



# Tema 4. Segmentación

Arquitectura de los Computadores

# Tema 4. Segmentación

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

## Objetivos:

- Mostrar al alumno conceptos relativos a segmentación.
- Proporcionar una clasificación de las arquitecturas segmentadas.
- Proponer varios niveles de aplicación de la segmentación.
- Profundizar en la segmentación del repertorio de instrucciones, utilizando la arquitectura MIPS como caso de estudio y manteniendo la continuidad con respecto a temas anteriores.
- Estudiar los cauces aritméticos
- Introducir las técnicas de optimación de unidades segmentadas
- Introducir conceptos de superescalares

# Tema 4. Segmentación

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- 1. Introducción
- 2. Segmentación del repertorio de instrucciones
- 3. Cauces aritméticos
- 4. Optimización de unidades segmentadas
- 5. Superescalares

# 4.1 Introducción

**Tema 4. Segmentación**

**Arquitectura de los Computadores**

# Segmentación

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

## 4.1. INTRODUCCIÓN

- Concepto de segmentación
- Clasificación de las arquitecturas segmentadas Niveles de aplicación.
- Análisis de prestaciones

# Concepto de segmentación

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- La segmentación es una de las claves que permite **aumentar el rendimiento** en los computadores.
- Analogía con una cadena de montaje industrial
  - La ejecución de un tarea se divide en etapas, cada elemento de la cadena se especializa en realizar una operación concreta.
- Explota el **paralelismo temporal**
  - Opera de forma serie para una pieza determinada
  - Ejecución de varias tareas simultáneas en diferentes etapas

# Concepto. Secuencial

Introducción

Segmentación  
del repertorio

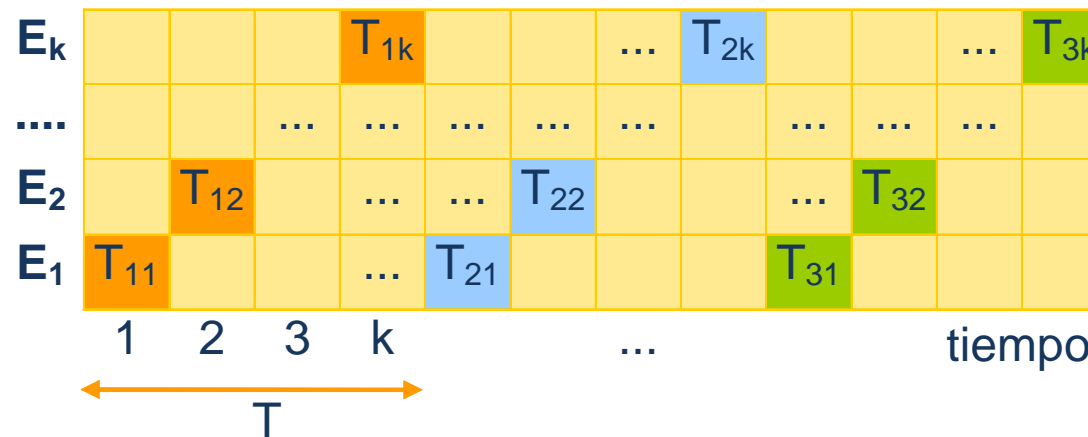
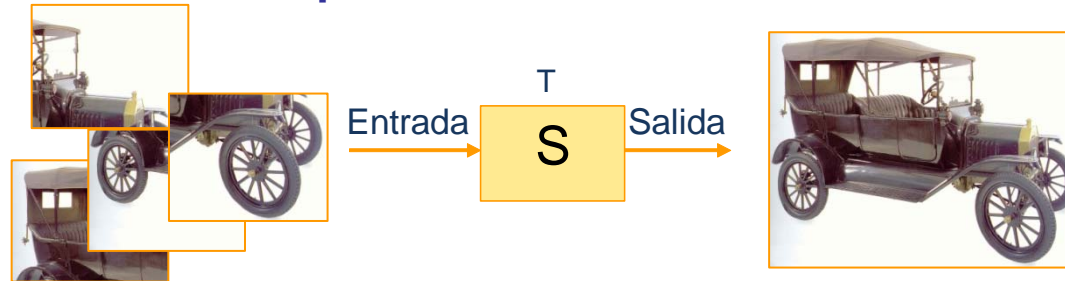
Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Una tarea puede realizarse cuando todos los **recursos** que necesita estén **disponibles**



- Equipo de trabajo, **trabajadores especializados** que descansan hasta que vuelvan a entrar los materiales
- Sólo un trabajador** (un recurso)

# Concepto. Secuencial

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

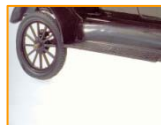
Segmentación

- Una tarea puede realizarse cuando todos los **recursos** que necesita estén **disponibles**



$E_k$				$T_{1k}$			...	$T_{2k}$			...	$T_{3k}$
....			...	...	...	...	...		...	...	...	
$E_2$		$T_{12}$		...	...	$T_{22}$			...	$T_{32}$		
$E_1$	$T_{11}$			...	$T_{21}$				$T_{31}$			
	1	2	3	k			...					tiempo

$\longleftrightarrow$   
 $T$





# Concepto. Secuencial

Introducción

Segmentación  
del repertorio

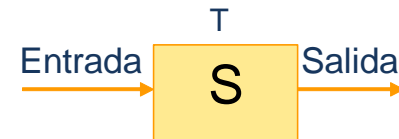
Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Una tarea puede realizarse cuando todos los **recursos** que necesita estén **disponibles**



$E_k$				$T_{1k}$			...	$T_{2k}$			...	$T_{3k}$
....			...	...	...	...	...		...	...	...	
$E_2$		$T_{12}$		...	...	$T_{22}$			...	$T_{32}$		
$E_1$	$T_{11}$			...	$T_{21}$				$T_{31}$			
	1	2	3	k			...					
				$T$		tiempo						



# Concepto. Secuencial

Introducción

Segmentación  
del repertorio

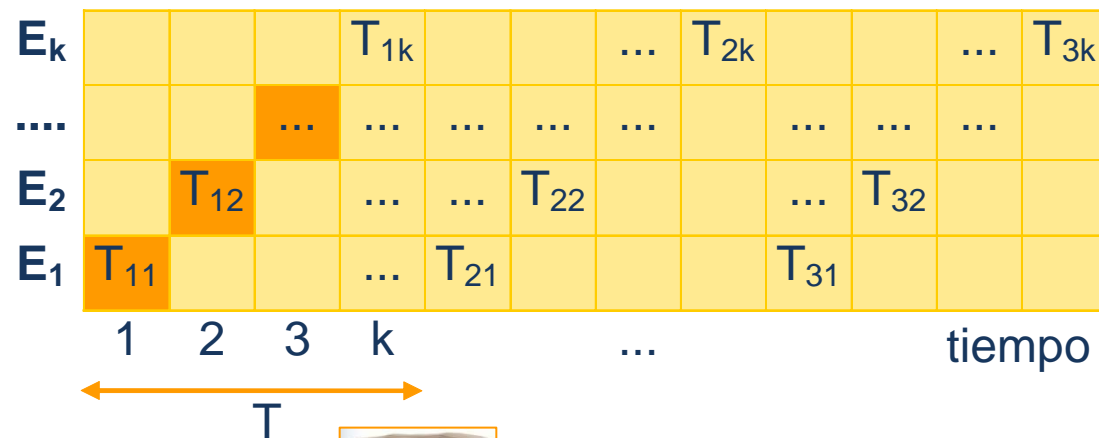
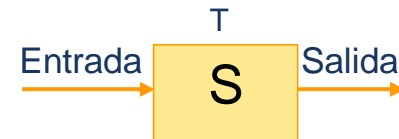
Cauces  
aritméticos

Optimización

Superescalares

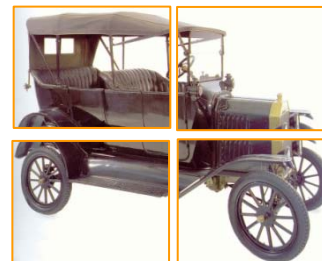
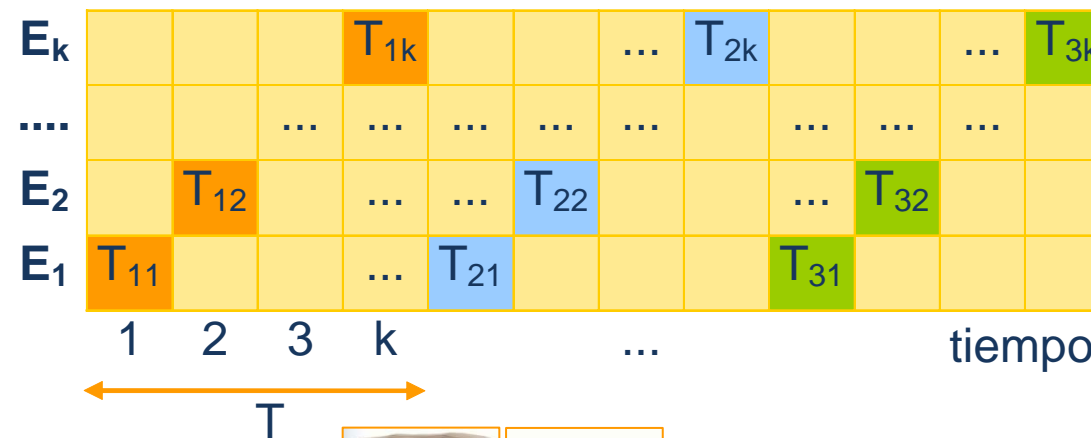
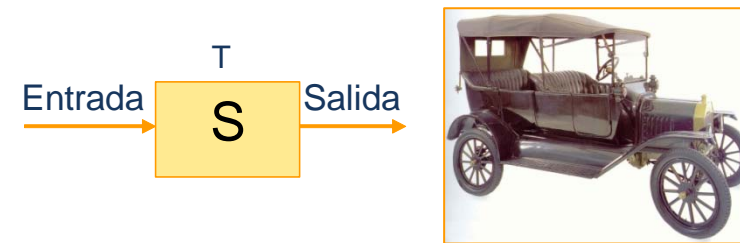
Segmentación

- Una tarea puede realizarse cuando todos los **recursos** que necesita estén **disponibles**



# Concepto. Secuencial

- Una tarea puede realizarse cuando todos los **recursos** que necesita estén **disponibles**



Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

# Concepto. Con segmentación

Introducción

Segmentación  
del repertorio

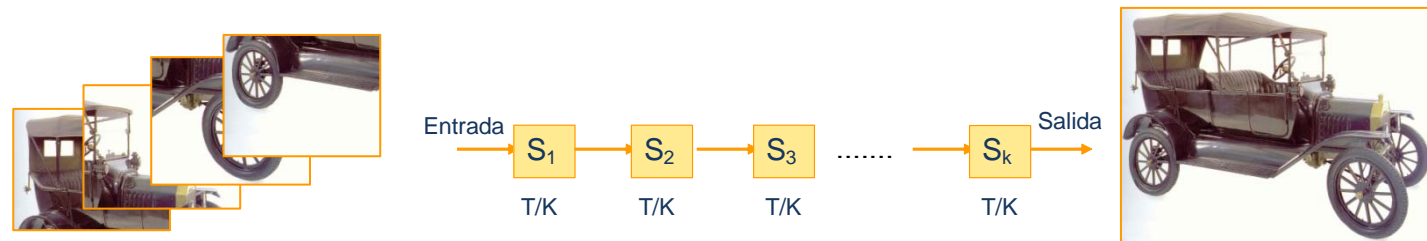
Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- El comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa



$E_k$				$T_{1k}$	$T_{2k}$	$T_{3k}$	$T_{4k}$				$T_{nk}$
....			...	...	...	...	...	...	...	...	...
$E_2$		$T_{12}$	$T_{22}$	$T_{32}$	$T_{42}$					$T_{n2}$	
$E_1$	$T_{11}$	$T_{21}$	$T_{31}$	$T_{41}$					$T_{n1}$		
	1	2	3	4			...	n			tiempo

- Cada trabajador pasa al siguiente trabajador de la cadena el estado actual del trabajo mientras puede comenzar a trabajar en el siguiente vehículo

# Concepto. Con segmentación

Introducción

Segmentación  
del repertorio

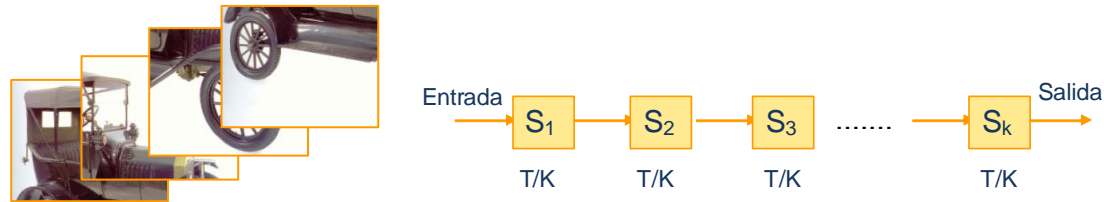
Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- El comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa



$E_k$				$T_{1k}$	$T_{2k}$	$T_{3k}$	$T_{4k}$				$T_{nk}$
....			...	...	...	...	...	...	...	...	...
$E_2$		$T_{12}$	$T_{22}$	$T_{32}$	$T_{42}$					$T_{n2}$	
$E_1$	$T_{11}$	$T_{21}$	$T_{31}$	$T_{41}$					$T_{n1}$		
	1	2	3	4				...	n		
	tiempo ciclos										



# Concepto. Con segmentación

Introducción

Segmentación  
del repertorio

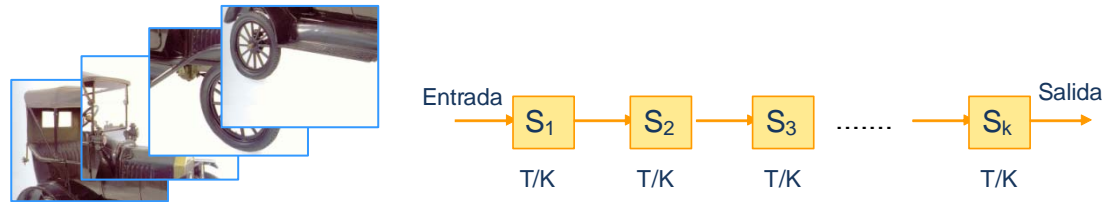
Cauces  
aritméticos

Optimización

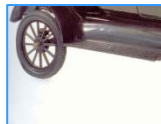
Superescalares

Segmentación

- El comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa



$E_k$				$T_{1k}$	$T_{2k}$	$T_{3k}$	$T_{4k}$				$T_{nk}$
....			...	...	...	...	...	...	...	...	...
$E_2$		$T_{12}$	$T_{22}$	$T_{32}$	$T_{42}$					$T_{n2}$	
$E_1$	$T_{11}$	$T_{21}$	$T_{31}$	$T_{41}$					$T_{n1}$		
	1	2	3	4				...	n		
											tiempo ciclos



# Concepto. Con segmentación

Introducción

Segmentación  
del repertorio

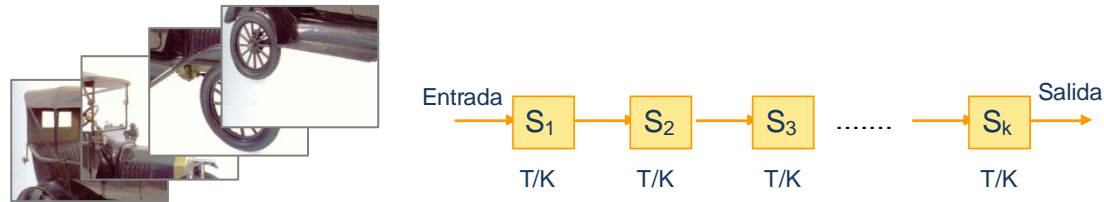
Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- El comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa



$E_k$				$T_{1k}$	$T_{2k}$	$T_{3k}$	$T_{4k}$				$T_{nk}$
....			...	...	...	...	...	...	...	...	...
$E_2$		$T_{12}$	$T_{22}$	$T_{32}$	$T_{42}$				$T_{n2}$		
$E_1$	$T_{11}$	$T_{21}$	$T_{31}$	$T_{41}$					$T_{n1}$		
	1	2	3	4				...	n		
	tiempo ciclos										



# Concepto. Con segmentación

Introducción

Segmentación  
del repertorio

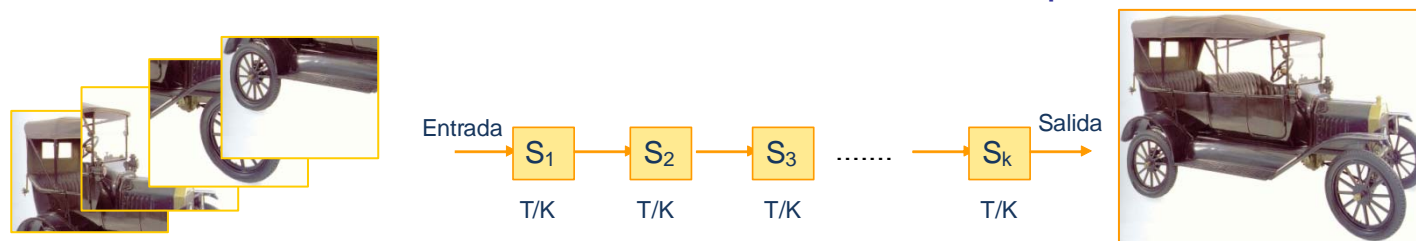
Cauces  
aritméticos

Optimización

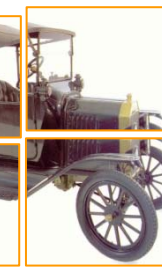
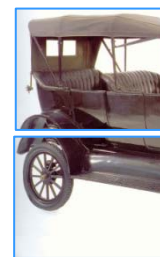
Superescalares

Segmentación

- El comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa



$E_k$				$T_{1k}$	$T_{2k}$	$T_{3k}$	$T_{4k}$				$T_{nk}$
....			...	...	...	...	...	...	...	...	...
$E_2$		$T_{12}$	$T_{22}$	$T_{32}$	$T_{42}$				$T_{n2}$		
$E_1$	$T_{11}$	$T_{21}$	$T_{31}$	$T_{41}$					$T_{n1}$		
	1	2	3	4				...	n		
	tiempo ciclos										





# Concepto. Con segmentación

Introducción

Segmentación  
del repertorio

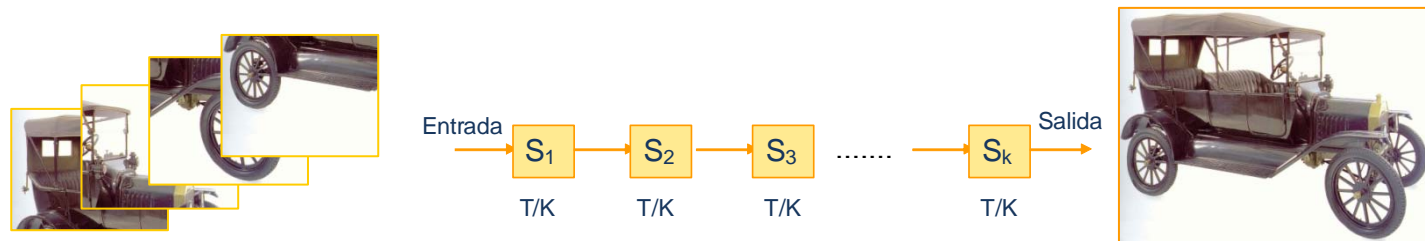
Cauces  
aritméticos

Optimización

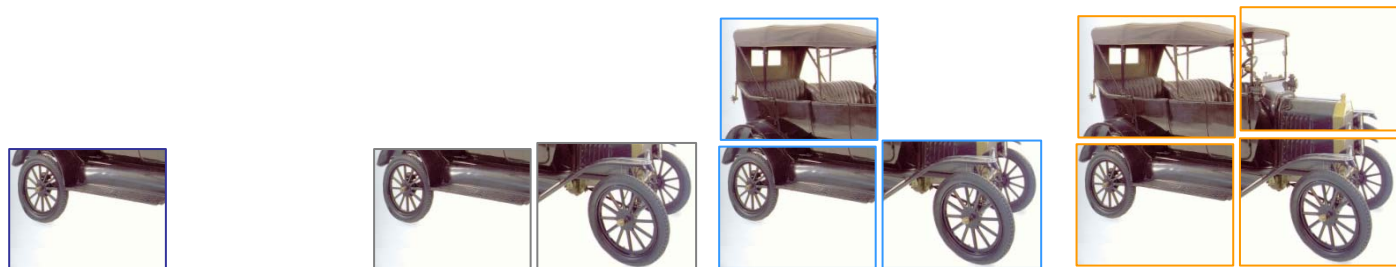
Superescalares

Segmentación

- El comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa



$E_k$				$T_{1k}$	$T_{2k}$	$T_{3k}$	$T_{4k}$				$T_{nk}$
....			...	...	...	...	...	...	...	...	...
$E_2$		$T_{12}$	$T_{22}$	$T_{32}$	$T_{42}$					$T_{n2}$	
$E_1$	$T_{11}$	$T_{21}$	$T_{31}$	$T_{41}$						$T_{n1}$	
	1	2	3	4				...	n		tiempo



# Concepto. Con segmentación

Introducción

Segmentación  
del repertorio

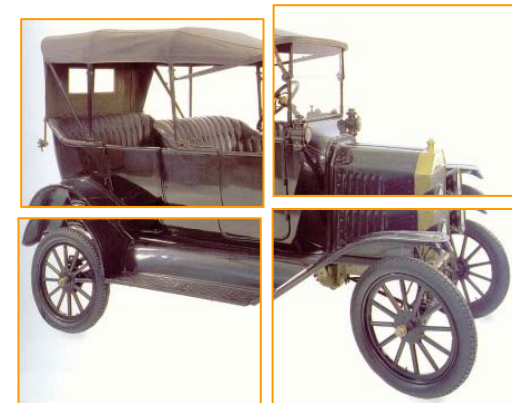
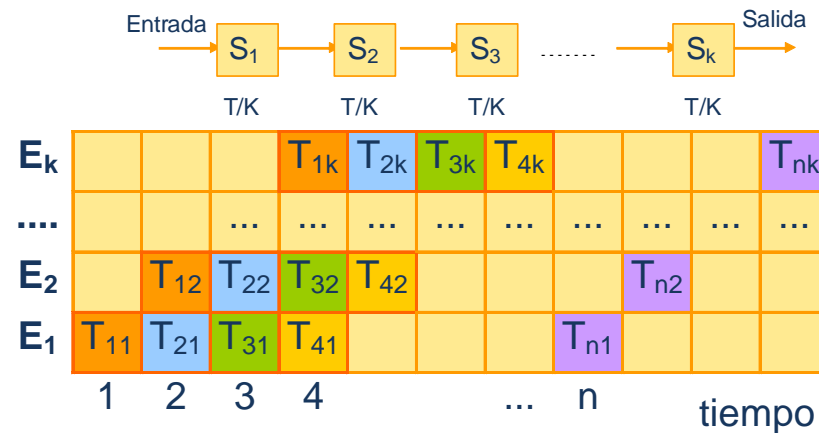
Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Factor determinante: descomposición tarea a realizar en etapas
  - Distribución uniforme del tiempo (caso ideal)
  - Etapas más lentas actúan como cuello de botella
  - Ajustar el ritmo de trabajo a la etapa más lenta
- Es necesario contemplar:
  - Para que cada trabajador pueda pasar el trabajo necesita tiempo para preparar y distribuir la parte del vehículo que lleva fabricada
  - También tener en cuenta que la nueva organización necesaria para controlar el proceso puede ser muy compleja



# Clasificación procesadores segmentados

Introducción

Segmentación del repertorio

Cauces aritméticos

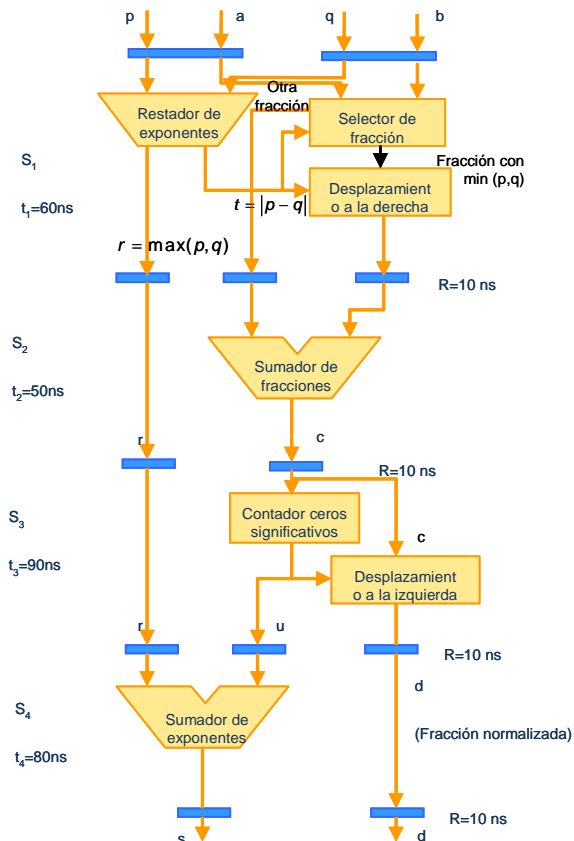
Optimización

Superescalares

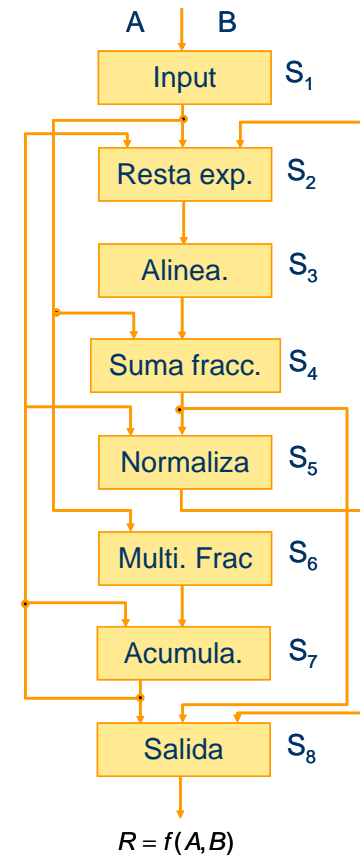
Segmentación

## Por tipo de cauce:

● **Unifunción:** ejecutan un único proceso (ejemplo: sumador).



● **Multifunción:** pueden ejecutar varios procesos (ejemplo: TI-ASC).



# Clasificación procesadores segmentados

Introducción

Segmentación del repertorio

Cauces aritméticos

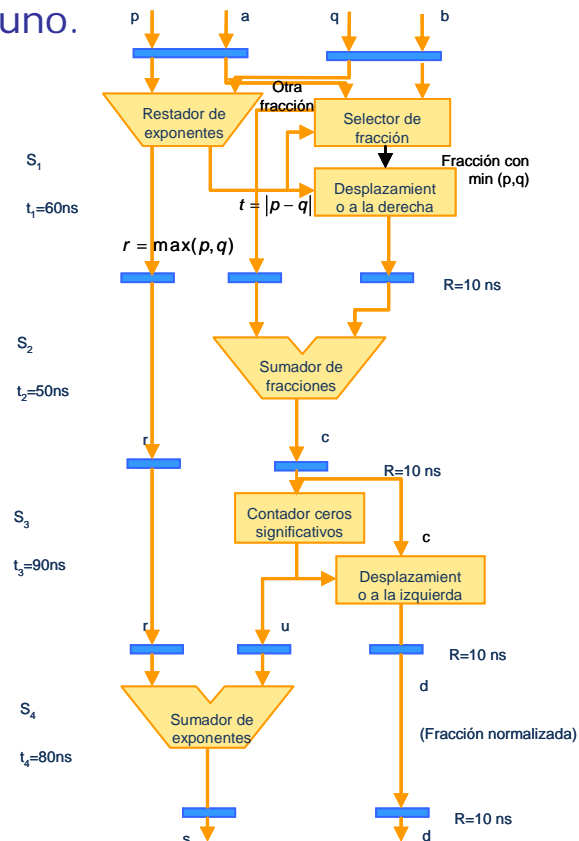
Optimización

Superescalares

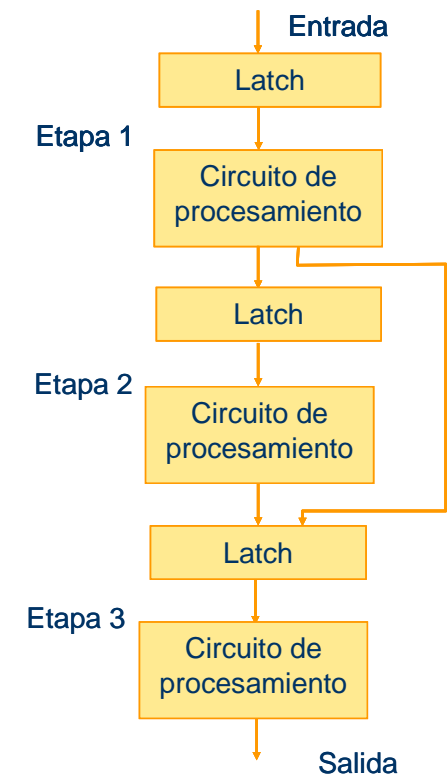
Segmentación

## Por tipo de cauce:

- **Estáticos:** en un instante determinado sólo pueden ejecutar uno.



- **Dinámicos:** pueden ejecutar simultáneamente varios procesos.



## Introducción

## Segmentación del repertorio

## Cauces aritméticos

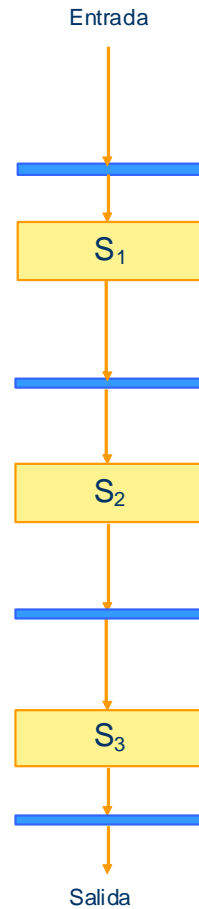
## Optimización

## Superescalares

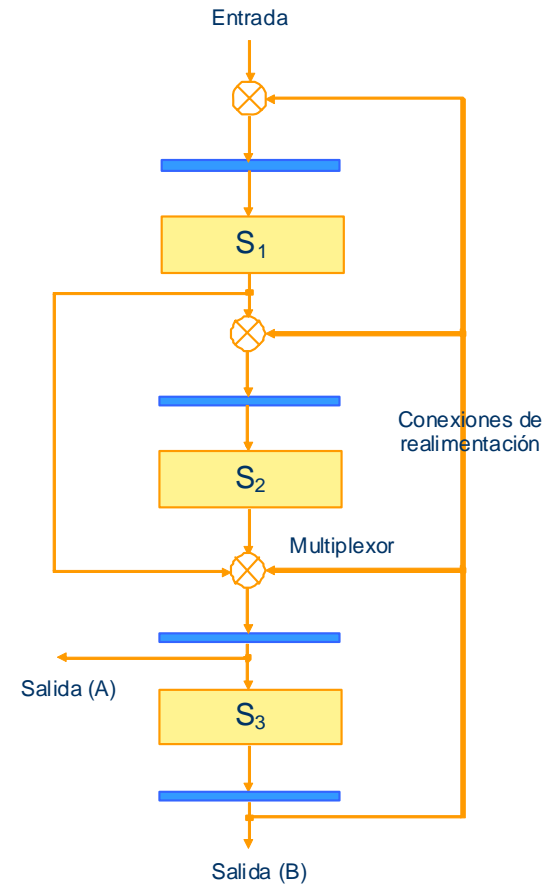
## Segmentación

### Por tipo de cauce:

- **Lineal:** a cada etapa sólo le puede seguir otra etapa concreta.



- No lineal:** se pueden establecer recorridos complejos de las etapas.



# Clasificación procesadores segmentados

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

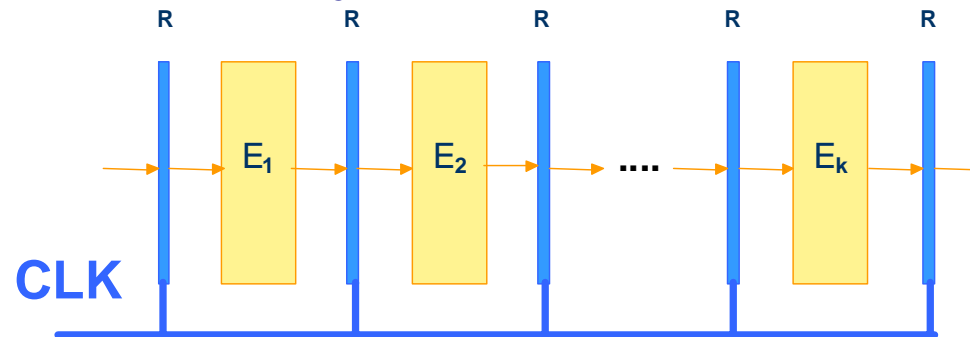
Optimización

Superescalares

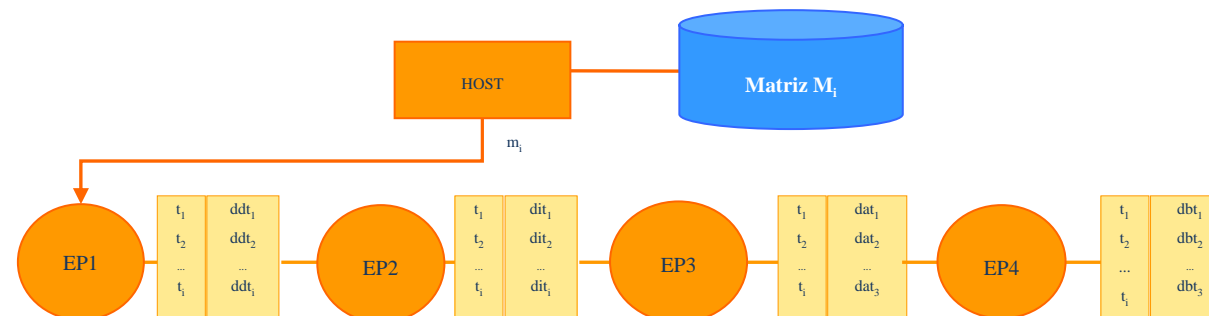
Segmentación

## Por tipo de cauce:

● **Síncrono:** Una señal de reloj.



● **Asíncrono:** Cada etapa funciona de forma autónoma.



● Protocolo comunicación entre etapas (semáforos, paso de mensajes).

# Clasificación procesadores segmentados

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

## ■ Por niveles de aplicación:

### ■ Segmentación aritmética

- Ejecutan una o varias operaciones de la ALU
- Pueden ser lineales (sumas) o no lineales (división). En este caso suelen ser cíclicos (bucles).
- Los procesadores actuales incluyen varias ALUs segmentadas y cada una se puede ocupar de varias operaciones.

### ■ Segmentación de instrucciones

- Suelen ser cauces lineales
- Algunas de sus fases pueden a su vez sub-segmentarse (uso de una ALU segmentada para la fase de ejecución)

### ■ Segmentación de procesadores

# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Se descompone cada operación aritmética en distintas etapas y se realiza un diseño encauzado de la ALU
- Técnica empleada por la mayoría de los computadores.
  - Unidad de punto flotante de tres etapas del clásico del MC68040.
  - La unidad de punto flotante del IBM 360/Model 91 con dos cauces. Un sumador de 2 etapas y un multiplicador de 6.
  - El cauce multifunción de 8 etapas del TI ASC.
  - Operaciones vectoriales del Cray-1 desde 1 a 14 etapas dependiendo del tipo de operación
  - Supercomputadores NEC SX2, Fujitsu VP400, Hitachi S820, Cray Y-MP
  - Procesadores NEC SX6 utilizado en los 640 nodos del Earh Simulator
  - Unidades punto flotante, MMX y SSE de microprocesadores de Intel y AMD



# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

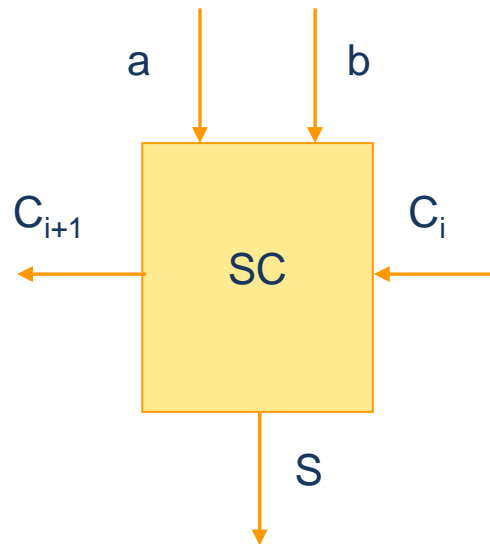
Optimización

Superescalares

Segmentación

- Sumador de tres bits con propagación de acarreo
  - Compararemos el caso secuencial frente al segmentado

## Sumador completo



# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

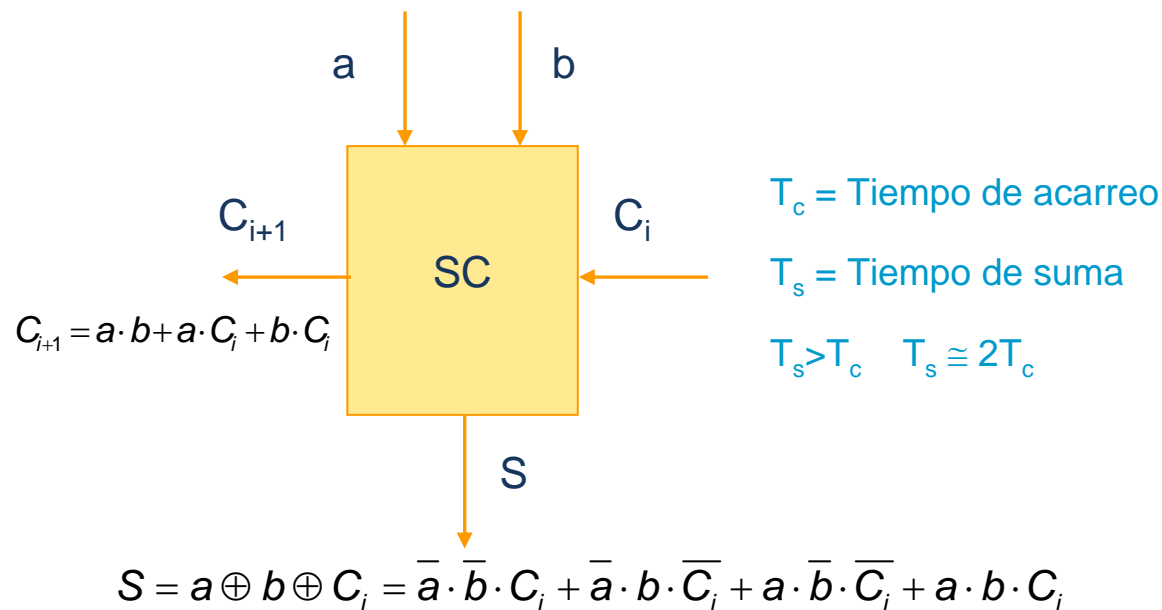
Superescalares

Segmentación

## Sumador de tres bits con propagación de acarreo

- Compararemos el caso secuencial frente al segmentado

### Sumador completo



# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

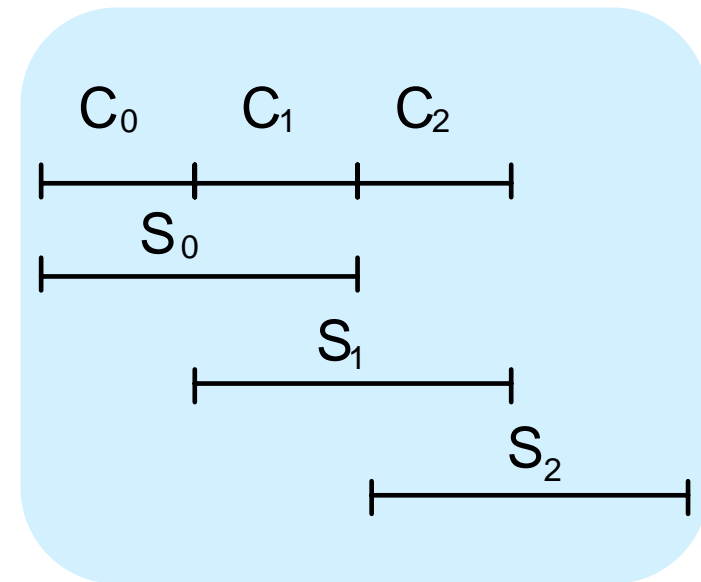
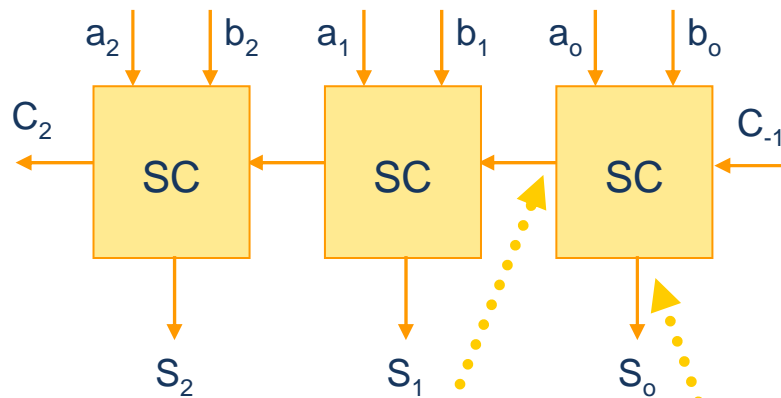
Optimización

Superescalares

Segmentación

## Sumador de tres bits con propagación de acarreo

### Caso secuencial



1º El acarreo

2º La suma

# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

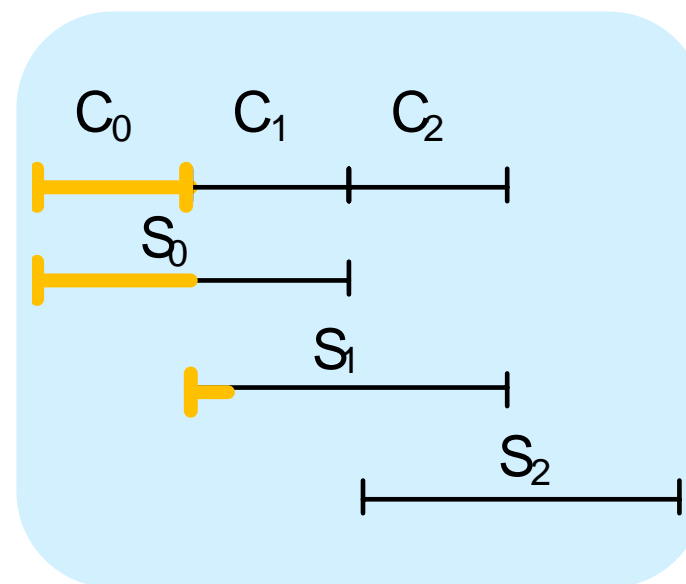
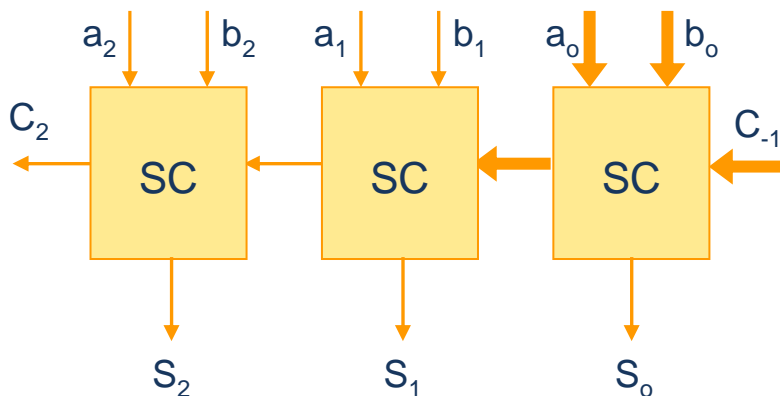
Optimización

Superescalares

Segmentación

Sumador de tres bits con propagación de acarreo

## Caso secuencial



# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

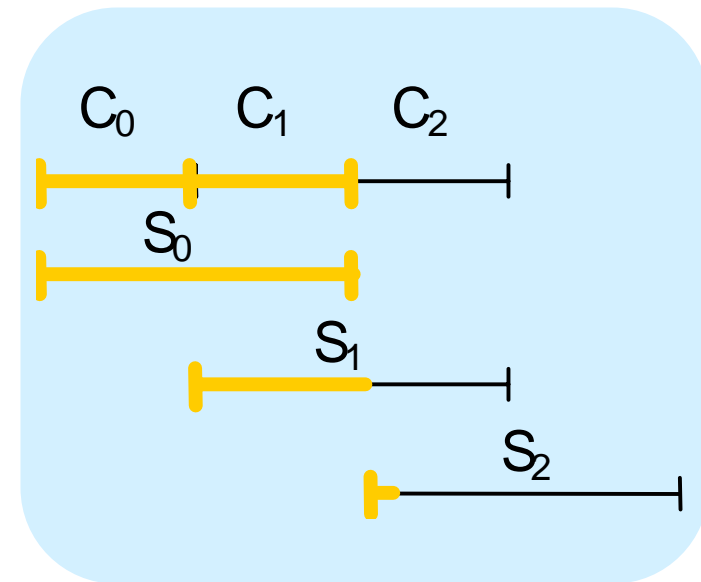
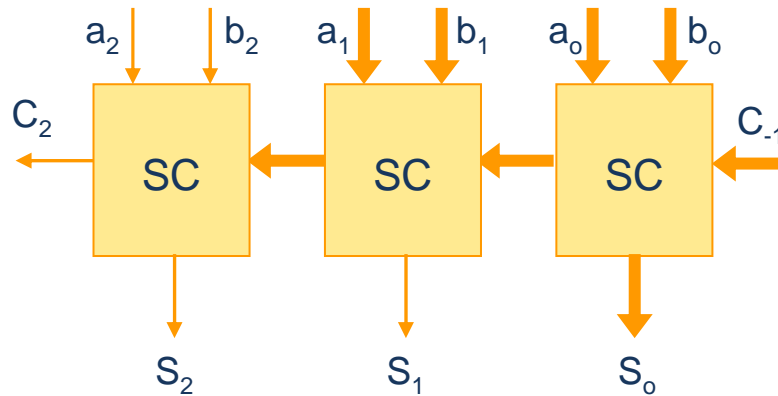
Optimización

Superescalares

Segmentación

Sumador de tres bits con propagación de acarreo

## Caso secuencial



# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

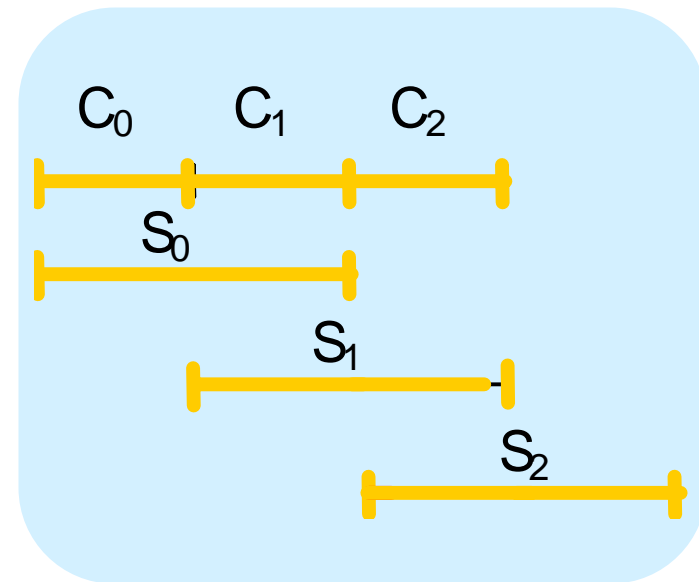
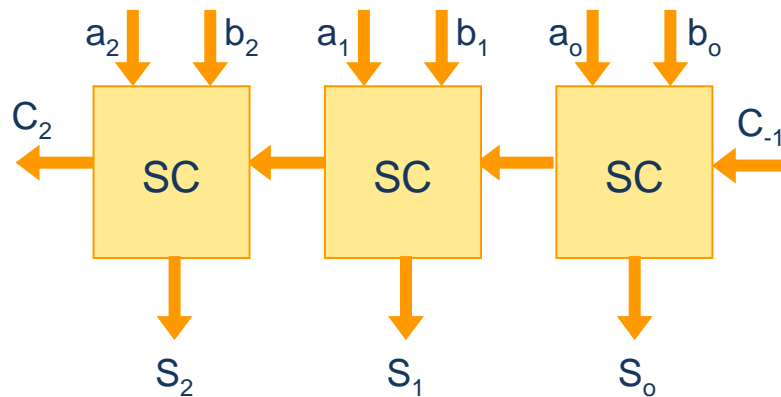
Optimización

Superescalares

Segmentación

Sumador de tres bits con propagación de acarreo

## Caso secuencial



$$T_{\text{suma}} = 2 T_c + T_s$$

# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

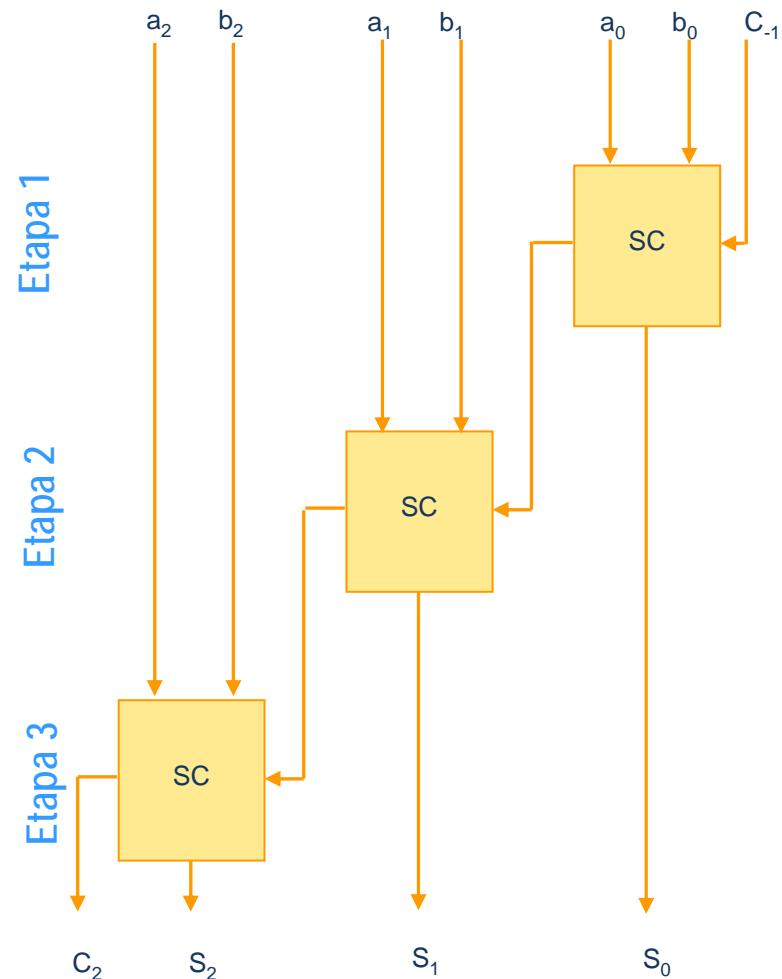
Superescalares

Segmentación

## Sumador de tres bits con propagación de acarreo

### Caso segmentado

- Bloque constructivo SC 1 bit
- Dividir en etapas cada una de las sumas parciales de los bits de los números



# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

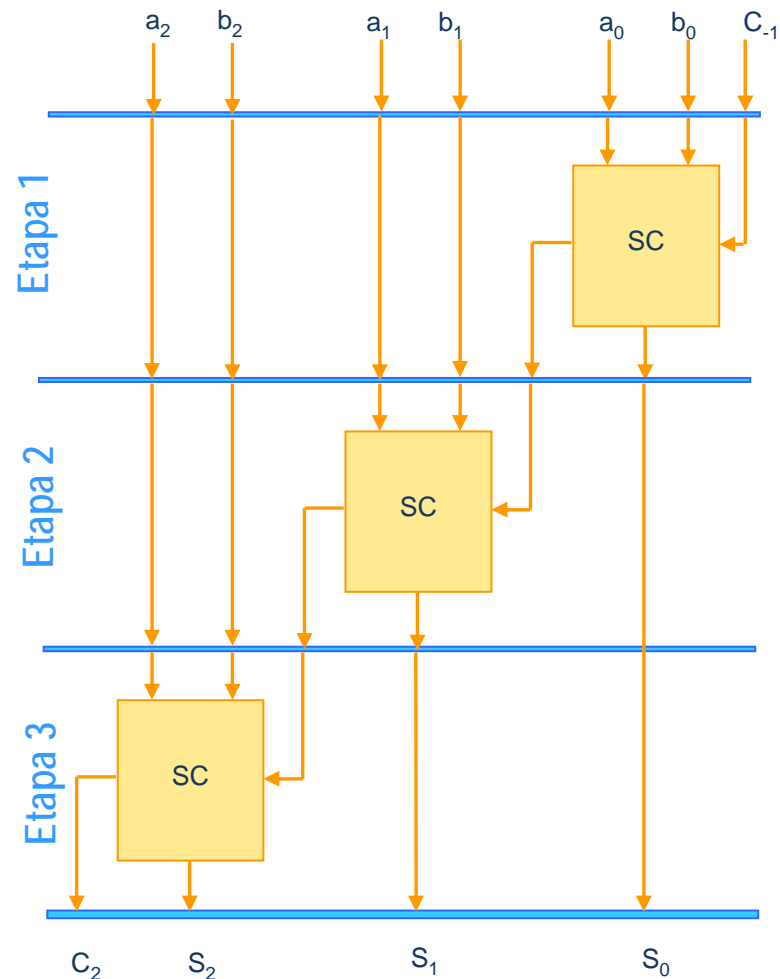
Superescalares

Segmentación

## Sumador de tres bits con propagación de acarreo

### Caso segmentado

- Bloque constructivo SC 1 bit
- Dividir en etapas cada una de las sumas parciales de los bits de los números
- Introducir registros intermedios entre cada una de las etapas para pasar datos (secuencial)





# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

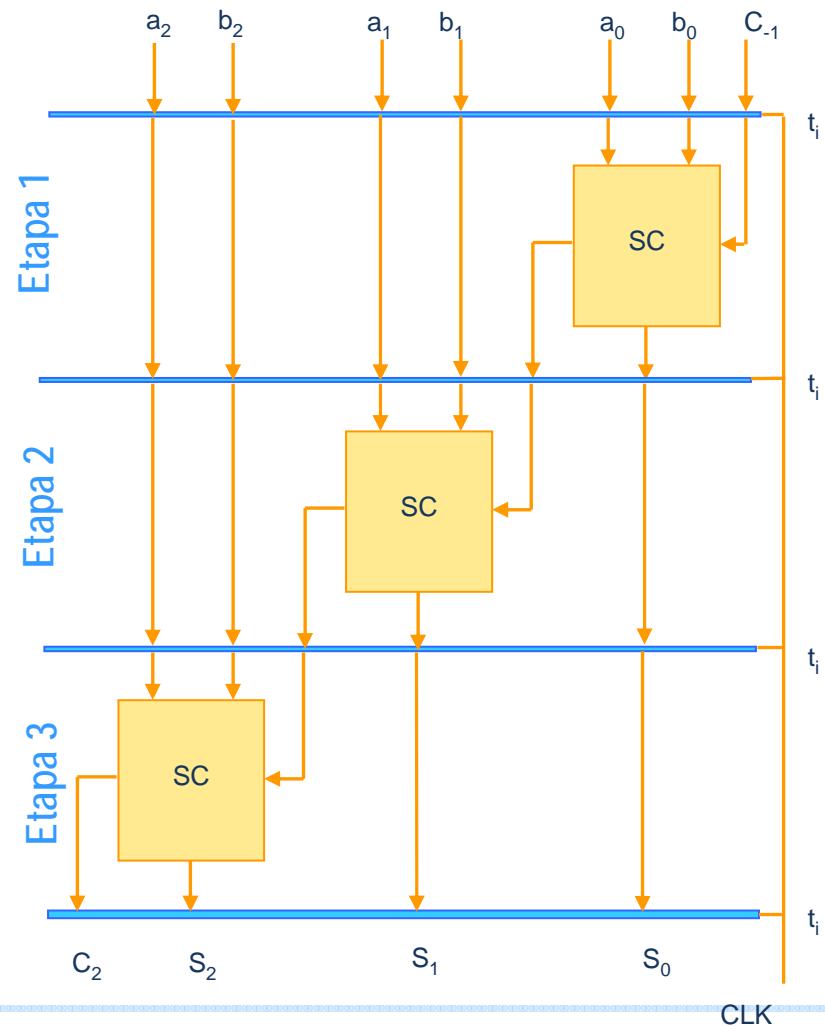
Segmentación

## Sumador de tres bits con propagación de acarreo

### Caso segmentado

- Bloque constructivo SC 1 bit
- Dividir en etapas cada una de las sumas parciales de los bits de los números
- Introducir registros intermedios entre cada una de las etapas para pasar datos (secuencial)
- (Ritmo) Duración del ciclo de reloj clk: tiempo de la etapa más lenta,  $T_s$

$$T_{\text{suma}} = 3 (T_s + t_i)$$



# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

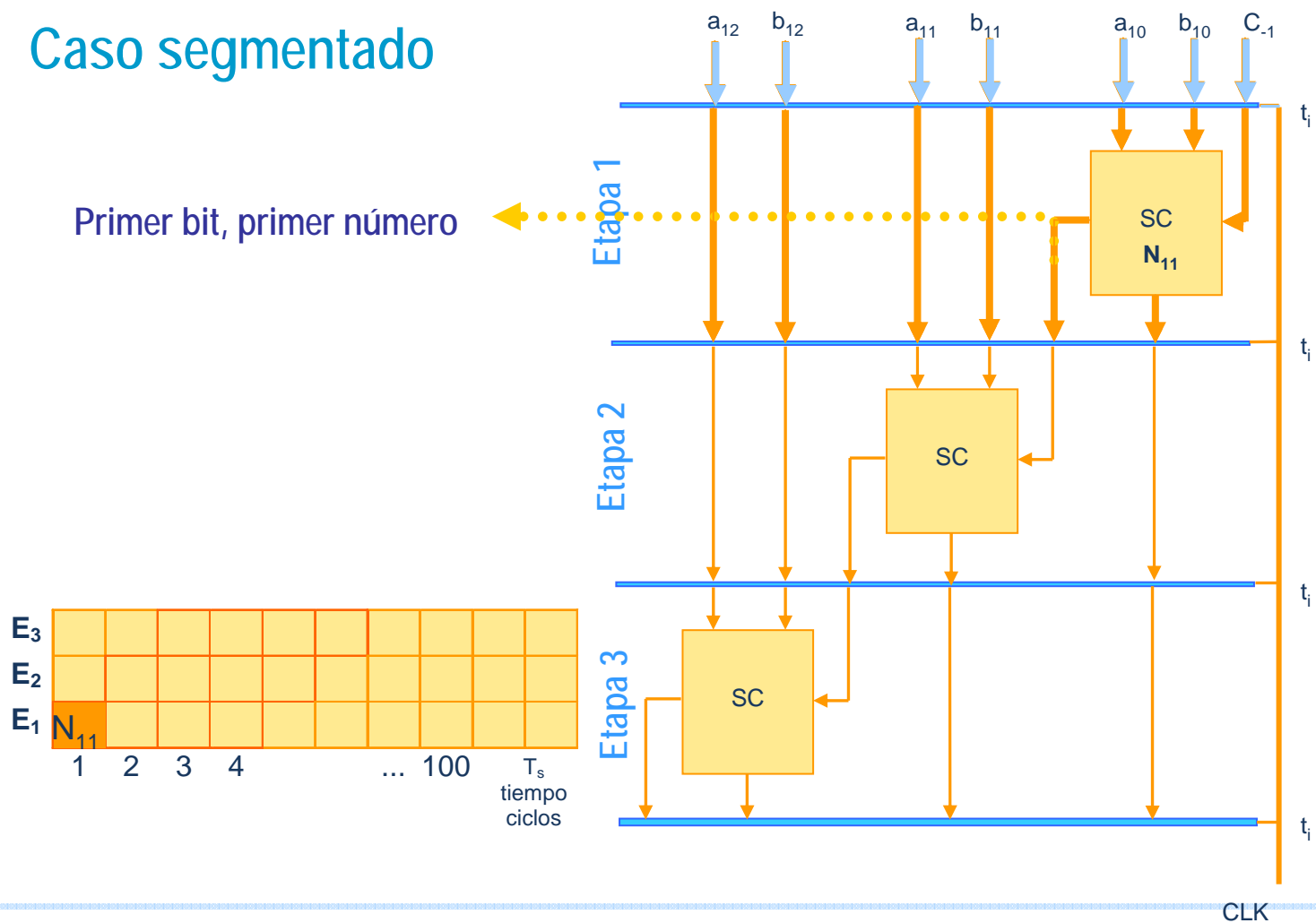
Superescalares

Segmentación

## Sumador de tres bits con propagación de acarreo

### Caso segmentado

Primer bit, primer número



# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

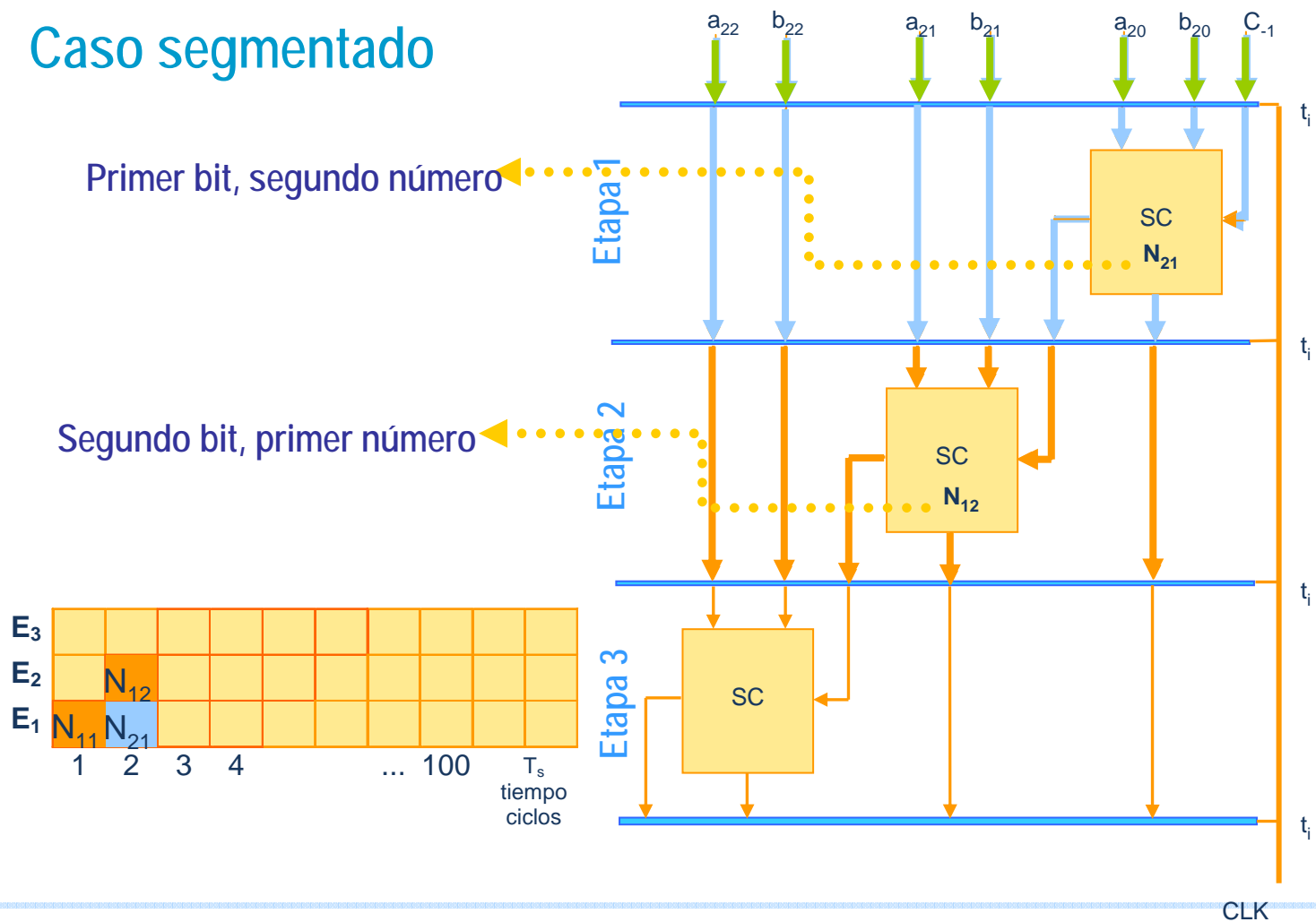
Optimización

Superescalares

Segmentación

## Sumador de tres bits con propagación de acarreo

### Caso segmentado



## Sumador de tres bits con propagación de acarreo

## Caso segmentado

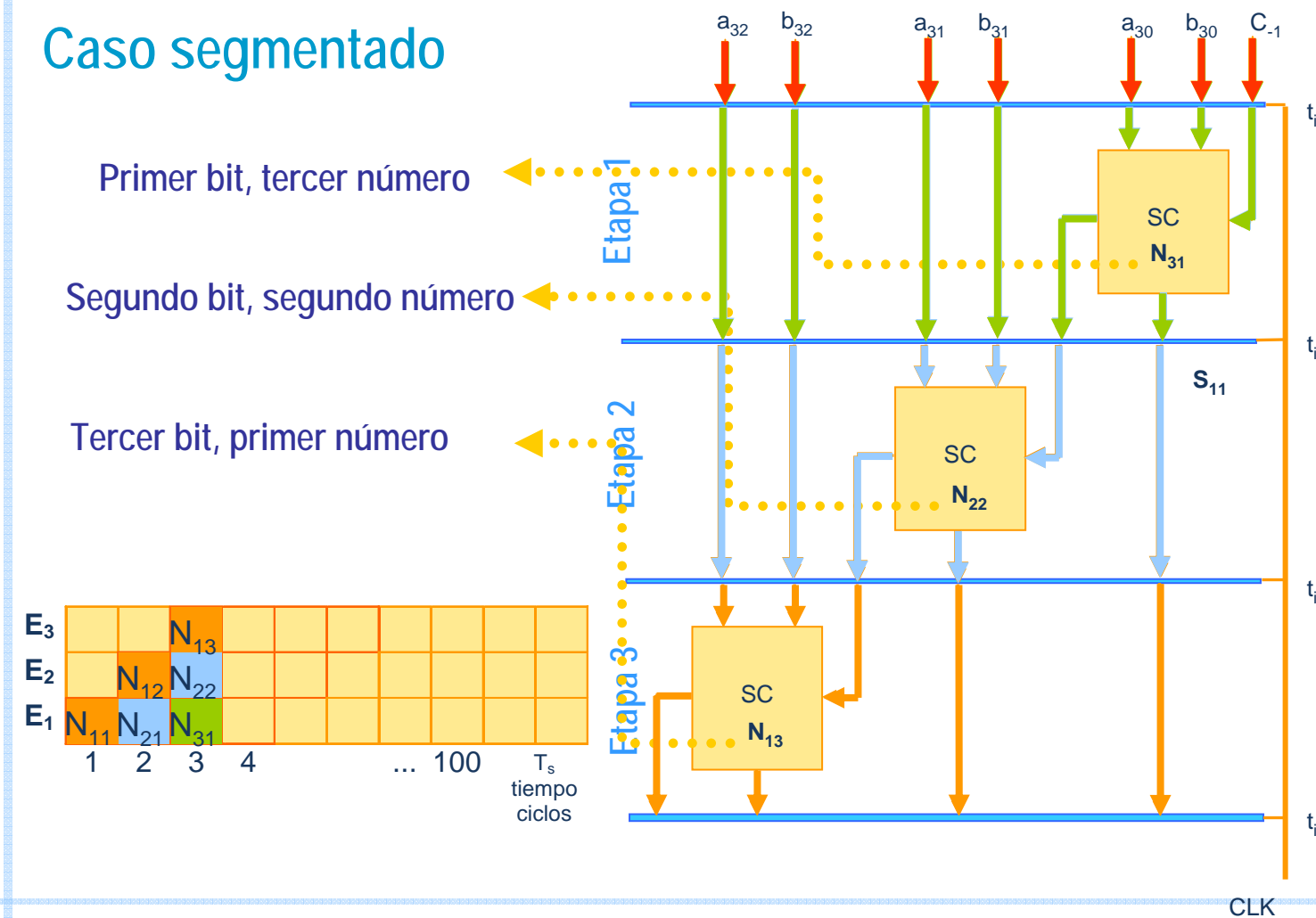


Diagrama de un arreglo de memoria de 3x100. Las filas están etiquetadas  $E_1$ ,  $E_2$ ,  $E_3$  y las columnas como 1, 2, 3, 4, ..., 100,  $T_s$ . Los elementos  $N_{11}$ ,  $N_{12}$ ,  $N_{13}$  están en la fila  $E_1$ ;  $N_{21}$ ,  $N_{22}$  están en la fila  $E_2$ ;  $N_{31}$  está en la fila  $E_3$ . Los elementos  $N_{11}$ ,  $N_{12}$ ,  $N_{13}$ ,  $N_{21}$ ,  $N_{22}$ ,  $N_{31}$  están resaltados en colores diferentes (naranja, azul, verde) para indicar accesos.

# Segmentación

# Segmentación aritmética

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Sumador de tres bits con propagación de acarreo
- El coste temporal de una suma aislada es incluso mayor.
- Tiempo secuencial para 100 números:

$$T_{\text{Secuencial}} = 100(2T_c + T_s) = 200T_s$$

$$\text{Si } T_c \approx \frac{1}{2}T_s$$

- Tiempo segmentado para 100 números:

$$T_{\text{segmentado}} = 3T_s + 99T_s = 102T_s$$

- Aceleración de aproximadamente 1,96 para 100 números

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

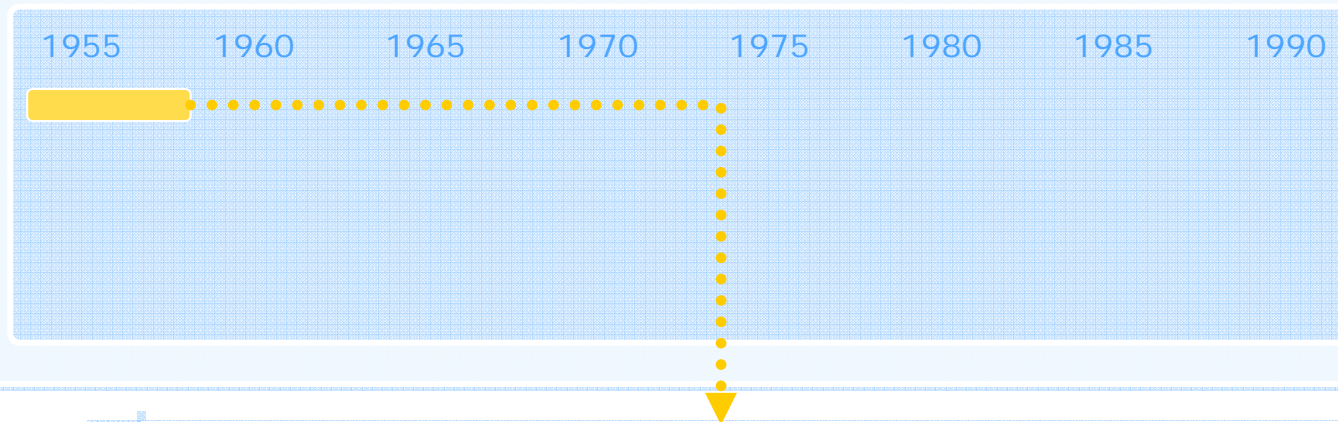
Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

## Hitos



**IBM 709 (1958): primer computador segmentado.** Computador de tubos de vacío y memorias de núcleo de ferrita

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

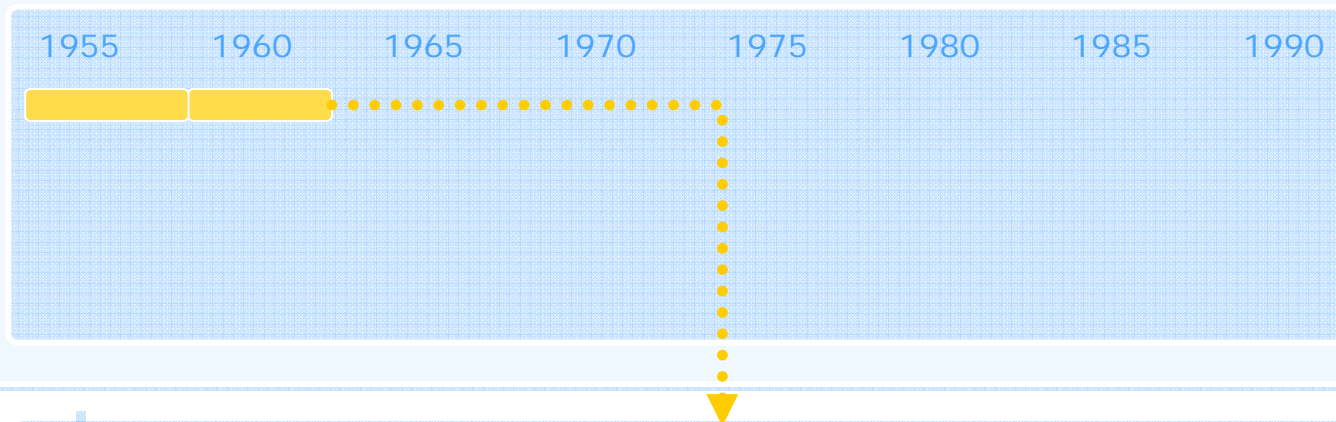
Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

## Hitos



IBM 7030 Stretch (1962): aparece por **primera vez el término segmentación**. Primer procesador de propósito general segmentado.



# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

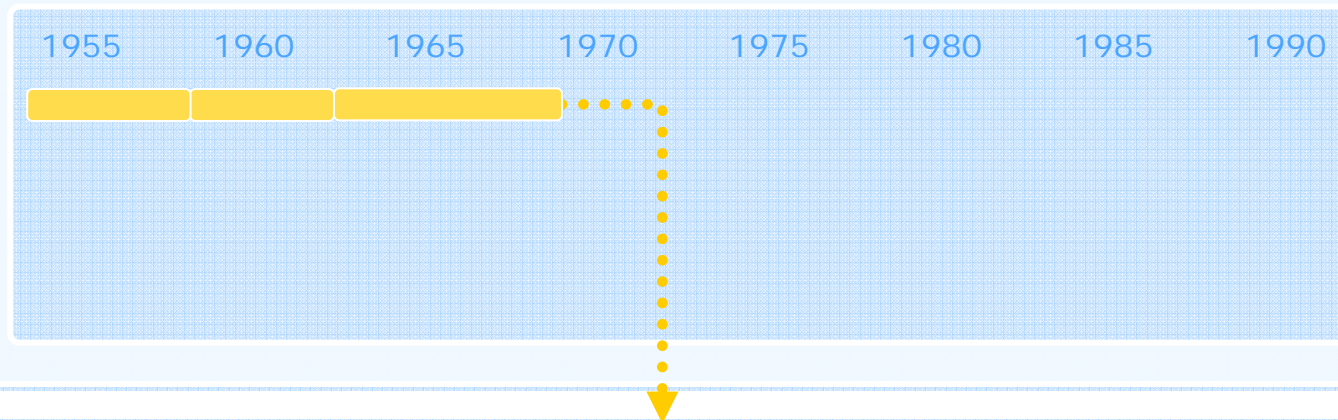
Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

## Hitos



Finales 60: la **segmentación** pasa a un **nivel secundario** dado el énfasis que se realiza en las máquinas orientadas a simplificar el software (VAX).

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

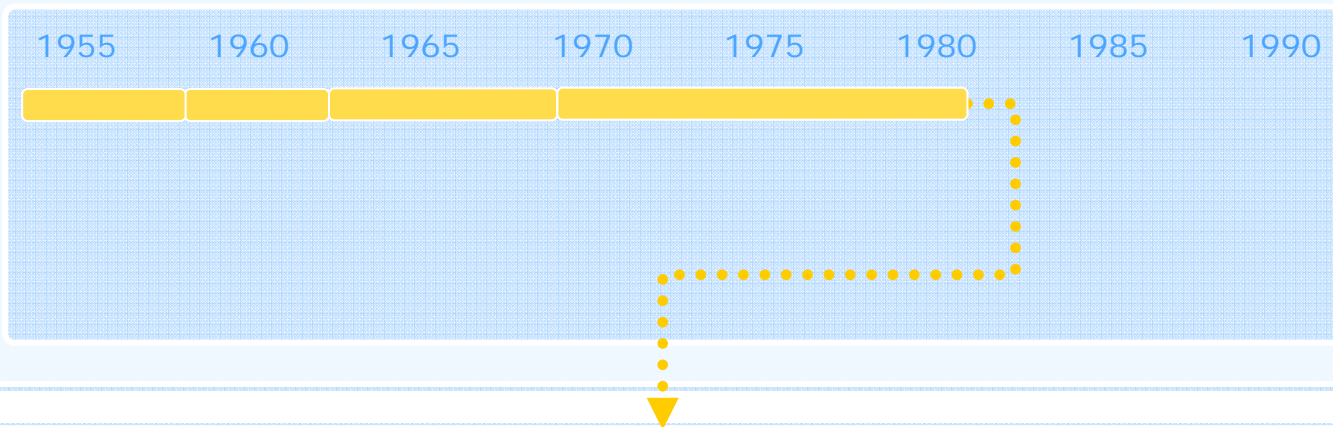
Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

## Hitos



**Años 80:** las **arquitecturas RISC** retoman la segmentación. Las características favorecen el diseño del cauce. Se diseñaron para ser procesadores segmentados

- MIPS R2000 de 5 etapas

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

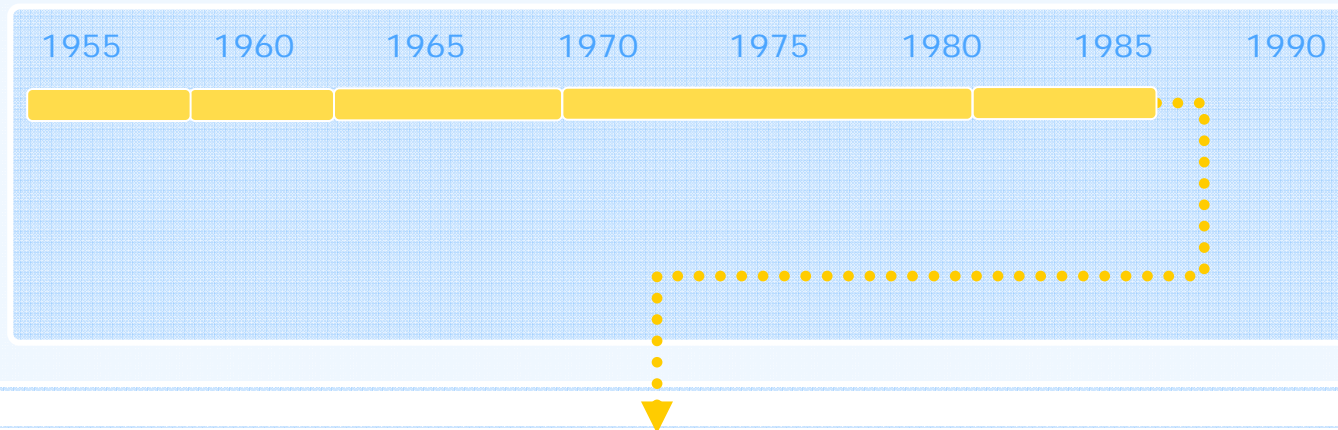
Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

## Hitos



Mediados de los 80: los microprocesadores incorporan segmentación de cauce internamente de forma generalizada

- Intel 80286 con 4 etapas
- Intel 80386 de 6 etapas
- Intel 80486 de 5 etapas

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

**Mediados de los 80:** Se introducen **más recursos** para trabajar de forma paralela

- MIPS R4000: uno de los primeros en incorporar un cauce profundo, supersegmentado de 8 etapas
- Alpha 21064: similar al MIPS R4000 para tratar enteros y una mayor segmentación para punto flotante
- Procesadores más modernos
  - MIPS R10000/120000
  - Sun Ultra Sparc III
  - PowerPC 603, G3 y G4
  - Alpha 21264.

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

Mediados de los 80: PCs traducen instrucciones CISC a microoperaciones RISC superescalares

- Pentium PRO, Pentium II, III y 4 de Intel
- K5, K6, K7, Athlon de AMD

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

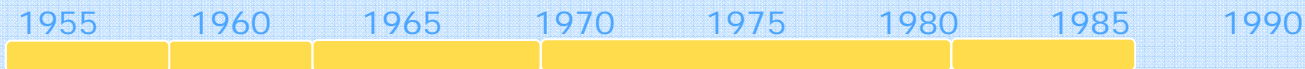
Optimización

Superescalares

Segmentación

- La instrucción a ejecutar se descompone en fases y se encauzan
- Esta técnica o su fundamento, se emplea en la mayoría de los procesadores para aumentar el rendimiento de la CPU

## Hitos



Los 90: **reutilización** de la técnica en microprocesadores **embebidos**.

- MIPS R4300 (1997). Fabricado por NEC. Gran cantidad de aplicaciones.
  - **Nintendo 64**.
  - Utiliza las 5 etapas MIPS.
- ARM de Advanced Risc Machines y SH de Hitachi. Rediseño de la arquitectura RISC para proporcionar un cociente prestaciones/potencia
  - **PDA's, TDT's** y dispositivos similares.

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

## Procesador MIPS

- **MIPS64** procesador **diseñado** expresamente para **trabajar de forma segmentada**.
- Su **simplicidad** hace más fácil demostrar los principios
  - Sencilla arquitectura de **carga almacenamiento**.
  - Sencillo repertorio de **instrucciones fácilmente decodificables**
  - Diseño de segmentación **eficiente**
  - Tres tipos de instrucciones: ALU, carga/almacenamiento, saltos

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

## ❖ Paso IF. Búsqueda de instrucción

## ❖ Paso ID. Decodificación de instrucción y búsqueda de registros

- ❖ Formato permite leer los registros ya que siempre están ubicados en las mismas posiciones
- ❖ Instrucción de salto: calcular la dirección y hacer la comprobación de los registros (dependerá de la implementación)

## ❖ Paso EX. Ejecución y cálculo de direcciones efectivas

- ❖ Los operandos en memoria sólo en las operaciones de carga y almacenamiento por lo que en esta fase se puede calcular la dirección de memoria.

## ❖ Paso MEM. Acceso a memoria

- ❖ Instrucción de carga se accede a memoria para la lectura del dato.
- ❖ Instrucción almacenamiento se guarda el valor del registro en memoria.

## ❖ Paso WB. Postescritura, escritura del resultado en un registro



# Segmentación de instrucciones

Introducción

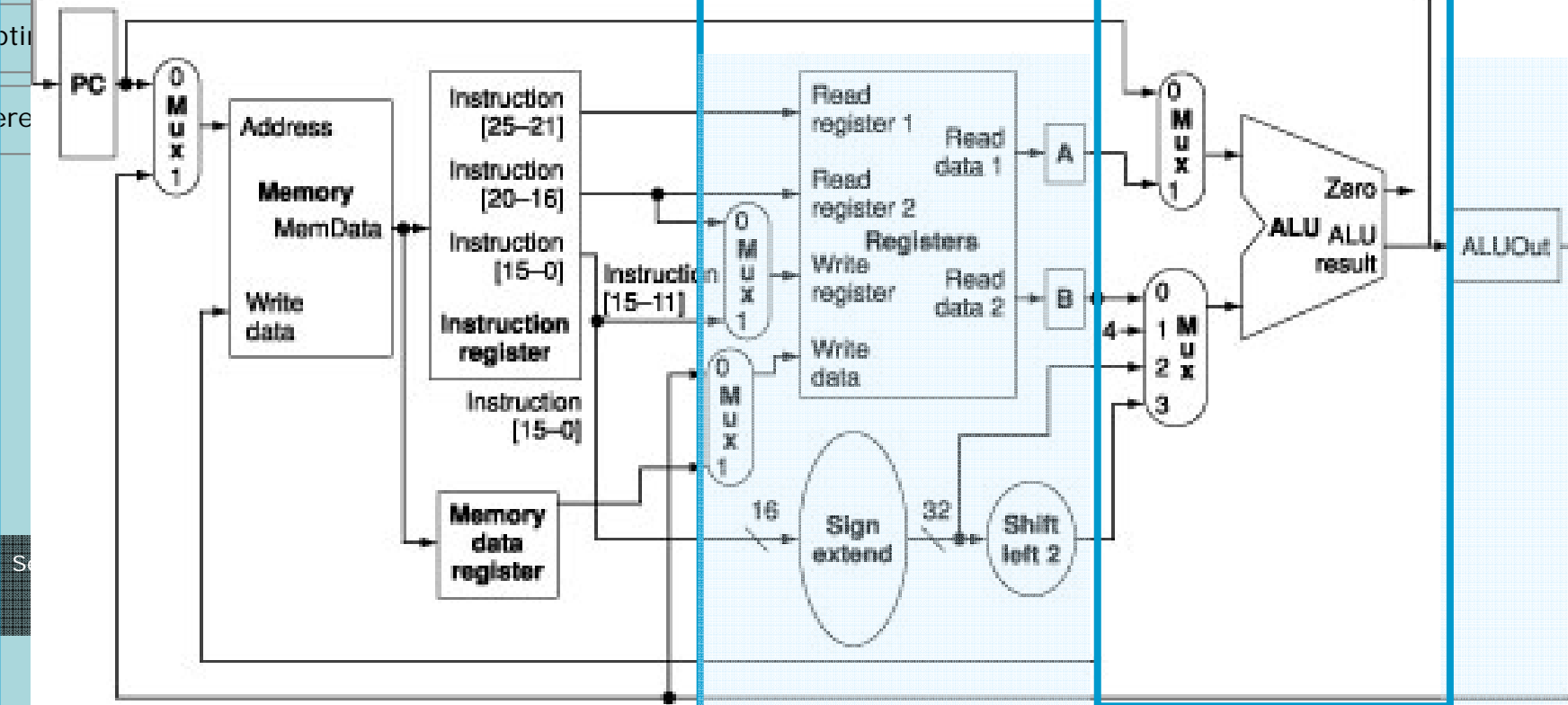
Segmentación  
del repertorio

Cauces

aritméticos

Optim

Supere



IF: Búsqueda de instrucción

$IR \leftarrow M[PC]; PC \leftarrow PC + 4$

Implementación multiciclo

# Introducción

## Segmentación del mercado

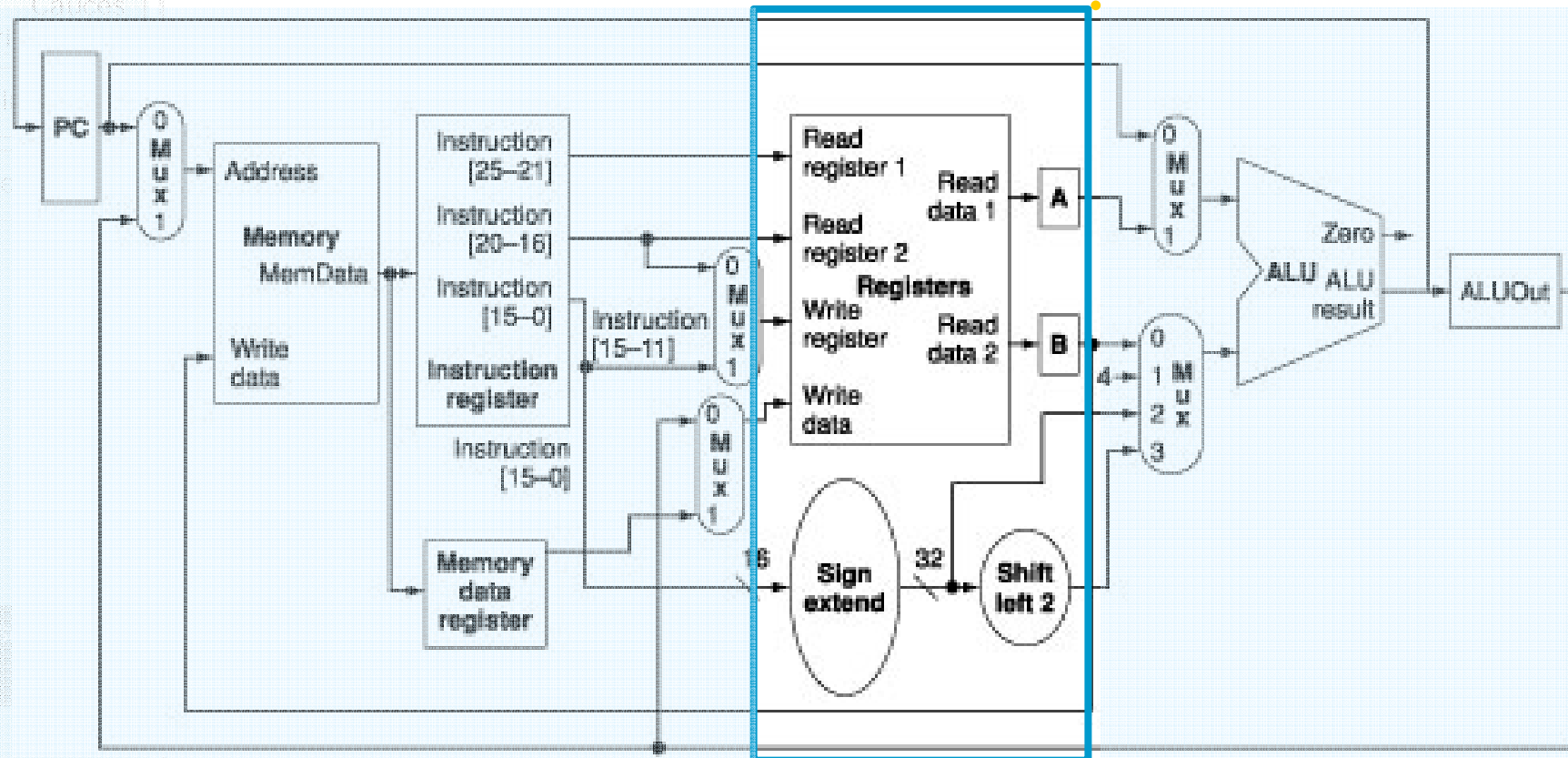
## Implementación multiciclo

Causes

Copyright © 2004 by John Wiley & Sons, Inc.

## Support

ID: Decodificación de instrucción y búsqueda de registros

$$A \leftarrow Rs1; B \leftarrow Rs2;$$


# Segmentación de instrucciones

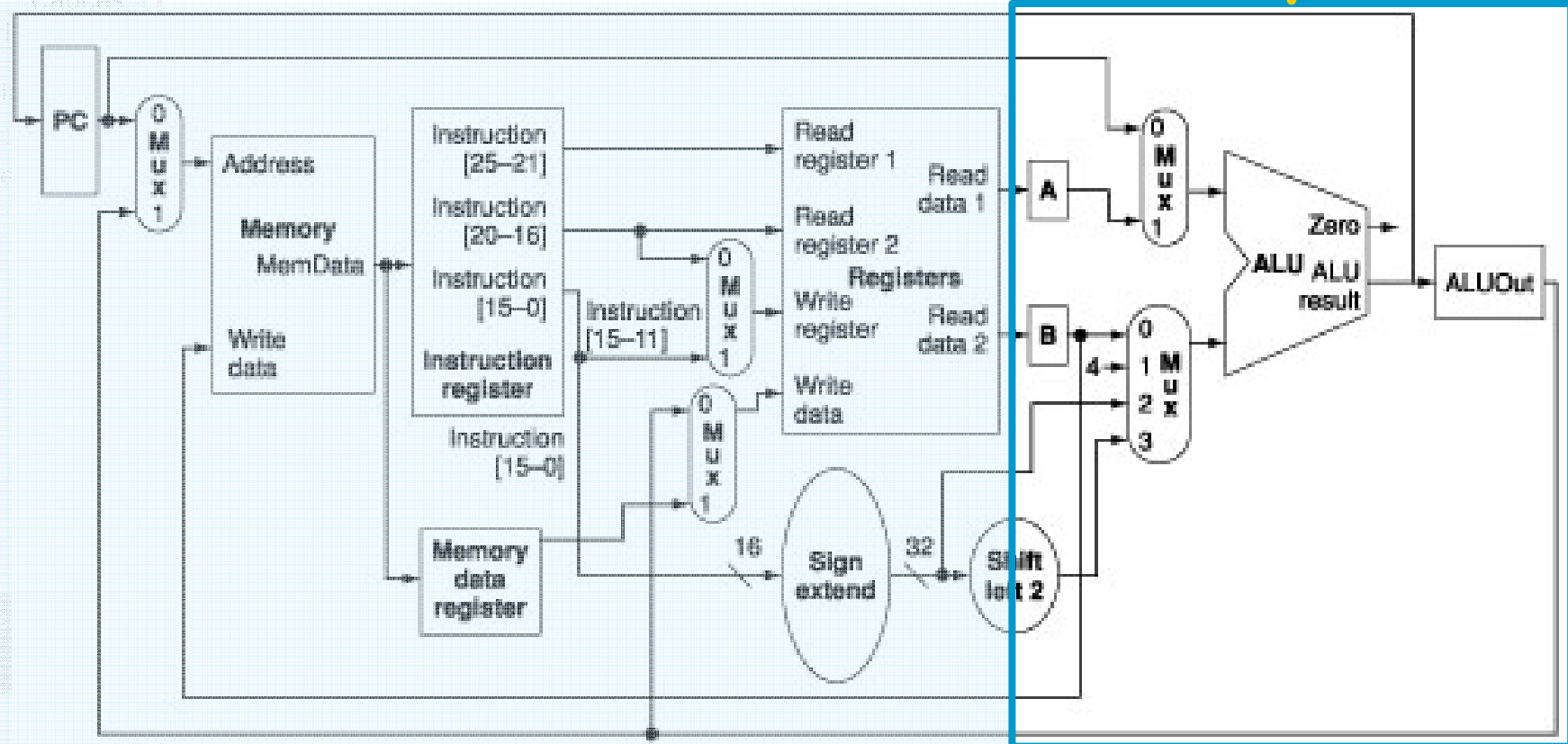
Introducción

Segmentación  
del registro

## Implementación multiciclo

EX: Cálculo de direcciones efectivas

$$\text{ALUOut} \leftarrow A + (\text{IR16})16\#\#\text{IR16}..31;$$



# Segmentación de instrucciones

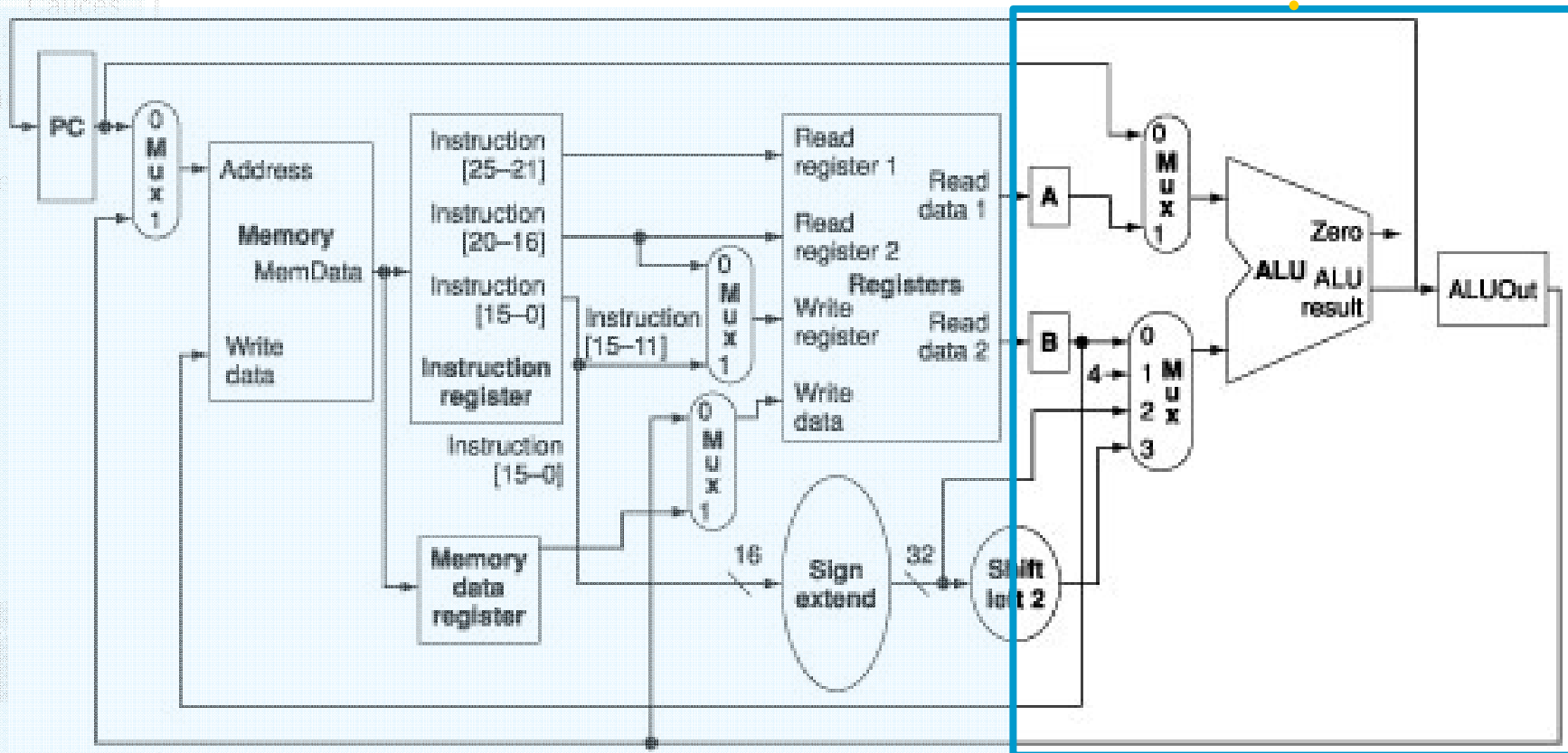
Introducción

Segmentación  
del registro

## Implementación multiciclo

EX: Ejecución

$ALUOutput \leftarrow A \text{ op } (B \text{ or } (IR16)16\#IR16..3)$



# Segmentación de instrucciones

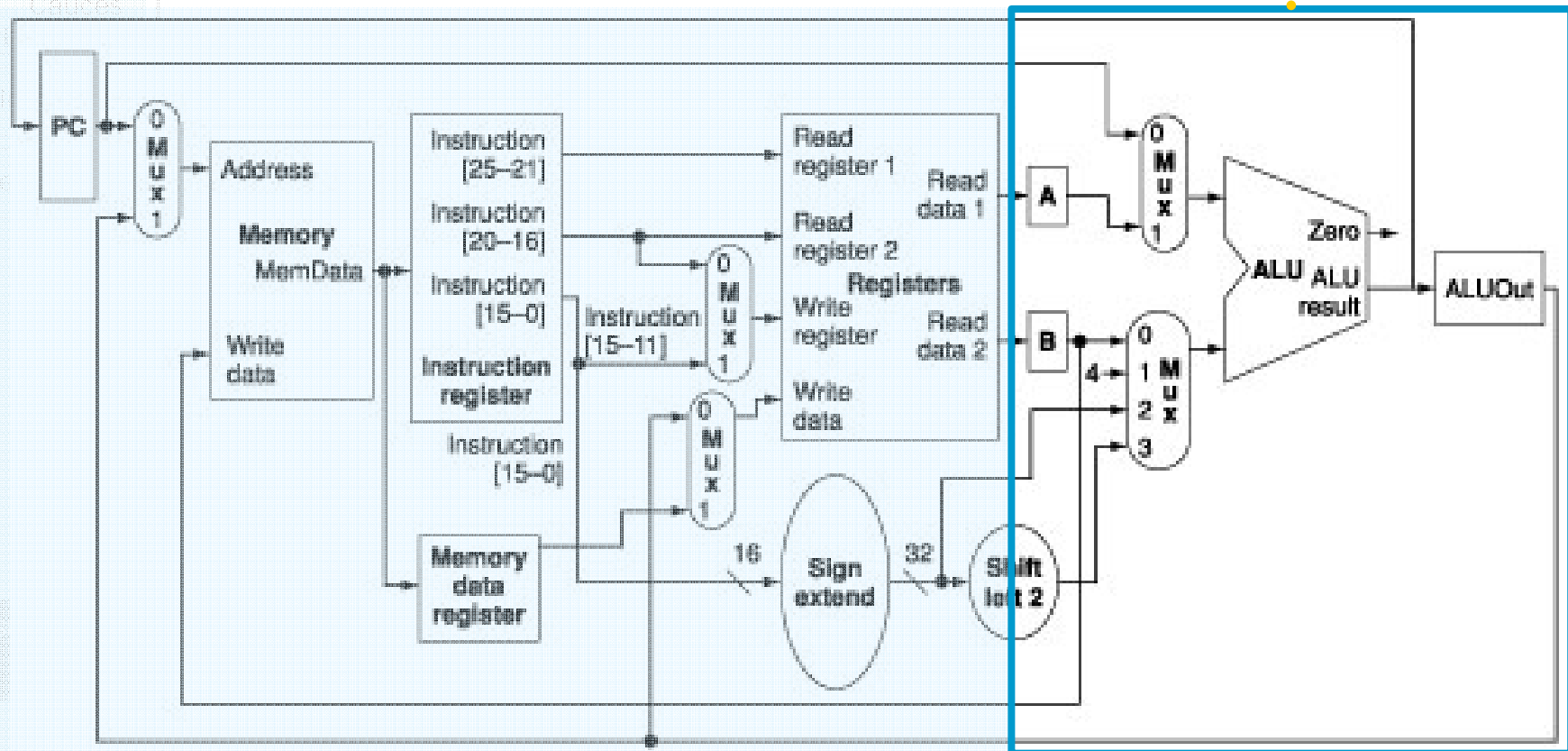
Introducción

Segmentación  
del registro

## Implementación multiciclo

EX: Salto/Bifurcación

$$\text{ALUOutput} \leftarrow \text{PC} + (\text{IR16})16\# \text{IR16..31}$$





# Segmentación de instrucciones

Introducción

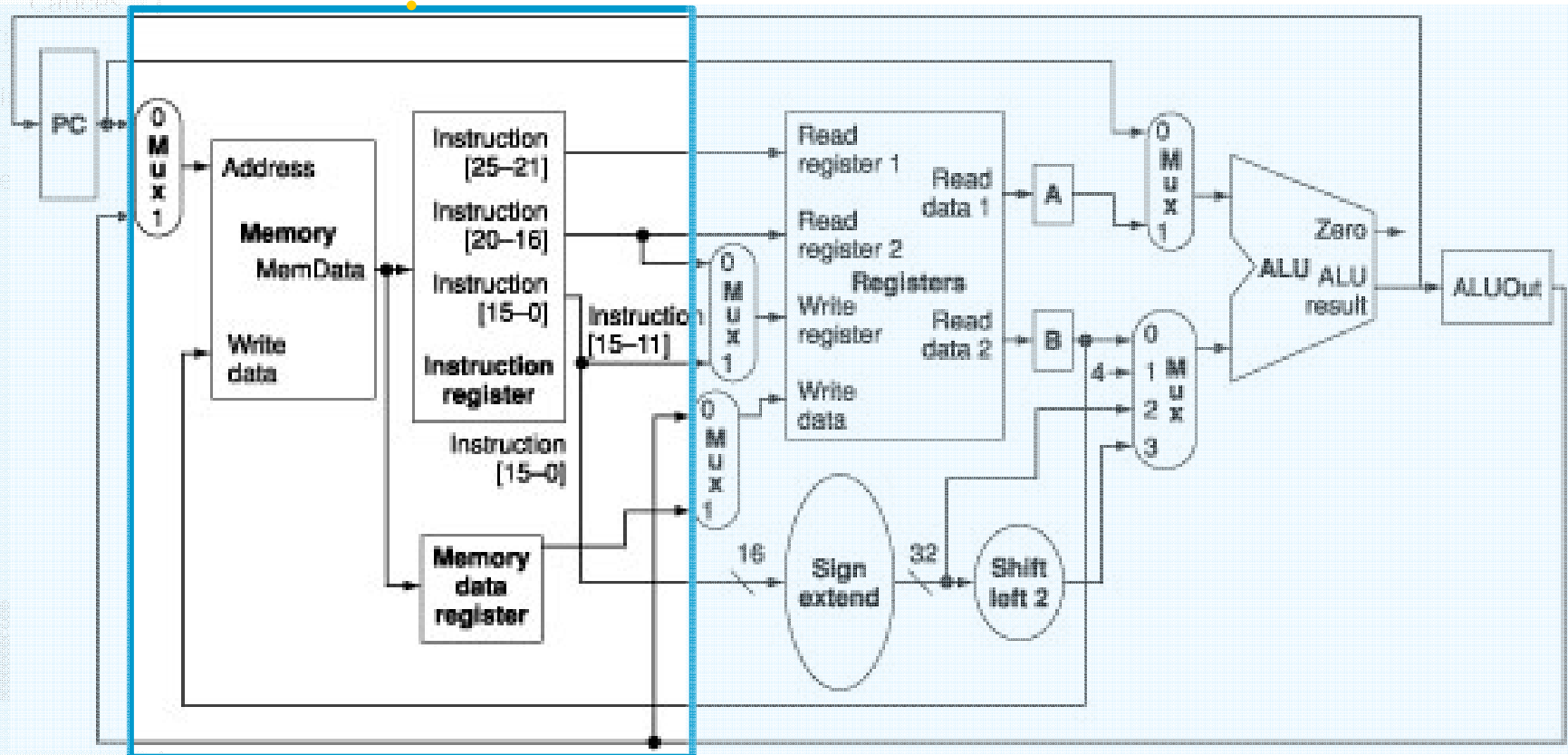
Segmentación  
del registro

## Implementación multiciclo

MEM: Referencia memoria

$MDR \leftarrow M[ALUOut]$

$M[ALUOut] \leftarrow B$



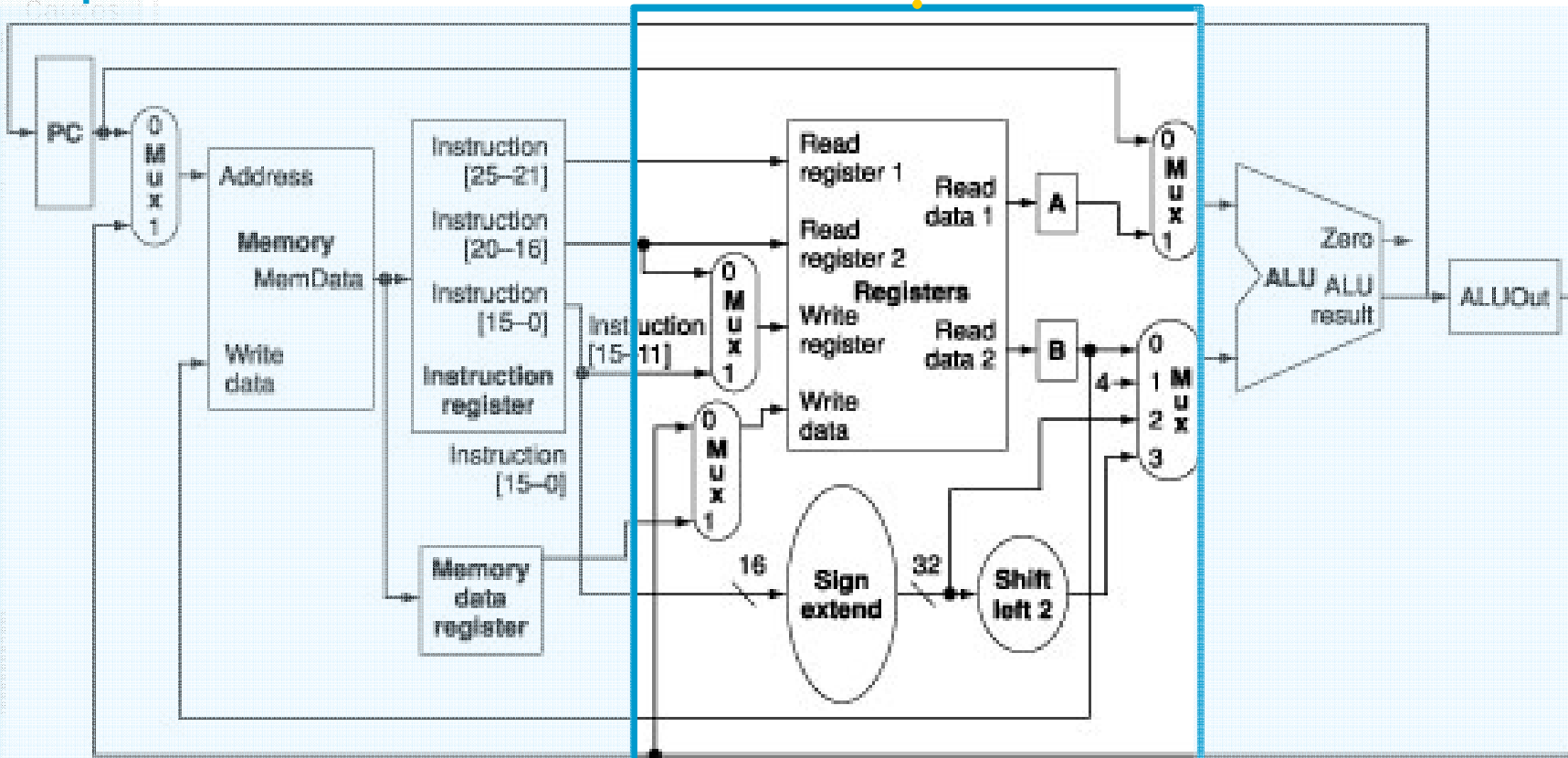
# Segmentación de instrucciones

Introducción

Segmentación  
del registro

## Implementación multiciclo

WB: Postescritura  
 $Rd \leftarrow \text{ALUOut o MDR}$



# Segmentación de instrucciones

- La segmentación consiste en solapar la ejecución de las instrucciones

## Implementación multiciclo

Ciclo reloj	1	2	3	4	5	6	7	8	9
Inst i	IF								
Inst i+1									
Inst i+2									
Inst i+3									
Inst i+4									

En el primer ciclo se carga una instrucción

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación



# Segmentación de instrucciones

- La segmentación consiste en solapar la ejecución de las instrucciones

## Implementación multiciclo

Ciclo reloj	1	2	3	4	5	6	7	8	9
Inst i	IF	ID							
Inst i+1		IF							
Inst i+2									
Inst i+3									
Inst i+4									

En el segundo ciclo se carga una nueva y se decodifica la anterior

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

# Segmentación de instrucciones

- La segmentación consiste en solapar la ejecución de las instrucciones

## Implementación multiciclo

Ciclo reloj	1	2	3	4	5	6	7	8	9
Inst i	IF	ID	EX						
Inst i+1		IF	ID						
Inst i+2			IF						
Inst i+3									
Inst i+4									

En el tercer ciclo se carga una nueva, se decodifica la anterior y se ejecuta la primera

Introducción

Segmentación del repertorio

Cauces aritméticos

Optimización

Superescalares

Segmentación

# Segmentación de instrucciones

- La segmentación consiste en solapar la ejecución de las instrucciones

## Implementación multiciclo

Ciclo reloj	1	2	3	4	5	6	7	8	9
Inst i	IF	ID	EX	MEM					
Inst i+1		IF	ID	EX					
Inst i+2			IF	ID					
Inst i+3				IF					
Inst i+4									

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

# Segmentación de instrucciones

- La segmentación consiste en solapar la ejecución de las instrucciones

## Implementación multiciclo

Ciclo reloj	1	2	3	4	5	6	7	8	9
Inst i	IF	ID	EX	MEM	WB				
Inst i+1		IF	ID	EX	MEM				
Inst i+2			IF	ID	EX				
Inst i+3				IF	ID				
Inst i+4					IF				

Cuando se llena el cauce se ejecuta una instrucción cada ciclo

Introducción

Segmentación del repertorio

Cauces aritméticos

Optimización

Superescalares

Segmentación

# Segmentación de instrucciones

Introducción

Segmentación del repertorio

Cauces aritméticos

Optimización

Superescalares

Segmentación

- La segmentación consiste en solapar la ejecución de las instrucciones

## Implementación multiciclo

Ciclo reloj	1	2	3	4	5	6	7	8	9
Inst i	IF	ID	EX	MEM	WB				
Inst i+1		IF	ID	EX	MEM	WB			
Inst i+2			IF	ID	EX	MEM	MEM		
Inst i+3				IF	ID	EX	MEM	WB	
Inst i+4					IF	ID	EX	MEM	WB

- Cada paso constituye una etapa de la segmentación.
- Cada ciclo, cinco instrucciones en ejecución
- La segmentación incrementa la **productividad** sin reducir el tiempo de ejecución de una instrucción individual.
- Los programas se ejecutan más rápido

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

## Ejemplo:

Considerar una máquina no segmentada con 5 pasos de ejecución cuyas duraciones son 50ns, 50ns, 60ns, 50ns, 50ns.

Suponer que debido al tiempo de preparación y sesgo de reloj, segmentar la máquina añade 5 ns de gasto a cada etapa de ejecución.

¿Cuánta velocidad se ganará con la segmentación en la frecuencia de ejecución de las instrucciones?

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

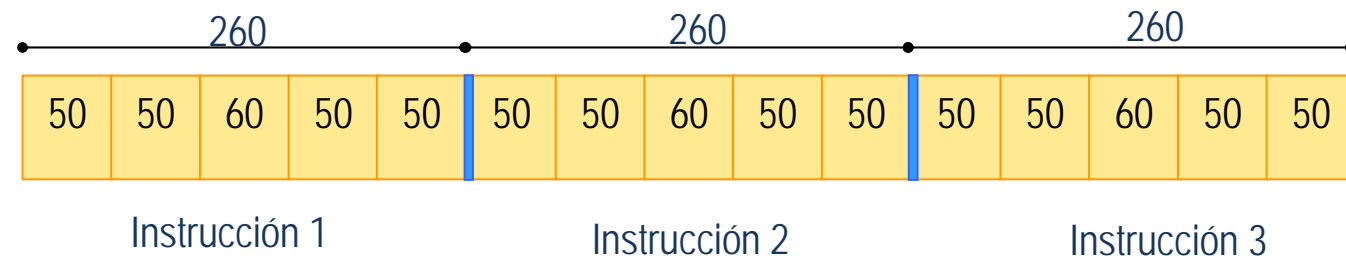
Cauces  
aritméticos

Optimización

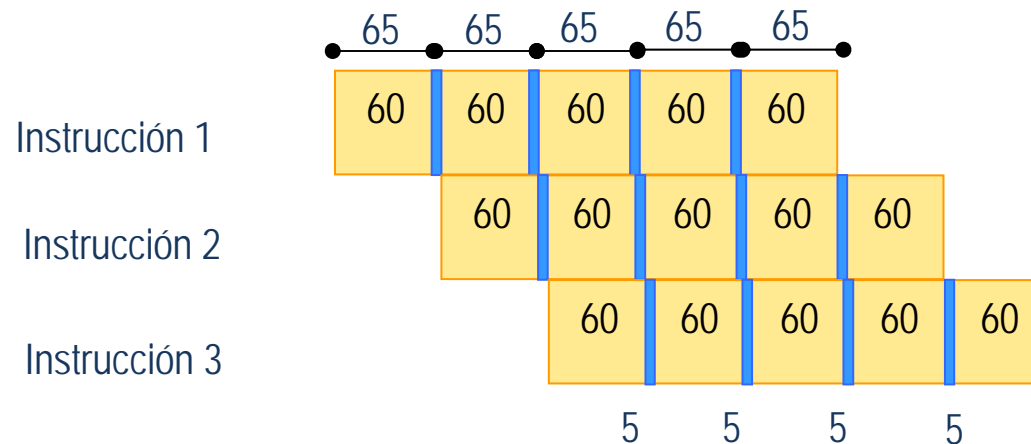
Superescalares

Segmentación

## Ejecución secuencial



## Ejecución segmentada



$$Aceleración = \frac{\text{Tiempo medio}_{\text{sin segmentación}}}{\text{Tiempo medio}_{\text{con segmentación}}} = \frac{260}{65} = 4$$

# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

## ⚙️ Riesgos de segmentación

- ⚙️ Riesgos estructurales. Surgen de conflictos de los recursos cuando el hardware no puede soportar todas las combinaciones de instrucciones en ejecución (p.ej. acceso a memoria, banco registros, etc).
- ⚙️ Riesgos por dependencias de datos. Surgen cuando una instrucción depende de los resultados de una instrucción anterior.
- ⚙️ Riesgos de control. Surgen de la segmentación de los saltos y otras instrucciones que cambian el PC.



# Segmentación de instrucciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

## ⚙️ Riesgos de segmentación

- ⚙️ Impiden que se ejecute la siguiente instrucción
- ⚙️ Pueden hacer necesario detener el cauce
- ⚙️ Cuando una instrucción se detiene las instrucciones posteriores a esta también lo hacen
- ⚙️ Detención disminuye la ganancia con respecto a la ideal

# Segmentación de procesadores

Introducción

Segmentación  
del repertorio

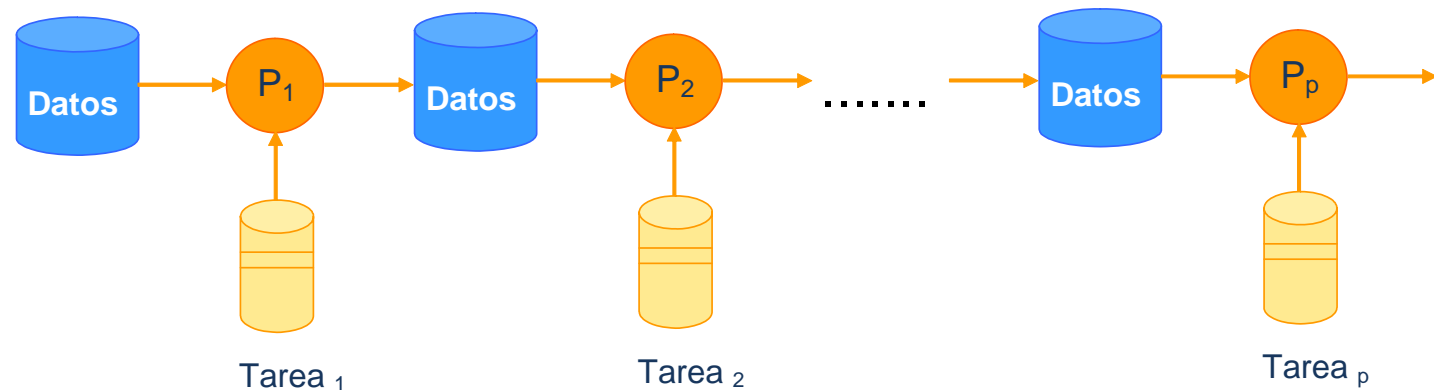
Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Las etapas del cauce están formadas por distintos procesadores que realizan, distintas operaciones sobre el flujo de datos
- Propio de la computación sistólica y utilizada en la implementación de algoritmos de alto coste.



# Segmentación de procesadores

Introducción

Segmentación  
del repertorio

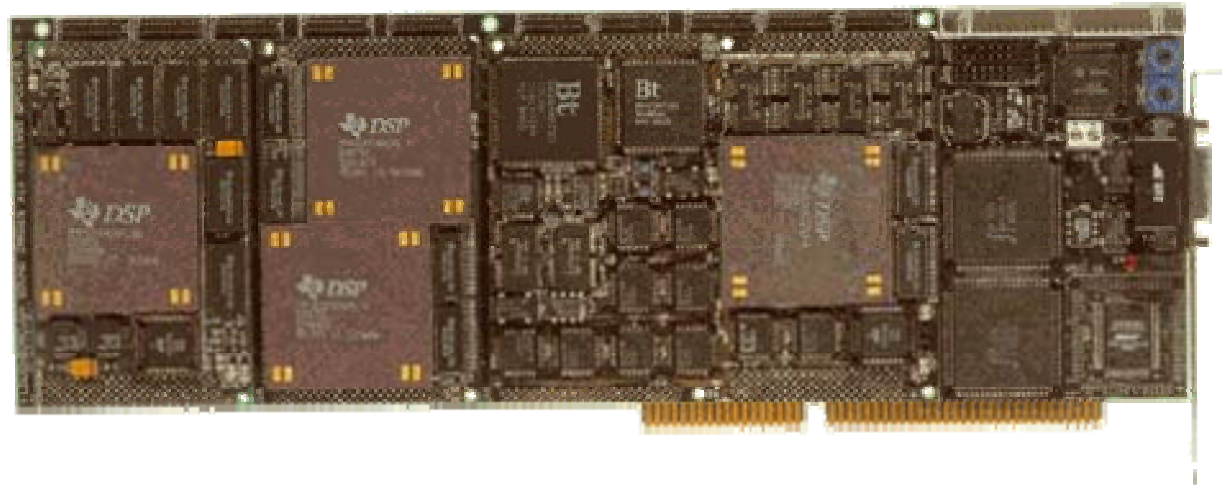
Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Ejemplo: Implementación de filtros de morfología matemática sobre DSPs (Digital Signal Processor). (Plataforma con 4 DSPs)



- La morfología matemática proporciona una herramienta para extraer componentes de la imagen tales como contornos, esqueletos y formas convexas.

# Análisis de prestaciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

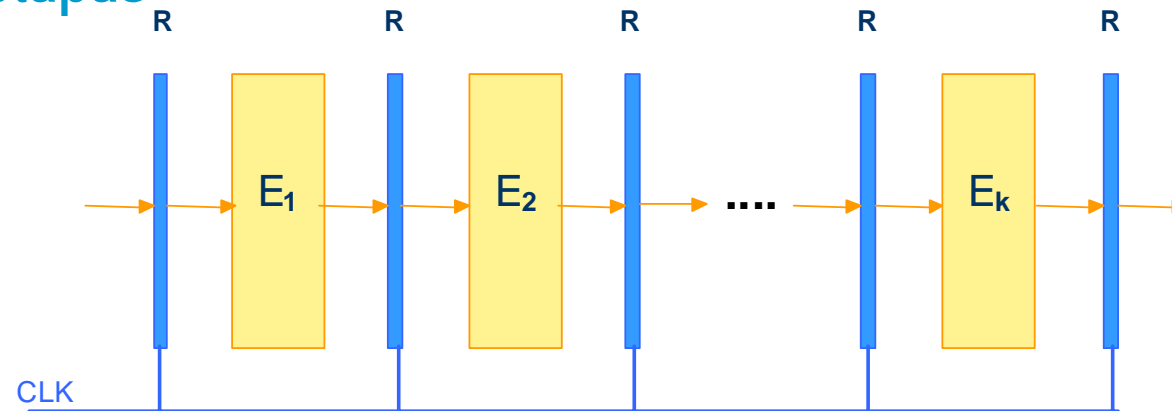
Optimización

Superescalares

Segmentación

## Unidad segmentada lineal síncrona

$K$  etapas



$R$  = Registros de almacenamiento intermedio

$E_i$  = Circuitos combinacionales para operaciones

CLK = Señal de reloj que controla el flujo de datos.

$t_i$  = Retardo temporal de cada etapa  $E_i$

$t_r$  = Es el retardo de cada registro  $R$ .

# Análisis de prestaciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

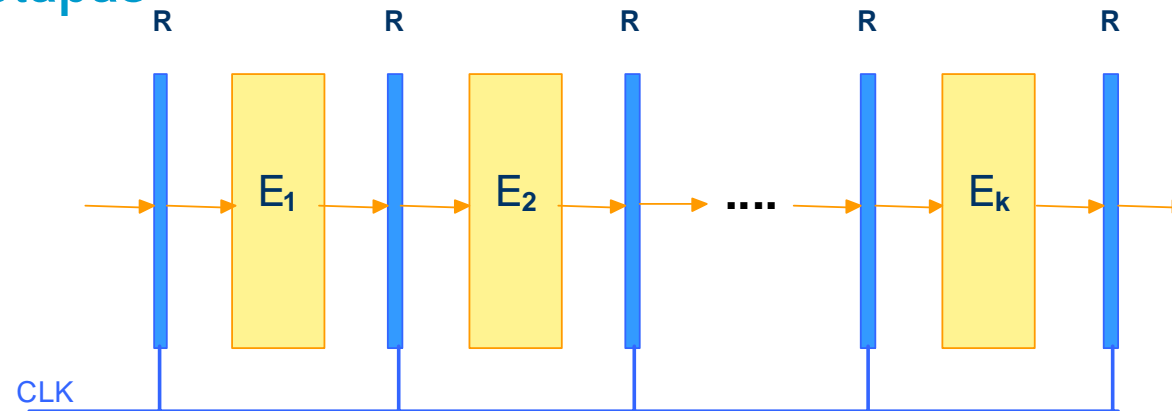
Optimización

Superescalares

Segmentación

## Unidad segmentada lineal síncrona

K etapas



### Periodo de reloj del cauce

$$CLK = \max \{t_i\}_{i=1}^k + t_r$$

- Sesgo de reloj (retardo del pulso). El instante de comienzo de las etapas debe incluir el retardo del pulso de reloj

$$CLK \geq \max \{t_i\}_{i=1}^k + t_r + s$$

# Análisis de prestaciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

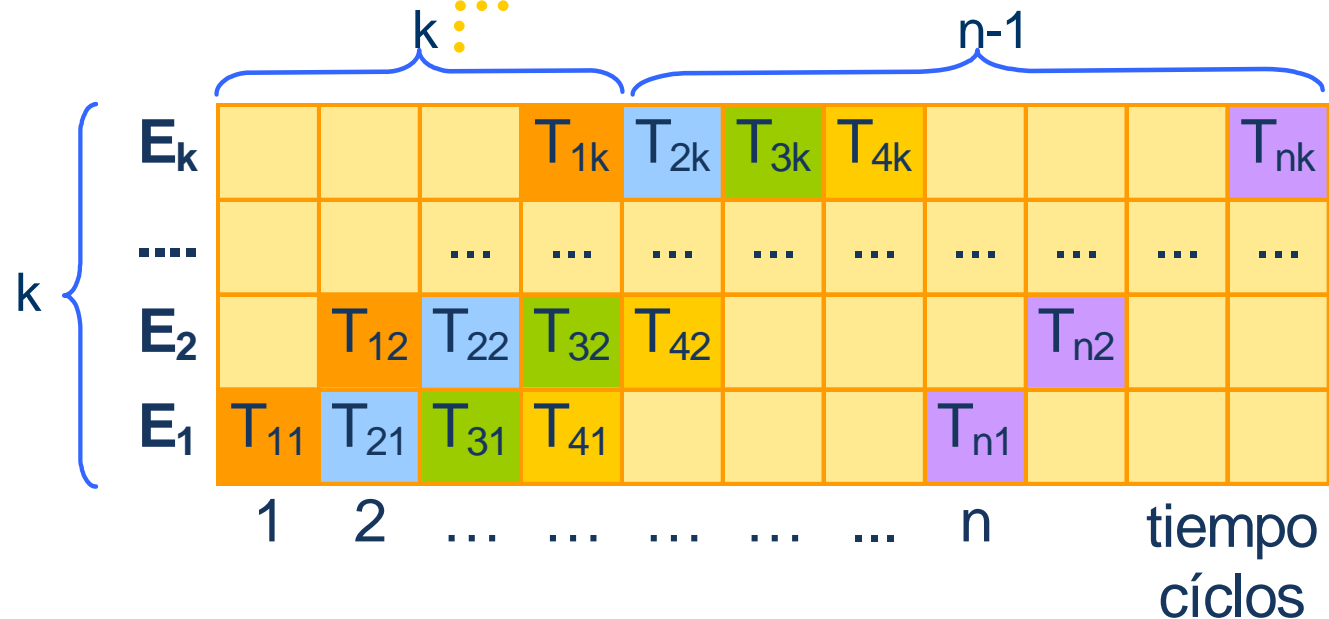
Superescalares

Segmentación

## Unidad segmentada lineal síncrona

Tiempo para procesar n tareas

$$T_{SEG} = k \cdot CLK$$



# Análisis de prestaciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

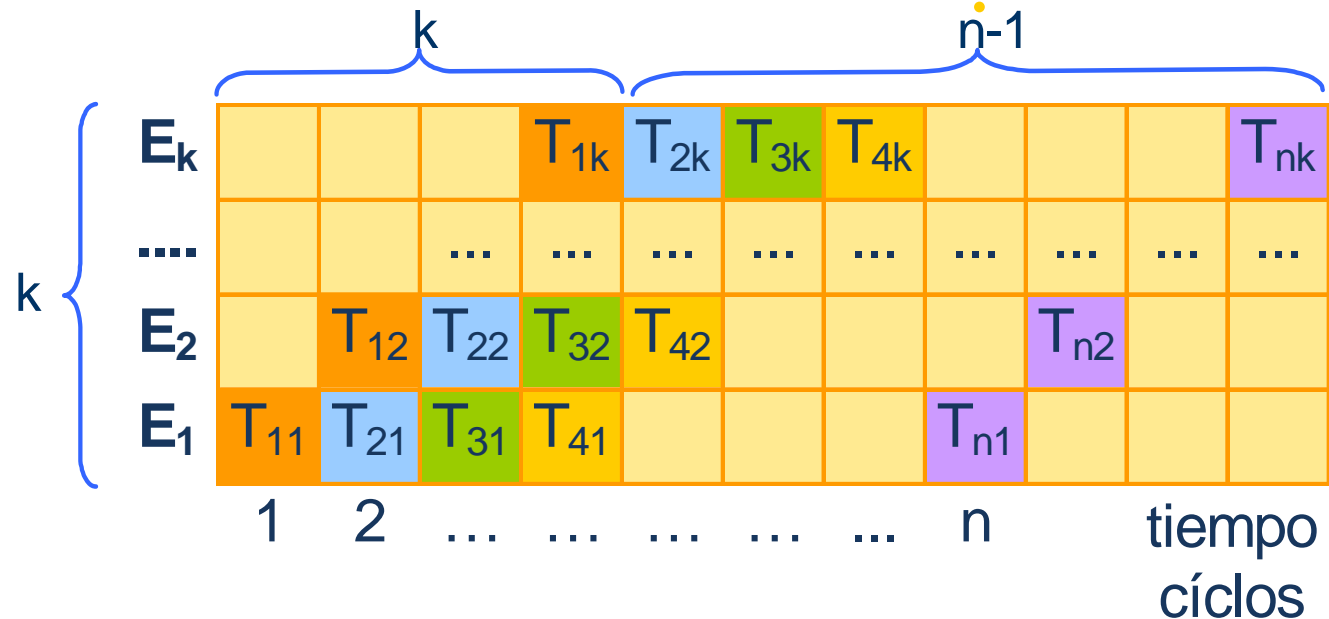
Superescalares

Segmentación

## Unidad segmentada lineal síncrona

Tiempo para procesar n tareas

$$T_{SEG} = k \cdot CLK + (n - 1) \cdot CLK$$



# Análisis de prestaciones

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

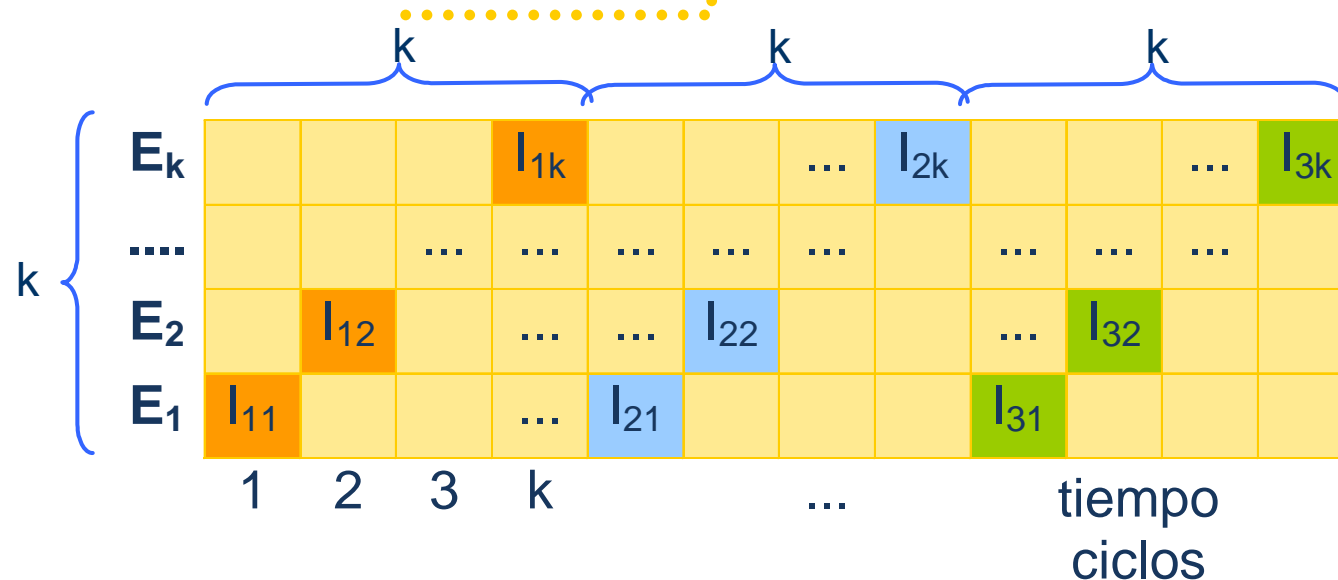
Superescalares

Segmentación

## Unidad segmentada lineal síncrona

Tiempo equivalente para un proceso no encauzado

$$T_{SEC} = K \cdot CLK \cdot n$$





# Ganancia de velocidad

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Incremento de velocidad de un cauce de  $k$  etapas respecto del proceso secuencial

$$G_k = \frac{T_{SEC}}{T_{SEG}} = \frac{n \cdot k \cdot CLK}{(k + n - 1)CLK} = \frac{n \cdot k}{k + n - 1}$$

- Cuando  $n \rightarrow \infty$

$$\lim_{n \rightarrow \infty} G_k = \lim_{n \rightarrow \infty} \frac{n \cdot k}{k + n - 1} = k$$

- En la práctica diversos condicionantes hacen que  $G_k < k$ 
  - Imposibilidad de la descomposición óptima de la tarea en  $k$  subtareas.
  - Dependencias entre datos e instrucciones.
  - Bifurcaciones en el programa.

# Eficiencia

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

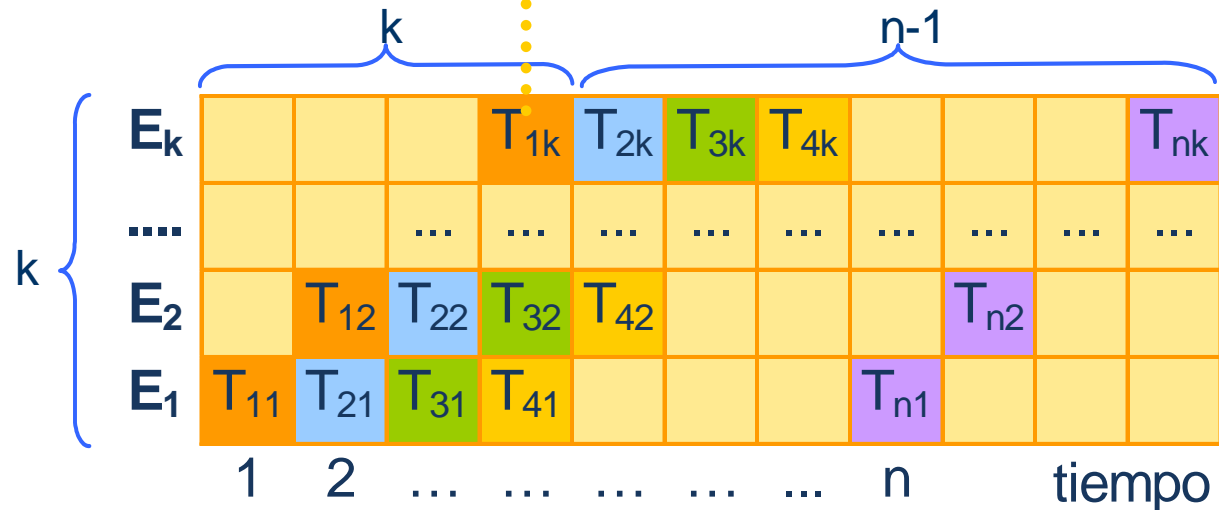
Optimización

Superescalares

Segmentación

- Nos ofrece idea del grado de utilización de los recursos
- Relación entre el número de tramos temporales ocupados y el número total de tramos en dicho periodo de tiempo  $T$

$$T_{ocupado} = k \cdot n \cdot clk$$



# Eficiencia

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

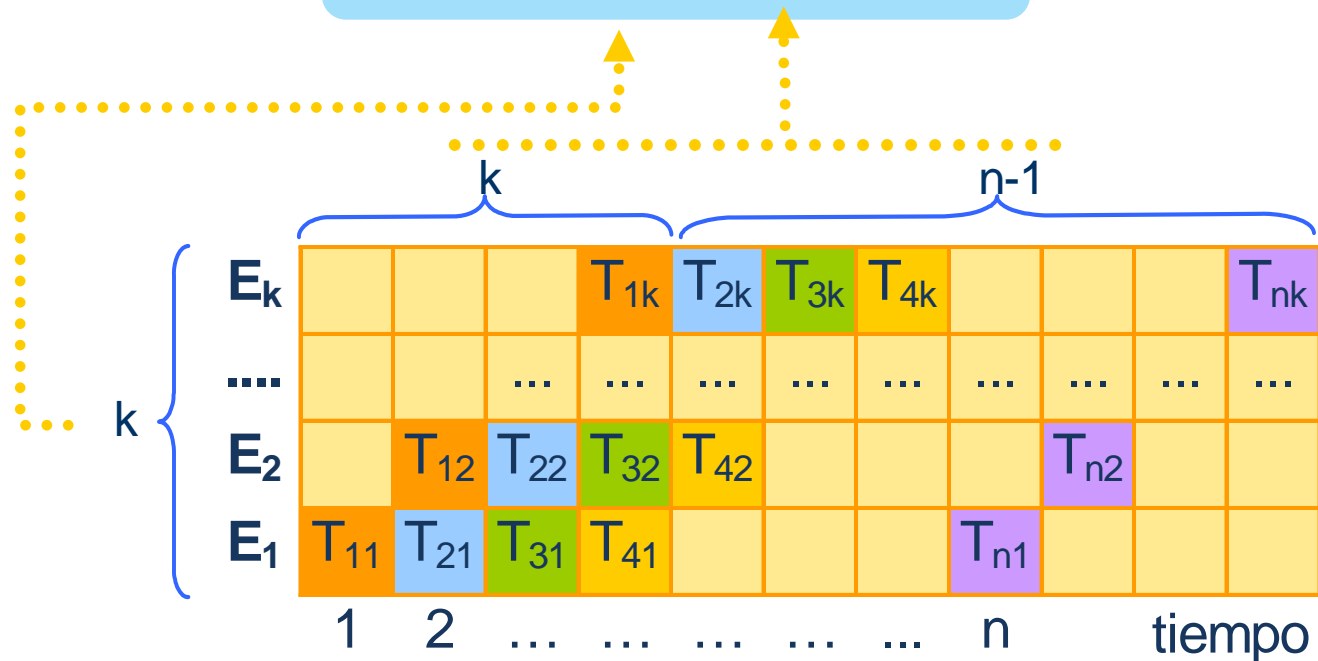
Optimización

Superescalares

Segmentación

- Nos ofrece idea del grado de utilización de los recursos
- Relación entre el número de tramos temporales ocupados y el número total de tramos en dicho periodo de tiempo  $T$

$$T_{total} = k \cdot (k + n - 1) \cdot clk$$



# Eficiencia

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Nos ofrece idea del grado de utilización de los recursos
- Relación entre el número de tramos temporales ocupados y el número total de tramos en dicho periodo de tiempo T

$$E_k = \frac{k \cdot n \cdot CLK}{k(k+n-1)CLK} = \frac{n}{k+n-1} = \frac{G_k}{k}$$

- Cuando  $n \rightarrow \infty$

$$\lim_{n \rightarrow \infty} E_k = \lim_{n \rightarrow \infty} \frac{n}{k+n-1} = 1$$

- En la práctica problemas hacen que  $E_k \ll 1$

# Productividad

Introducción

Segmentación  
del repertorio

Cauces  
aritméticos

Optimización

Superescalares

Segmentación

- Número de datos o instrucciones que puede procesar por unidad de tiempo

$$P_k = \frac{n}{(k+n-1)CLK} = \frac{E_k}{CLK}$$

- Cuando  $n \rightarrow \infty$

$$\lim_{n \rightarrow \infty} P_k = \lim_{n \rightarrow \infty} \frac{n}{(k+n-1)CLK} = \frac{1}{CLK}$$

- La cota máxima coincide con la frecuencia de funcionamiento y corresponde a la aparición de un resultado cada periodo de reloj