Módulo 3: Diseño del procesador MIPS Segmentado

Tema 3.1 Segmentación

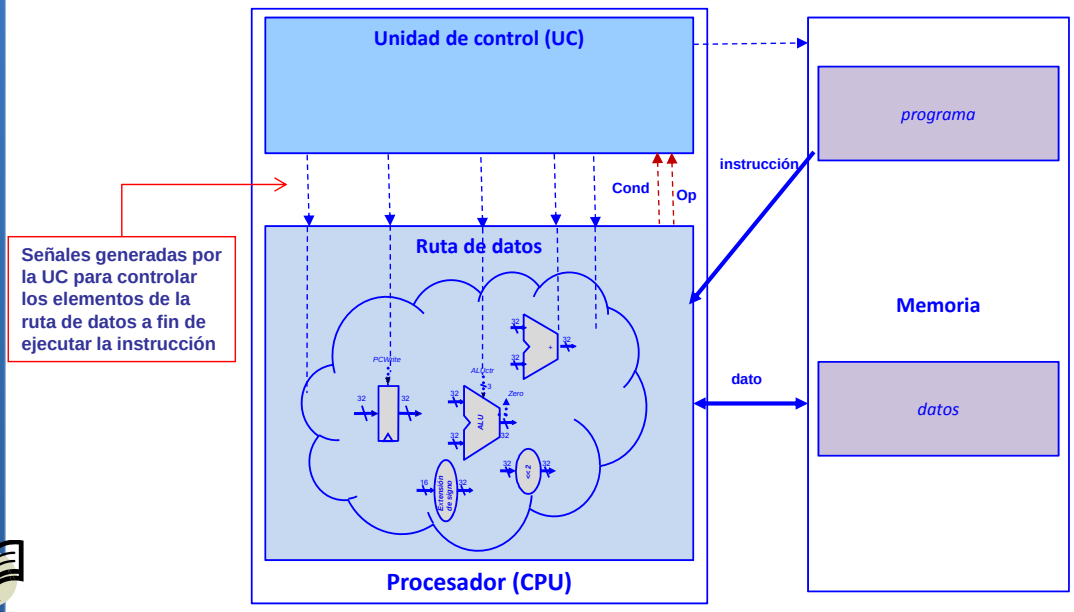
1. Introducción
2. MIPS
3. Segmentación
4. Riesgos estructurales
5. Riesgos de datos
6. Riesgos de control
7. Riesgos de control con riesgos LDE
8. Resumen

- **Bibliografía**
  - Hennessy Patterson Apendice A, 4ª ed.

1. Introducción

Esquema general de la arquitectura de un computador: modelo Von Neuman

- ✓ Las instrucciones son ejecutadas sobre la ruta de datos
- ✓ La UC genera las señales que gobiernan los elementos de la ruta de datos



2. MIPS: Repertorio de instrucciones implementado

- **Instrucciones aritmético-lógicas con operandos en registros** (tipo R)
  - add rd, rs, rt — rd ← rs + rt, PC ← PC + 4
  - sub rd, rs, rt — rd ← rs - rt , PC ← PC + 4
  - and rd, rs, rt — rd ← rs and rt , PC ← PC + 4
  - or rd, rs, rt — rd ← rs or rt , PC ← PC + 4
  - slt rd, rs, rt — (si (rs < rt) entonces (rd ← 1) en otro caso (rd ← 0)), PC ← PC+4

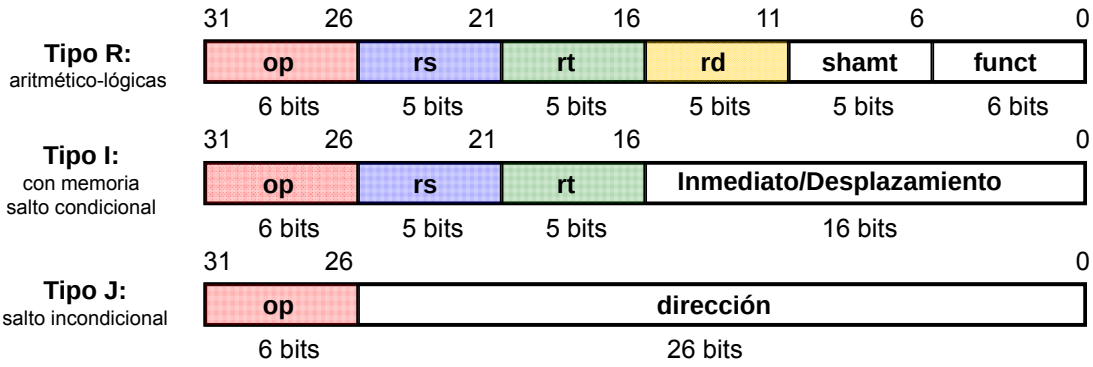
- **Instrucciones con referencia a memoria** (tipo I)
  - lw rt, inmed(rs) — rt ← Memoria(rs + SignExt(inmed)) , PC ← PC + 4
  - sw rt, inmed(rs) — Memoria(rs + SignExt(inmed)) ← rt , PC ← PC + 4

- **Instrucciones de salto condicional** (tipo I)
  - beq rs, rt, inmed — si (rs = rt) entonces (PC ← PC + 4 + 4·SignExt(inmed)) en otro caso PC ← PC + 4

2. MIPS: Repertorio de instrucciones implementado

Arquitectura MIPS (DLX)

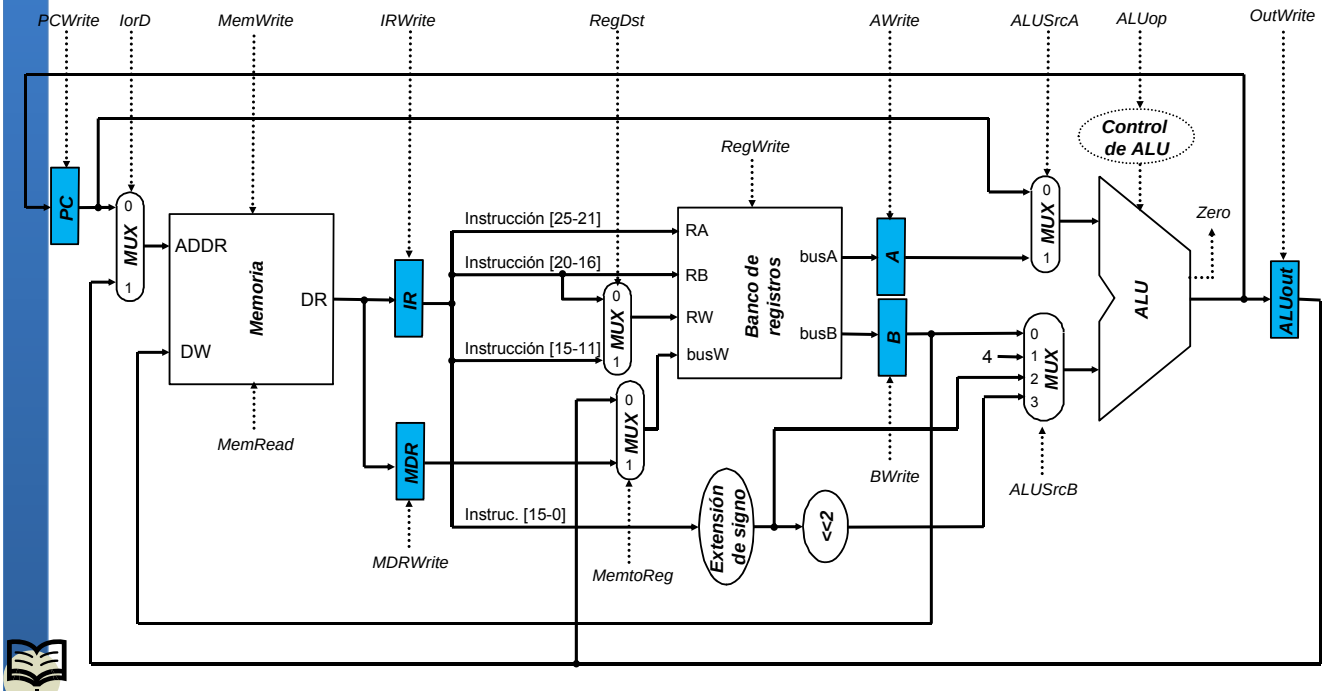
- ✓ Instrucciones de longitud fija: 32 bits
- ✓ Tres formatos de instrucción diferentes:



- ✓ Significado de los campos:
  - **op:** identificador de instrucción
  - **rs, rt, rd:** identificadores de los registros fuentes y destino
  - **shamt:** cantidad a desplazar (en operaciones de desplazamiento)
  - **funct:** selecciona la operación aritmética a realizar
  - **inmediato:** operando inmediato o desplazamiento en direccionamiento indirecto
  - **dirección:** dirección destino del salto

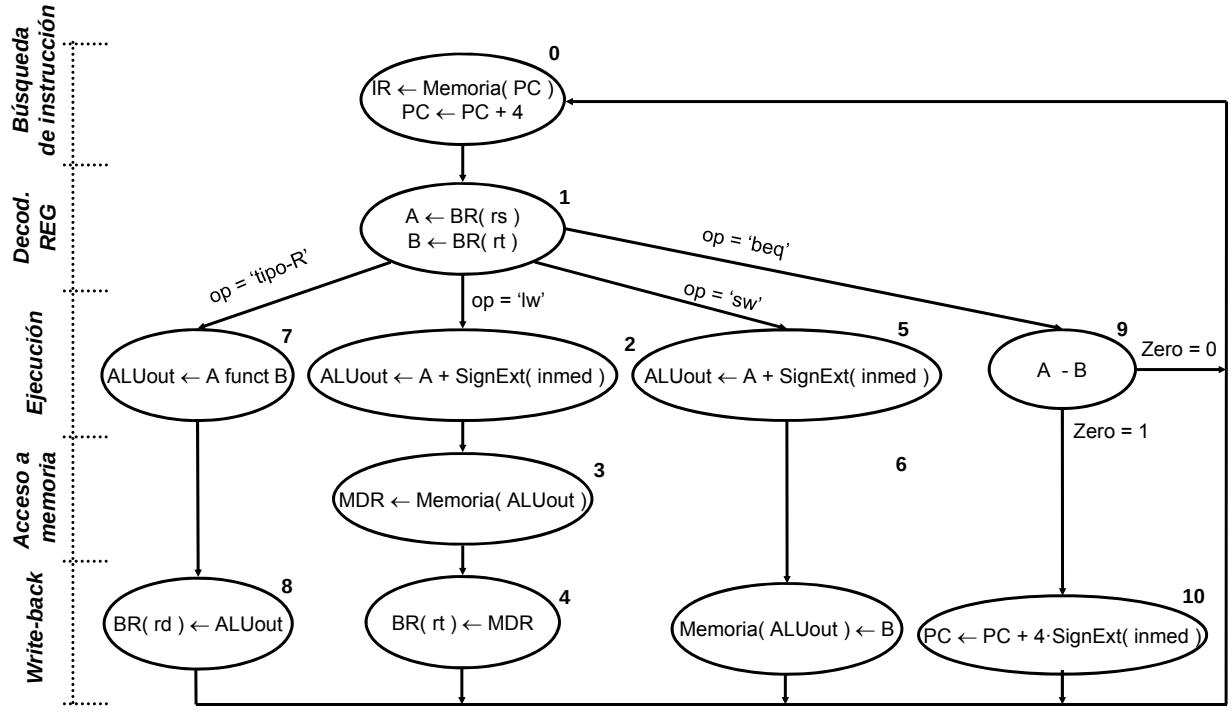
2. MIPS Multiciclo

Ruta de datos (Multiciclo)



2. MIPS Multiciclo

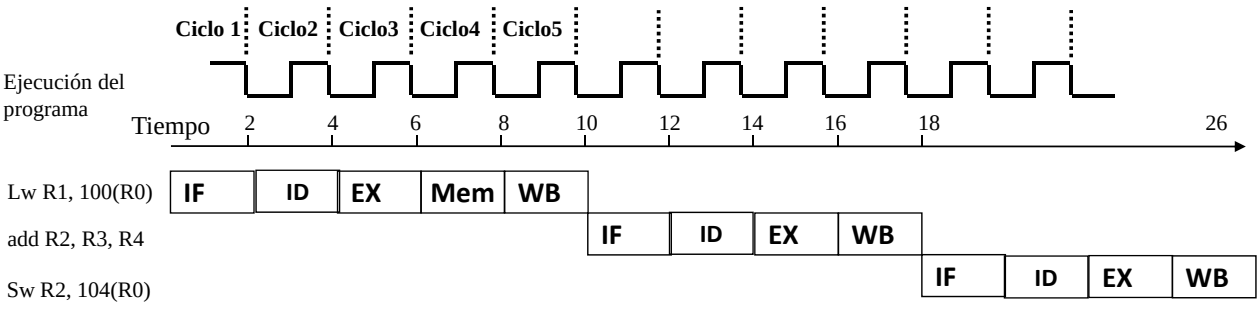
Diagrama de estados del control



2. MIPS Multiciclo

Ejecución de instrucciones

- Instrucciones **Load tiene 5 fases** , utilizan todas las etapas
- **El resto** de instrucciones **tienen 4 fases**, no usan la fase Mem



26ns

3. Segmentación

Cómo mejorar el rendimiento del procesador: Aplicar segmentación

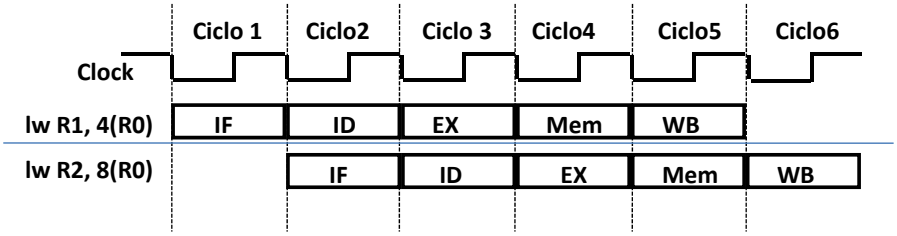
- Es una técnica de implementación en la que múltiples instrucciones se solapan en ejecución
  - **Cada etapa opera en paralelo con otras etapas pero sobre instrucciones diferentes**

- Se puede aplicar porque las fases por la que pasa una instrucción no usan todas las componentes de la ruta de datos
  - **Explota el paralelismo a nivel de instrucción**

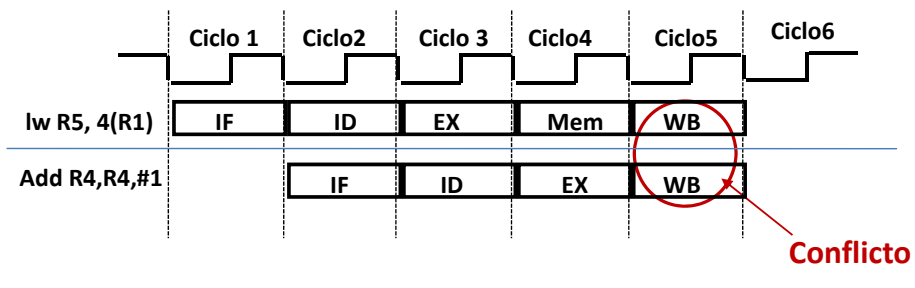
3. Segmentación

Ejecución Procesador Segmentado

- Ejemplo 1



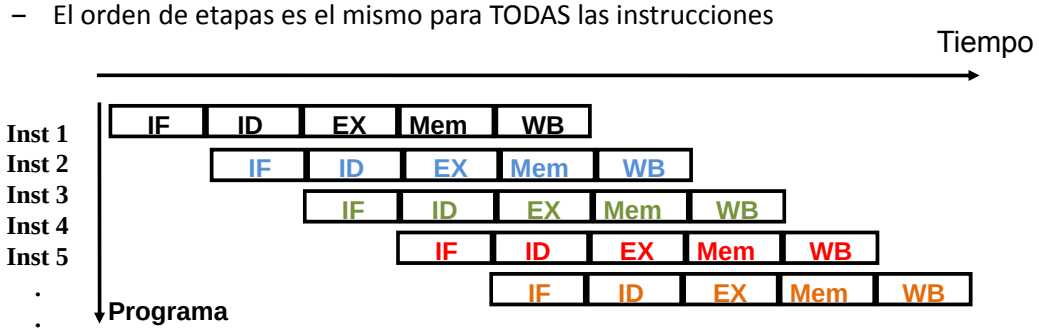
- Ejemplo 2



3. Segmentación

¿ Cómo evitar este conflicto?

- No puede haber más de una instrucción intentando acceder a una etapa
- **TODAS las instrucciones deben pasar por TODAS las etapas**
  - **El ciclo de reloj viene determinado por la etapa más lenta**
- El orden de etapas es el mismo para TODAS las instrucciones

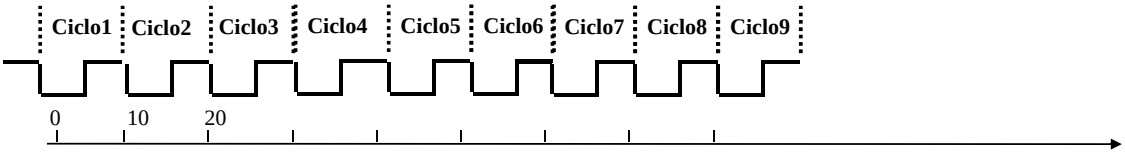


- A partir del ciclo 5
  - Sale una instrucción cada ciclo de reloj
  - **CPI=1**
- **Los 4 primeros ciclo se llaman de llenado del pipeline y se pueden despreciar.**
- **CPI_{Ideal}=1**

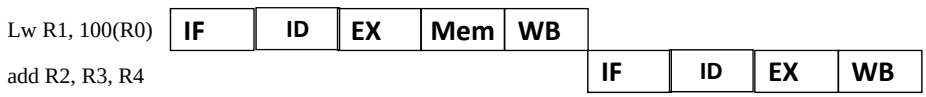
3. Segmentación

Segmentado frente a Multiciclo

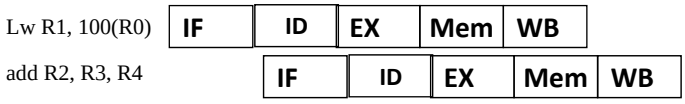
Si se ejecutan **100 instrucciones**
- **Multiciclo**
  10ns/ciclo x 4.6 CPI x 100inst =**4600ns**
- **Segmentado**
  10ns/ciclo x (1CPIx100inst+4 llenado) =**1040ns**



Multiciclo



Segmentado



3. Segmentación

¿Qué dificulta la segmentación?

- **Riesgos**
  - Aparecen situaciones que impiden que en cada ciclo se inicie la ejecución de una nueva instrucción
  - **Tipos:**
    - **Estructurales**
      - Se producen cuando dos instrucciones tratan de utilizar el mismo recurso en el mismo ciclo
    - **De datos**
      - Se intenta utilizar un dato antes de que esté preparado. Mantenimiento del orden estricto de lecturas y escrituras
    - **De control**
      - Se intenta tomar una decisión sobre una condición todavía no evaluada

  ✓ **Los riesgos se deben *detectar y resolver***

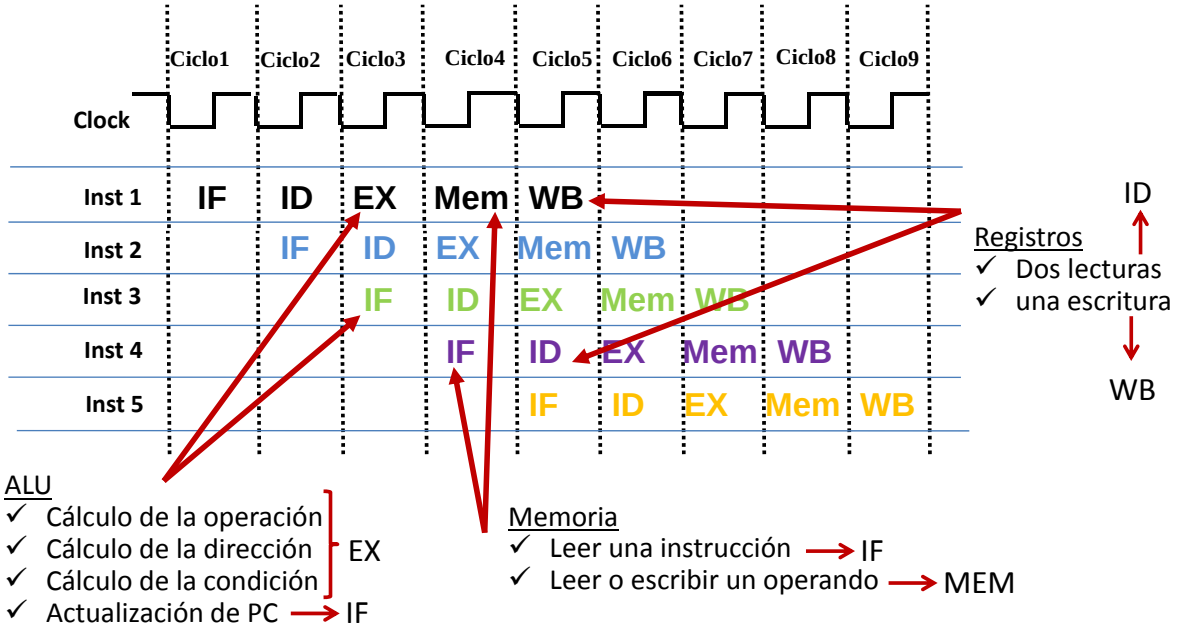
- **Gestión de interrupciones**

- **Ejecución fuera de orden**

4. Segmentación: Riesgos estructurales

Riesgos estructurales

✓ *Se producen cuando dos instrucciones tratan de utilizar el mismo recurso en el mismo ciclo*



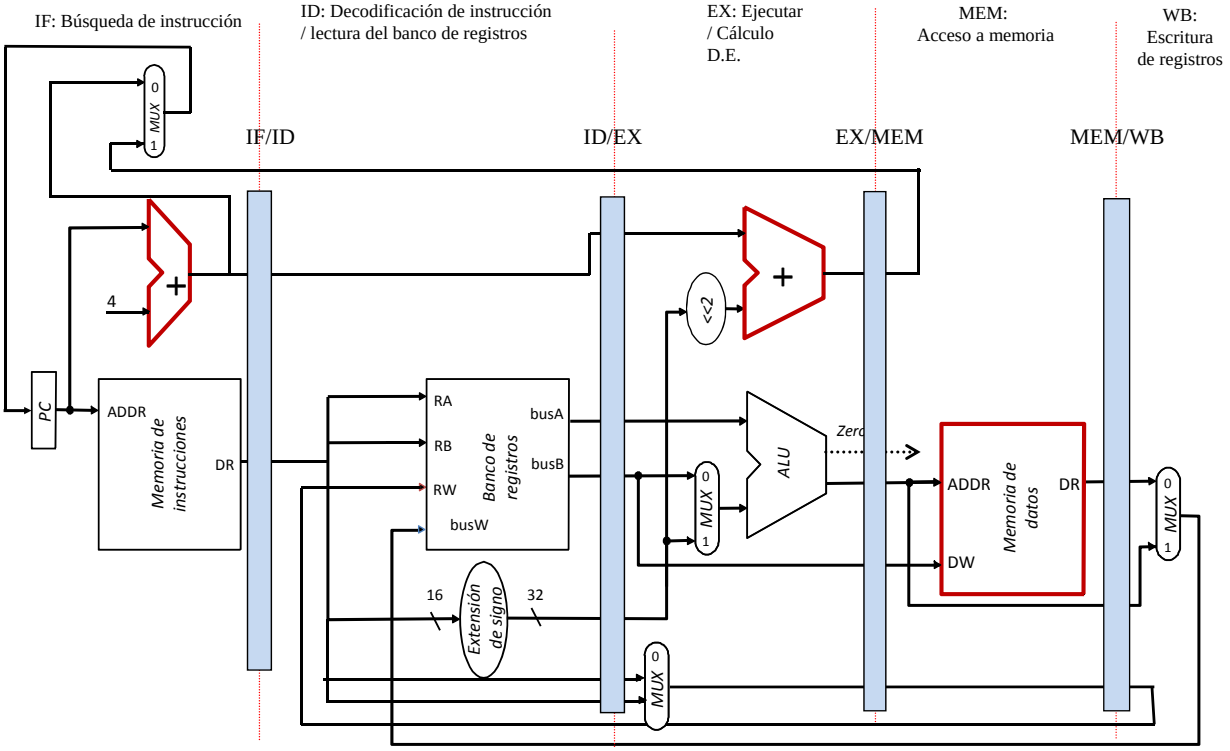
Objetivo:

✓ **Ejecutar sin conflicto cualquier combinación de instrucciones**

4. Segmentación: Riesgos estructurales

¿Cómo resolver los riesgos estructurales?

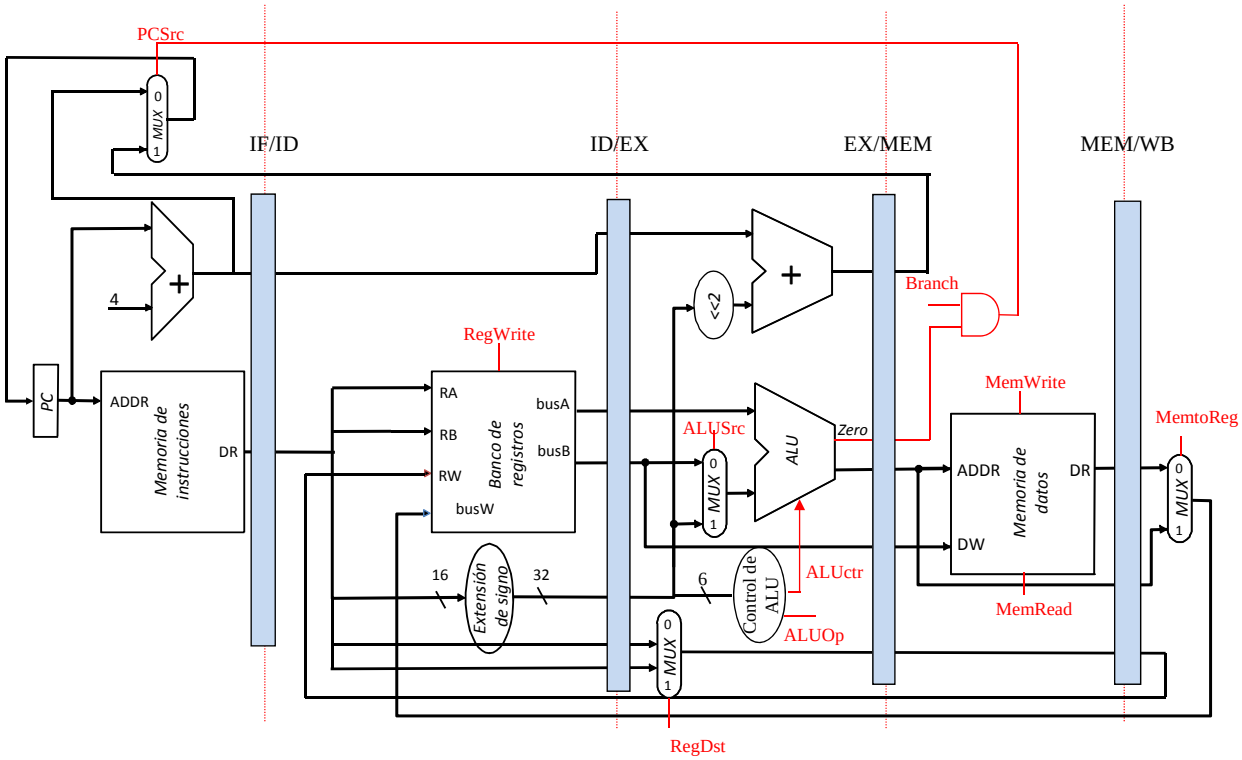
- ✓ **Duplicar los recursos que se necesitan en el mismo ciclo.**
- ✓ **El B.R no es problema porque tiene puertos para leer de dos registros y escribir en un registro a la vez.**



4. Segmentación: Riesgos estructurales

Diseño del control segmentado

- ✓ El PC y los diferentes registros de desacoplo (IF/ID ..) se deben cargar en cada ciclo, por tanto no necesitan una orden específica de carga



4. Segmentación: Riesgos estructurales

Diseño de control segmentado

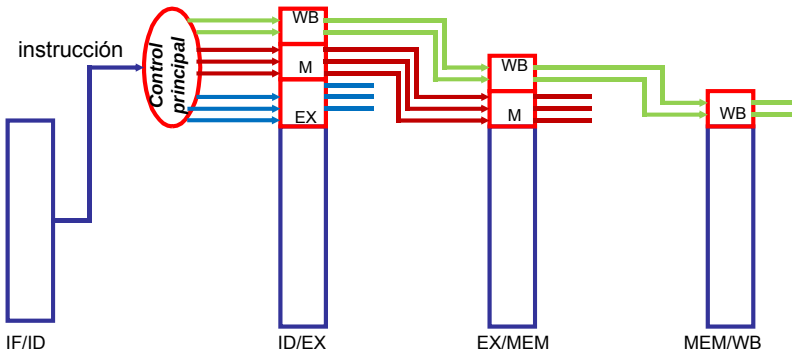
Control de la ALU

op	funct	ALUop	ALUctr
100011 (lw)	XXXXXX	00	010+
101011 (sw)		00	010 +
000100 (beq)		01	110-
000000 (tipo-R)	100000 (add)	10	010+
	100010 (sub)	10	110-
	100100 (and)	10	000
	100101 (or)	10	001
	101010 (slt)	10	111

Control principal

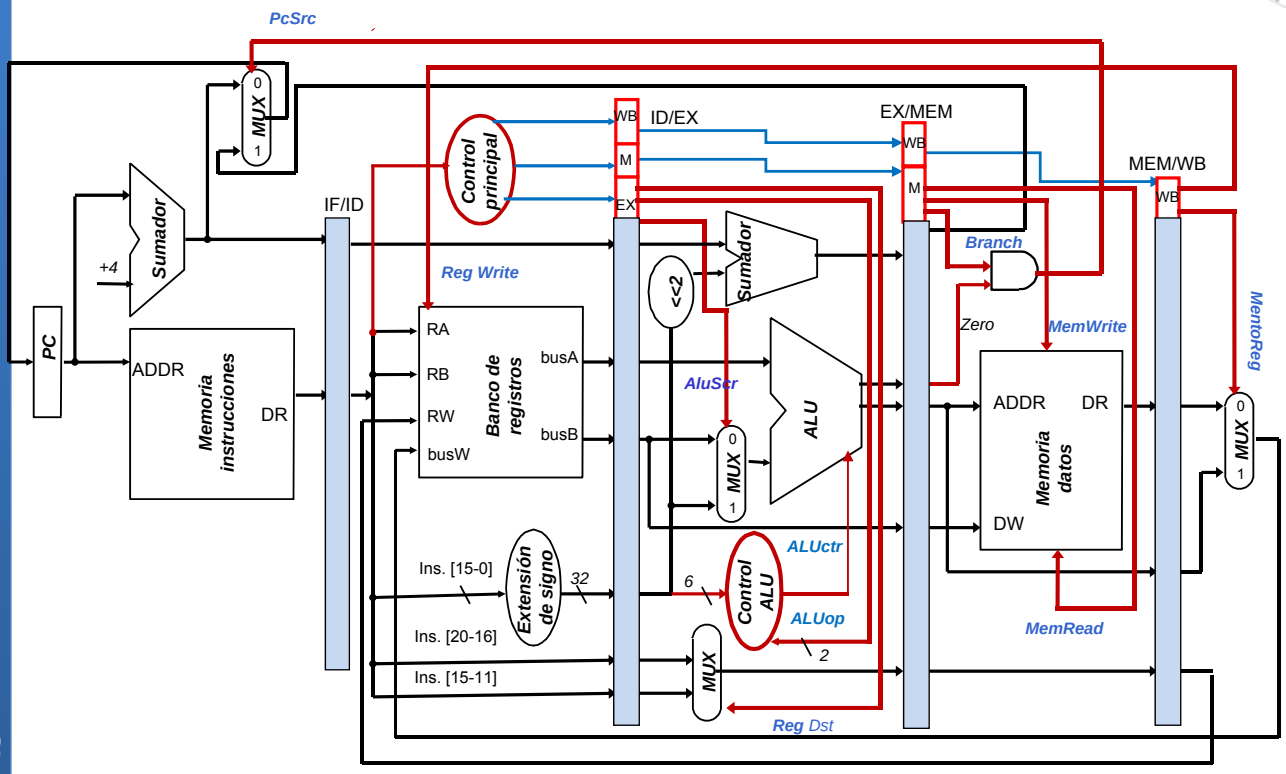
op	RegDst	ALUSrc	ALUop	MemRead	MemWrite	Branch	RegWrite	MemtoReg
100011 (lw)	0	1	00	1	0	0	1	0
101011 (sw)	X	1	00	0	1	0	0	X
000100 (beq)	X	0	01	0	0	1	0	X
000000 (tipo-R)	1	0	10	0	0	0	1	1

El control de la ALU se determina por ALUop (que depende del tipo de instrucción) y el campo de función en las instrucciones de tipo-R



4. Segmentación: Riesgos estructurales

Ruta de datos del procesador segmentado sin riesgos estructurales



5. Segmentación : Riesgos de datos

Riesgos de datos:

- Se produce un riesgo si existe dependencia entre instrucciones que se ejecutan concurrentemente
- Este tipo de riesgos aumentan en operaciones multiciclo
- **Tres tipos** diferentes:
  - ✓ **Lectura después de escritura** **(LDE)**
  - ✓ **Escritura después de lectura** **(EDL)**
  - ✓ **Escritura después de escritura** **(EDE)**

5. Segmentación : Riesgos de datos

Lectura después de escritura (LDE)

*Add **r1**,r2,r3 – escribe el registro r1*

*Add r4,**r1**,r2—lee el registro r1*

- **Se produce riesgo si r1 se lee antes de que lo escriba la primera instrucción**

Escritura después de lectura (EDL)

*Add r1,**r4**,r3 – lee el registro r4*

*Add **r4**,r5,r2—escribe el registro r4*

- **Se produce riesgo si r4 se escribe antes de que lo lea la primera instrucción**

Escritura después de escritura (EDE)

*Add **r4**,r2,r3 – escribe el registro r4*

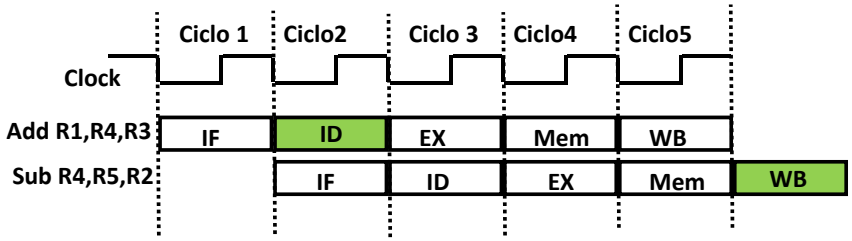
*Add **r4**,r1,r2—escribe el registro r4*

- **Se produce riesgo si r4 de la segunda instrucción se escribe antes de que lo escriba la primera instrucción**

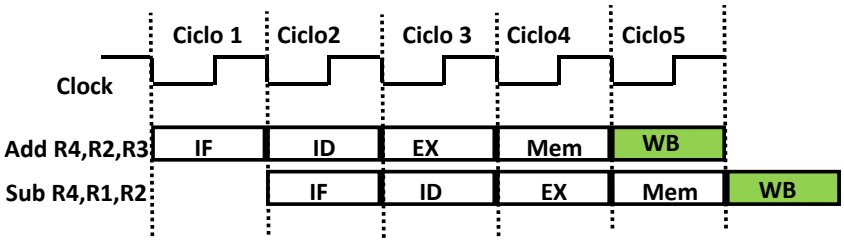
5. Segmentación : Riesgos de datos

Los riesgos EDL y EDE

- Escritura después de lectura: EDL



- Escritura después de escritura: EDE

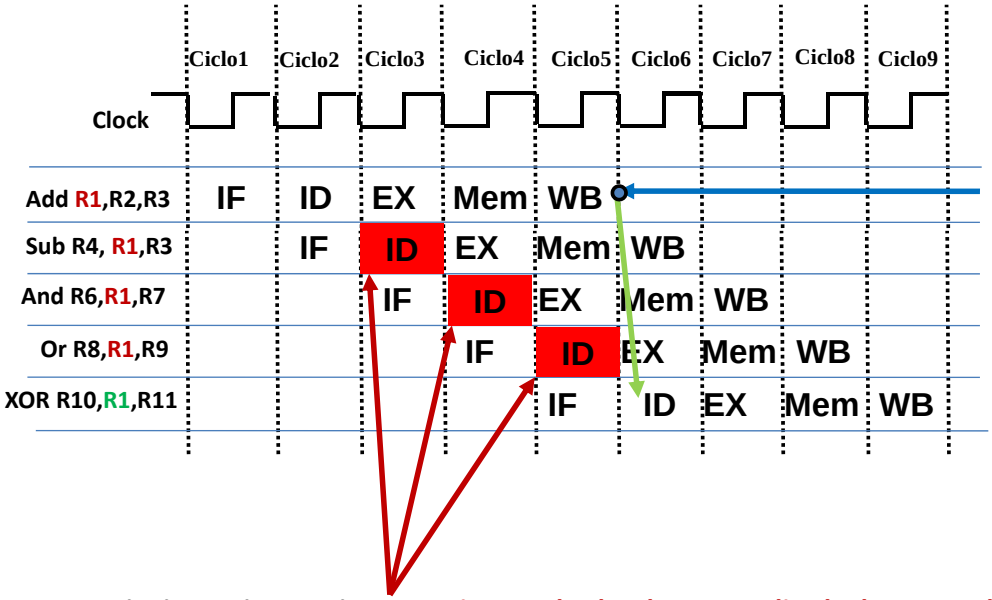


- **Estos riesgos No se dan en el pipeline lineal**
  - Se leen los registros en el final de la segunda etapa
  - Las instrucciones escriben en el BR en la última etapa
  - Todas las instrucciones tienen igual duración

5. Segmentación: Riesgos de datos

Riesgos LDE

- Se dan en la siguiente situación



R1 se escribe al empezar el ciclo6

Las 3 siguientes instrucciones no tienen el valor de r1 actualizado, leen un valor antiguo

5. Segmentación: Riesgos de datos

¿Cómo resolver los riesgos LDE?

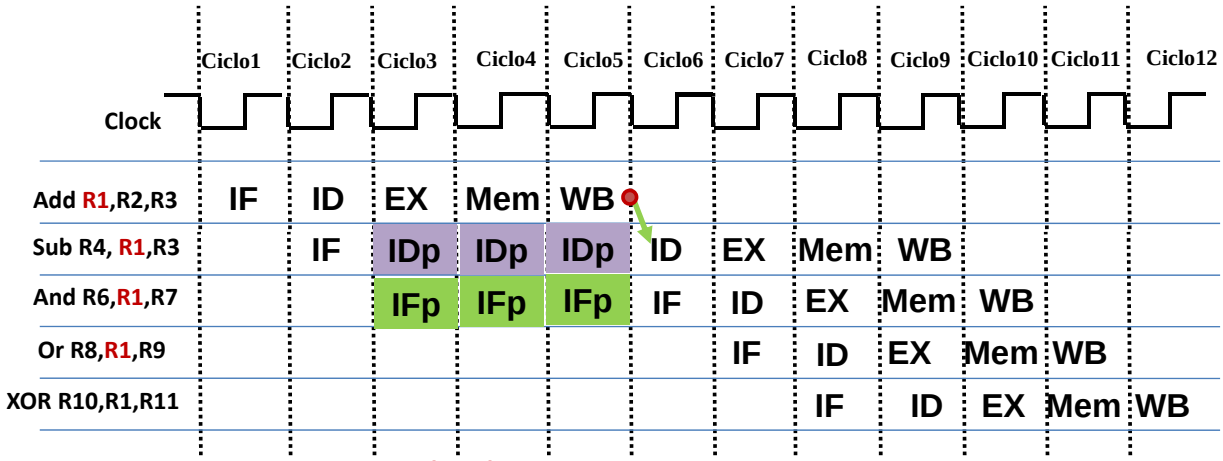
- **Solución 1:** Detener el pipeline
  - **¿En qué etapa se realiza la parada?**
    - Depende del diseño de la R.D
    - Nosotros vamos a suponer que **las paradas se realizan en decodificación**
    - **Si se realiza en otra etapa lo especificará el enunciado del problema**
  - **¿Cómo afectan las paradas a la ejecución de un programa?**
    - Las instrucciones que están en etapas anteriores a la etapa de parada también se paran
    - Las instrucciones que están en etapas posteriores a la etapa de parada siguen ejecutándose

- **Solución 2:** Reordenar código
  - **Si se puede, no siempre es posible**
  - Puede minimizar las paradas

5. Segmentación: Riesgos de datos

¿Cómo resolver los riesgos LDE?

- **Solución 1:** Detener el pipeline (en la etapa de decodificación)



3 ciclos de espera

Sólo se está ejecutando la primera instrucción

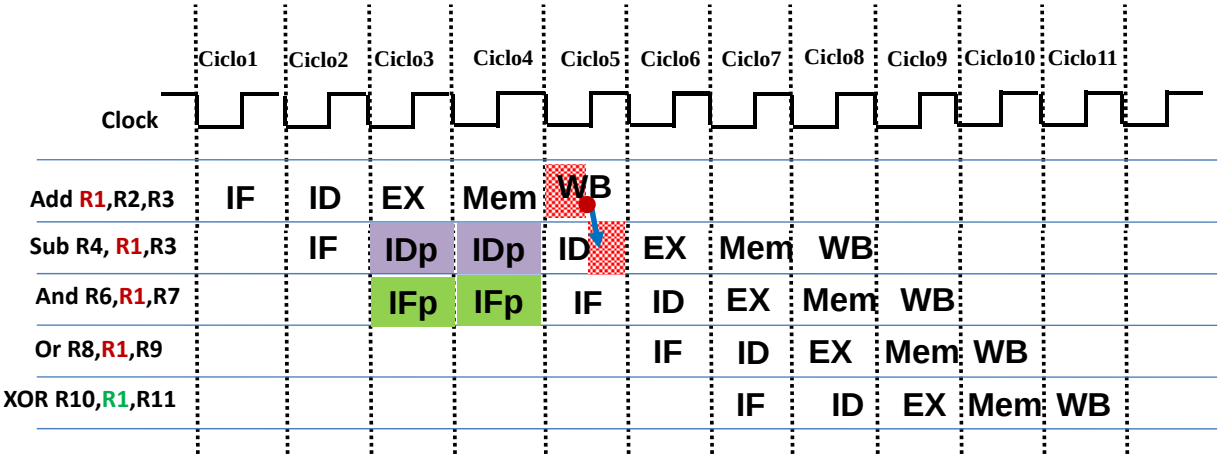
IDp **parada por riesgo LDE** entre instrucción 1 e instrucción 2

IFp **parada por fallo estructural**, la etapa está ocupada por la instrucción anterior

5. Segmentación: Riesgos de datos

¿Cómo resolver los riesgos LDE?

– **Añadimos una mejora**: cambiamos el **Banco de Registros** por uno que **escribe en la primera mitad del ciclo y lee en la segunda mitad del ciclo**



2 ciclos de espera

Sólo se está ejecutando la primera instrucción

IDp **parada por riesgo LDE** entre instrucción 1 e instrucción 2

IFp **parada por fallo estructural**, la etapa está ocupada por la instrucción anterior

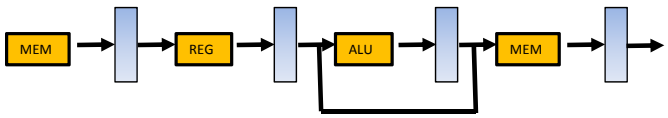
5. Segmentación : Riesgos de datos

¿Cómo resolver los riesgos LDE?

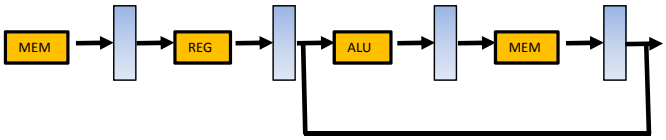
– **Solución 3: Cortocircuito (forwarding)**

– **Enviar el dato cuando esté calculado a las etapas que lo necesiten sin esperar a WB**

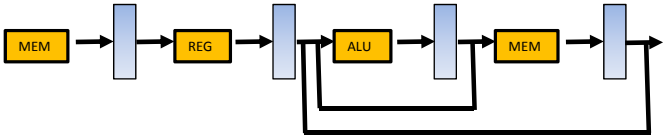
- Situación:
  add **r1**,r2,r3
  sub r4,**r1**,r3
  and r6,r5,r7



- Situación:
  add **r1**,r2,r3
  sub r4,r5,r3
  and r6,**r1**,r7



- Situación:
  add **r1**,r2,r3
  sub r4,**r1**,r3
  and r6,**r1**,r7



- **Para implementar el cortocircuito necesitamos dos caminos de datos:**
  – Desde el registro de pipeline EX/MEM (salida de la ALU) a entrada ALU
  – Desde el registro de pipeline MEM/WB (salida de la memoria) a entrada ALU

5. Segmentación : Riesgos de datos

¿Cómo resolver los riesgos LDE?

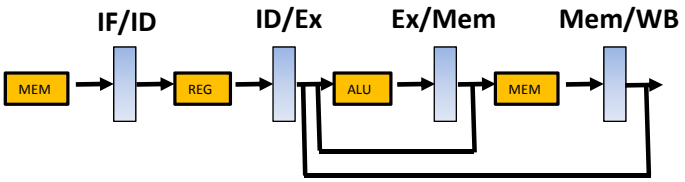
– **Solución 3:** Cortocircuito (forwarding)

Información necesaria para implementar el cortocircuito:

Registro a escribir en última etapa (Rd en Tipo-R y Rt en Lw)
- EX/MEM.Rd
- MEM/WB.Rd

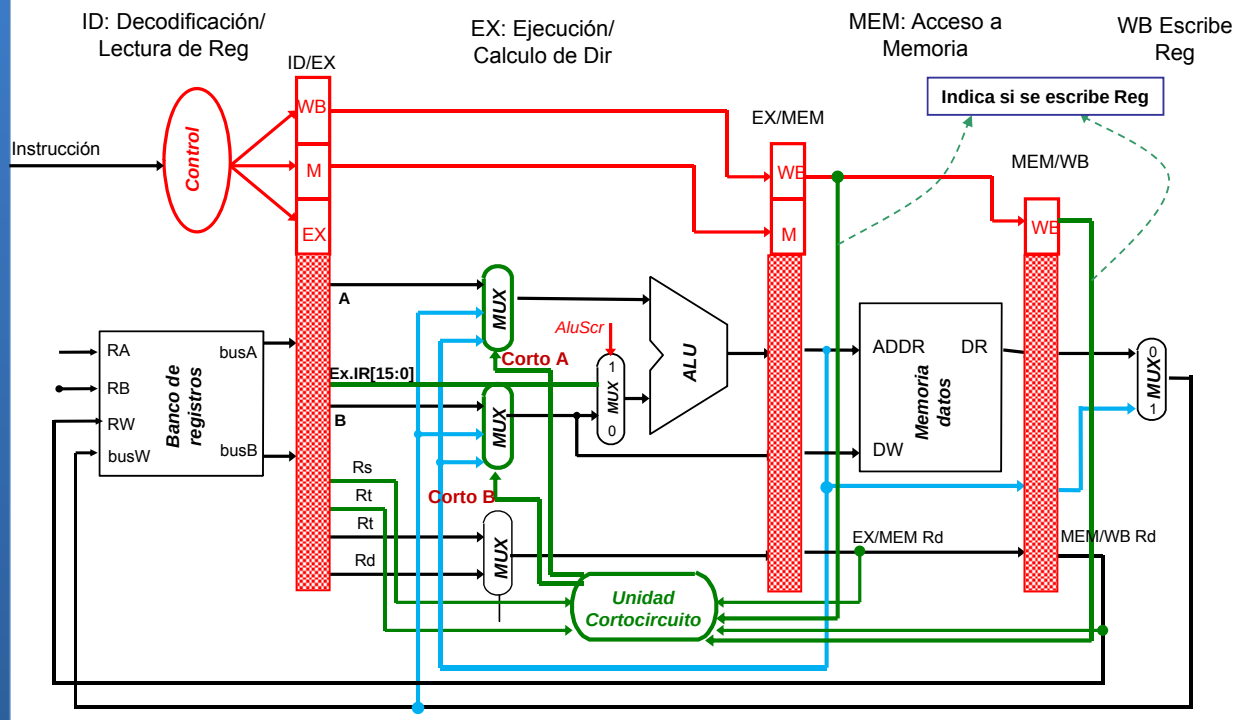
Registros que se leen en segunda etapa (Rs y Rt)
- ID/EX.Rt
- ID/EX.Rs

Información sobre si se escribe en el banco de registros
- EX/MEM.RegWrite
- MEM/WB.RegWrite



5. Segmentación : Riesgos de datos

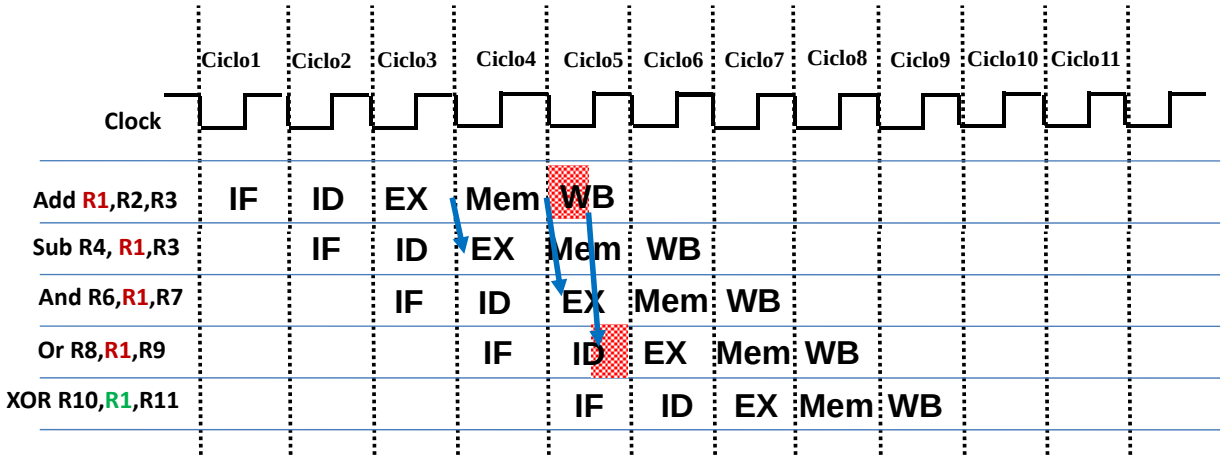
Riesgos LDE: Implementación del cortocircuito



5. Segmentación: Riesgos de datos

¿Cómo resolver los riesgos LDE?

– **Solución 3:** Cortocircuito (forwarding)

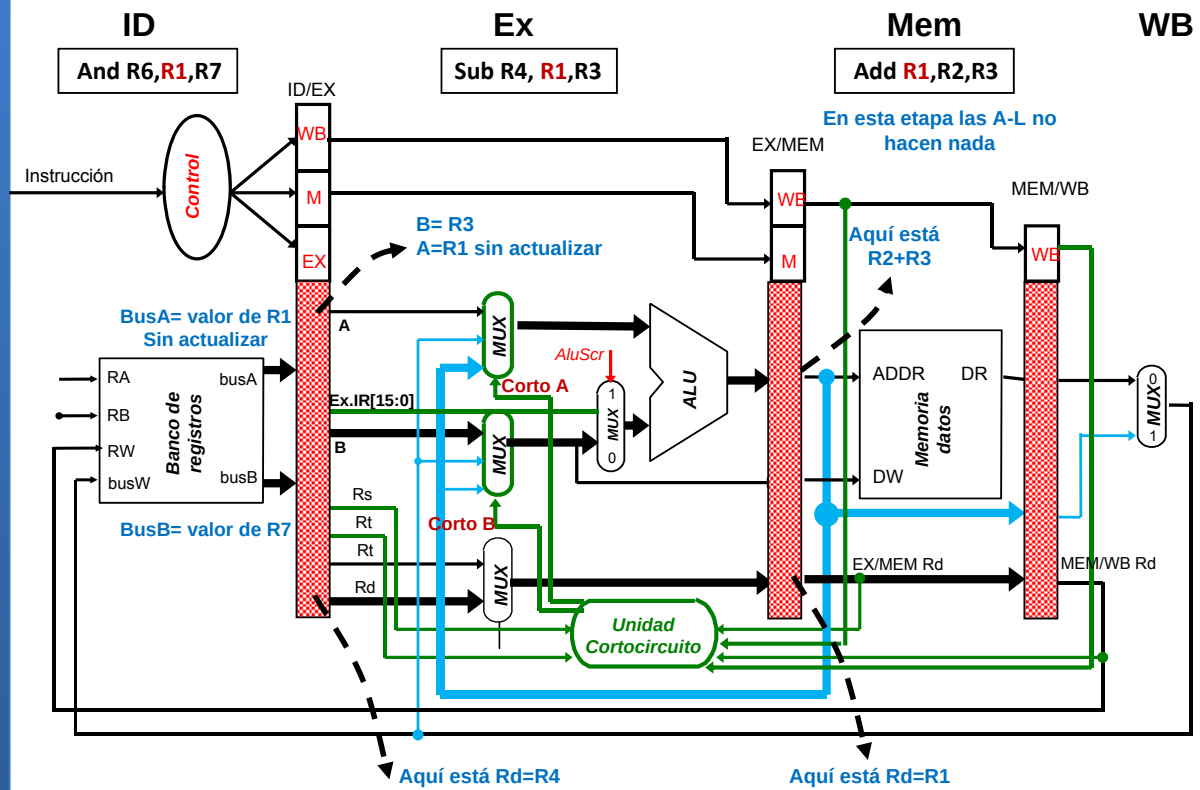


No hay ciclos de espera

Se pueden ejecutar todas las instrucciones sin problemas

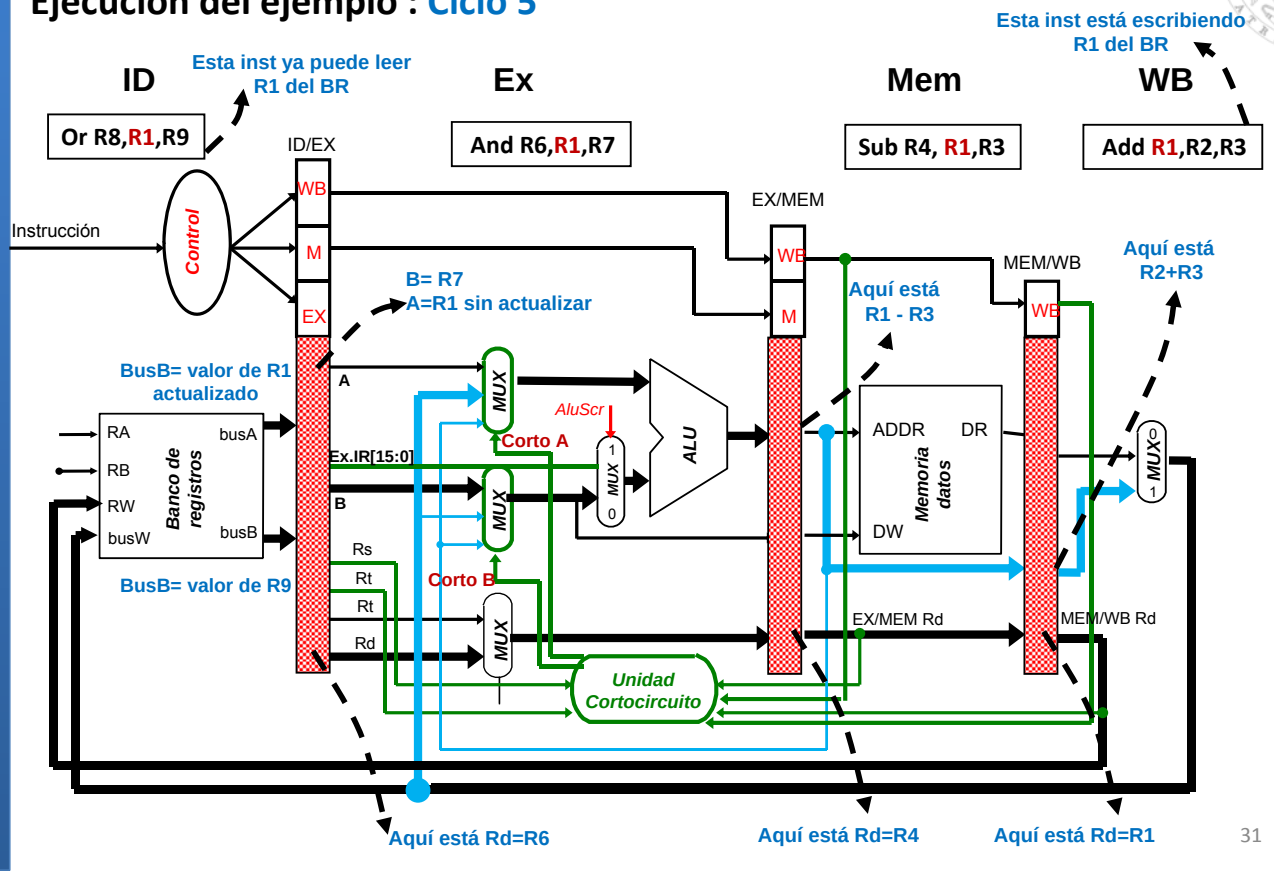
5. Segmentación : Riesgos de datos

Ejecución del ejemplo: Ciclo 4



5. Segmentación : Riesgos de datos

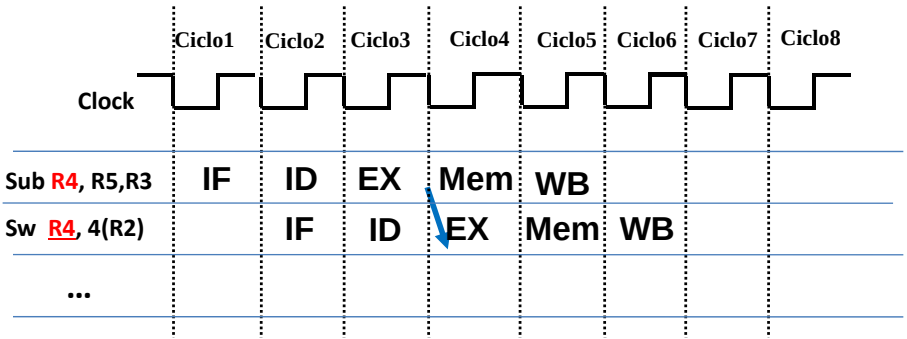
Ejecución del ejemplo : Ciclo 5



5. Segmentación: Riesgos de datos

Riesgo LDE: ejemplo 1 con instrucción store

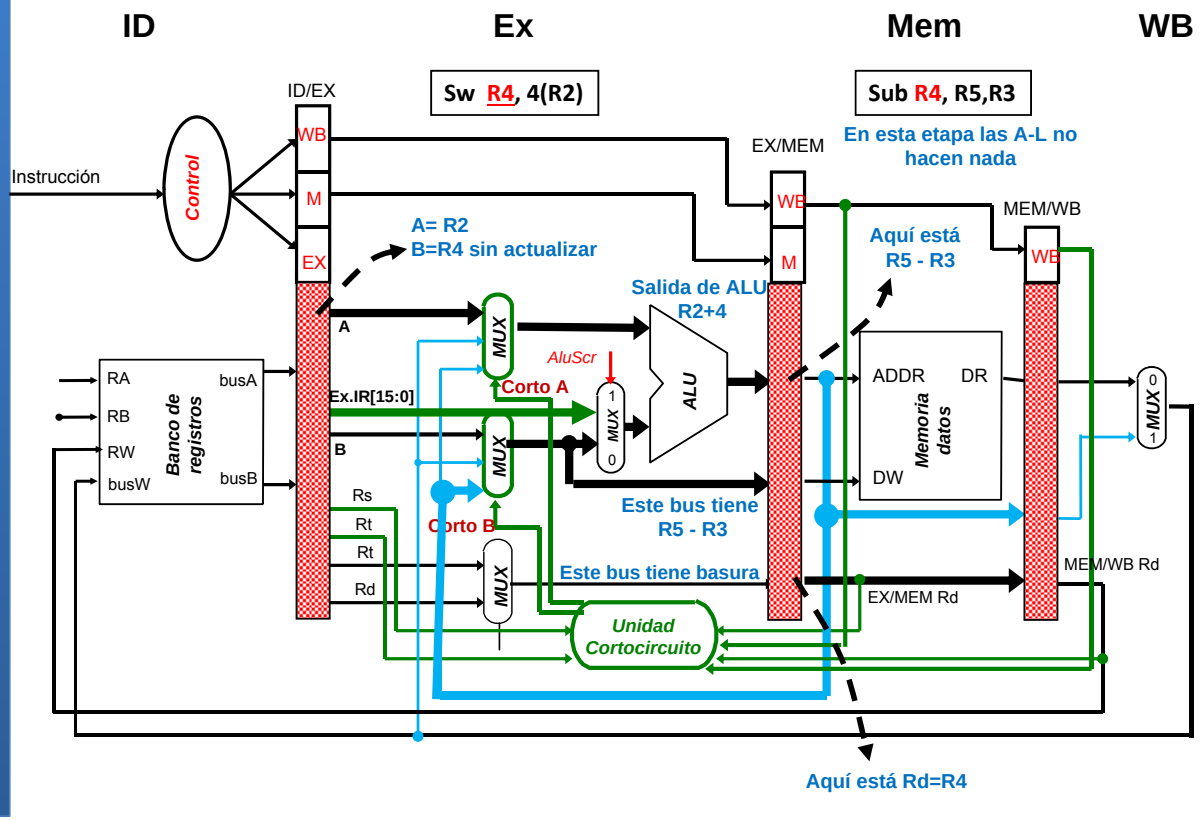
- El store escribe en memoria en la etapa Mem pero en nuestra ruta de datos **el dato que va a escribir** en memoria lo **tiene que tener en la etapa Ex**
  - **Se puede resolver con cortocircuito**



No hay ciclos de espera

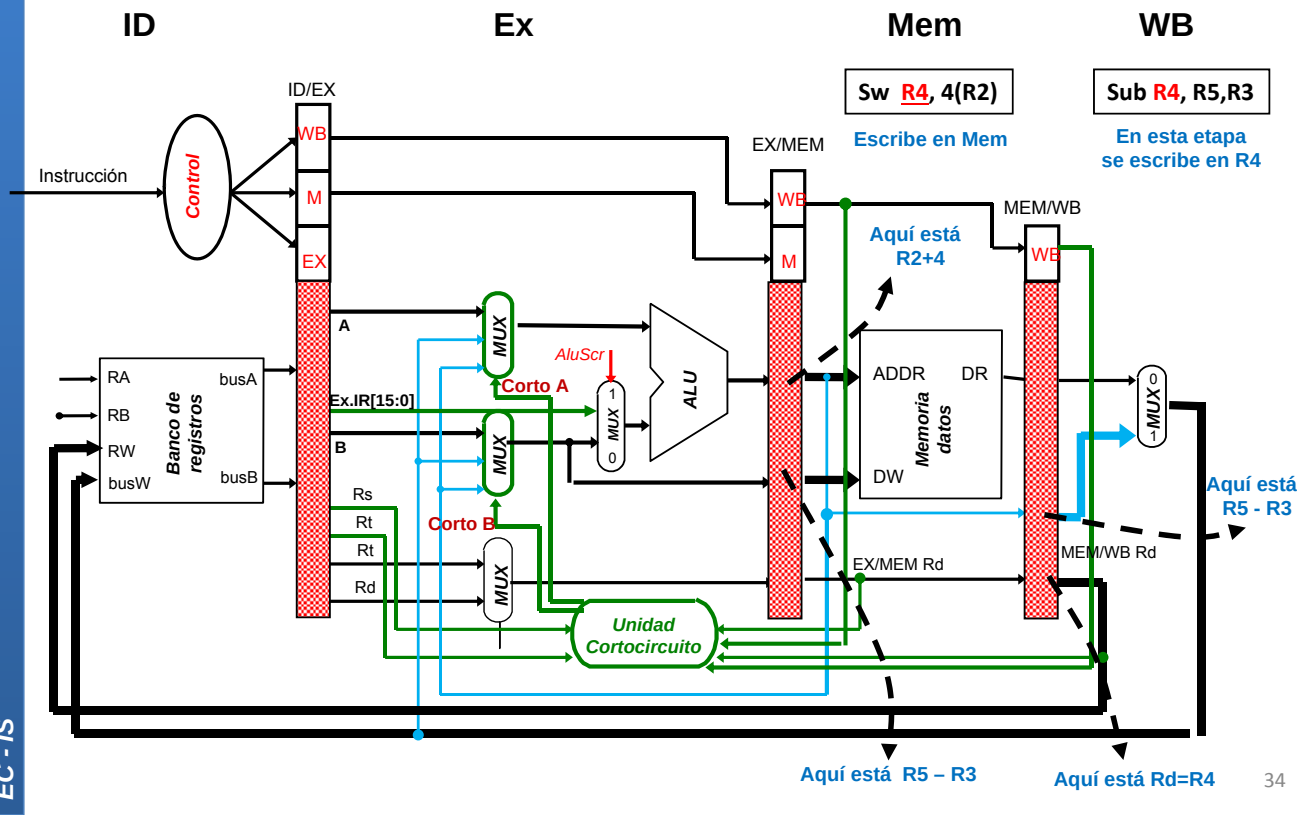
5. Segmentación : Riesgos de datos

Ejecución del ejemplo 1: Ciclo 4



5. Segmentación : Riesgos de datos

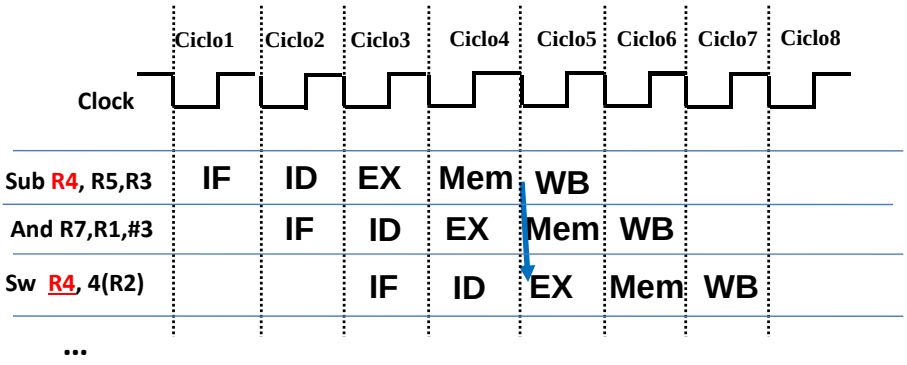
Ejecución del ejemplo 1: Ciclo 5



5. Segmentación: Riesgos de datos

Riesgo LDE: ejemplo 2 con instrucción store

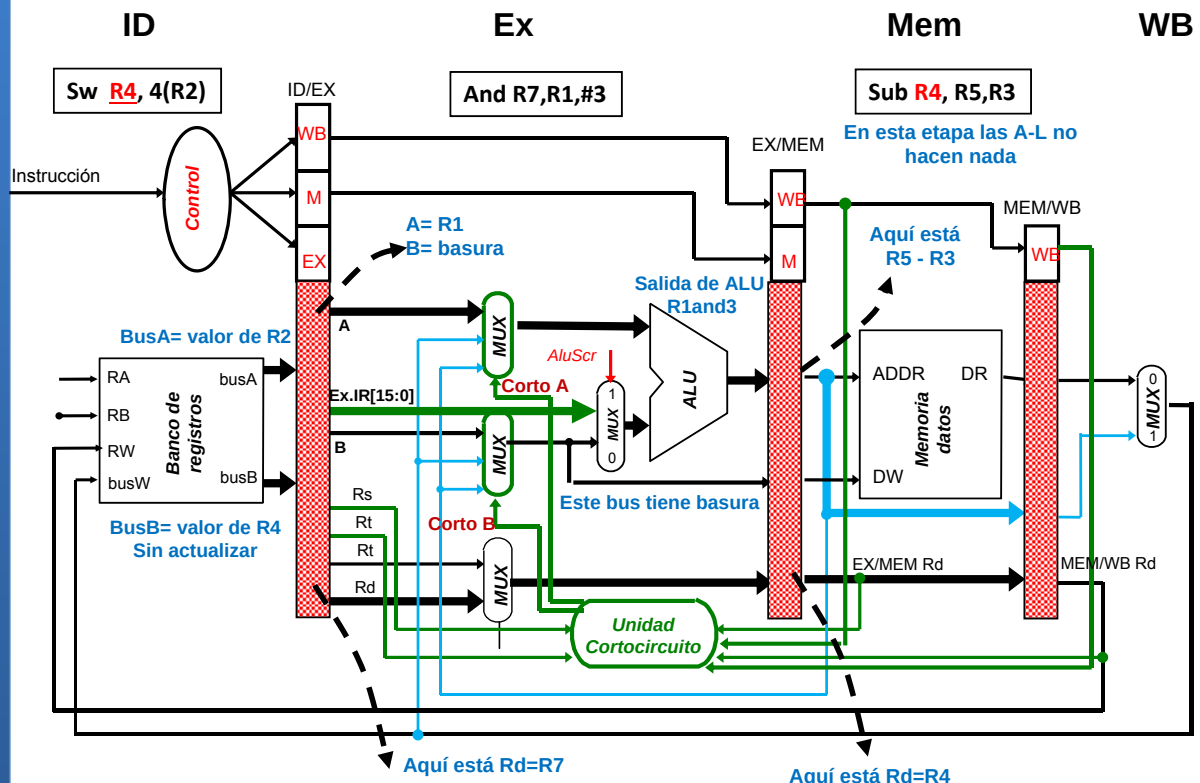
- El store escribe en memoria en la etapa Mem pero en nuestra ruta de datos **el dato que va a escribir** en memoria lo **tiene que tener en la etapa Ex**
  - **Se puede resolver con cortocircuito**



No hay ciclos de espera

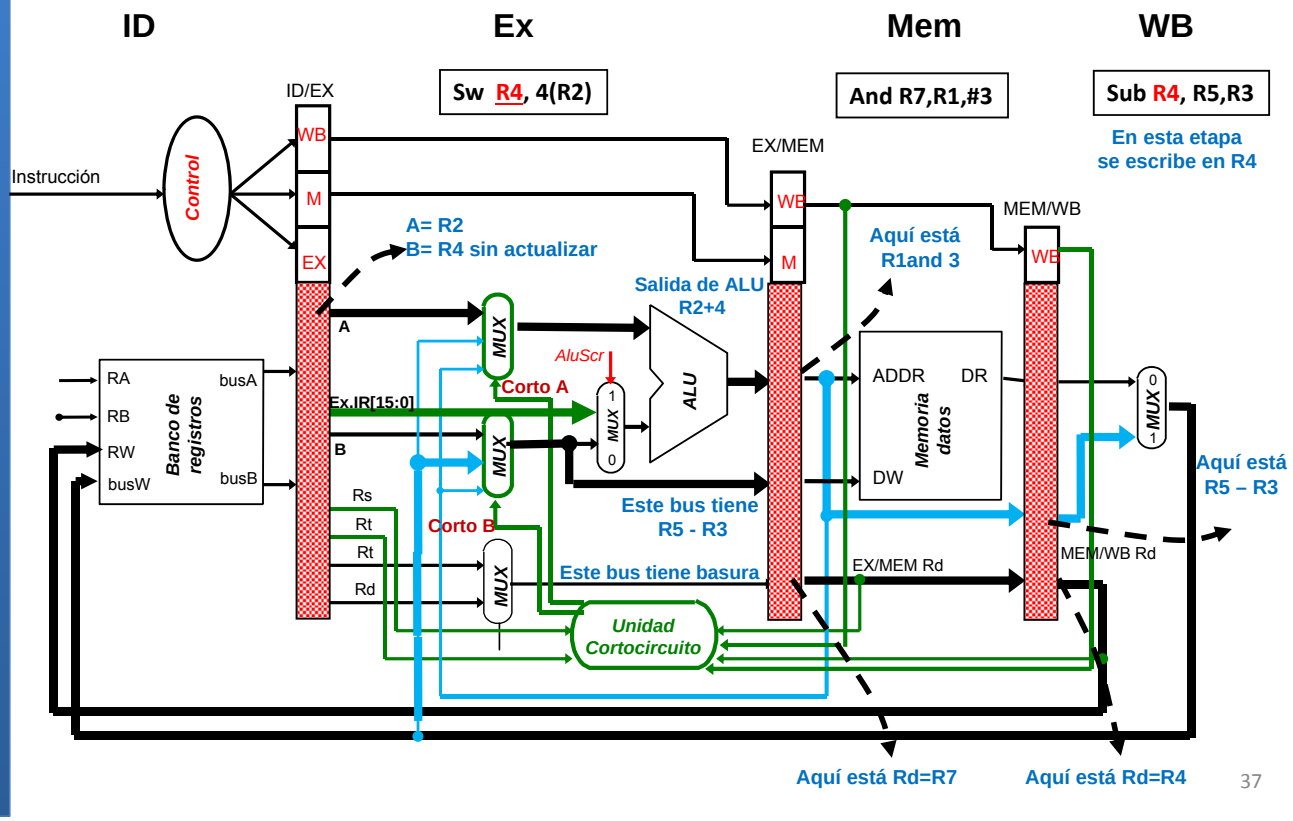
5. Segmentación : Riesgos de datos

Ejecución del ejemplo 2: Ciclo 4



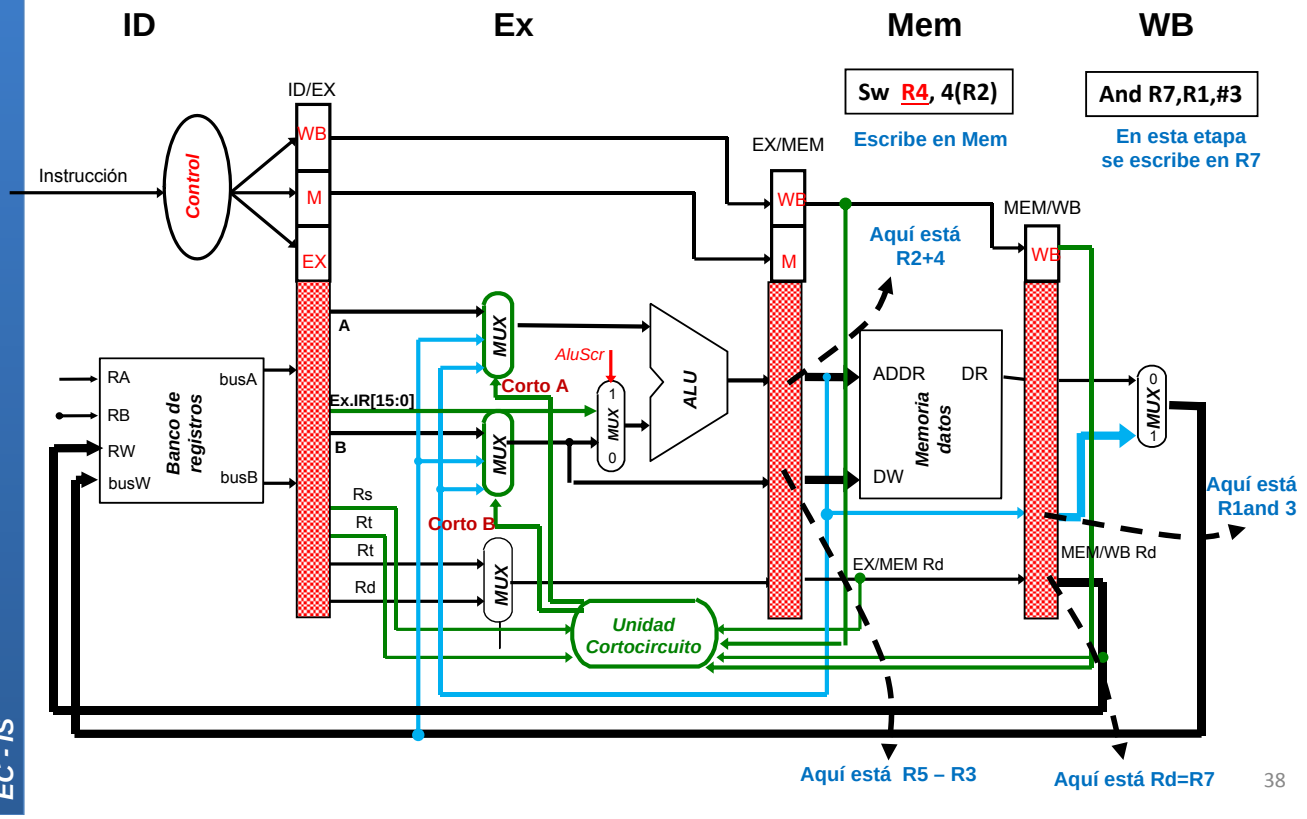
5. Segmentación : Riesgos de datos

Ejecución del ejemplo 2: Ciclo 5



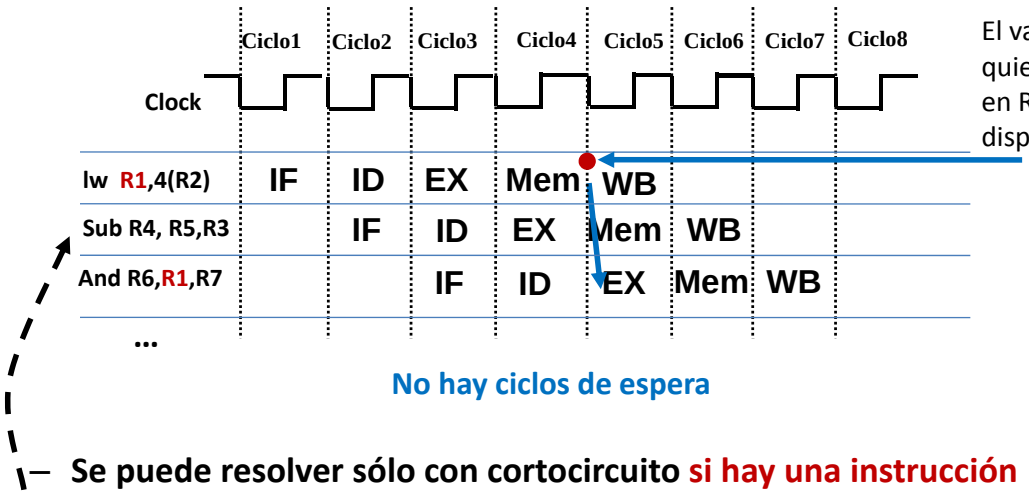
5. Segmentación : Riesgos de datos

Ejecución del ejemplo 2: Ciclo 6



5. Segmentación: Riesgos de datos

Riesgo LDE: caso particular, **el dato lo proporciona un Load**



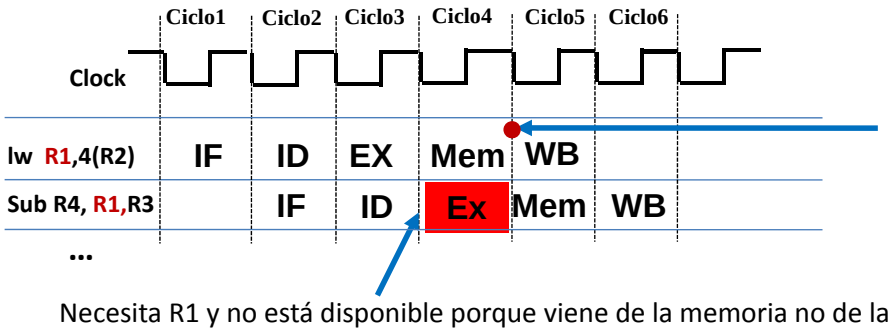
No hay ciclos de espera

– **Se puede resolver sólo con cortocircuito si hay una instrucción intermedia**

T2

5. Segmentación: Riesgos de datos

Riesgo LDE: caso particular, **el dato lo proporciona un Load**



Necesita R1 y no está disponible porque viene de la memoria no de la ALU

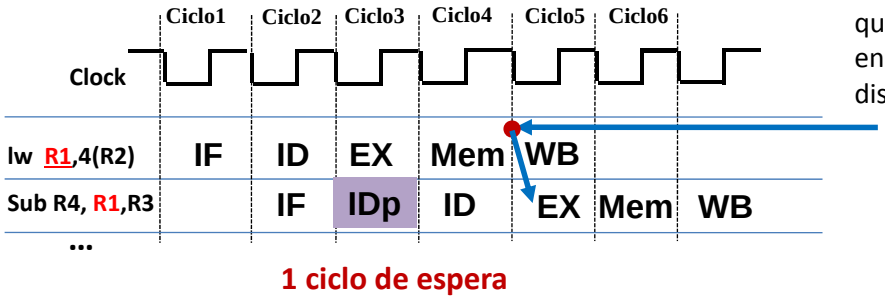
– **No se puede resolver sólo con el cortocircuito**

EC - IS

5. Segmentación: Riesgos de datos

Riesgo LDE: caso particular, **el dato lo proporciona un Load**

- **Solución HW:** Detección del riesgos y parada del procesador un ciclo



El valor que se quiere guardar en R1 está disponible

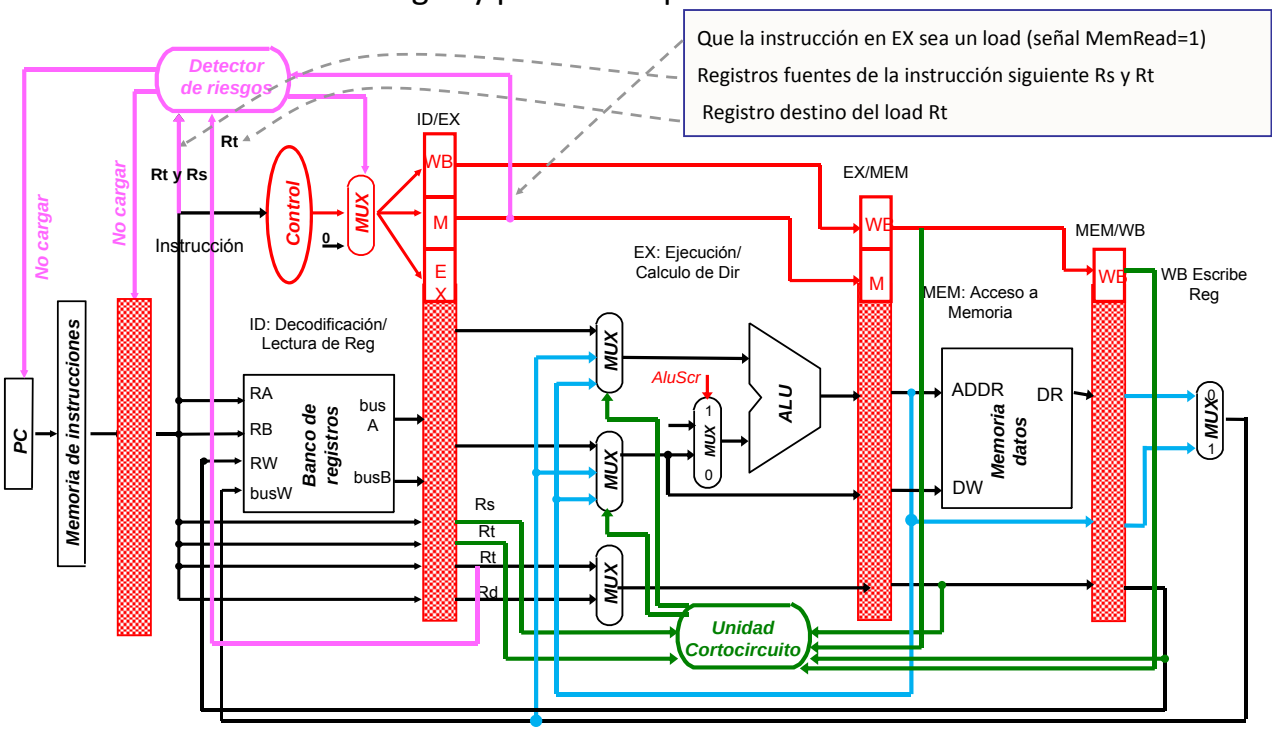
1 ciclo de espera

IDp **parada por riesgo LDE** entre instrucción 1 e instrucción 2

5. Segmentación : Riesgos de datos

Implementación de la Solución HW:

- Detección de riesgos y parada del procesador un ciclo



5. Segmentación : Riesgos de datos

Riesgo LDE: caso particular, **el dato lo proporciona un Load**

- **Solución SW**: Anticipar el Load en la planificación de instrucciones que hace el compilador

a := b + c
d := e - f

Código original:

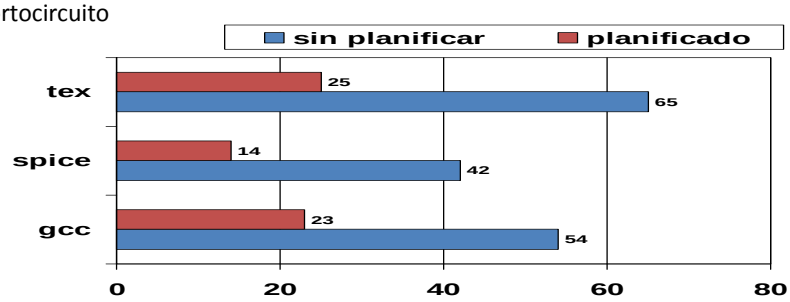
```
LW   Rb,b
LW   Rc,c
ADD  Ra,Rb,Rc
SW   a,Ra
LW   Re,e
LW   Rf,f
SUB  Rd,Re,Rf
SW   d,Rd
```

Código con planificación de instrucciones

```
LW   Rb,b
LW   Rc,c
LW   Re,e
ADD  Ra,Rb,Rc
LW   Rf,f
SW   a,Ra
SUB  Rd,Re,Rf
SW   d,Rd
```

Los stores no presentan riesgo LDE porque existe cortocircuito

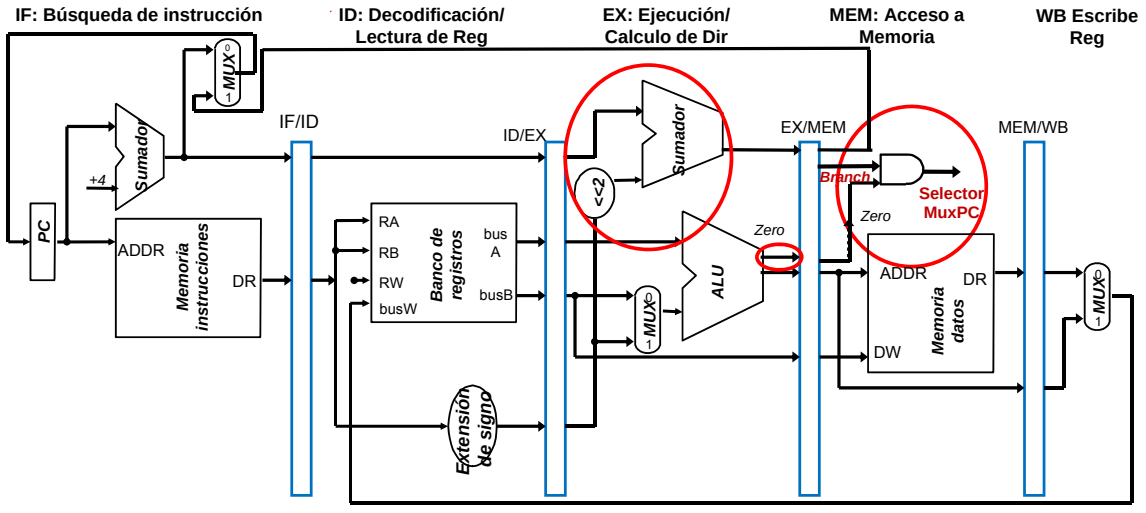
% de load que provocan un ciclo de parada



6. Segmentación : Riesgos de control

Riesgos de control: **¿Por qué aparecen?**

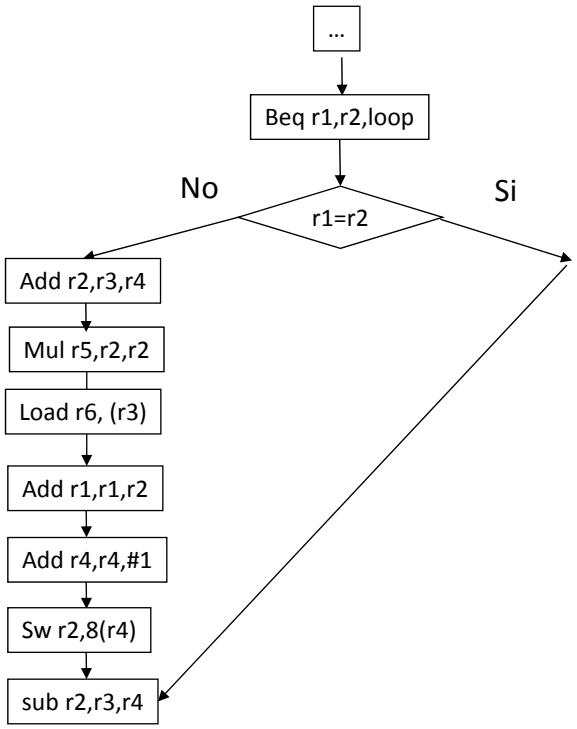
- Para saltar se necesita:
  - Haber calculado la **dirección de salto**
  - Saber **si se cumple la condición** de salto
- **Aparece el riesgo porque**
  - En la ruta de datos que hemos estudiado **ambos datos se calculan en Ex**
  - **Si el salto se toma, la dirección del salto se carga en PC al final de Mem,** cuando llega el flanco de reloj



6. Segmentación : Riesgos de control

Ejemplo:

```
        ...
        Beq r1,r2,loop
        Add r2,r3,r4
        Mul r5,r2,r2
        load r6, (r3)
        Add r1,r1,r2
        Add r4,r4,#1
        Sw   r2,8(r4)
Loop    sub r2,r3,r4
```

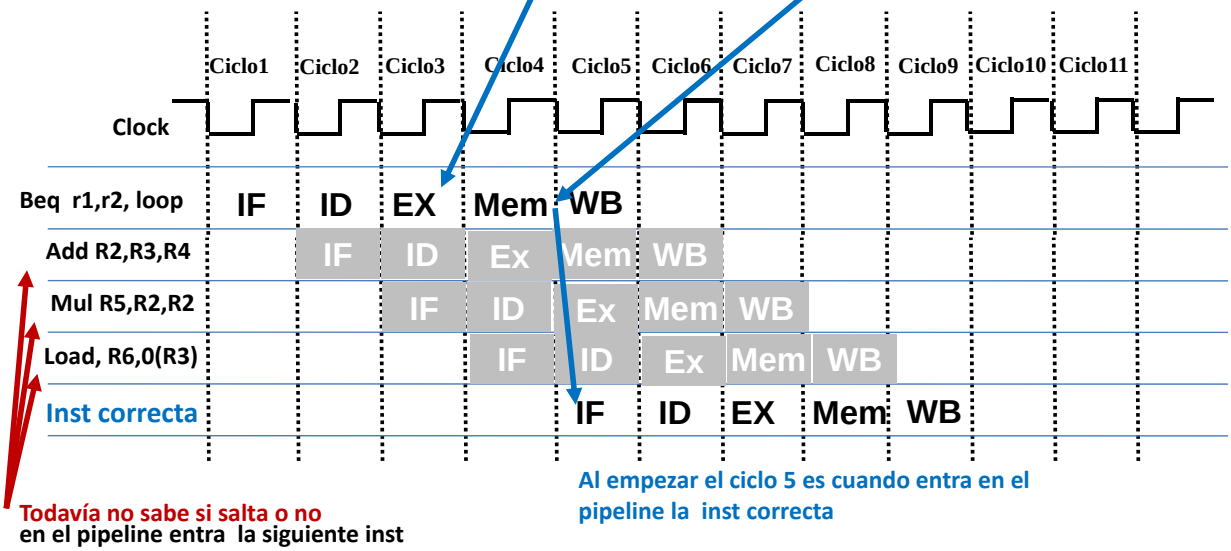


6. Segmentación: Riesgos de control

Ejemplo:

El cálculo de la **dirección de salto** y la evaluación de **la condición** se realizan en la **etapa EX**

La nueva dirección se carga en PC cuando llega el flanco de reloj **al entrar en WB**



Todavía no sabe si salta o no en el pipeline entra la siguiente inst

Al empezar el ciclo 5 es cuando entra en el pipeline la inst correcta

Se han ejecutado instrucciones que a lo mejor no debían ejecutarse

6. Segmentación : Riesgos de control

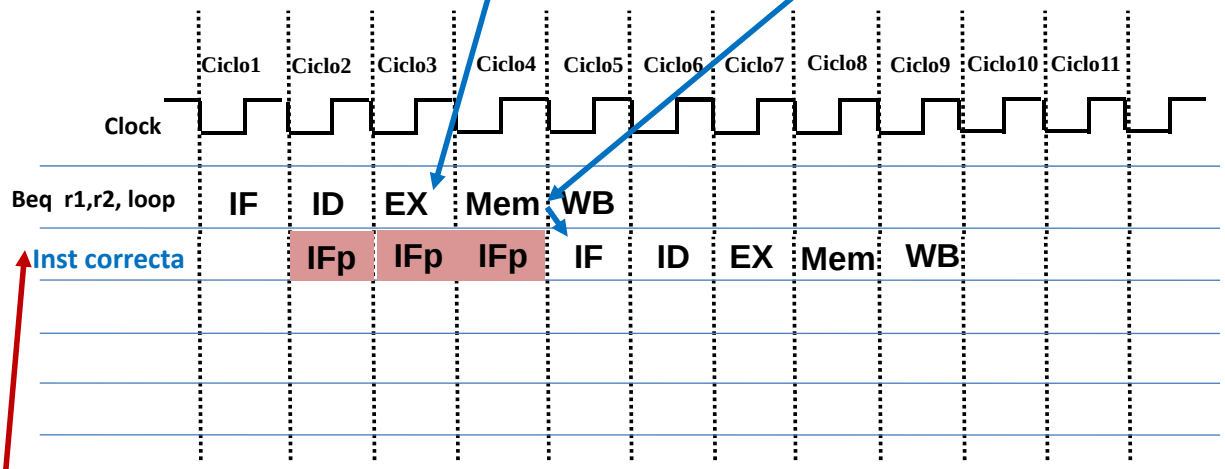
¿Cómo solucionar los riesgos de control?

– **Solución 1**: Hay que **esperar tres ciclos** para saber cuál es la instrucción siguiente al salto

- **Parando el pipeline**

El cálculo de la **dirección de salto** y la evaluación de **la condición** se realizan en la **etapa EX**

La nueva dirección se carga en PC cuando llega el flanco de reloj **al entrar en WB**



Hasta que no se sepa si salta o no el pipeline está parado

Al empezar **el ciclo 5** es cuando entra en el pipeline la inst correcta

3 ciclos de penalización

T2

6. Segmentación : Riesgos de control

¿Cómo solucionar los riesgos de control?

– **Solución 2**:

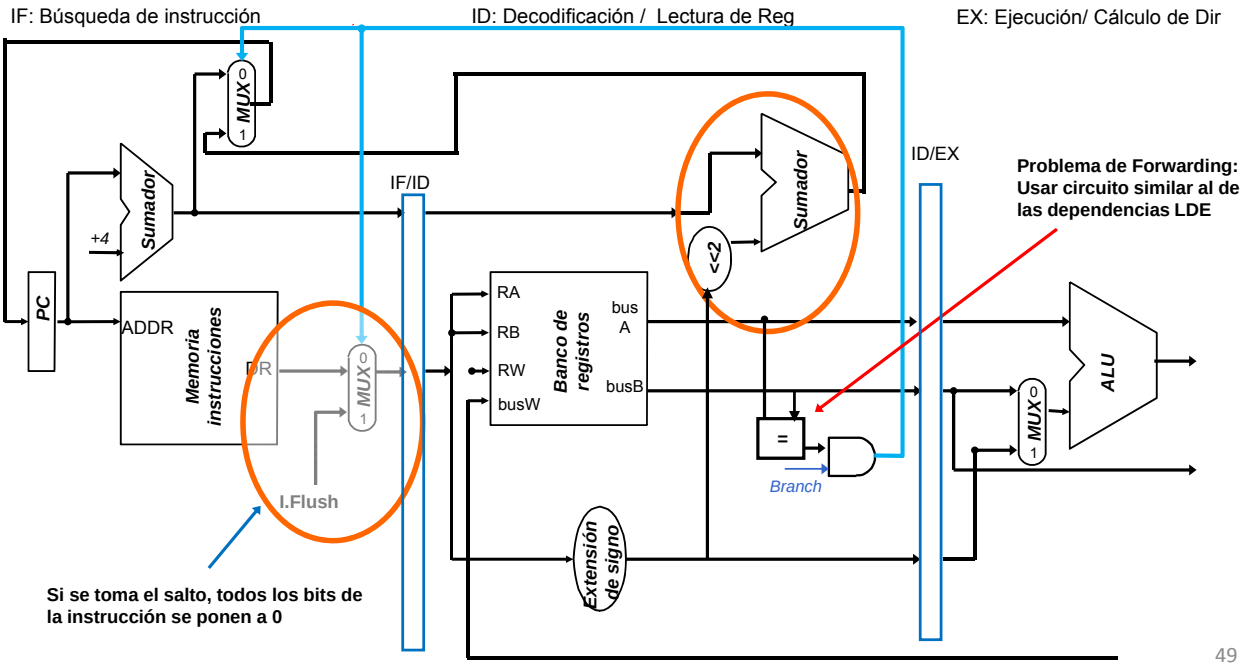
- Desplazar el **cálculo de la dirección** y la evaluación de **la condición** a la etapa **ID**
- Sólo hay que **esperar un ciclo** para saber la instrucción que sigue a la de salto. Este ciclo de espera se puede implementar:
  - **HW: Se introduce la siguiente instrucción**
    - **Si el salto se realiza:** se elimina la instrucción introducida
      - **Se ponen "0" en las etapas del pipeline para que no se realice ninguna acción**
      - ***1 ciclo de penalización***
    - **Si el salto NO se realiza:** se introduce en el pipeline la instrucción correcta
      - ***Ningún ciclo de penalización***
  - **SW: Saltos retardados**
    - Ejecutar instrucciones ***independientes del salto*** durante el ciclo de retardo
      - **Se ejecutarán siempre**, tanto si el salto se realiza como si no
    - ***Si es posible encontrar instrucciones que se tengan que ejecutar siempre***
      - ***Ningún ciclo de penalización***

EC - IS

6. Segmentación : Riesgos de control

Ruta de datos con la implementación HW de la solución 2:

- Calcular la **dirección de salto** y la evaluación de la **condición** en la **etapa ID**
- Eliminar la instrucción siguiente si el salto no se realiza



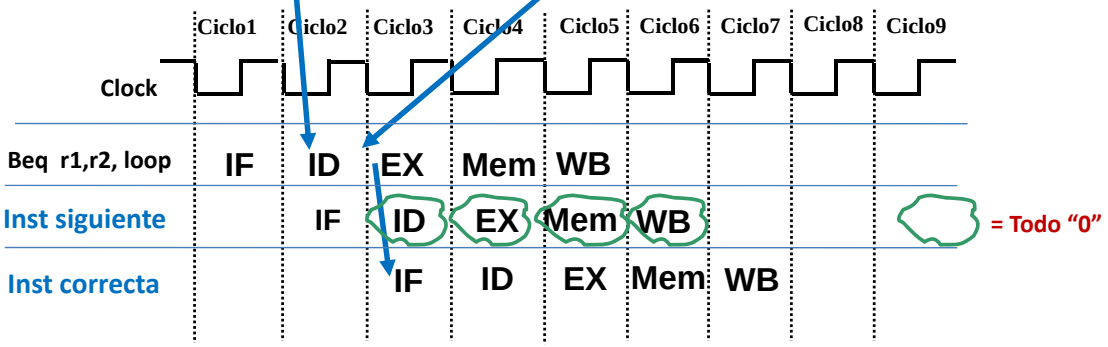
6. Segmentación : Riesgos de control

Ejemplo: Aplicando la Solución 2 con implementación HW

Si el salto se realiza

El cálculo de la **dirección de salto** y la evaluación de **la condición** se realizan en la **etapa ID**

La nueva dirección se carga en PC cuando llega el flanco de reloj **al entrar en Ex**



La Inst siguiente NO es la Inst correcta (se ponen "0")

Al empezar el ciclo 3 es cuando entra en el pipeline la inst correcta

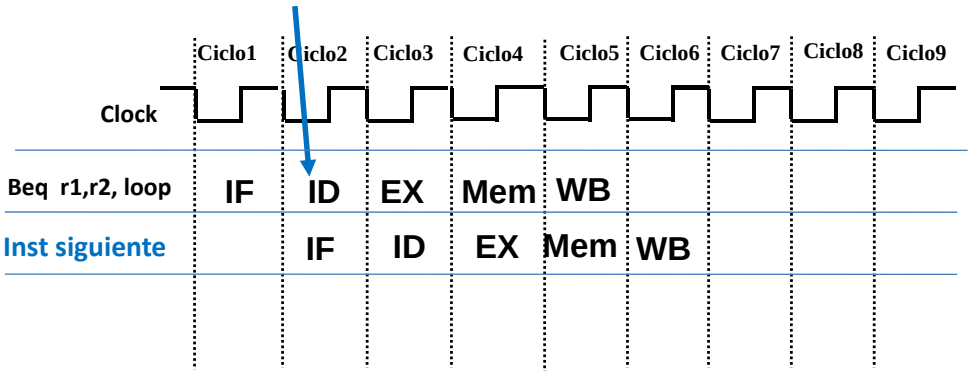
1 ciclos de penalización

6. Segmentación : Riesgos de control

Ejemplo: Aplicando la Solución 2 con implementación HW

Si el salto NO se realiza

El cálculo de la **dirección de salto** y la evaluación de **la condición** se realizan en la **etapa ID**



La Inst siguiente es la Inst correcta

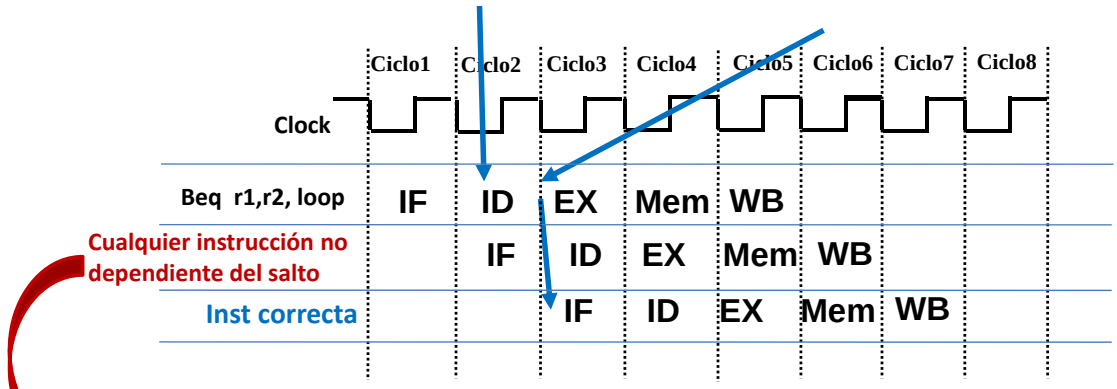
0 ciclos de penalización

6. Segmentación : Riesgos de control

Ejemplo: Aplicando la Solución 2 con implementación SW: Saltos retardados

El cálculo de la **dirección de salto** y la evaluación de **la condición** se realizan en la **etapa ID**

La nueva dirección se carga en PC cuando llega el flanco de reloj **al entrar en Ex**



No hay ciclos de penalización

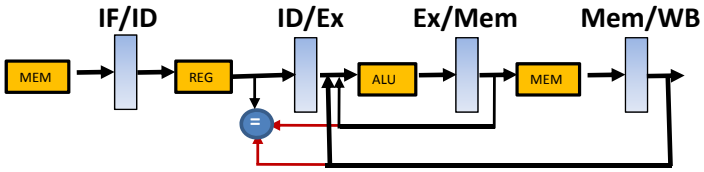
- **Lo ideal es elegir una instrucción que se tenga que ejecutar siempre**
- **Si no es posible, elegir una instrucción cuya ejecución no afecte al resultado**

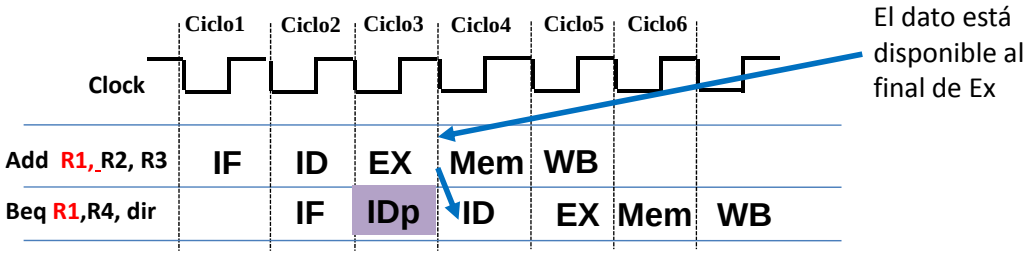
7. Riesgo de control + Riesgos de LDE

La instrucción de salto necesita un dato que proporciona la instrucción anterior

– **Problema**:
- El salto evalúa la condición en la etapa ID
- El cortocircuito tal y como está implementado no se lo proporciona

– **Solución**: **Ampliar el cortocircuito a la etapa ID**





1 ciclo de espera

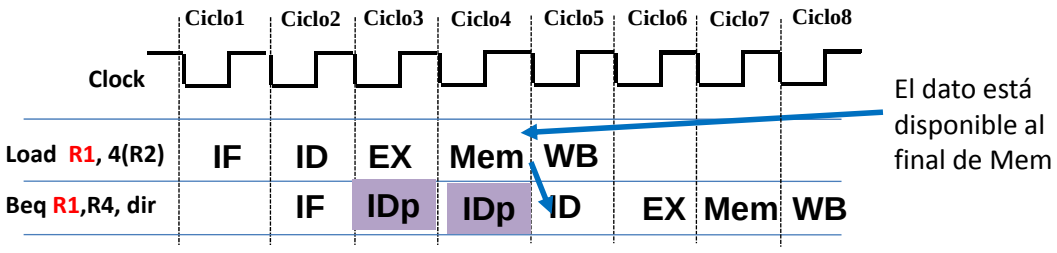
IDp **parada por riesgo LDE** entre instrucción 1 e instrucción 2

T2

7. Riesgo de control + Riesgos de LDE

La instrucción de salto necesita un dato que proporciona la instrucción anterior

– Si la instrucción que proporciona el dato es un Load



2 ciclos de espera

IDp **parada por riesgo LDE** entre instrucción 1 e instrucción 2

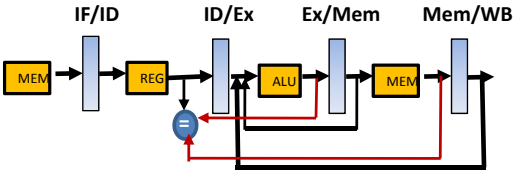
EC - IS

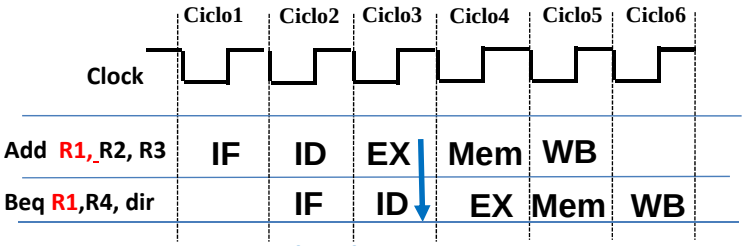
7. Riesgo de control + Riesgos de LDE

La instrucción de salto necesita un dato que proporciona la instrucción anterior

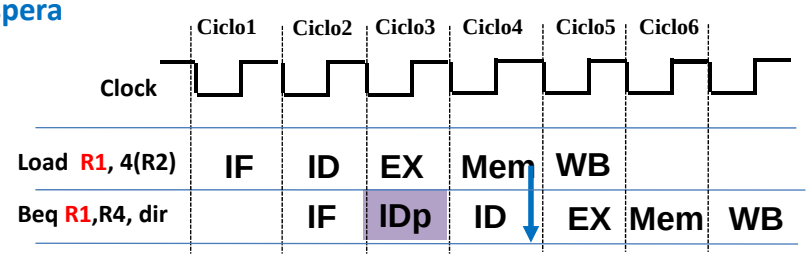
– Si el forwarding es combinacional

– El dato no se anticipa desde los registros del pipeline sino desde la salida de la ALU en el caso de una A-L o desde la salida de DM en el caso del load





0 ciclos de espera



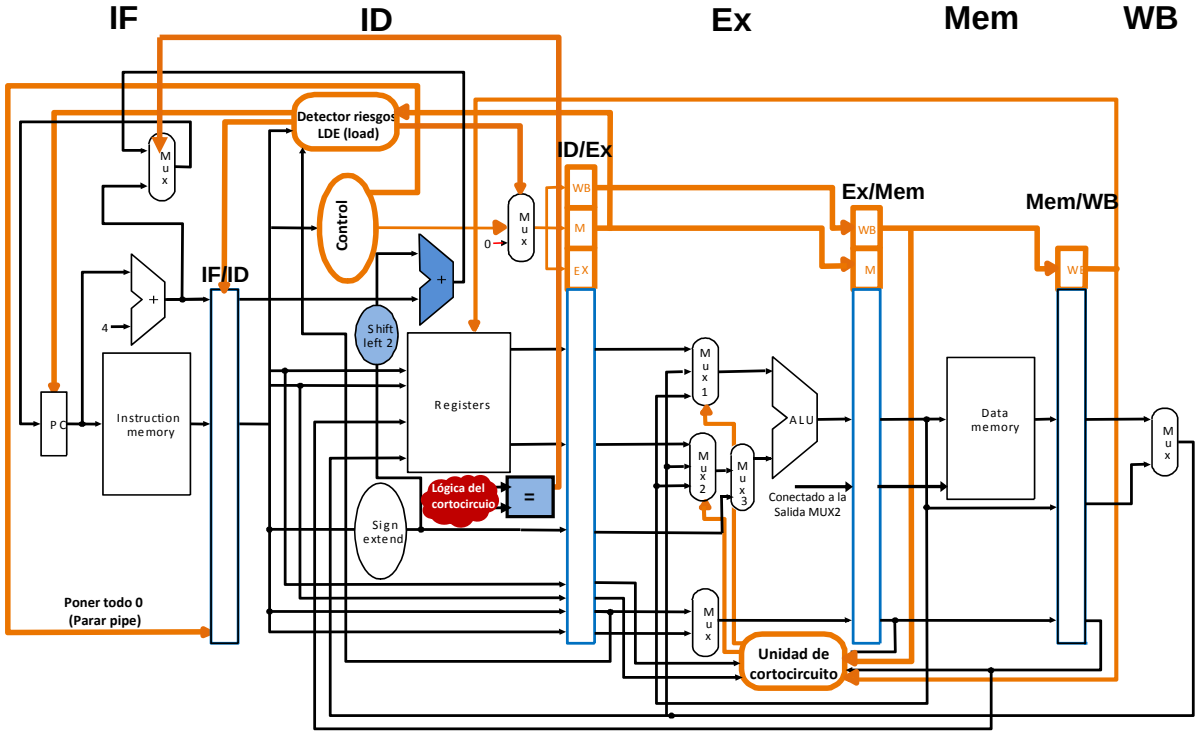
1 ciclo de espera

8. Resumen

Ruta datos completa del procesador segmentado

– Tiene implementadas las soluciones óptimas para solucionar los riesgos vistos hasta ahora

– **La entradas del comparador vienen de dos Mux los cuales eligen entre las salidas del B.R o el cortocircuito**



8. Resumen

Procesador segmentado

- ✓ Todas las instrucciones tienen igual duración
- ✓ Rendimiento ideal, una instrucción por ciclo CPI=1
- ✓ Riesgos estructurales y de datos EDE y EDL se resuelven por construcción
- ✓ Riesgos LDE en instrucciones tipo-R se solucionan con el cortocircuito
- ✓ Riesgos LDE en instrucciones de *load* implican paradas del procesador
  - ✓ Ayuda del compilador planificando las instrucciones
- ✓ Riesgos de control se solucionan:
  - HW: Si el salto se realiza se introduce en el pipeline una instrucción NOP (= "0")
  - SW: con Saltos retardados
- ✓ **Las instrucciones empiezan y terminan en orden**