Arquitectura e Ingeniería de Computadores

Tema 2 – Segmentación y superescalares

Ingeniería en Informática

Departamento de Tecnología Informática y Computación

AIC – Tema 2 Segmentación y superescalares

(Parte A – Segmentación)

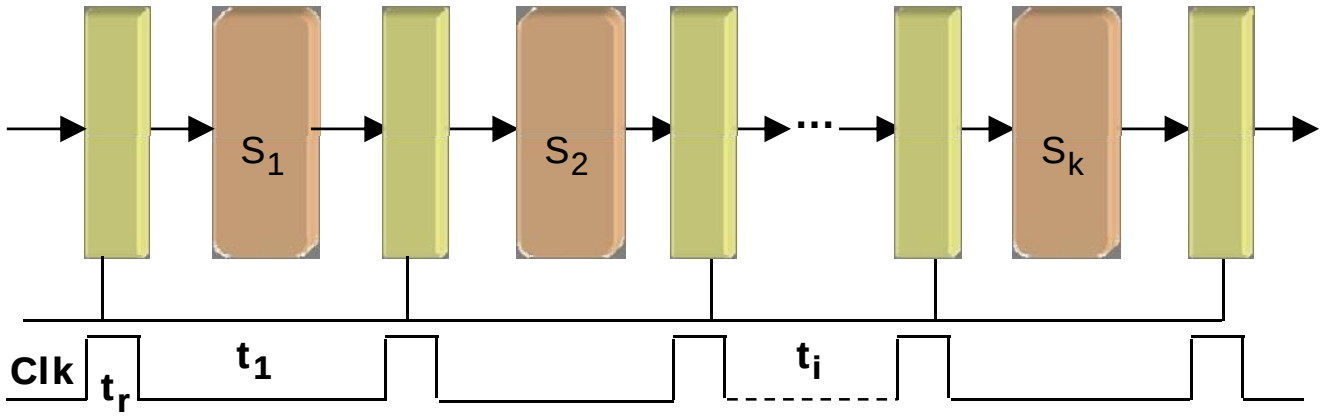
Objetivos:

- Repasar conceptos básicos de segmentación y superescalabilidad.
- Aprender técnicas de gestión de riesgos de datos, control y estructurales.
- Comprender el diseño de procesadores segmentados.
- Conocer mecanismos de optimización de cauces funcionales.

AIC – Tema 2 Segmentación y superescalares

(Parte A – Segmentación)

- **Segmentación (pipelining):** división de un proceso en un conjunto de etapas especializadas.
  - Identificación de fases en el procesamiento de una tarea.
  - Rediseño para implementar cada fase de forma independiente al resto.
  - Paralelismo por etapas (el sistema procesa varias tareas al mismo tiempo aunque sea en etapas distintas).
  - Se aumenta el número de tareas que se completan por unidad de tiempo.



Segmentación: **Historia.**

- Primer proceso segmentado: acelerar las operaciones en coma flotante.
- Primer computador con segmentación: IBM 7030 (100 veces más rápido que el anterior 704).
- Años 70: se convierte en una característica propia de los procesadores vectoriales.
- A principio de los 80 los procesadores RISC se proponen con la mejora de rendimiento gracias a la segmentación de cauce.
- A mediados de los 80 la segmentación de cauce se extiende a todos los procesadores, incorporándola a los procesadores embebidos en los años 90.

Prestaciones:

- **Ganancia.** Suponemos que TLI (tiempo de latencia de inicio) = T, tiempo que tarda en ejecutarse una operación en una unidad sin segmentar.

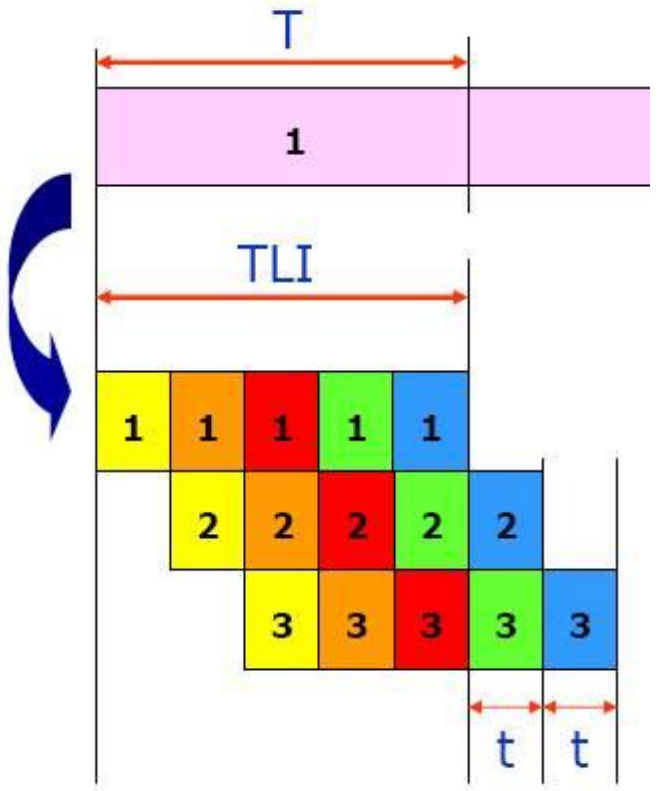
  TLI = k·t, siendo **k** el nº de etapas del cauce, y **t** la duración de cada etapa

$$G_k = \frac{T_1}{T_k} = \frac{k \cdot n \cdot t}{k \cdot t + (n-1) \cdot t} =$$

$$= \frac{k \cdot n}{k + n - 1}$$

$$\lim_{n \to \infty} G_k = k$$

Normalmente, ***¿TLI>T ó TLI<T?***



Prestaciones:

- **Productividad:** número de operaciones que se ejecutan por unidad de tiempo.

$$P_k = \frac{n}{TLI + (n-1)\cdot t} \qquad \lim_{n\to\infty} P_k = \frac{1}{t}$$

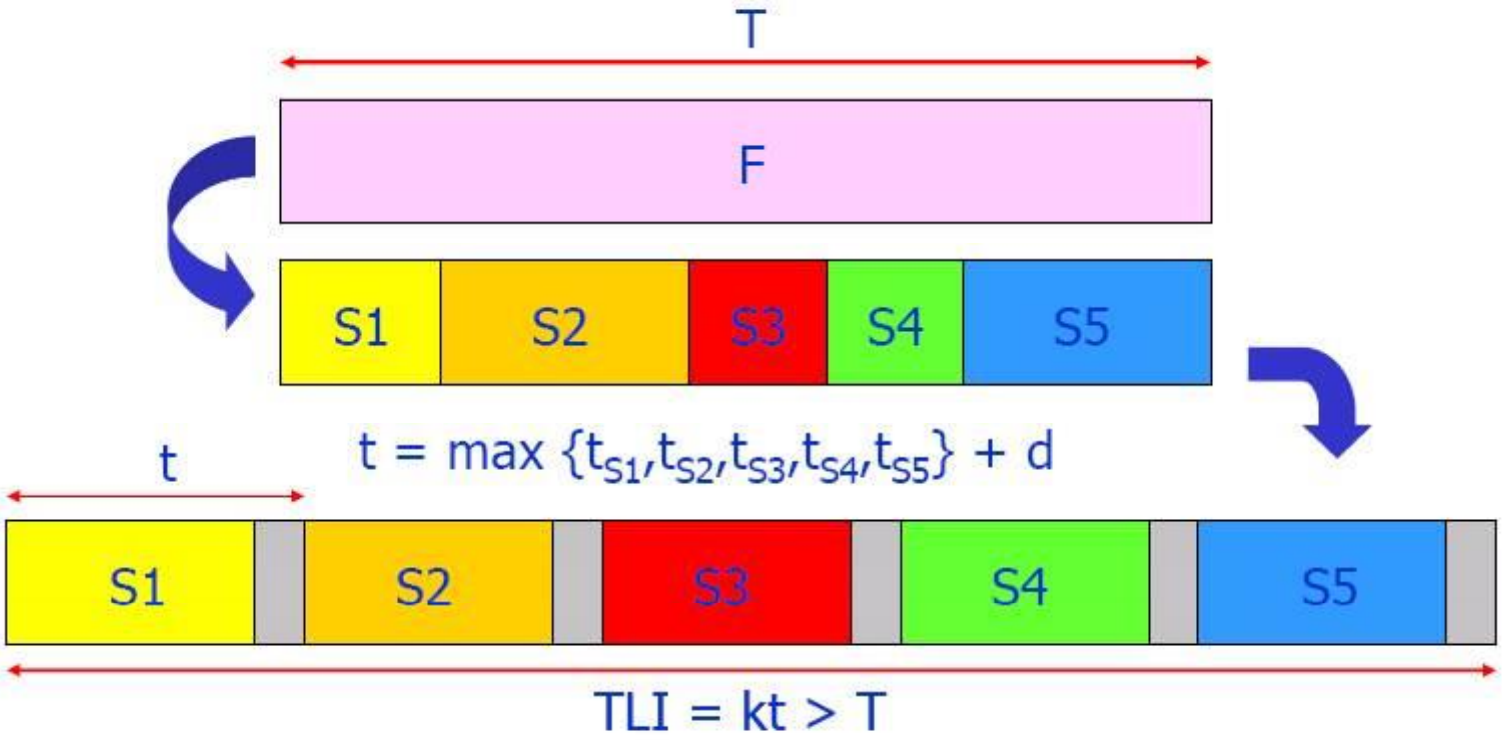
- **Eficiencia:** relación entre la ganancia de velocidad que proporciona el cauce y el número de etapas del mismo.

$$E_k = \frac{G_k}{k} = \frac{n \cdot T}{k\cdot(TLI + (n-1)\cdot t)} \qquad \lim_{n\to\infty} E_k = 1$$

AIC – Tema 2 Segmentación y superescalares

(Parte A – Segmentación)

- **Prestaciones:** es importante diseñar el cauce de manera que las etapas tengan una duración similar. Si existe una etapa con una duración mucho mayor que las otras, se convertirá en el cuello de botella que condiciona la duración de todas las etapas, y la ganancia que se podría alcanzar se vería bastante limitada.



Prestaciones:

- **Ganancia en el caso real:**

$$G_k = \frac{T_{sin\_segmentar}}{T_{segmentado}} = \frac{n \times T}{TLI + (n-1) \times t}$$

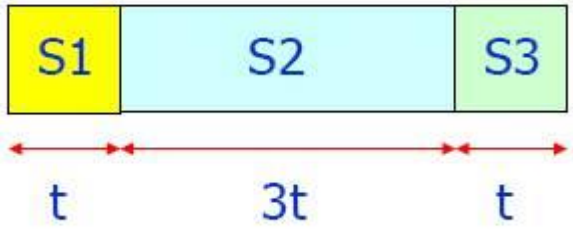
$$G_{max} = \lim_{n \to \infty} \frac{n \times T}{(k \times t) + (n-1) \times t} = \frac{T}{t} < k$$

TLI = kt

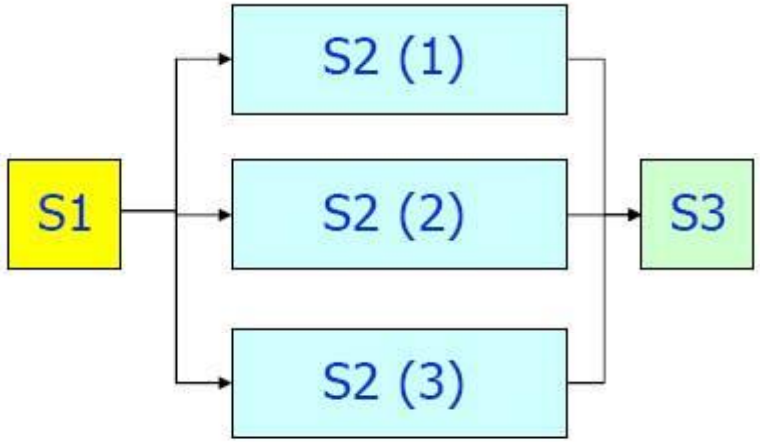
T < TLI = kt

Prestaciones:

- Solución a la existencia de tiempos diferentes de ejecución por etapas:



$$G_{max} = \frac{T}{3t}$$

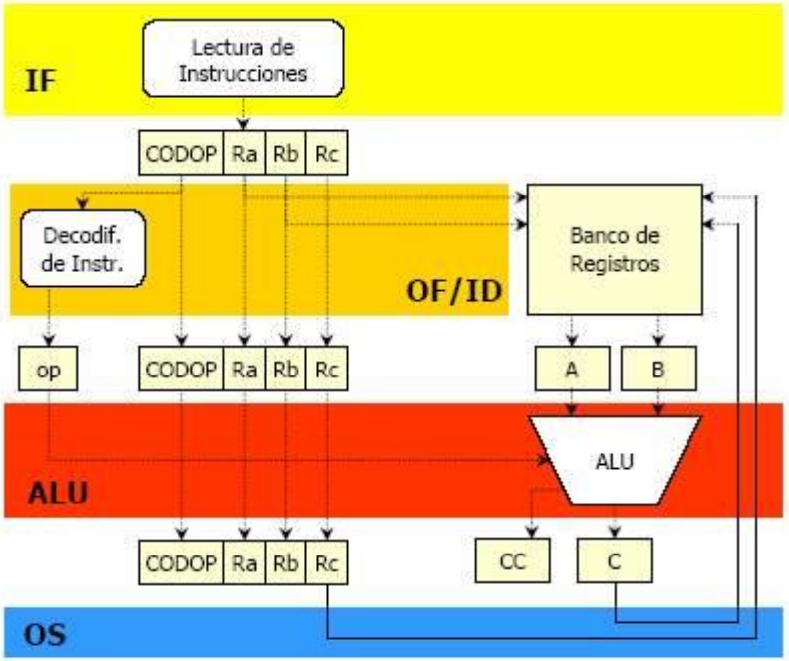


	t	t+1	t+2	t+3	t+4	t+5	t+6	t+7
S1	1	2	3	4				
S2(1)		1	1	1	4	4	4	
S2(2)			2	2	2			
S2(3)				3	3	3		
S3					1	2	3	4

$$G_{max} = \frac{T}{t}$$

AIC – Tema 2 Segmentación y superescalares

- **Diseño de un cauce segmentado:** partiremos de un cauce sencillo que poco a poco iremos complicando a medida que vayan apareciendo dificultades, considerando entonces las alternativas planteadas para resolverlos.



El Fichero de Registros ha de tener **dos puertos de lectura** (para poder leer los dos operandos en la etapa OF/ID) y **uno de escritura** (para poder escribir el resultado de instrucciones anteriores en la etapa OS)

Se elige una instrucción que utilice todas las etapas del cauce pero lo suficientemente simple para poder iniciar el diseño que luego se va a ir complicando: **C ← A operación B**

Etapas:

IF (Instruction Fetch) Se capta la instrucción

OF/ID (Operand Fetch/Inst. Decod.) Se captan los operandos y se decodifica la instrucción.

ALU Operación con la ALU

OS (Operand Store) El resultado se guarda en el fichero de registros

AIC – Tema 2 Segmentación y superescalares

- Riesgo (hazard):
  - Cualquier condición que pueda interrumpir el flujo continuo de instrucciones a través del cauce.
- Tipos de riesgos:
  - **Riesgos de datos.** Se producen por dependencias entre operandos y resultados de instrucciones distintas.
  - **Riesgos de control.** Se originan a partir de instrucciones de salto condicional que, según su resultado, determinan la secuencia de instrucciones que hay que procesar tras ellas.
  - **Riesgos estructurales o colisiones.** Se producen cuando instrucciones diferentes necesitan el mismo recurso al mismo tiempo.

AIC – Tema 2 Segmentación y superescalares

- Tipos de riesgos de datos:

RAW (Read After Write)

R2 := R1 + R2

R1 := R2 + R3

WAR (Write After Read)

R2 := R1 + R2

R1 := R2 + R3

WAW (Write After Write)

R2 := R1 + R2

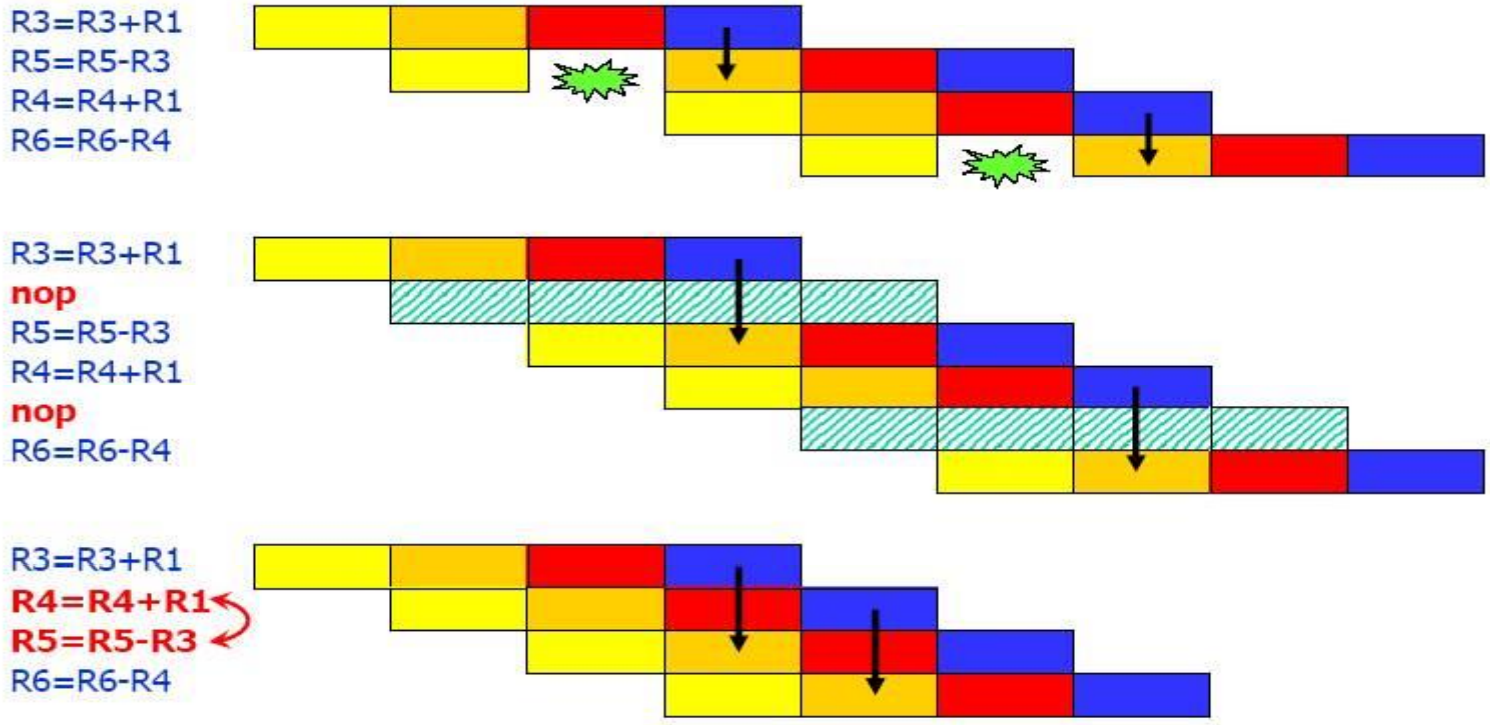
R2 := R4 + R3

AIC – Tema 2 Segmentación y superescalares

Soluciones a los riesgos de datos (Tipo **RAW**):

- **Reorganización de código:**
  - **Intercambio de instrucciones.**
  - **Introducción de instrucciones NOP** (reduce prestaciones del cauce).

(se supone **escritura** en flanco de subida y **lectura** en bajada)

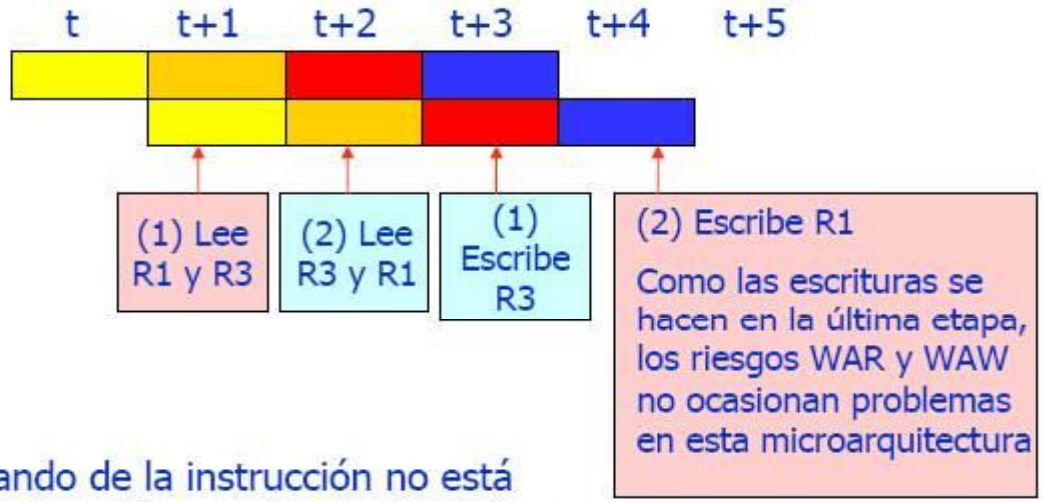


AIC – Tema 2 Segmentación y superescalares

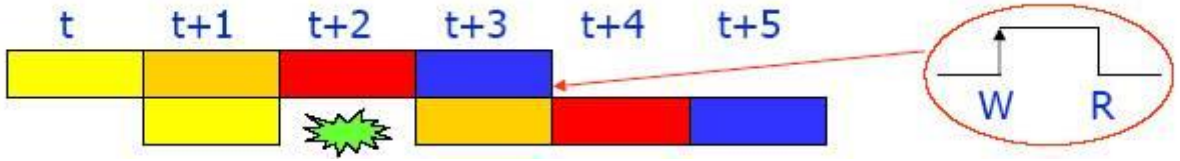
Soluciones a los riesgos de datos:

- **Interbloqueo entre etapas:** Se introducen elementos HW en el cauce que detectan la existencia de dependencias y detienen el cauce el número de ciclos necesarios (pérdida de rendimiento).

(1) R3 = R3 + R1
(2) R1 = R3 + R1



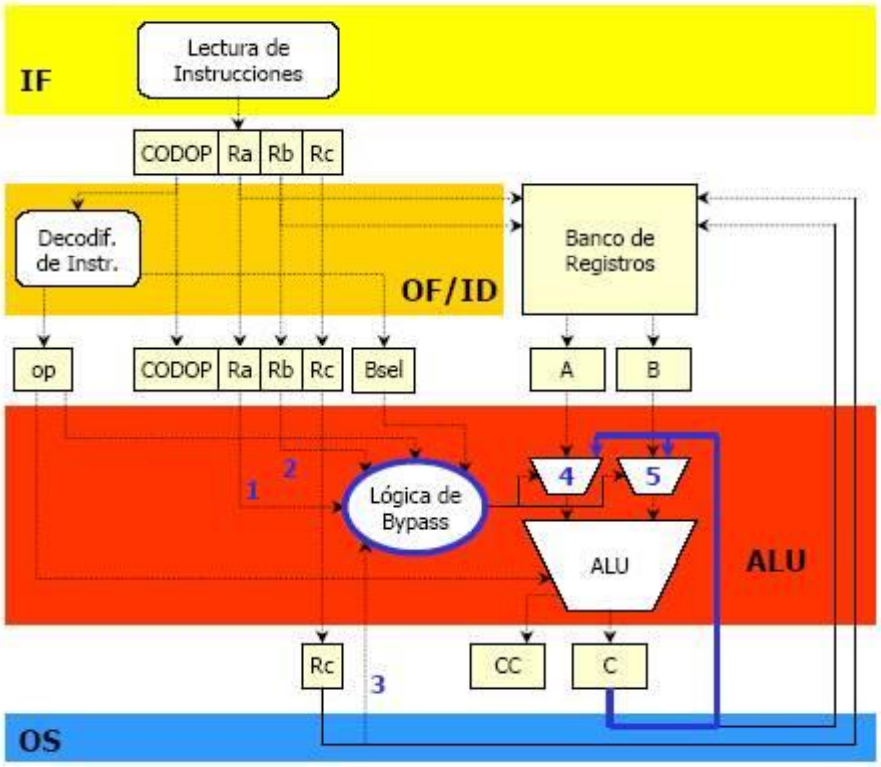
Si se detecta que el operando de la instrucción no está disponible se bloquea la instrucción un número de ciclos suficiente



AIC – Tema 2 Segmentación y superescalares

Soluciones a los riesgos de datos:

- **Atajos (bypass paths or forwardings):** Se aprovecha HW de la técnica anterior para habilitar caminos auxiliares (buses), y así anticipar el dato a la etapa que lo necesite antes de ser escritos en el banco de registros.



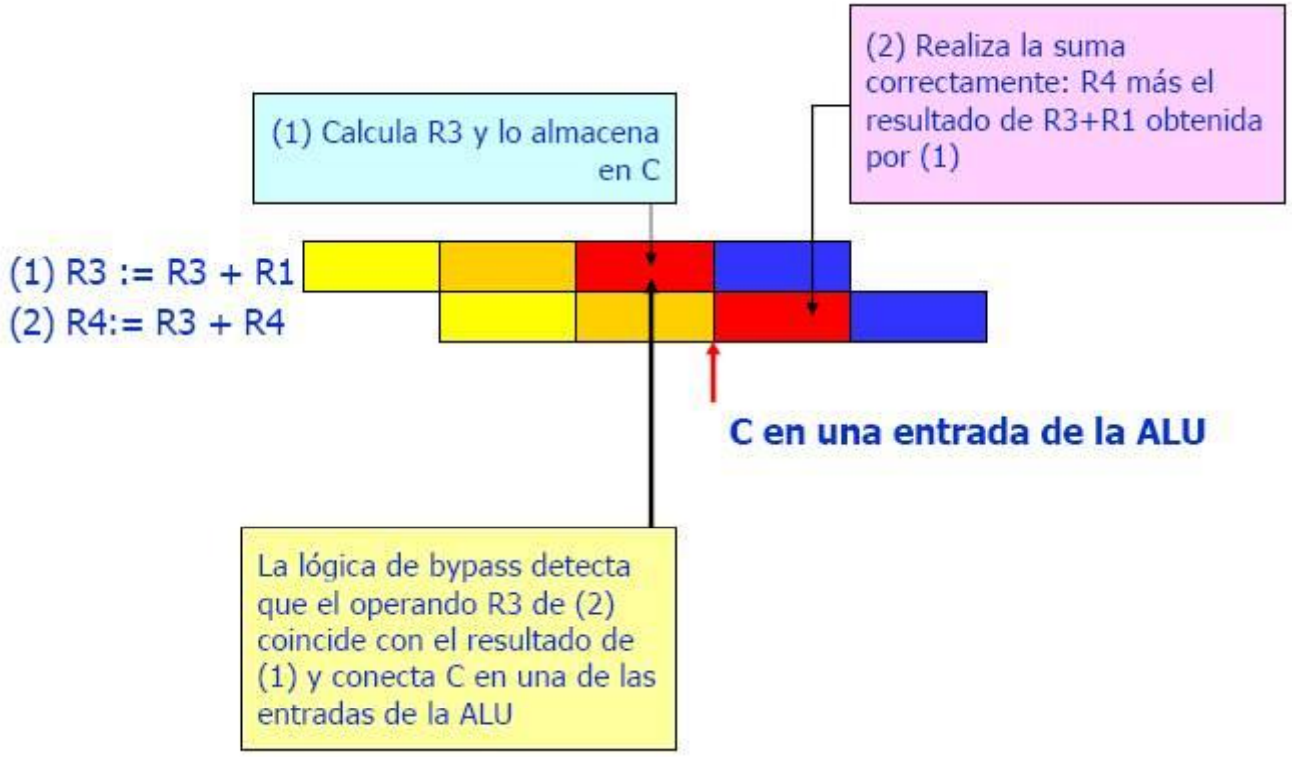
La **lógica de bypass** compara el destino de una instrucción **(3)** con los operandos de la siguiente **(1)** y **(2).**

En el caso de coincidencia entre (3) y (1) ó (3) y (2), se actúa sobre los multiplexores **(4)** o **(5)**, respectivamente para que se cargue en la entrada correspondiente de la ALU el resultado de la instrucción al mismo tiempo que se almacena en el banco de registros

AIC – Tema 2 Segmentación y superescalares

Soluciones a los riesgos de datos:

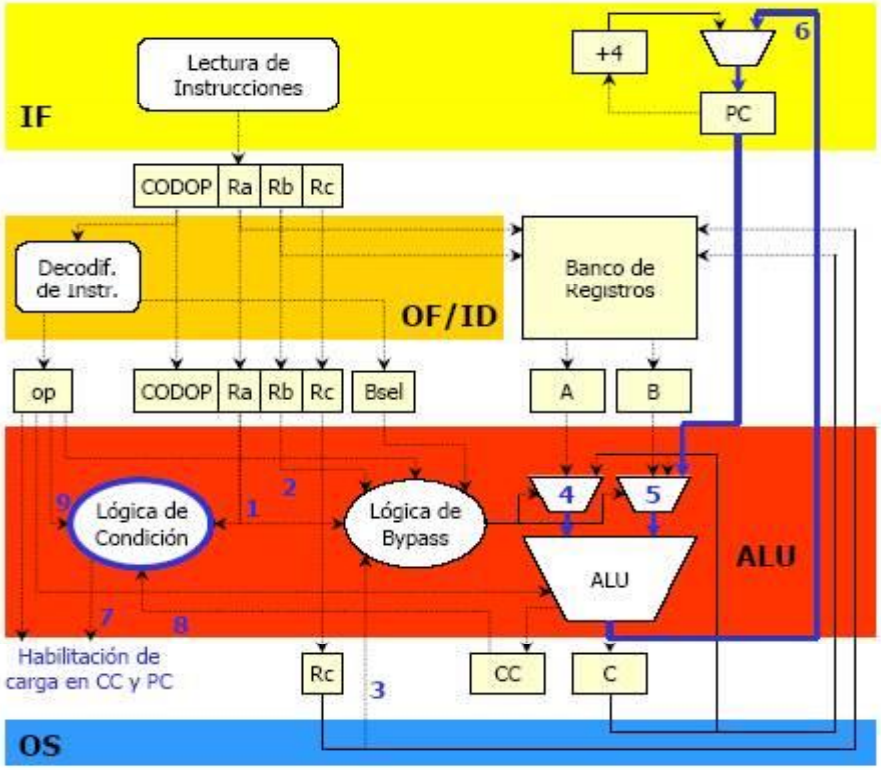
- **Atajos (Ejemplo):**



AIC – Tema 2 Segmentación y superescalares

Riesgos de control: **Instrucciones de salto**.

- Se considera el formato **bcnd Ra** :
  - **cnd** es la condición que debe cumplir la instrucción anterior.
  - **Ra** almacena el desplazamiento respecto al valor de **PC.**



El valor de PC se incrementa cada ciclo de reloj. Si se produce el salto **el valor de PC se carga (6) con la dirección obtenida al sumar el valor de PC actual con el OFFSET en el registro indicado por Ra (que se ha cargado en A).** PC es uno de los operandos de la ALU

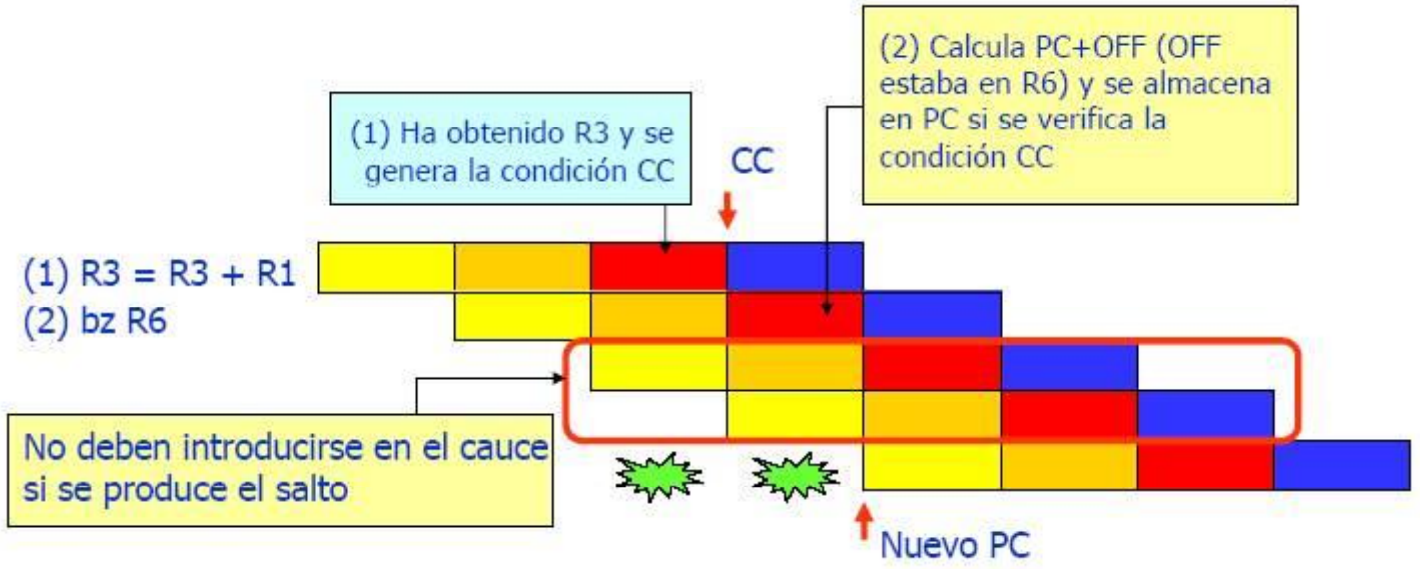
Las señales de habilitación de carga **(7)** permiten que los registros **PC** y **CC** se puedan cargar con los valores que proporciona la ALU

La determinación de si se produce un salto o no, la hace la **lógica de condición** en función del registro de **estado CC (8) y del código de salto (9)**

AIC – Tema 2 Segmentación y superescalares

Riesgos de control: **Soluciones.**

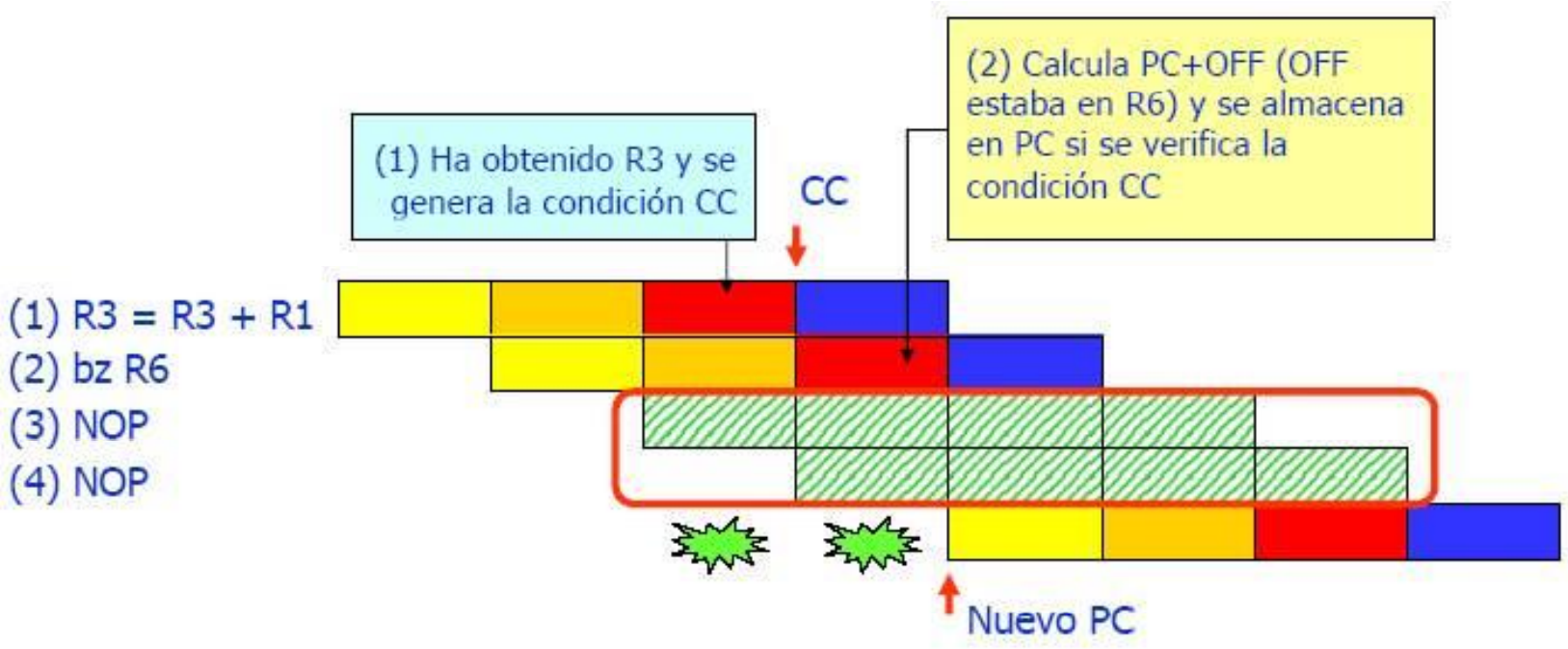
- **Abortar operaciones:**



AIC – Tema 2 Segmentación y superescalares

Riesgos de control: **Soluciones.**

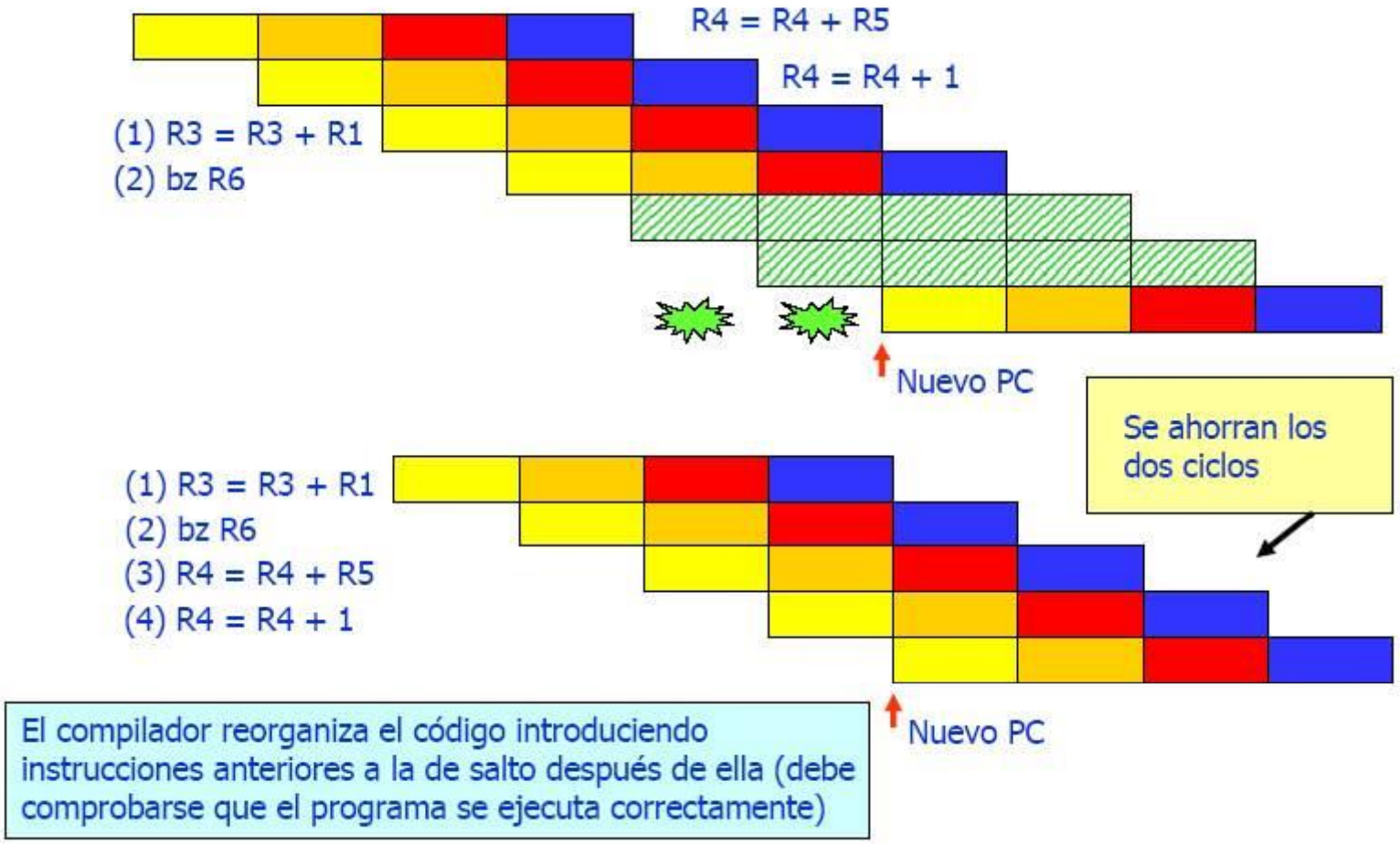
- **Bloqueos del cauce o uso de NOP:**
  - Tarea del compilador.
  - Se pierde rendimiento (perdemos ciclos de procesamiento).



AIC – Tema 2 Segmentación y superescalares

Riesgos de control: **Soluciones.**

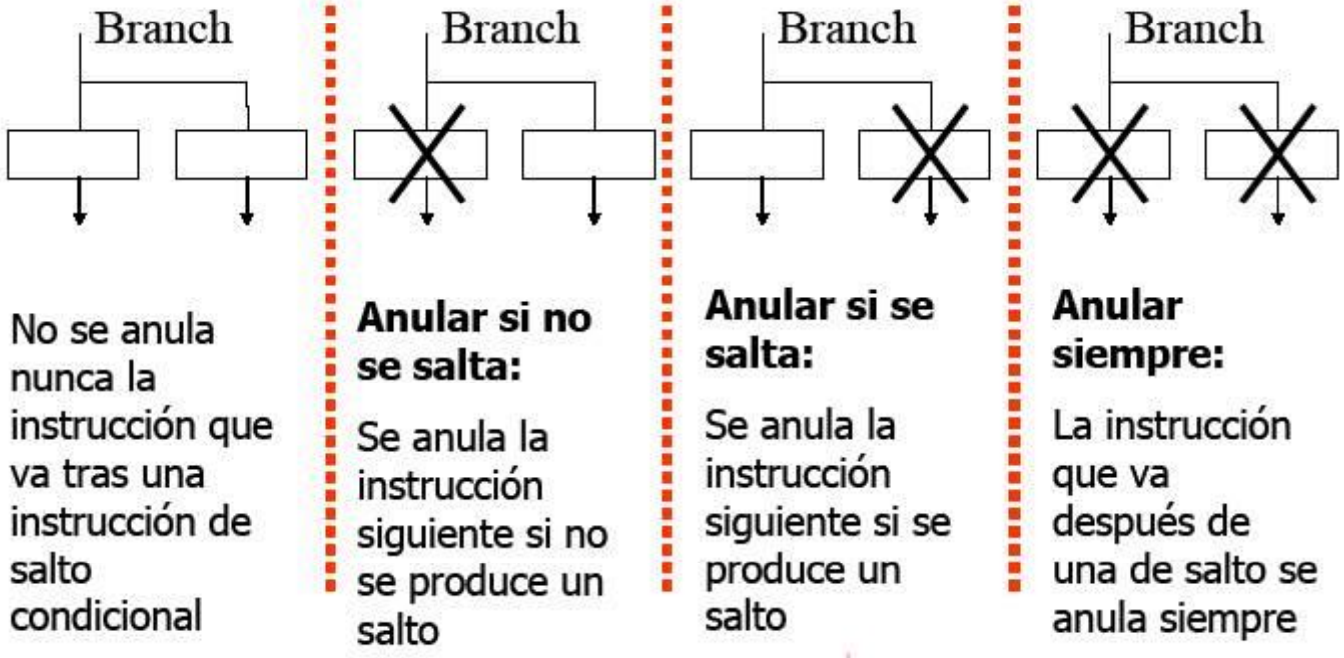
- **Salto retardado (Delayed branch):**



AIC – Tema 2 Segmentación y superescalares

Riesgos de control: **Soluciones.**

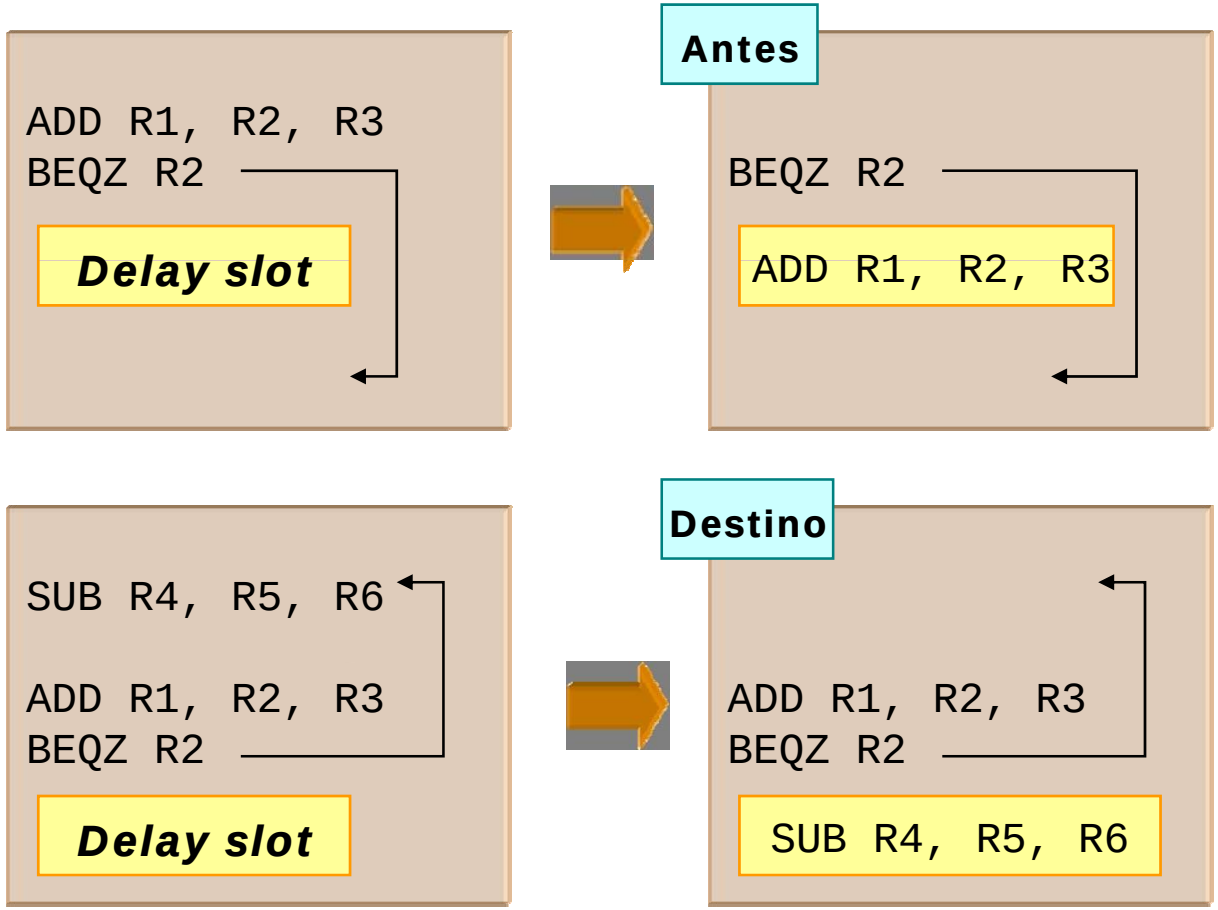
- **Salto retardado (Delayed branch):**
  - Depende de las características del procesador en lo que respecta a la política de anulación de instrucciones introducidas en el cauce erróneamente en los saltos.



AIC – Tema 2 Segmentación y superescalares

Riesgos de control: **Soluciones.**

- **Salto retardado (Delayed branch):**

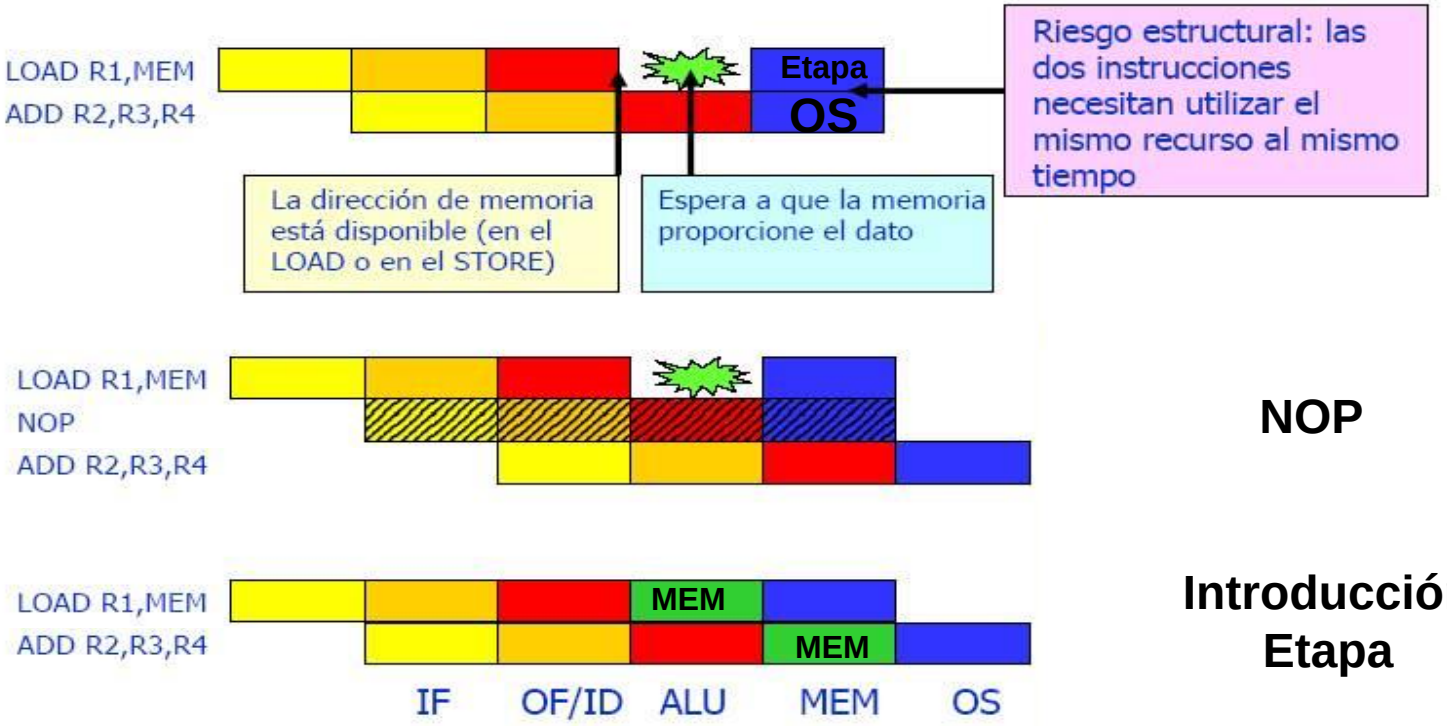


Riesgos estructurales: Instrucciones de acceso a memoria.

- Supongamos instrucciones de carga y almacenamiento ***LOAD Rd, Mem*** y ***STORE Rd, Mem.***
  - **Rd** es el registro donde se carga el dato de memoria.
  - **Mem** es el registro que almacena la dirección de memoria.
- Nuevos registros:
  - **DMAR** (Data Memory Address Register) almacena la dirección de memoria.
  - **SDMR** (Store Data Memory Register) almacena el dato a escribir.
  - **LDMR** (Load Data Memory Register) almacena el dato que se obtiene.
- Colisión debido a la necesidad de un ciclo adicional para leer el dato o escribir el dato en memoria. Puede colisionar con otra instrucción que no sea de acceso a memoria, la cual utilizaría un ciclo menos de reloj, produciéndose una solicitud de recurso (por ejemplo: etapa OS) que esta siendo usado por otra instrucción.

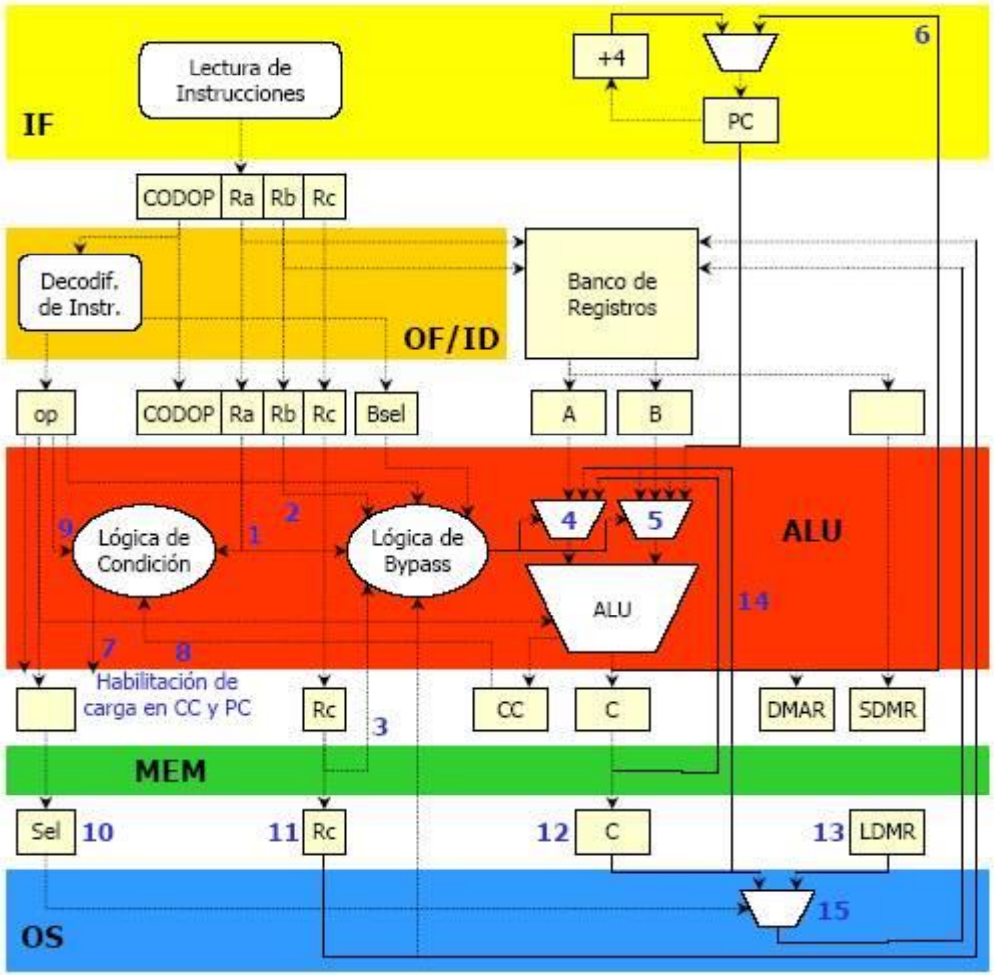
AIC – Tema 2 Segmentación y superescalares

- Riesgos estructurales por acceso a memoria: **Solución.**
  - Modificaciones hardware en la memoria.
  - Introducción de NOP.
  - Modificaciones en el cauce: introducción de una etapa.



AIC – Tema 2 Segmentación y superescalares

- Riesgos estructurales por acceso a memoria: **Introducción de la etapa *MEM*.**



El acceso a memoria se realiza mediante instrucciones LOAD/STORE.

Se añaden los registros **DMAR** y **SDMR** y **LDMR**.

DMAR para las direcciones de memoria (se actualiza al final de la etapa de operación con la ALU que puede utilizarse en los modos de direccionamiento.

SDMR tiene los datos a almacenar en los STOREs al final de la etapa ALU

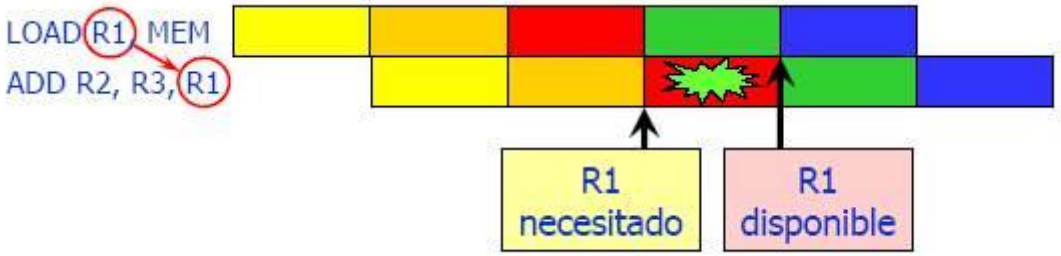
LDMR tiene el dato que se lee de memoria en los LOADs

Se incrementa el cauce en una etapa de espera –no hace nada- **(10, 11, 12, 13, 14)** para esperar que la memoria responda en los LOADs. El multiplexor **(15)** permite almacenar el resultado del LOAD o de la ALU

AIC – Tema 2 Segmentación y superescalares

Acceso a memoria: **Dependencia RAW.**

- Se produce un retardo denominado *retardo de uso de carga (load-use delay).*

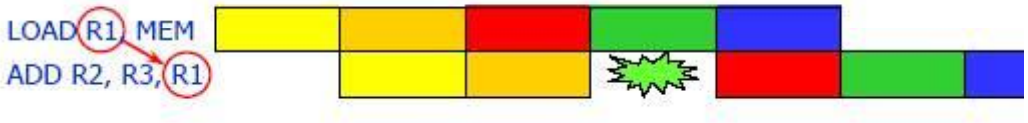


AIC – Tema 2 Segmentación y superescalares

Retardo de uso de carga: **Soluciones.**

- Introducción de instrucciones independientes (programador o compilador).

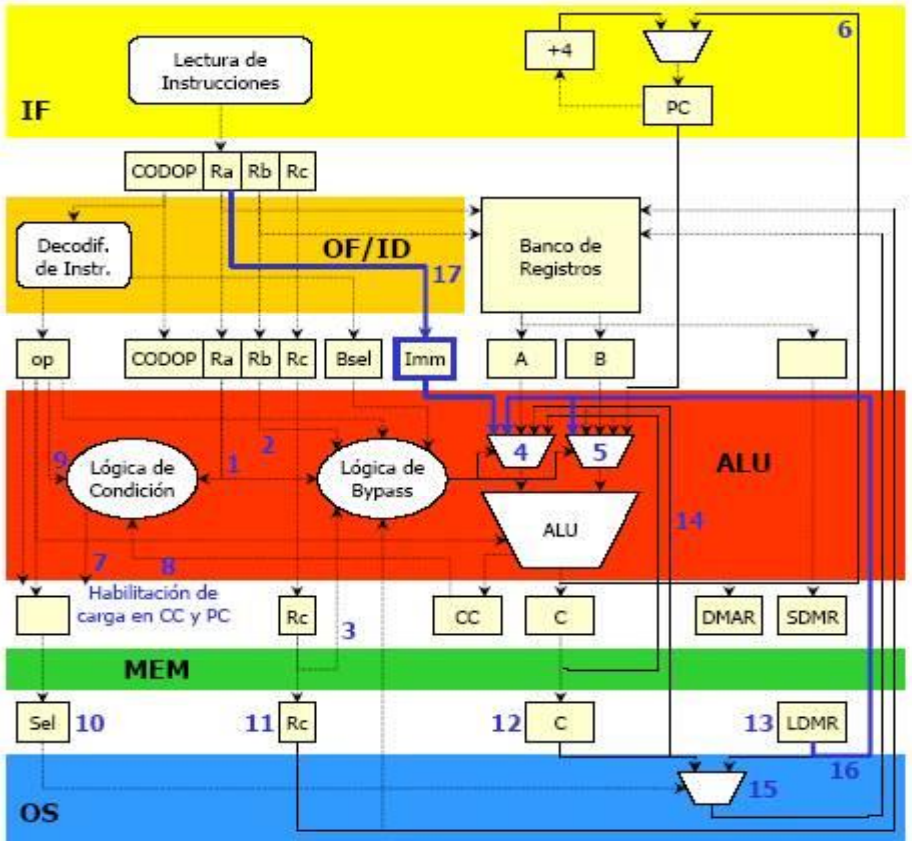
- Bloqueo de cauce.



AIC – Tema 2 Segmentación y superescalares

Retardo de uso de carga: **Soluciones.**

- Añadiendo circuitería (bypass) de forma que las instrucciones que necesiten un dato leído de memoria puedan disponer dé él lo antes posible.



Se ha añadido un camino de bypass **(16)** entre el registro LDMR y la ALU para minimizar el retardo carga-uso del procesador.

También se añade la posibilidad de operar con valores inmediatos mediante la adición del registro Imm **(17)**, que posibilita alimentar a la ALU con un dato almacenado en la instrucción

AIC – Tema 2 Segmentación y superescalares

- **Dimensiones de diseño de un procesador segmentado: Clasificación.**
  - **Organización del cauce.**
    - Número de etapas.
    - Subtarea que implementa cada etapa.
    - Distribución de la secuencia de etapas.
    - Uso de caminos de bypass.
    - Temporización del cauce.
  - **Resolución de dependencias.**
    - Estática (compilador).
    - Dinámica (elementos hardware).
    - Combinada.

AIC – Tema 2 Segmentación y superescalares

- Organización del cauce: **Número de etapas.**

Número de etapas en procesadores:

RISC I (1982): 2

RISC II (1983): 3

α21064 (1993): 6-10

Pentium (93) : 6-7 (INT), 8-10 (FP)

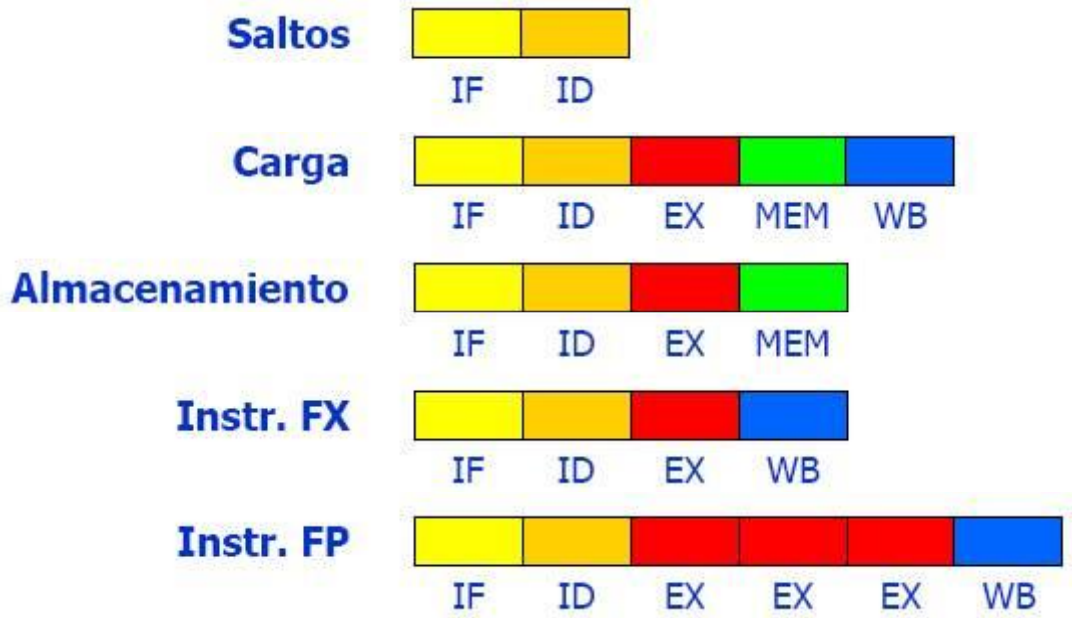
MC68060: 9

Cuantas más etapas tiene un cauce, mayor puede ser la ganancia (en el caso de un comportamiento ideal), aunque pueden aparecer más dependencias (al haber más instrucciones en el cauce) y el coste asociado a dependencias no resueltas es mayor (se pierden más ciclos)

AIC – Tema 2 Segmentación y superescalares

Organización del cauce: **Subtarea de cada etapa.**

- Aspecto esencial en el diseño del cauce:
  - ¿Adelantar el procesamiento del salto a las primeras etapas?
  - ¿Etapa específica para el acceso a memoria?
  - ¿Comparten etapa el acceso a memoria y la ejecución de operaciones aritméticas?



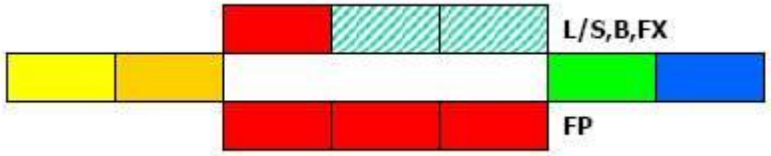
AIC – Tema 2 Segmentación y superescalares

- Organización del cauce: **Distribución de la secuencia de etapas (I).**

RISC I (82), RISC II (83), MIPS (81), MIPS-X (86), IBM801 (78)

Cauce Único

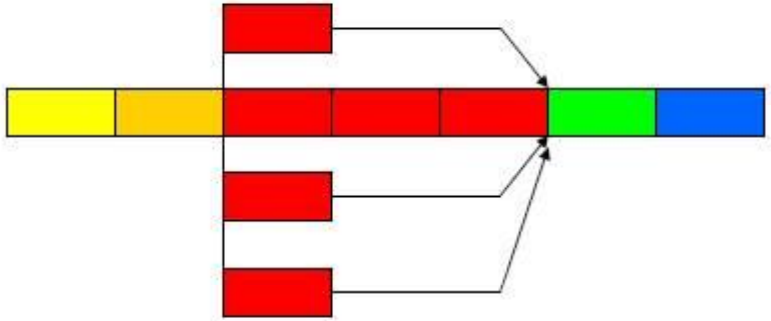
Cauces en los que no hay unidades funcionales para aritmética de coma flotante



Z80000 (84), R2000 (87), R3000 (88), AMD29000 (87), i486 (88)

Cauce Doble

Se fuerza que las instrucciones terminen en orden



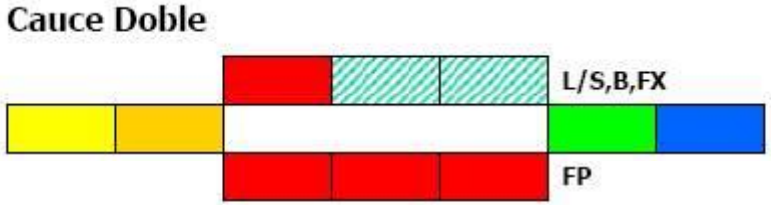
Fundamentalmente en Procesadores Superescalares

Múltiples Cauces

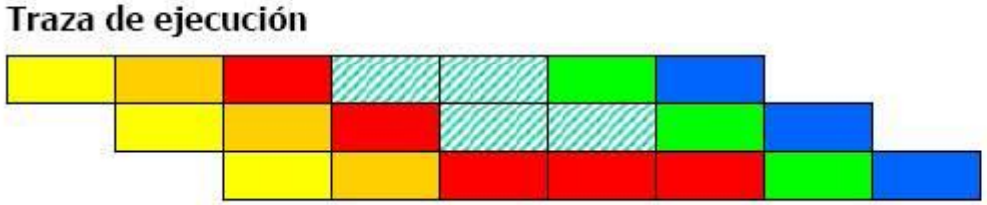
Se permite que las instrucciones terminen desordenadamente (se debe comprobar que no se causan errores)

AIC – Tema 2 Segmentación y superescalares

- Organización del cauce: **Distribución de la secuencia de etapas (II).**



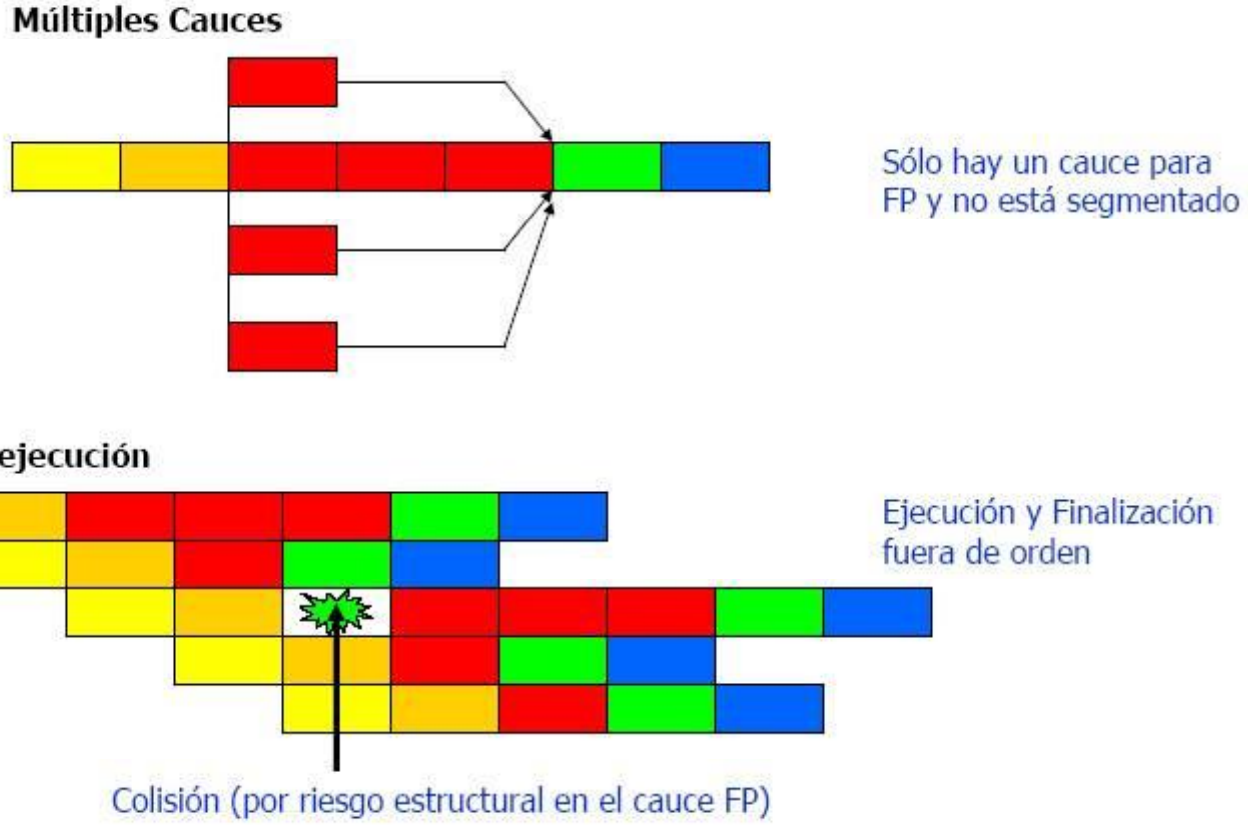
Se alarga el cauce corto para facilitar el mantenimiento de la consistencia secuencial



Ejecución y Finalización en orden

AIC – Tema 2 Segmentación y superescalares

- Organización del cauce: **Distribución de la secuencia de etapas (III).**



AIC – Tema 2 Segmentación y superescalares

- Organización del cauce: **Uso de atajos.**
  - Es posible adelantar los datos necesarios entre etapas para que las dependencias no tengan efectos.
- Organización del cauce: **Temporización.**
  - **Síncrona.** Se utiliza una señal de reloj común a todas la etapas para gestionar el paso de las instrucciones de una etapa a otra.
  - **Asíncrona.** Se utilizan líneas de sincronización entre etapas consecutivas para implementar un protocolo de conformidad (handshaking).
    - Más adecuada cuando no se conoce con exactitud la duración de las etapas o ésta puede cambiar según la instrucción que se procese.