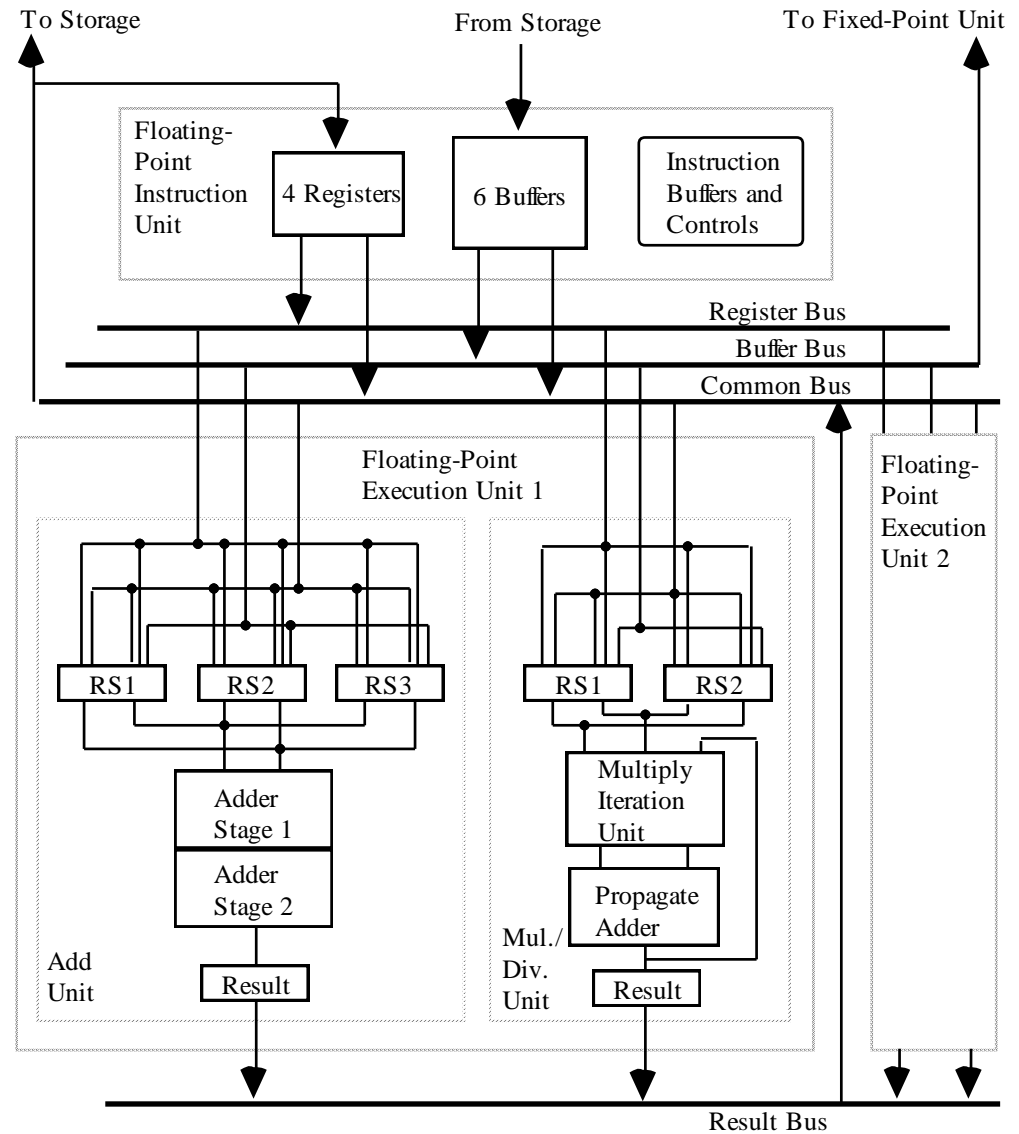
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Estructura de la unidad de ejecución en coma flotante del IBM 360/91



Algunos ejemplos: TI-ASC

Introducción

Segmentación
del repertorio

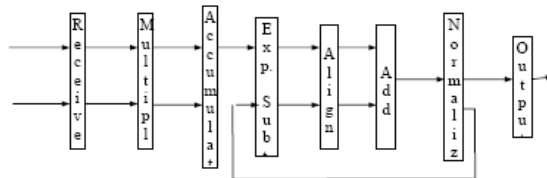
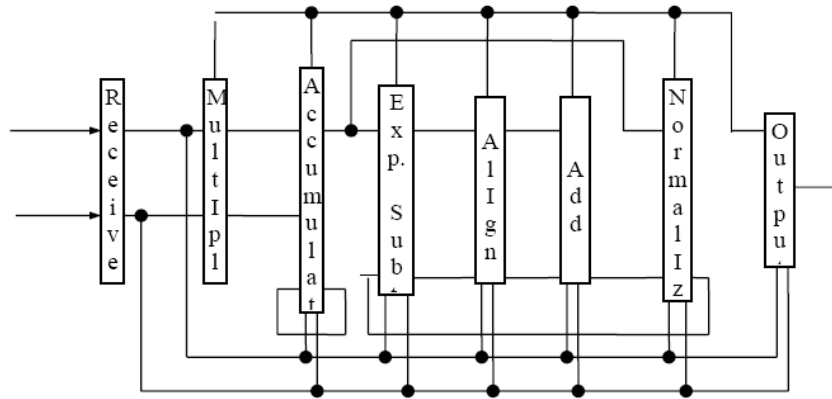
Cauces
aritméticos

Optimización

Superescalares

Segmentación

- TI-ASC (Texas Instruments Advanced Scientific Computer):
 - primer procesador vectorial instalado con cauces multifunción. La segmentación aritmética del ASC consiste en 8 etapas. Puede realizar funciones aritméticas en coma fija o coma flotante y muchas operaciones de desplazamientos lógicos sobre operandos escalares y vectores de longitud 16, 32 o 64 bits.



Multiplicación en coma flotante



Suma en coma fija

Algunos ejemplos: MC68040

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- La unidad de coma flotante del MC68040 consta de un cauce segmentado de tres etapas:
 - La sección de la mantisa y la sección del exponente son esencialmente dos cauces segmentados separados de 3 etapas cada uno.
 - La sección de la mantisa puede realizar operaciones de suma o multiplicación en coma flotante, en simple y doble precisión.
 - La sección de la mantisa, etapa 1 utiliza registros de 64 bits. La etapa 2 contiene un multiplicador de array que puede utilizarse repetidamente para realizar multiplicaciones largas de las mantisas. La etapa 3 contiene registros para almacenar mantener los resultados antes de guardarlos en los registros de la etapa 1 para ser utilizados por otras operaciones.
 - En la sección del exponente, la etapa1 utiliza un sumador para comparar la magnitud relativa de los exponentes. En la etapa 3 se reajusta el exponente utilizando otro sumador . El valor final se lleva de los registros de etapa 3 a los de la etapa 1, preparados para ser utilizados de nuevo.

4.4 Optimización de unidades segmentadas

Tema 4. Segmentación

Arquitectura de los Computadores

Optimización de unidades segmentadas

Introducción

Segmentación
del repertorio




Cauces
aritméticos

Optimización

Superescalares

Segmentación

4.4. OPTIMIZACIÓN DE UNIDADES SEGMENTADAS

-  4.4.1. Introducción
-  4.4.2. Tablas de reserva
-  4.4.3. Análisis de diagramas de estado

Introducción

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

➤ Cauce lineal:

- cada etapa está conectada sólo con una etapa anterior y otra posterior. Las entradas a procesar (instrucciones, operandos,...) se pueden introducir en el cauce al ritmo de una cada ciclo de reloj.

➤ Cauce no lineal:

- Hay etapas que necesitan varios ciclos de reloj.
- Algunas etapas se vuelven a reutilizar por una misma operación.
- Una misma operación (instrucción,...) puede utilizar más de una etapa al mismo tiempo.
- El orden en que se visitan las etapas y las etapas que se visitan puede cambiar de una operación a otra (cauces multifuncionales)
- Pueden existir dependencias entre las operaciones que se introducen en el cauce, de forma que el orden en que una operación visite las etapas cambie dinámicamente (cauces dinámicos multifuncionales)

Introducción

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Consecuencias de la no linealidad:
 - No se pueden introducir datos en cada ciclo de reloj; es necesario sincronizar la entrada de datos con el flujo interno para evitar colisiones.
- Tabla de reserva:
 - Describe el uso que una operación, instrucción,...hace de las distintas etapas del cauce en su paso por el mismo.
 - Permite determinar el tiempo de latencia de la operación, instrucción,...

Tablas de reserva

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Contiene información sobre la ocupación de cada etapa en cada uno de los ciclos a medida que un elemento progresa por el cauce.

Ciclos de reloj →

Etapas ↓

	1	2	3	4	5	6
A	X			X		
B		X				
C			X		X	X

- Las filas se corresponden con las etapas del proceso
- Las columnas indican el número de ciclos necesarios para completar la tarea
- Múltiples marcas en una fila indican una utilización repetida de la misma etapa en diferentes ciclos

Tablas de reserva

Introducción

Segmentación
del repertorio

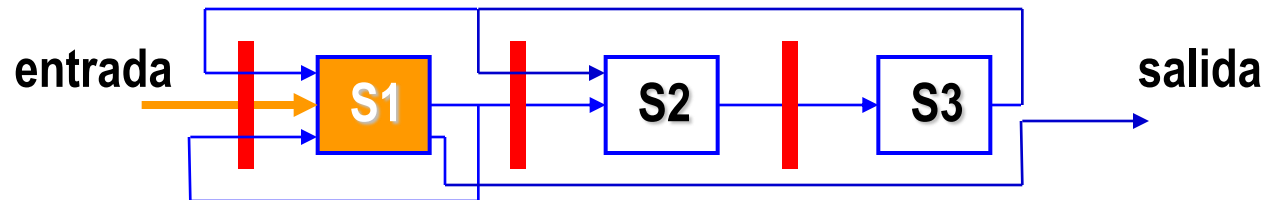
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Máquina X



Ciclos →

etapas →

	0	1	2	3	4	5	6	7
S1	X							
S2								
S3								

Tablas de reserva

Introducción

Segmentación
del repertorio

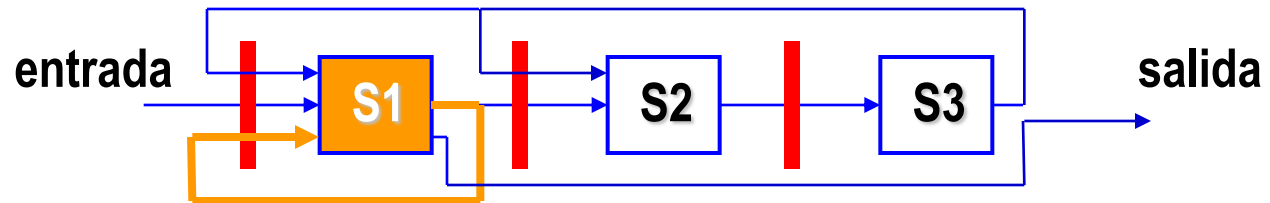
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Máquina X



Ciclos →

etapas →

	0	1	2	3	4	5	6	7
S1	X	X						
S2								
S3								

Tablas de reserva

Introducción

Segmentación
del repertorio

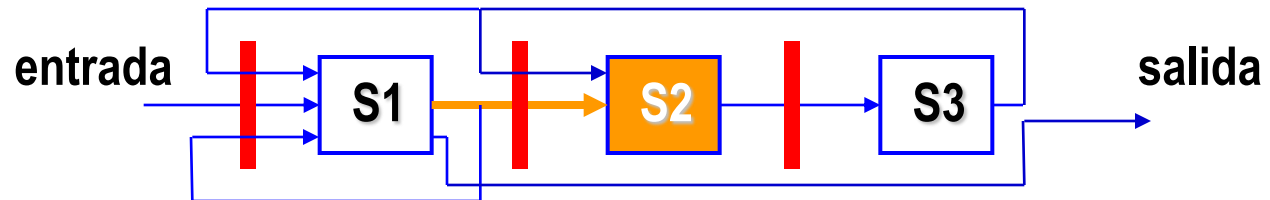
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Máquina X



Ciclos →

etapas →

	0	1	2	3	4	5	6	7
S1	X	X						
S2			X					
S3								

Tablas de reserva

Introducción

Segmentación
del repertorio

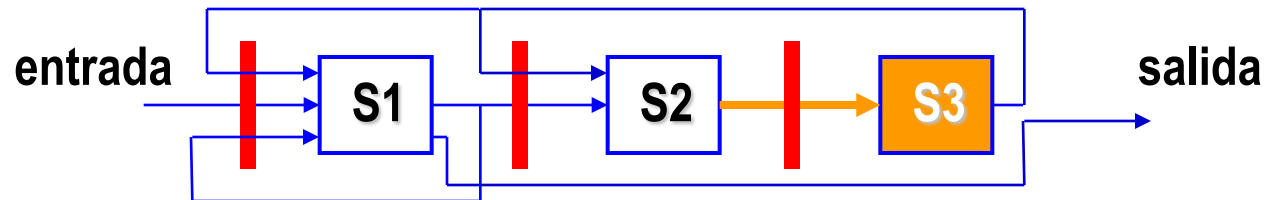
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Máquina X



Ciclos →

etapas →

	0	1	2	3	4	5	6	7
S1	X	X						
S2			X					
S3				X				

Tablas de reserva

Introducción

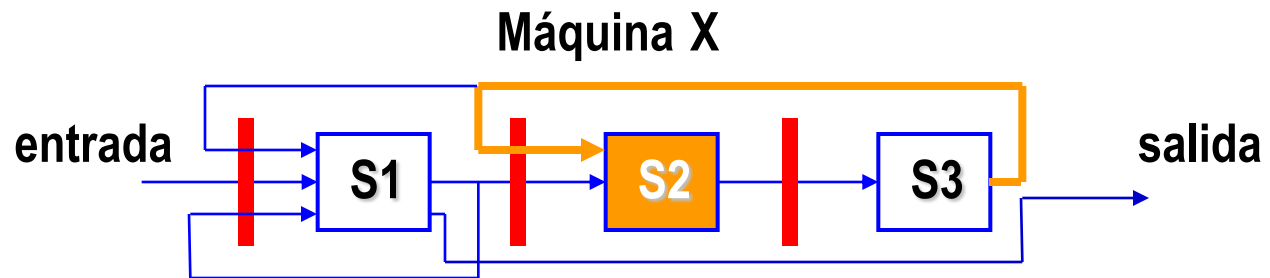
Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superscalares

Segmentación



Ciclos →

etapas →

	0	1	2	3	4	5	6	7
S1	X	X						
S2			X		X			
S3				X				

Tablas de reserva

Introducción

Segmentación
del repertorio

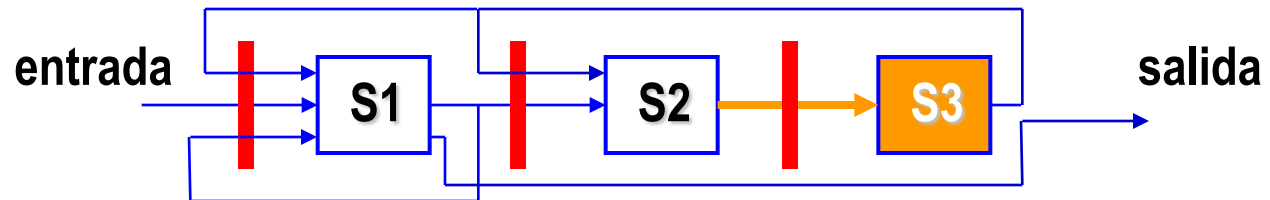
Cauces
aritméticos

Optimización

Superescales

Segmentación

Máquina X



Ciclos →

etapas →

	0	1	2	3	4	5	6	7
S1	X	X						
S2			X		X			
S3				X		X		

Tablas de reserva

Introducción

Segmentación
del repertorio

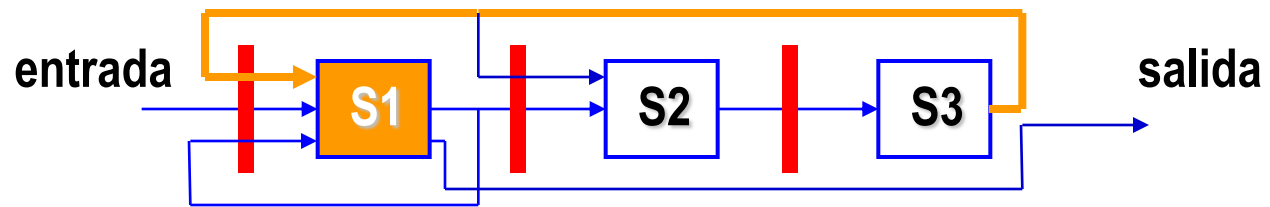
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Máquina X



Ciclos →

etapas →

	0	1	2	3	4	5	6	7
S1	X	X					X	
S2			X		X			
S3				X		X		

Tablas de reserva

Introducción

Segmentación
del repertorio

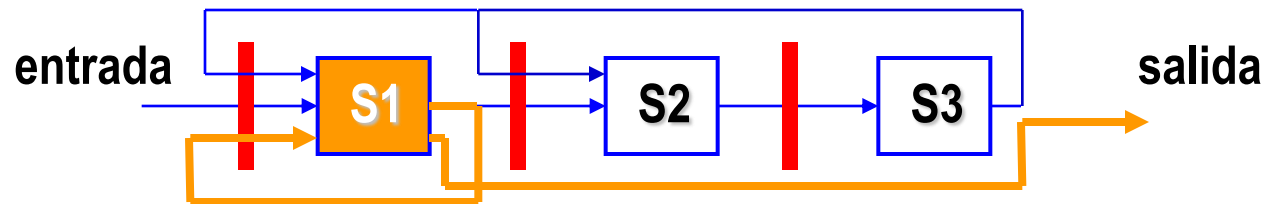
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Máquina X



Ciclos →

etapas →

	0	1	2	3	4	5	6	7
S1	X	X					X	X
S2			X		X			
S3				X		X		

Tablas de reserva

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Definiciones:

- ⚙️ Tiempo de computo: Unidades de tiempo para completar una iteración de la tabla de reserva.
- ⚙️ Latencia: número de ciclos entre dos iniciaciones consecutivas
- ⚙️ Una latencia $k \rightarrow$ dos iniciaciones separadas por k ciclos
- ⚙️ Colisión: conflicto de recursos entre dos iniciaciones
- ⚙️ Latencias que causan colisión \rightarrow latencias prohibidas

Tablas de reserva

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

🏠 Ejemplo: Colisión con latencias 2 y 5 al iniciar X2

Colisión con latencia 2

S1	X1		X2			X1		X2 X1
S2		X1		X2 X1		X2		
S3			X1		X2 X1		X2 X1	

Colisión con latencia 5

S1	X1					X2 X1		X1
S2		X1		X1			X2	
S3			X1		X1		X1	X2

Latencias Prohibidas: 2, 5

Tablas de reserva

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Definiciones:

- Seuencia de latencia: secuencia de latencias permitidas entre sucesivas iteraciones. **Ejemplo: 3,8**
- Ciclo de latencia: Una secuencia de latencia que se repite indefinidamente. **Ejemplo: (3,8) → 3,8,3,8,3,8,....**
- Latencia media (o de hecho productividad): unidades de tiempo promedio entre iniciaciones.
- Mínima Latencia media (MAL) (=máxima productividad) : latencia media mínima más pequeña obtenida mediante una estrategia de control.

Vector de colisión

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Vector binario cuya componente i es **1** si, y sólo si, se produce una colisión cuando se arranca la tarea la segunda vez i ciclos después de haberse iniciado la primera.
- Se utiliza el vector de colisiones para planificar las tareas en el pipeline.

$$C = (C_1, C_2, \dots, C_{m-1}, C_m)$$

$C_i = 1$ si la latencia i causa colisión (latencia prohibido)

$C_i = 0$ si la latencia i se permite

$C_m = 1$ (siempre) latencia máxima prohibida

Latencia máxima prohibida : $m \leq n-1$

n = número de columnas en la tabla de reserva

Vector de colisión

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

■ Ejemplo:

■ Latencia prohibida: 2, 4, 5, 7

■ Vector de colisiones: 0101101

■ Latencia prohibida: 2, 4

■ Vector de colisiones: 0101

Vector de colisión

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Obtención del vector de colisión al iniciar la segunda tarea 2 de la tabla de reserva de la máquina X

i = 0	0	1	2	3	4	5	6	7
S1	1 2	1 2					1 2	1 2
S2			1 2		1 2			
S3				1 2		1 2		
i = 1	0	1	2	3	4	5	6	7
S1	1	1 2	2				1	1 2
S2			1	2	1	2		
S3				1	2	1	2	
			Vector de Colisión = 1xxxxxx					
i = 2	0	1	2	3	4	5	6	7
S1	1	1	2	2			1	1
S2			1		1 2		2	
S3				1		1 2		2
			Vector de Colisión = 11xxxxx					
i = 3	0	1	2	3	4	5	6	7
S1	1	1		2	2		1	1
S2			1		1	2		2
S3				1		1	2	

Vector de Colisión = 110xxxx

Vector de colisión

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Obtención del vector de colisión al iniciar la segunda tarea 2 de la tabla de reserva de la máquina X

i = 4	0	1	2	3	4	5	6	7
S1	1	1			2	2	1	1
S2			1		1		2	
S3				1		1		2
Vector de Colisión = 1100xxx								
i = 5	0	1	2	3	4	5	6	7
S1	1	1				2	1 2	1
S2			1		1			2
S3				1		1		
Vector de Colisión = 11001xx								
i = 6	0	1	2	3	4	5	6	7
S1	1	1					1 2	1 2
S2			1		1			
S3				1		1		
Vector de Colisión = 110011x								
i = 7	0	1	2	3	4	5	6	7
S1	1	1					1	1 2
S2			1		1			
S3				1		1		

Vector de Colisión = 1100111

Vector de colisión

Ejemplo: Obtened el vector de colisión para la tabla de reserva de la figura

	1	2	3	4	5	6
A	X					X
B		X				X
C			X			
D				X	X	

Vector de Colisión = 10011

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Planificación de la tarea

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Planificar las tareas de la máquina segmentada:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S1	1	1					1	1														
S2			1		1																	
S3				1		1																

❖ ¿Cuándo iniciar la próxima tarea sin colisiones?

❖ ¿Cómo conseguir la máxima productividad?

Planificación de la tarea

Introducción

Segmentación
del repertorio


Cauces
aritméticos

Optimización

Superescalares

Segmentación

Planificar las tareas de la máquina segmentada:




	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S1	1	1		2	2		1	1		2	2	3	3		4	4		3	3		4	4
S2			1		1	2		2						3		3	4		4			
S3				1		1	2		2						3		3	4		4		

Estrategia *avara* (*Greedy strategy*)

Secuencia de latencias = $\langle 3, 8, 3, 8, \dots \rangle$

Latencia media = $(3+8)/2 = 5.5$ ciclos



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S1	1	1			2	2	1	1	3	3	2	2	4	4	3	3	5	5	4	4	6	6
S2			1		1		2		2		3		3		4		4		5		5	
S3				1		1		2		2		3		3		4		4		5		5

Iniciación óptima

Secuencia de latencias = $\langle 4, 4, 4, 4, \dots \rangle$

Latencia media = 4 ciclos

Diagrama de estado

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Los diagramas de estado se utilizan para mostrar los diferentes estados de un pipeline en un instante determinado.
- Una vez se tiene el diagrama de estados, es fácil planificar la entrada de los datos al cauce segmentado para que no tengan colisiones.
- Para crear el diagrama de estados, el estado inicial es siempre el vector de colisión inicial

Diagrama de estado

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

1100111

Vector de colisión inicial

- Un 0 en el vector indica una latencia permitida: es decir, se puede iniciar una nueva operación
- esta nueva operación puede hacer no permitidas algunas latencias más: el cauce cambia de estado
- el nuevo vector de colisión se obtiene desplazando el inicial a la izquierda tantas componentes como la latencia que se ha utilizado introduciendo ceros por la derecha, y haciendo la OR del resultado con el vector inicial
- los estados del cauce se representan por los vectores de colisión del cauce cuando se encuentra en ese estado
- los estados están relacionados entre sí por las latencias que llevan de uno a otro

Diagrama de estado

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

1100111

Vector de colisión inicial

Latencia 3 permitida

V. C. inicial: 1100111

Despl.: 0011100

Or: -----
1111111

Latencia 4 permitida

V. C. inicial: 1100111

Despl.: 1110000

Or: -----
1110111

1111111

1110111

Diagrama de estado

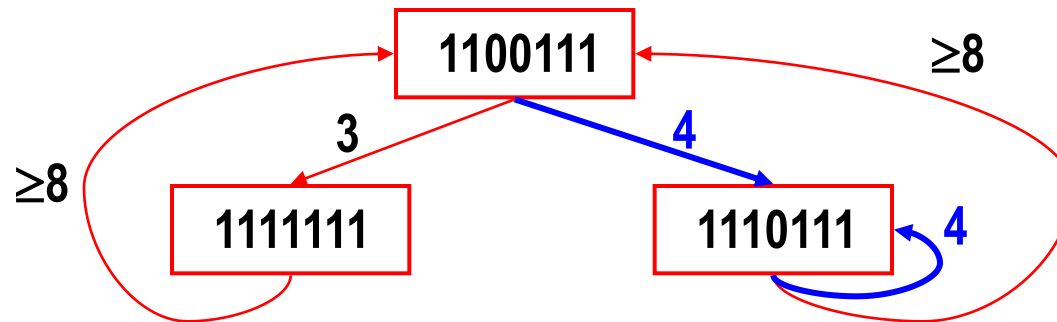
Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares



Secuencia de latencias = $\langle 4, 4, 4, 4, 4, \dots \rangle$

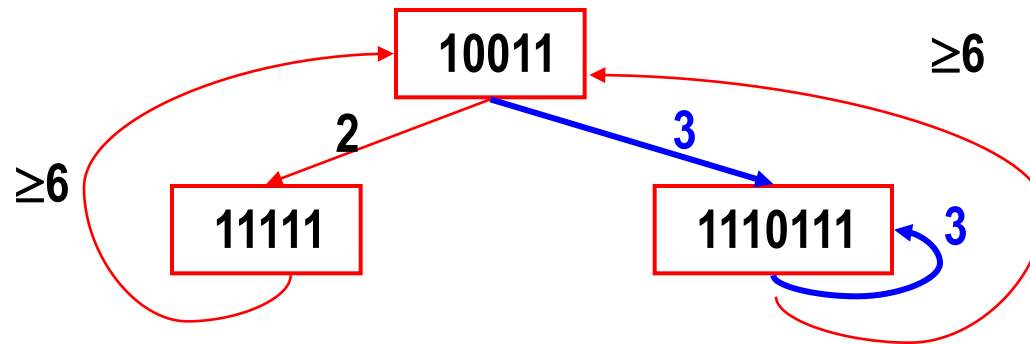
Ciclo de latencia = (4)

Latencia media = 4 = MAL

Segmentación

Diagrama de estado

Ejemplo: Obtened el diagrama de estados para el vector de colisión: 10011



Estrategia avara:

Secuencia de latencias = <2, 6, 2, 6, 2, ..>

Latencia media=(4)

Estrategia óptima:

Secuencia de latencias = <3, 3, 3, 3, 3, ..>

Ciclo de latencia = (3)

Latencia media = 3 = MAL

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Optimización por inserción de retardo

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- A veces se puede modificar la tabla de reserva sin que cambie la estructura del pipeline pero que incremente la productividad (MAL decrece)
- La inserción de un retardo en la tabla refleja la inserción de un registro delante o detrás de la lógica de un estado.
- Consiste en insertar un retardo en ciertas filas de la tabla de reserva.

Optimización por inserción de retardo

Introducción

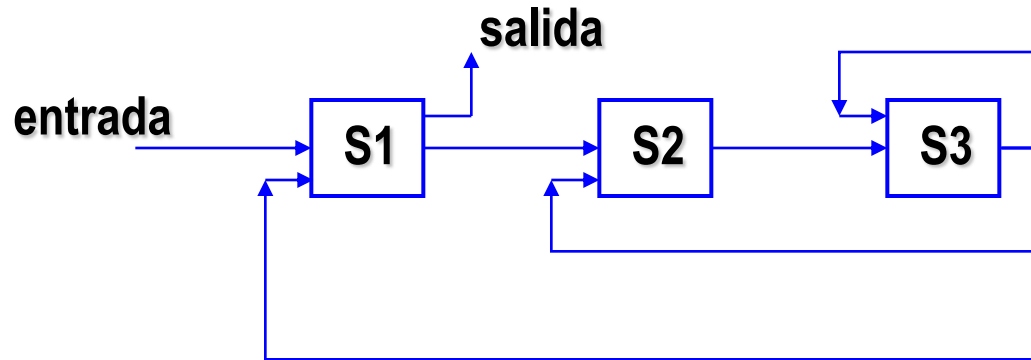
Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación



	0	1	2	3	4
S1	X				X
S2		X		X	
S3			X	X	

Vector de colisión

1101

Latencia del Pipeline = 5 ciclos

Optimización por inserción de retardo

Introducción

Segmentación
del repertorio

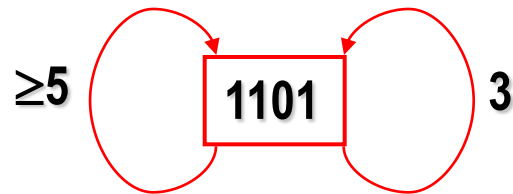
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Diagrama de estado



- ❖ MAL=3 ¿Se puede mejorar?
- ❖ El mayor rendimiento se obtendrá cuando se consiga que la latencia entre dos operaciones consecutivas sea el máximo número de marcas en una fila de la tabla de reserva (en este caso 2).

Optimización por inserción de retardo

Introducción

Segmentación
del repertorio

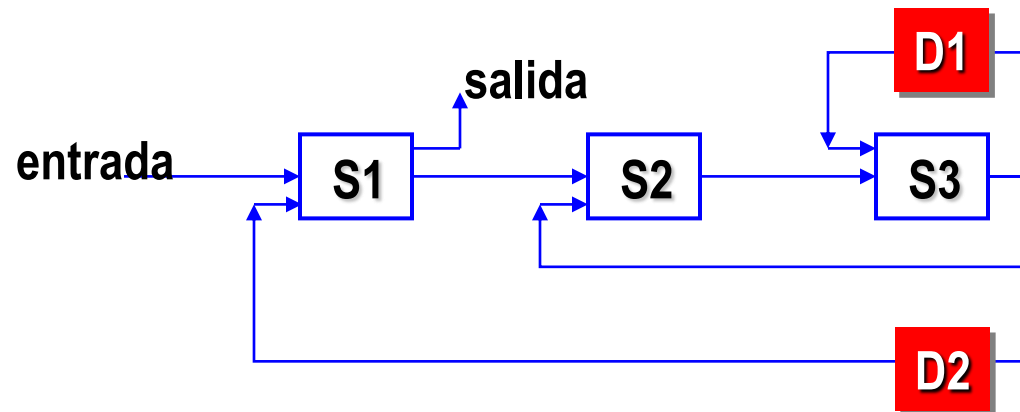
Cauces
aritméticos

Optimización

Superescales

Segmentación

Inserción de retardos sin cómputo



	0	1	2	3	4	5	6
S1	X				X		X
S2		X		X			
S3			X	X	X		

Vector de Colisión

010001

Latencia del Pipeline = 7 ciclos

Optimización por inserción de retardo

Introducción

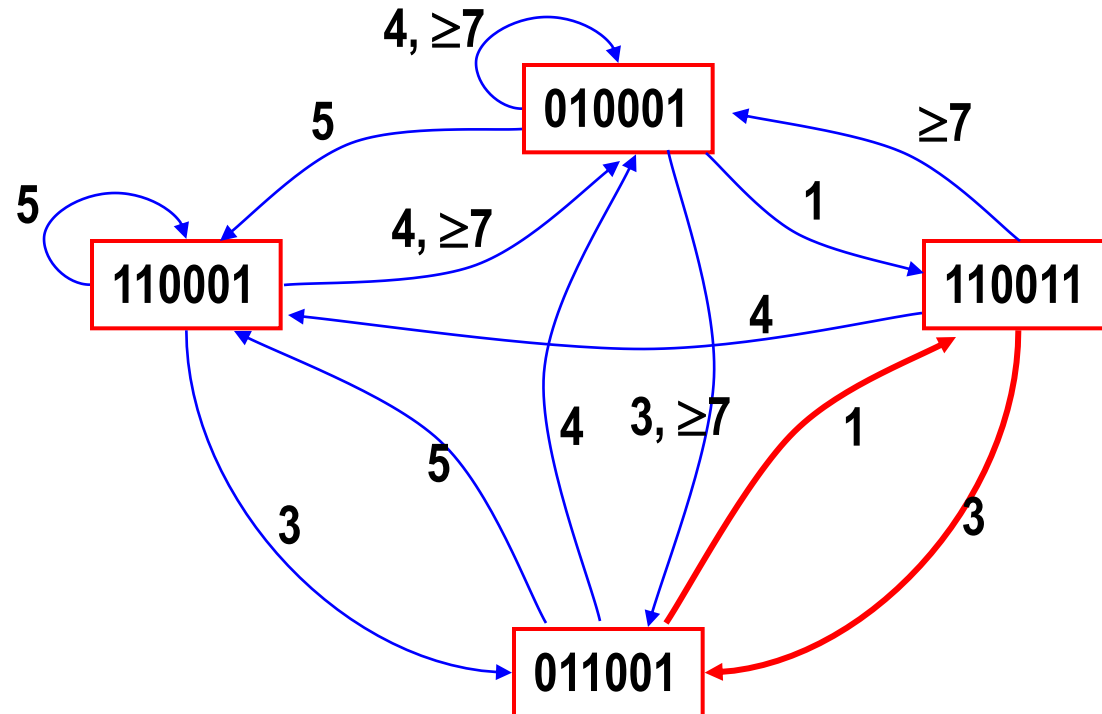
Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación



Nueva secuencia de latencias = $\langle 3, 1, 3, 1, 3, 1, \dots \rangle$

$MAL = (3+1)/2 = 2$

Cauces multifunción

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- La tabla de reserva especifica la ocupación del cauce para cada función
- Da lugar a múltiples vectores de colisión. Por ejemplo si el cauce puede ejecutar dos funciones, el número de vectores de colisión es 4: V_{11} , V_{12} , V_{21} , V_{22}
- El vector V_{ab} indica las latencias prohibidas y permitidas cuando se quiere lanzar la operación **a** después de la operación **b**
- Se construyen las matrices de colisiones cuyas filas son los vectores de colisión correspondientes.
 - Ejemplo, M_1 estaría formado por M_{11} y M_{21}
- El estado del cauce se representa ahora por matrices

Cauces multifunción

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

🔧 Ejemplo: Pipeline ejecuta dos funciones

	0	1	2	3	4
S1	A			A	
S2		A			
S3			A		A

	0	1	2	3	4
S1		B			B
S2				B	
S3	B		B		

Vectores de colisión:

$$\begin{aligned} V_{AA} &= (0110) & V_{AB} &= (1011) \\ V_{BA} &= (1010) & V_{BB} &= (0110) \end{aligned}$$

Matrices de colisión:

$$M_A = \begin{pmatrix} V_{AA} \\ V_{BA} \end{pmatrix} = \begin{pmatrix} 0110 \\ 1010 \end{pmatrix} \quad M_B = \begin{pmatrix} V_{AB} \\ V_{BB} \end{pmatrix} = \begin{pmatrix} 1011 \\ 0110 \end{pmatrix}$$

Cauces multifunción

Introducción

Segmentación
del repertorio

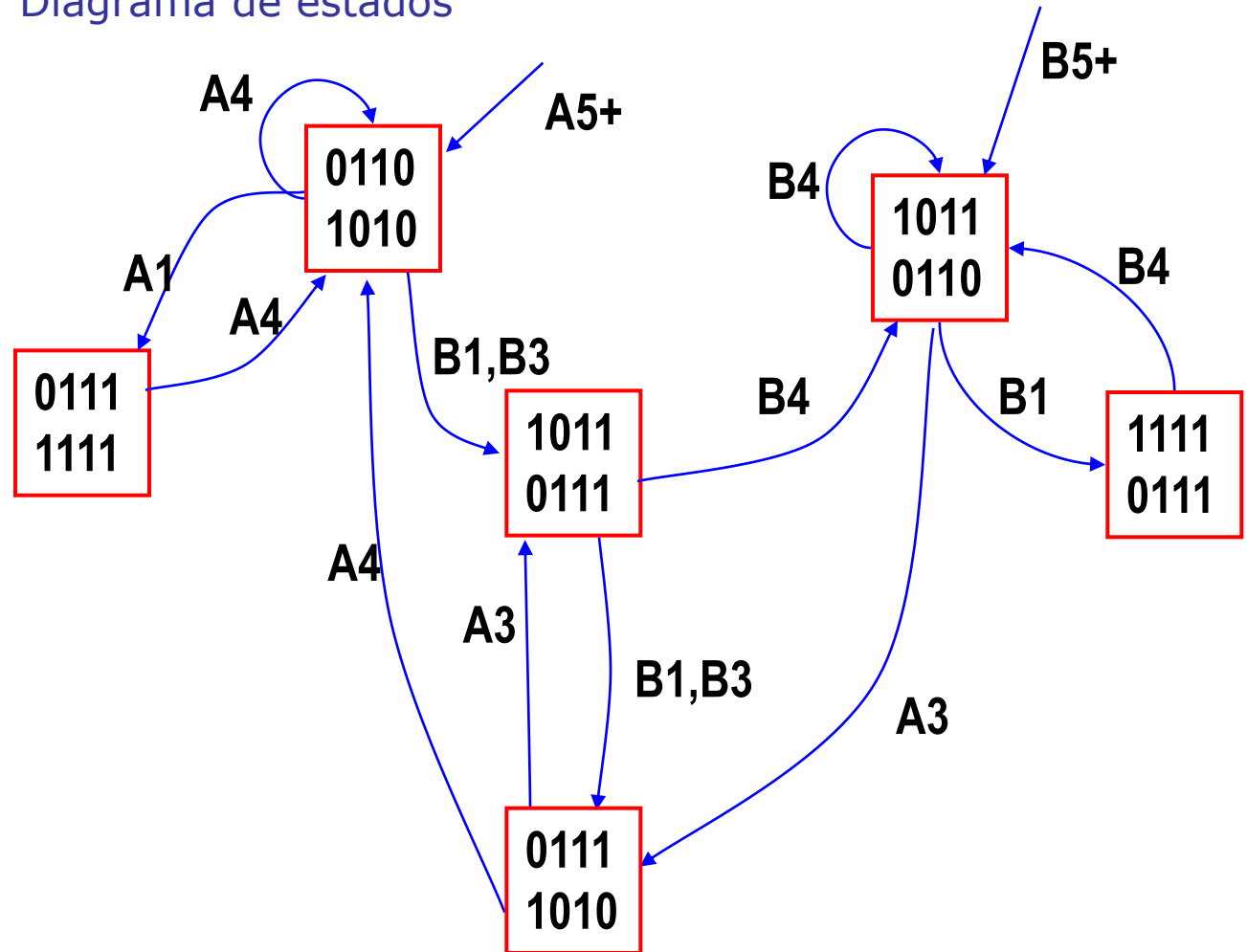
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Diagrama de estados



Cauces multifunción

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- ❖ Diferencias del diagrama de estados multifunción:
- ❖ Los nodos en lugar de estar identificados por un vector, están identificados por una matriz.
- ❖ El estado inicial no es único, sino que hay tantos estados iniciales como funciones pueda efectuar el procesador segmentado.
- ❖ En los arcos que conectan los nodos debe indicarse, además de la latencia, la operación que se ejecuta.

Tablas de reserva

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Contiene información sobre la ocupación de cada etapa en cada uno de los ciclos a medida que un elemento progresa por el cauce.

Ciclos de reloj →

Etapas ↓

	1	2	3	4	5	6
A	X			X		
B		X				
C			X		X	X

- Las filas se corresponden con las etapas del proceso
- Las columnas indican el número de ciclos necesarios para completar la tarea

4.5 Superescalares

Tema. Segmentación

Arquitectura de los Computadores

Superescalares

Introducción

Segmentación
del repertorio




Cauces
aritméticos

Optimización

Superescalares

Segmentación

4.5. SUPERESCALARES

-  4.5.1. Introducción
-  4.5.2. Procesadores superescalares
-  4.5.4. Procesadores VLIW

Introducción

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Segmentación: Ejecutar múltiples instrucciones en paralelo.
- ¿Cómo aumentar el incremento de productividad de la segmentación de instrucciones? -> aumentar el paralelismo a nivel de instrucción (ILP).
- Dos aproximaciones:
 - Aumentar la profundidad del pipeline
 - Menos trabajo por etapa -> ciclo de reloj más corto
 - Ejecución múltiple
 - Replicar etapas segmentadas->múltiples pipelines
 - Iniciar varias instrucciones en cada ciclo de reloj
 - $CPI < 1$ -> se utiliza IPC (Instrucciones por ciclo)
 - En la práctica, problemas con las dependencias

Introducción

Introducción

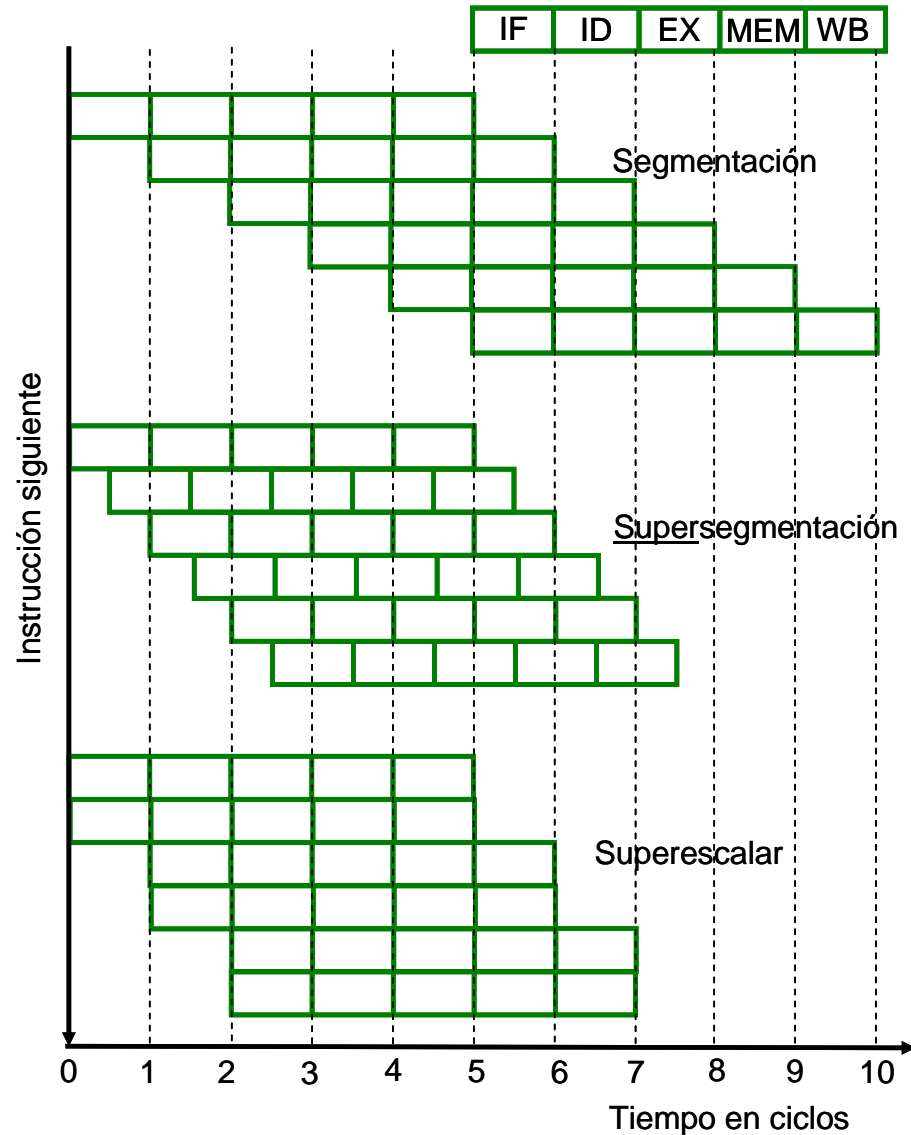
Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superscalares

Segmentación



Introducción

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Procesadores supersegmentados : aumento del número de etapas
- Procesadores superescalares: Se emite más de una instrucción por ciclo de reloj. Las instrucciones emitidas a la vez deben ser independientes
 - El HW se encarga de la planificación (dinámica) de instrucciones
- Procesadores VLIW (Very Long Instruction Word) Varias operaciones independientes en instrucciones muy largas
 - El compilador se encarga de la planificación (estática) de instrucciones
 - Frecuencias de trabajo bajas para limitar el ancho de banda de memoria

Procesadores Superescalares

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Características

- ejecución de varias instrucciones en paralelo
- emisión de varias instrucciones por ciclo
- productividad de mas de 1 instrucción por ciclo

Requisitos

- existencia de paralelismo en el código a nivel de instrucciones

```
ADD R1,R2,R3
SUB R4,R5,R6
AND R7,R8,R9
Paralelismo de grado 3
```

```
ADD R1,R2,R3
SUB R4,R5,R1
AND R7,R8,R4
Paralelismo de grado 1
```

- existencia de paralelismo en el hardware (para poder explotar el anterior)

Procesadores Superescalares

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Limitaciones

- RIESGOS: estructurales / dependencia de datos / control
- mayor concurrencia -> conflictos ↑ (ganancia real mas lejos de la ideal)
- mayor influencia de los huecos de retardo (carga / salto)

Procesadores Superescalares

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Los procesadores superescalares están diseñados para explotar más el paralelismo a nivel de instrucciones en los programas de usuario
- Solo las instrucciones independientes pueden ejecutarse en paralelo sin que provoquen estados de espera
- La cantidad de paralelismo a nivel de instrucciones varía ampliamente dependiendo del tipo de código que va a ejecutarse
- El procesador ha de ser capaz de identificar el paralelismo en las instrucciones (ILP) y organizar la captación, decodificación y ejecución de las instrucciones en paralelo.

Procesadores Superescalares

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- El procesador localiza las instrucciones que pueden introducirse en el cauce y ejecutarse.
- 3 ordenaciones importantes:
 - Orden en que se captan las instrucciones
 - Orden en que se ejecutan
 - Orden en que actualizan los registros y memoria
- Cuanto más sofisticado sea el procesador, menos limitación en las ordenaciones anteriores.
- Restricción: El resultado debe ser correcto.

Procesadores Superescalares

Introducción

Segmentación
del repertorio

Cauces
aritméticos

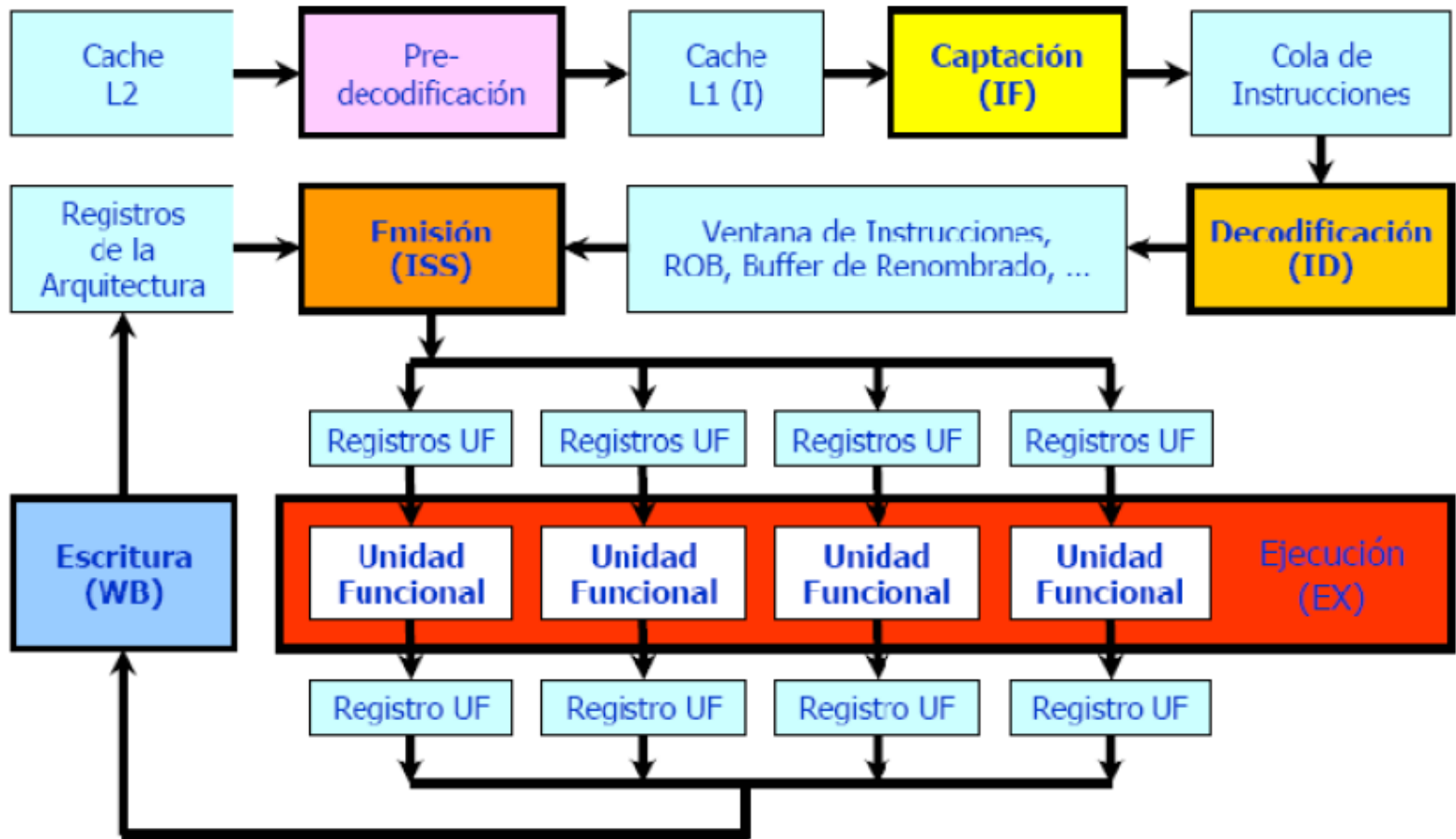
Optimización

Superescalares

Segmentación

- Políticas de emisión de instrucciones:
 - Emisión en orden y finalización en orden
 - Emisión en orden y finalización desordenada
 - La emisión para cuando encuentra un conflicto
 - La finalización desordenada requiere una política de emisión más compleja
 - Emisión desordenada y finalización desordenada
 - Para la emisión desordenada se necesita desacoplar etapas de decodificación y ejecución (mediante un buffer llamado ventana de instrucciones)
- Método para mejorar conflictos en los almacenamientos fuera de orden: renombramientos de registros.

Etapas de un cauce superescalar



Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Procesadores Superescalares

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Etapas del procesamiento escalar:

Captación de Instrucciones (IF):

- Capaz de leer varias instrucciones por ciclo desde la caché.
- Las instrucciones pasan a una cola en el orden que se han captado.

Decodificación de instrucciones (ID)

- Se introducen tantas instrucciones como sea capa de decodificar por ciclo
- Se almacenan a la espera de que estén disponibles los operandos y las unidades funcionales que implementan las operaciones

Emisión (Issue)

- Determina que instrucciones pueden pasar a ejecutarse entre la que tienen disponibles tanto sus datos como la unidad funcional requerida

Procesadores Superescalares

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Etapas del procesamiento escalar:
- Ejecución (EX)
 - Se implementa a través de distintas unidades que realizan las operaciones codificadas en la instrucción. El número de unidades de procesamiento disponibles determina el número máximo de instrucciones que pueden ejecutarse al mismo tiempo
- Escritura (WB)
 - Almacena los resultados en el banco de registros

Algunas arquitecturas

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- PowerPC 604
 - Seis unidades independientes de ejecución
 - Unidad de ejecución de salto
 - Unidad de load/store
 - 3 unidades de enteros
 - Unidad de coma flotante
 - Emisión en orden
 - Renombrado de registros
- PowerPC 620
 - Proporciona además emisión fuera de orden
- Pentium
 - Tres unidades independientes de ejecución
 - 2 unidades de enteros
 - Unidad de coma flotante
 - Emisión en orden

Algunas arquitecturas

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- La complejidad de la planificación dinámica y la especulación requieren potencia.
- Varios núcleos no requieren tanta potencia

Microprocessor	Año	Frecuencia reloj	Etapas Pipeline	Inst. emitidas	Fuera de orden/ especulación	Núcleos	Potencia
i486	1989	25MHz	5	1	No	1	5W
Pentium	1993	66MHz	5	2	No	1	10W
Pentium Pro	1997	200MHz	10	3	Yes	1	29W
P4 Willamette	2001	2000MHz	22	3	Yes	1	75W
P4 Prescott	2004	3600MHz	31	3	Yes	1	103W
Core	2006	2930MHz	14	4	Yes	2	75W
UltraSparc III	2003	1950MHz	14	4	No	1	90W
UltraSparc T1	2005	1200MHz	6	1	No	8	70W

Procesadores VLIW

Introducción

Segmentación
del repertorio

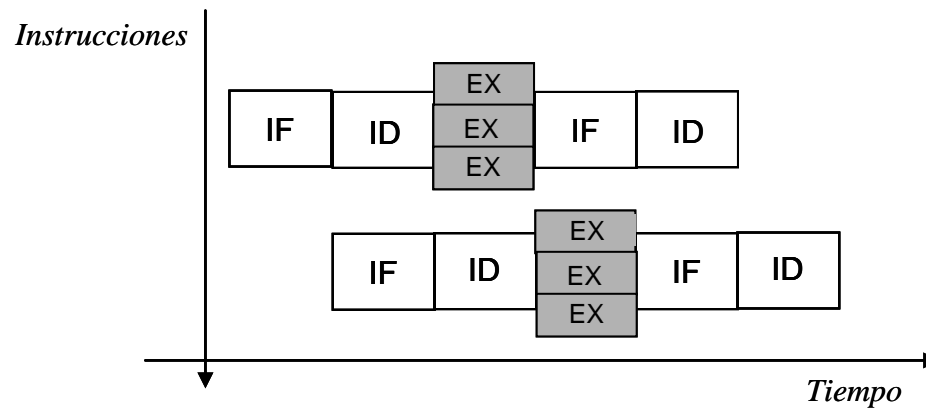
Cauces
aritméticos

Optimización

Superescalares

Segmentación

- ❖ VLIW = Very Long Instruction Word
- ❖ Comparten con los superescalares:
 - ❖ Son procesadores segmentados que puede emitir instrucciones en cada ciclo
 - ❖ Disponen de varias unidades de ejecución por lo que se pueden ejecutar varias operaciones simultáneamente



- ❖ Mientras en los superescalares el paralelismo lo planifica el procesador, en los procesadores VLIW el paralelismo se indica explícitamente en cada una de las instrucciones que se captan (cada instrucción incluye las operaciones que se realizan simultáneamente)

Procesamiento VLIW

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- ❖ Cada palabra de instrucción (entre 112 y 128 bits) está constituida por un conjunto de subpalabras o *slots*, y cada una de tales palabras puede codificar una operación.
- ❖ Las operaciones codificadas en cada subpalabra deben de poder ejecutarse en paralelo (sin dependencias entre ellas).
- ❖ El computador dispone de las unidades funcionales requeridas para su ejecución simultánea.
- ❖ El compilador es el encargado de ubicar las distintas operaciones que deben realizarse en un programa en las distintas subpalabras de las instrucciones VLIW teniendo en cuenta las dependencias y los recursos del procesador.
- ❖ En cada ciclo se emitirá una instrucción VLIW
- ❖ Hardware más sencillo que el de un superescalar

Procesadores VLIW

Introducción

Segmentación
del repertorio

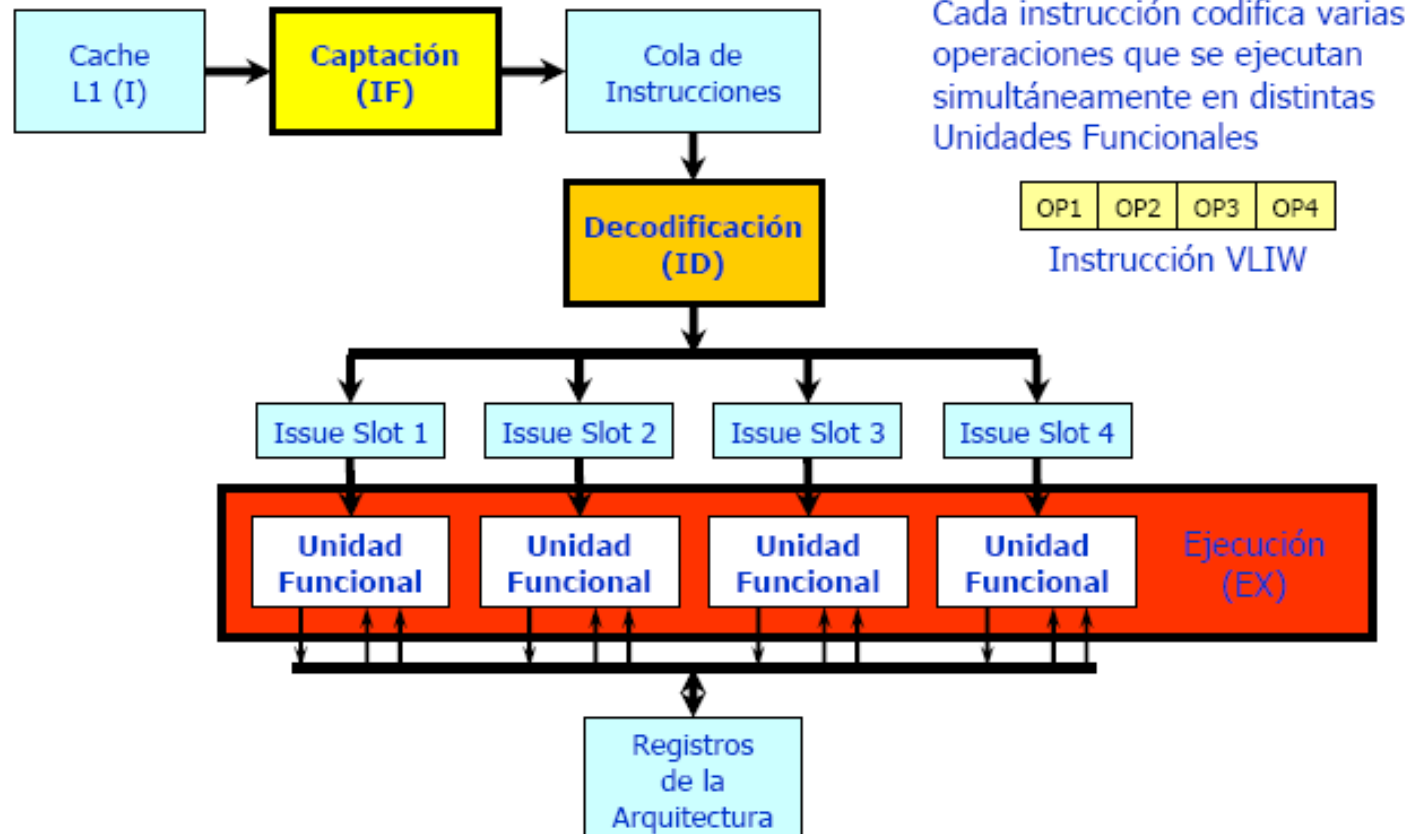
Cauces
aritméticos

Optimización

Superescalares

Segmentación

Esquema de bloques de un procesador sencillo VLIW



Procesadores VLIW

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

- Se trata de una arquitectura segmentada en la que las instrucciones que se captan pasan a una cola desde la que se emitirá una instrucción por ciclo a las unidades de ejecución. Cada instrucción VLIW codifica varias operaciones que se emitirán en el mismo ciclo a las correspondientes unidades funcionales.
- Tras decodificarse la instrucción VLIW, cada una de las operaciones pasa a la ventana de emisión (*issue slot*) correspondiente para empezar a ejecutarse en el ciclo siguiente.
- Las **ventajas** de la aproximación VLIW crece a medida que se pretende emitir más instrucciones por ciclo (el hardware adicional para un superescalar que emita dos instrucciones por ciclo es relativamente pequeño, pero crece a medida que se pretenden emitir más instrucciones por ciclo)

Procesadores VLIW

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

❖ Desventajas de VLIW

- ❖ Incompatibilidad entre distintas implementaciones o versiones.
El código VLIW no correrá bien con un número de unidades funcionales distinto o con eventos no planificados con diferentes latencias (ej: fallos en caché)
- ❖ Complejidad del compilador
- ❖ Si el compilador no encuentra instrucciones independientes de cada tipo ha de dejar la correspondiente subpalabra sin contenido pero ocupando espacio en memoria
- ❖ Los programas VLIW ocupan más espacio que los escalares

Procesadores VLIW

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescales

Segmentación

El papel del compilador

Planificación estática

Necesita asistencia de compilador, que puede realizar renombrados, reorganizaciones de código, etc, para mejorar el uso de los recursos disponibles, el esquema de predicción de saltos,...

↓
VLIW

Planificación dinámica

Requiere menos asistencia del compilador pero más coste hardware.
Facilita la portabilidad del código entra la misma familia de procesadores

↓
Superescalaar

El compilador construye paquetes de instrucciones (ventanas de emisión) sin dependencias, de forma que el procesador no necesita comprobarlas explícitamente.

Ejemplo de código VLIW

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Loop: LD F0, 0(R1)
ADD F4, F0, F2
SD 0(R1),F4
SUBI R1, R1,8
BNEZ R1, Loop

Código normal



Código VLIW

Referencia a Memoria 1	Referencia a Memoria 2	Operación FP 1	Operación FP 2	Inst. op/branch	Clock
LD F0, 0(R1)	LD F6, -8(R1)	NOP	NOP	NOP	1
LD F10, -16(R1)	LD F14, -24(R1)	NOP	NOP	NOP	2
LD 18, -32(R1)	LD F22, -40(R1)	ADD F4,F0,F2	ADD F8,F6,F2	NOP	3
LD F26, -48(R1)	NOP	ADD F12,F10,F2	ADD F16,F14,F2	NOP	4
NOP	NOP	ADD F20,F18,F2	ADD F24,F22,F2	NOP	5
SD 0(R1), F0	SD -8(R1), F4	ADD F28,F26,F2	NOP	NOP	6
SD -16(R1), F12	SD -24(R1), F16	NOP	NOP	NOP	7
SD -32(R1), F20	SD -40(R1), F24	NOP	NOP	SUBI R1,R1,#48	8
SD -0(R1), F28	NOP	NOP	NOP	BNÈZ R1,LOOP	9

Código instrucción VLIW

Inst memoria 1	Inst memoria 2	Inst coma flot 1	Inst coma flot 2	Inst entera o salto
----------------	----------------	------------------	------------------	---------------------

Ejemplo Arquitectura IA-64: ITANIUM-2 (INTEL)

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

Registros

- 128 registros de enteros ó generales, cada uno de 64 bits de longitud.
- 128 registros de punto flotante, cada uno de 80 bits de longitud.
- 64 registros de 1bit (predicado)
- 128 registros de aplicación
- Otros registros

Instrucciones largas: 128bits

Bundle (paquete): Conjunto de 3 instrucciones y un template



← 41 bits →



Indica en que unidad funcional se ejecuta cada operación. Y la dependencia entre este bundle y los siguientes

Ejemplo Arquitectura IA-64: ITANIUM-2 (INTEL)

Introducción

Segmentación
del repertorio

Cauces
aritméticos

Optimización

Superescalares

Segmentación

