Clase 23 - Paralelismo Parte 2

IIC2343 - Arquitectura de Computadores

Profesor:
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl

Resumen de la clase pasada

Arquitectura de Computadores: Mejoras y extensiones

En lo que hemos visto hasta ahora

- Una máquina programable que ejecuta programas
- Interacción con dispositivos de entrada y salida

Lo que nos queda ver **mejoras:**

- Mejora los accesos a memoria 
- Mejora en lo que respecta a la ejecución de más de un programa 
- Mejora en lo que respecta a la ejecución de instrucciones en la CPU ?

Flujo de instrucción: General

1- Lectura de instrucción desde memoria (**Fetch**)

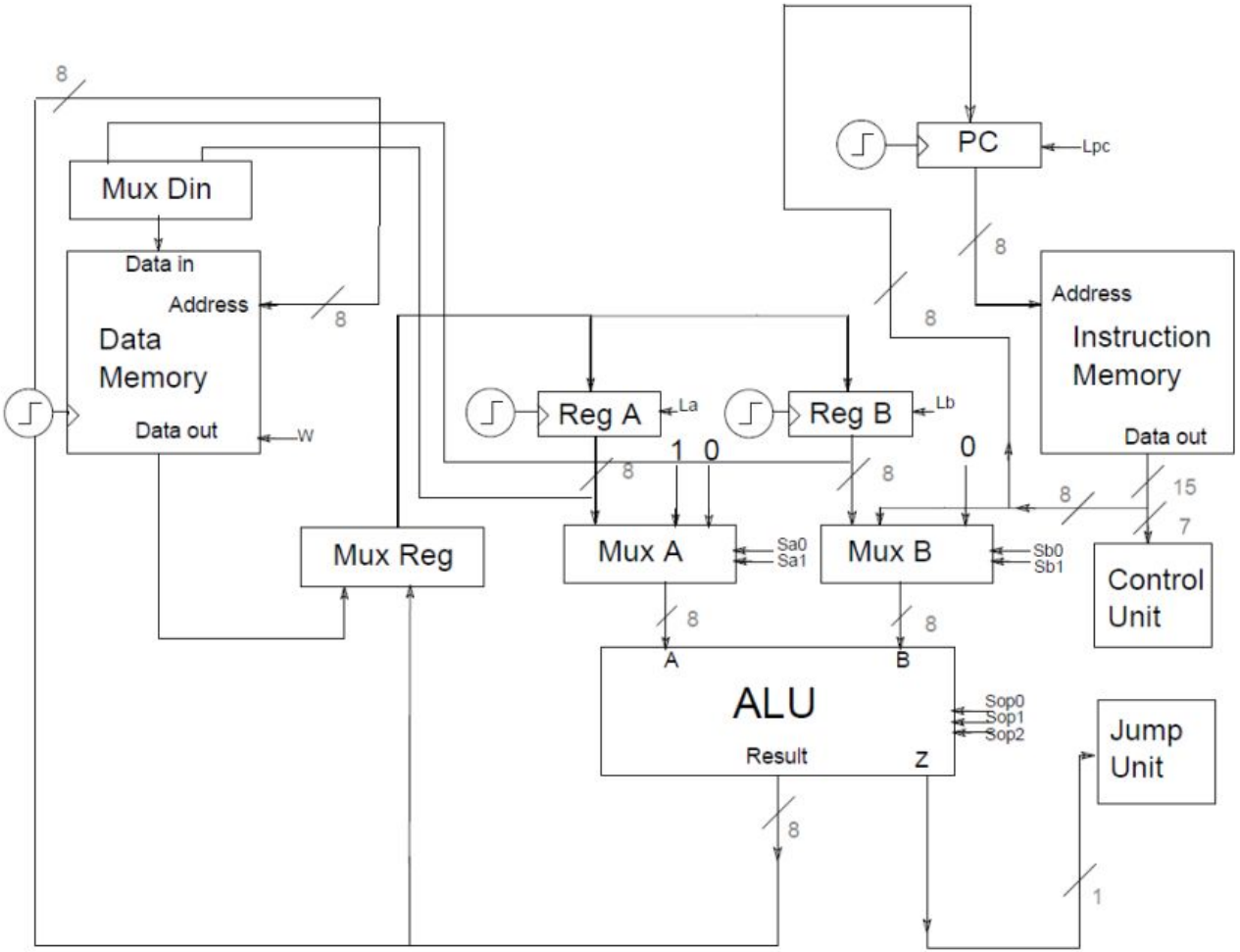
2-Decodificar instr. en unidad de control (**Decode**)

3-Ejecutar en ALU (**Execute**)

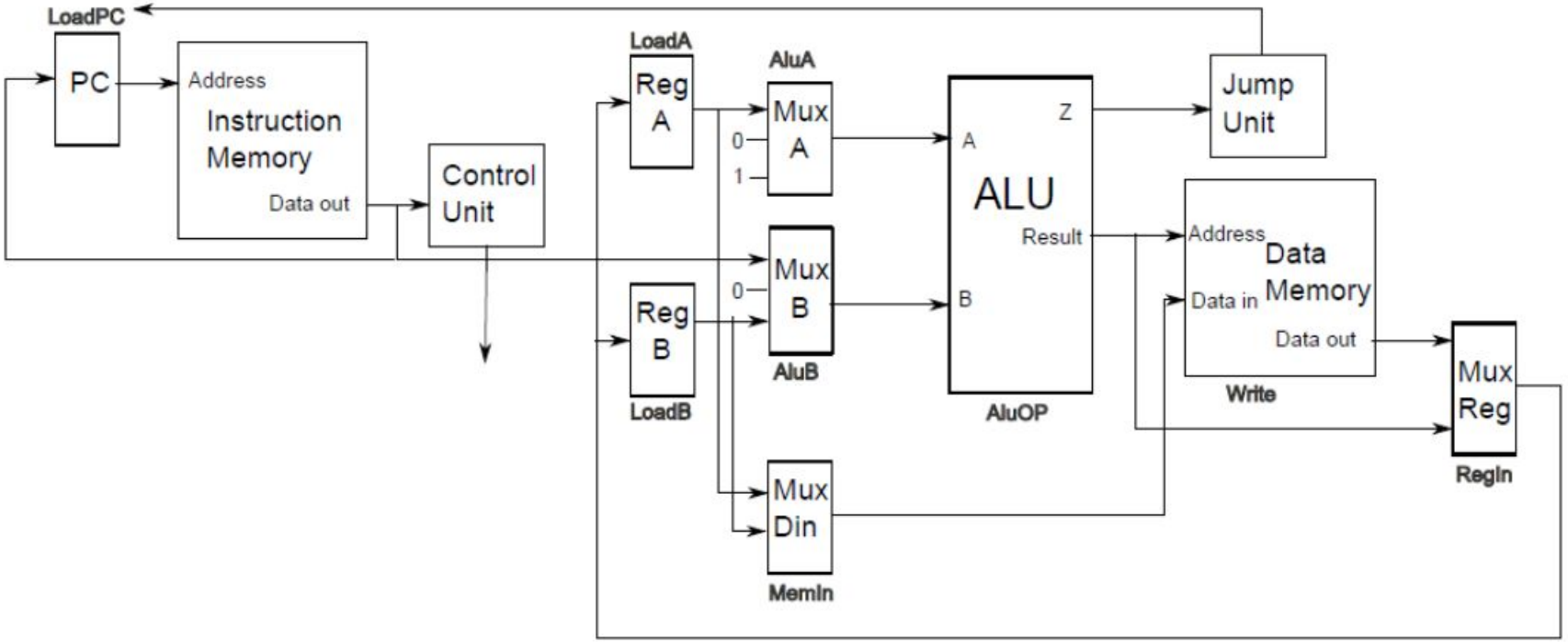
4- Acceder a memoria (**Mem**)

5- Escribir resultado en registro (**Write Back**)

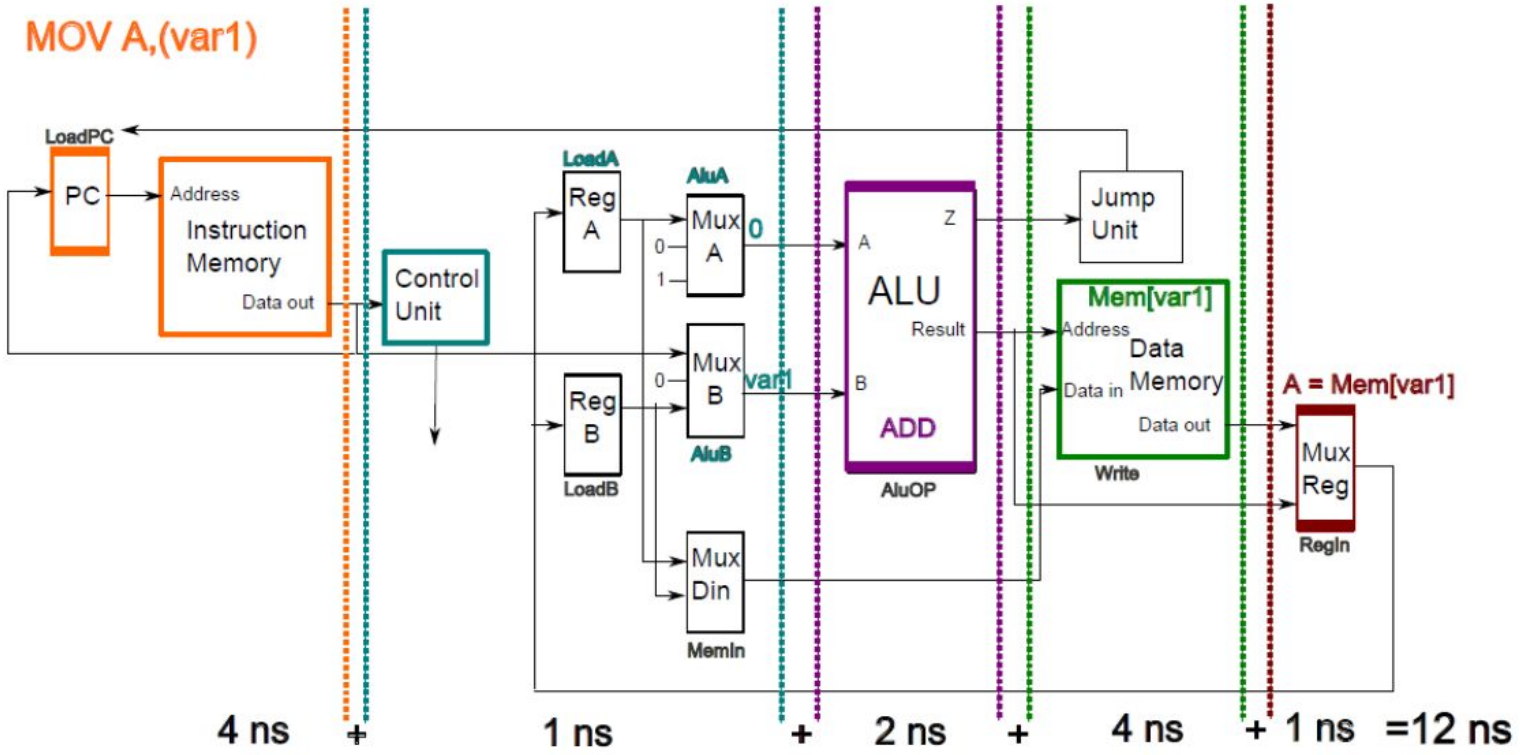
Nuevo Computador



Nuevo Computador Re-organizado

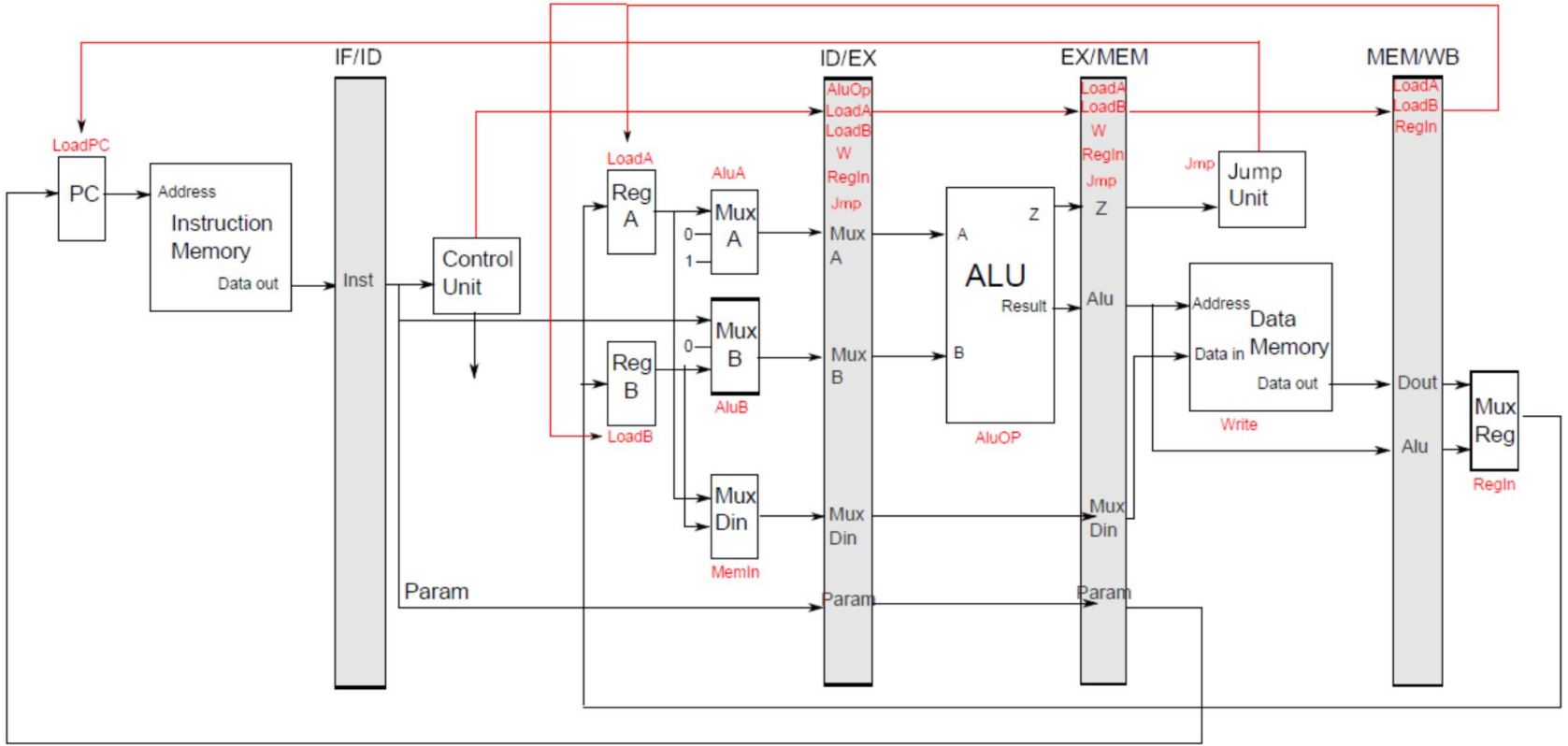


Nuevo Computador Re-organizado: Tiempo MOV A,(var)

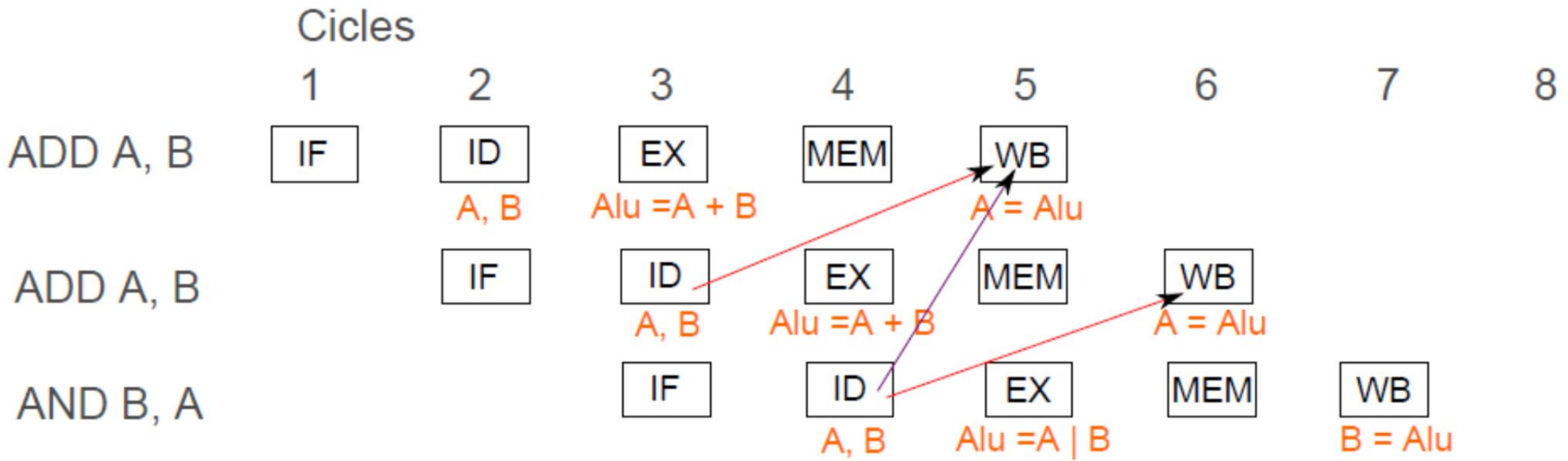


¿Dudas?

Computador básico con pipeline



Computador básico con pipeline: ¿Consistencia?

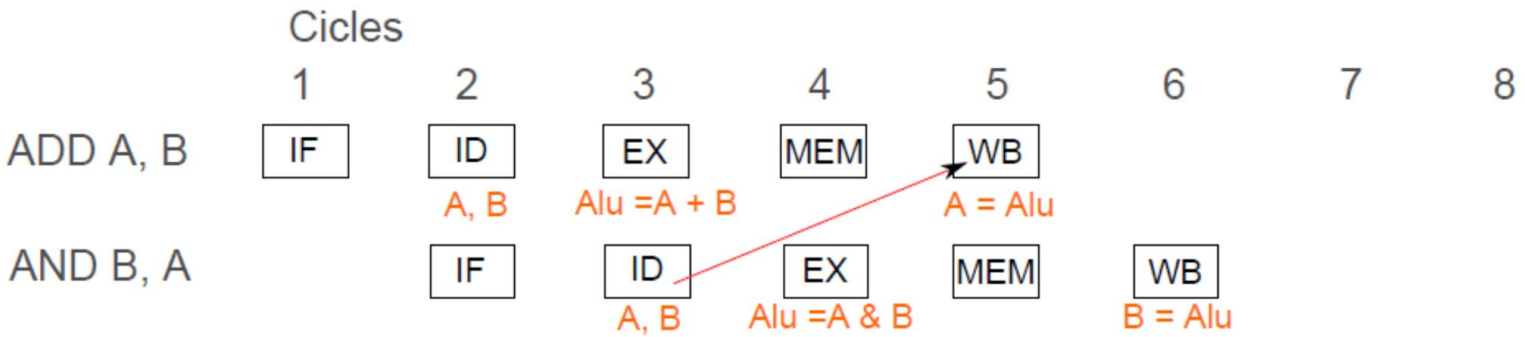


Computador básico con pipeline: Hazards

- Al implementar un pipeline, pueden existir problemas por la dependencia entre instrucciones. Estos se conocen como hazards y existen tres tipos:
  - Hazards de datos
  - Hazards de control
  - Hazards estructurales

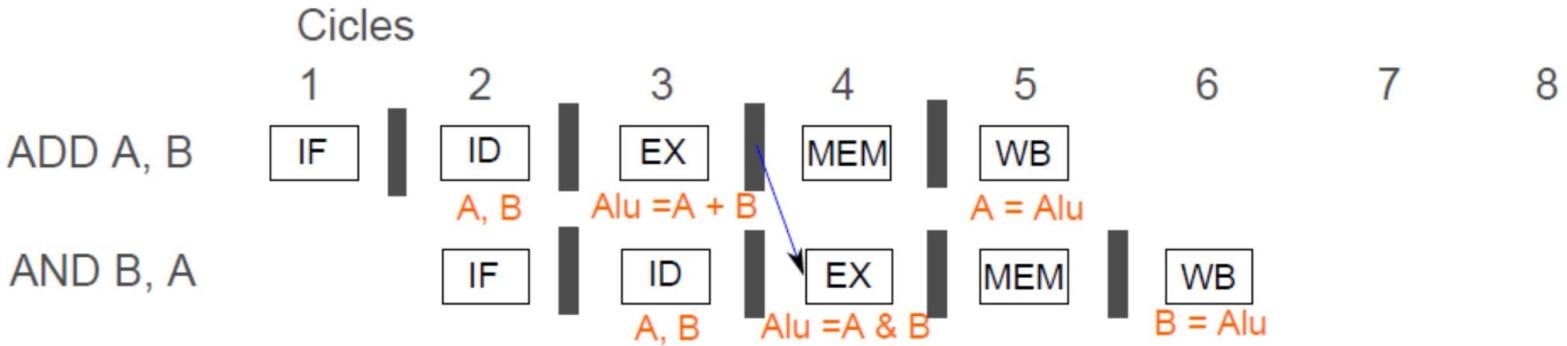
Computador básico con pipeline: Hazards de datos

- Estos ocurren cuando existen dependencias de datos entre instrucciones



Computador básico con pipeline: Hazards de datos

- Podemos solucionarlo con algo llamado **forwarding**



Computador básico con pipeline: Forwarding Unit 1

- Agregaremos una nueva unidad llamada **Forwarding Unit 1** que detectará el caso anterior de la siguiente forma

  if (EX/MEM.LoadA == 1 and ID/EX.AluA == A)

  or (EX/MEM.LoadB == 1 and ID/EX.AluB == B)

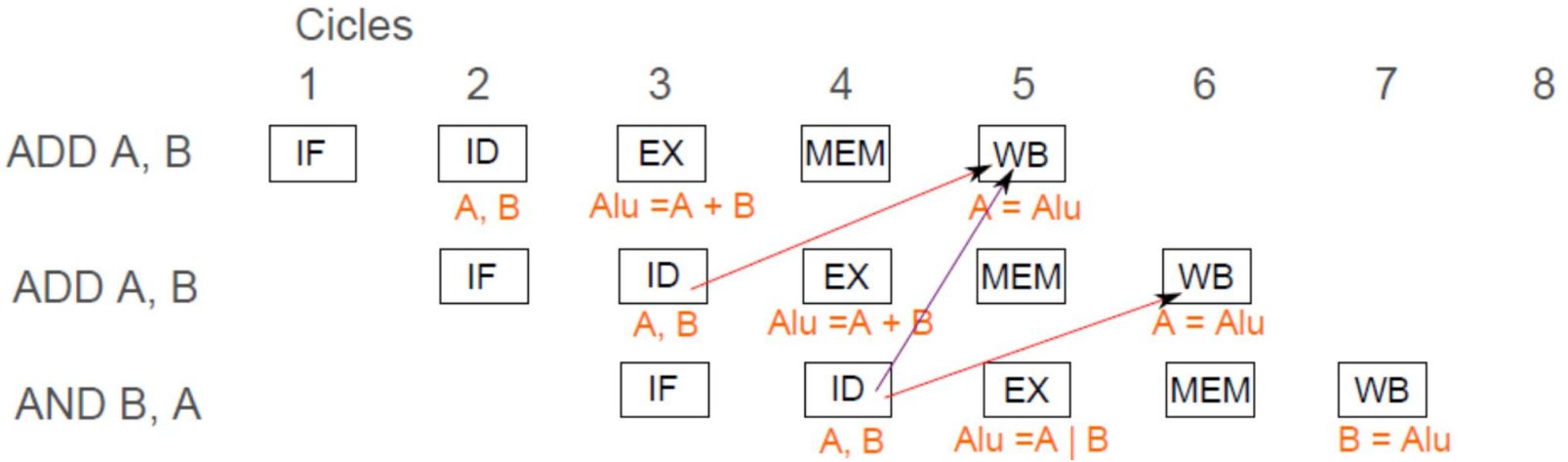
- Forwarding de EX/MEM.Alu a etapa EX según el registro.

  if (MEM/WB.LoadA == 1 and ID/EX.AluA == A and EX/MEM.LoadA == 0)

  or (MEM/WB.LoadB == 1 and ID/EX.AluB == B and EX/MEM.LoadB == 0)

- Forwarding de MEM/WB.Alu a etapa EX según el registro.

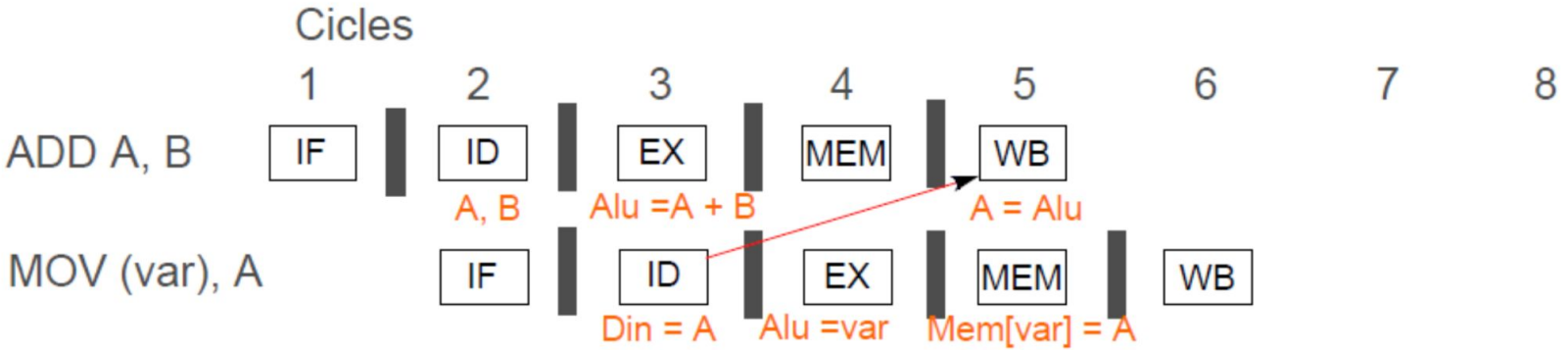
Computador básico con pipeline: Hazards de datos



¿Dudas?

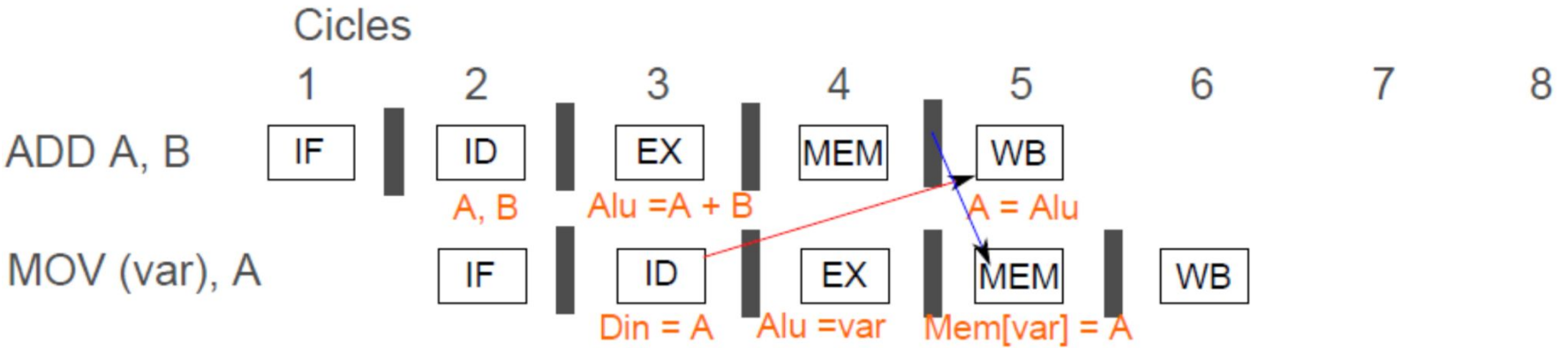
Computador básico con pipeline: Hazards de datos

- Estos ocurren cuando existen más dependencias de datos entre instrucciones



Computador básico con pipeline: Hazards de datos

- Podemos solucionarlo también con **forwarding !**



Computador básico con pipeline: Forwarding Unit 2

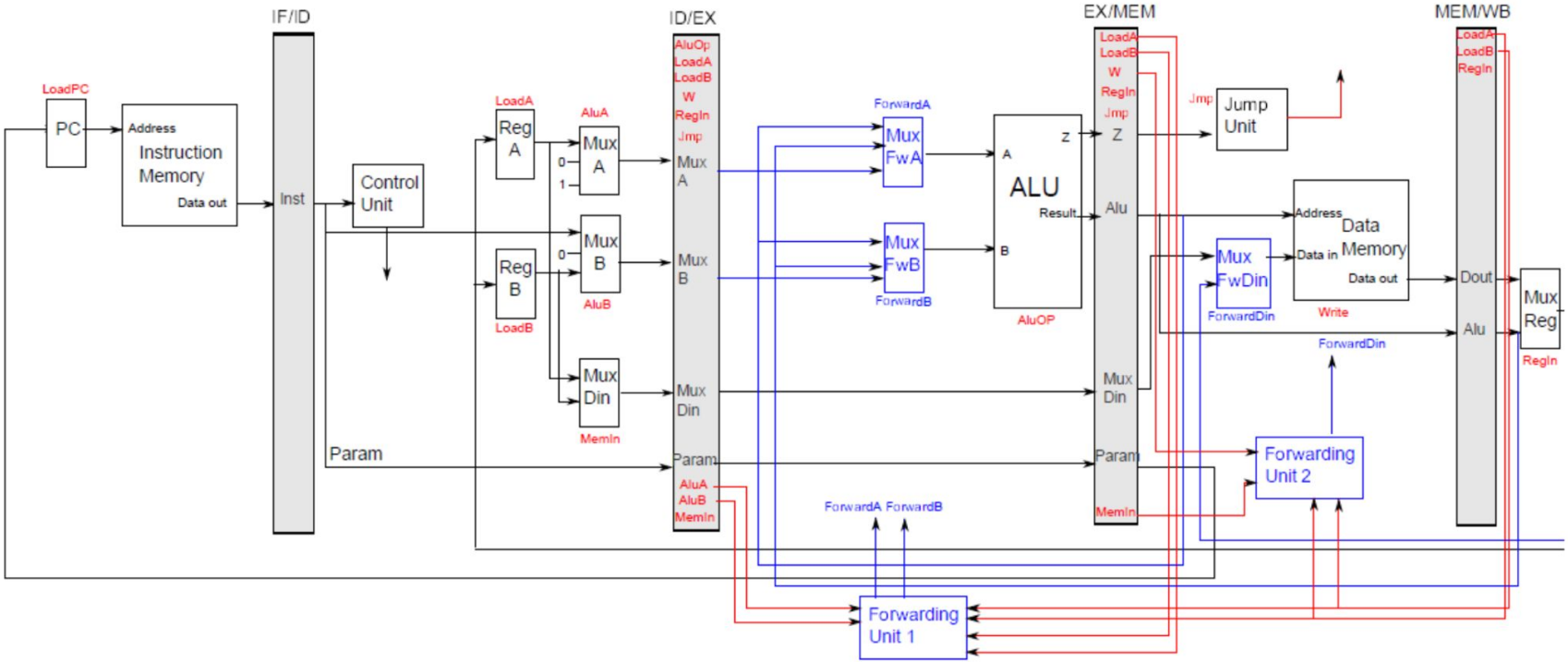
- Agregaremos una nueva unidad llamada **Forwarding Unit 2** que detectará el caso anterior de la siguiente forma:

  if (MEM/WB.LoadA == 1 and EX/MEM.MemIn == A and EX/MEM.Write == 1)

  or (MEM/WB.LoadB == 1 and EX/MEM.MemIn == B and EX/MEM.Write == 1)

- Forwarding de WB.Mux Reg a etapa MEM según el registro

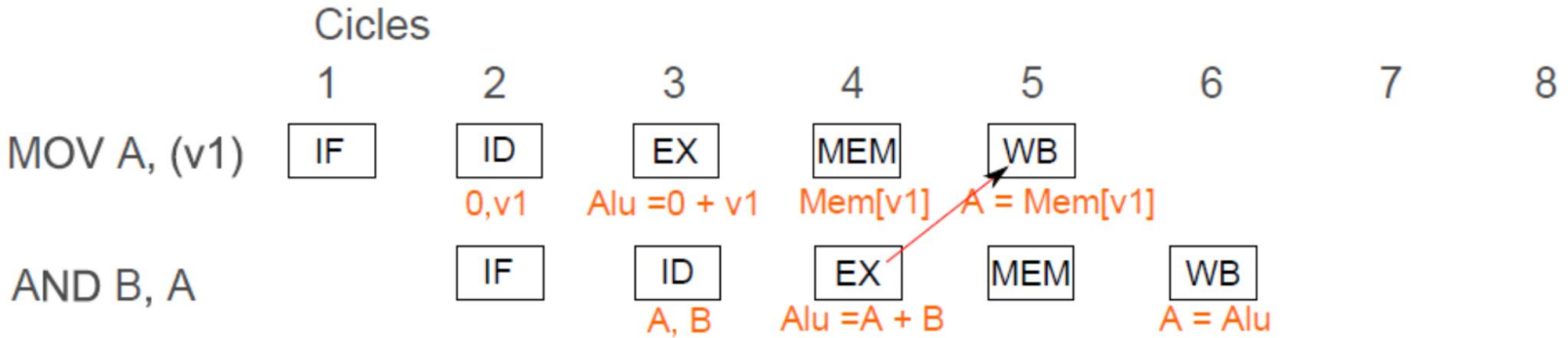
Computador básico con pipeline: Hazards de datos



¿Dudas?

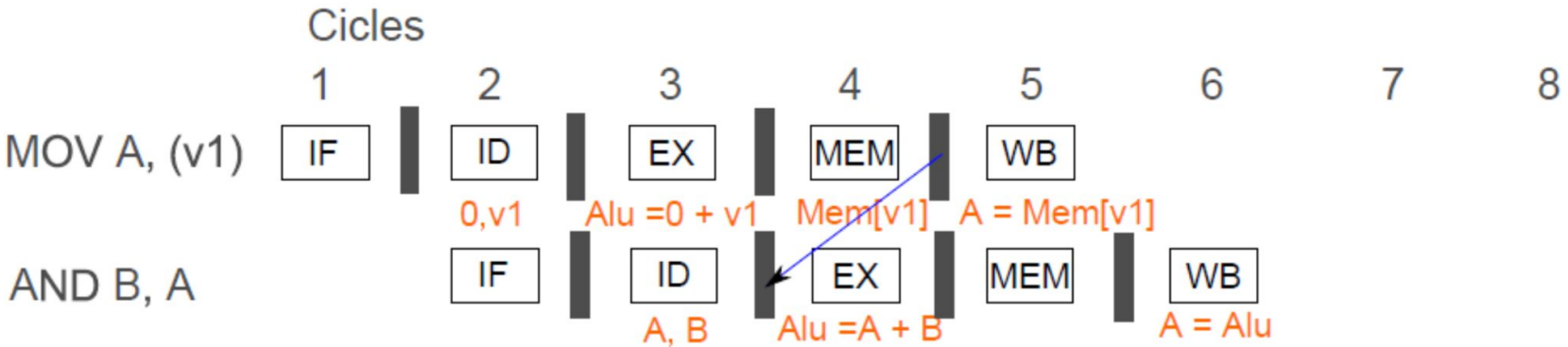
Computador básico con pipeline: Hazards de datos

- ¿Qué hacemos en este caso ?



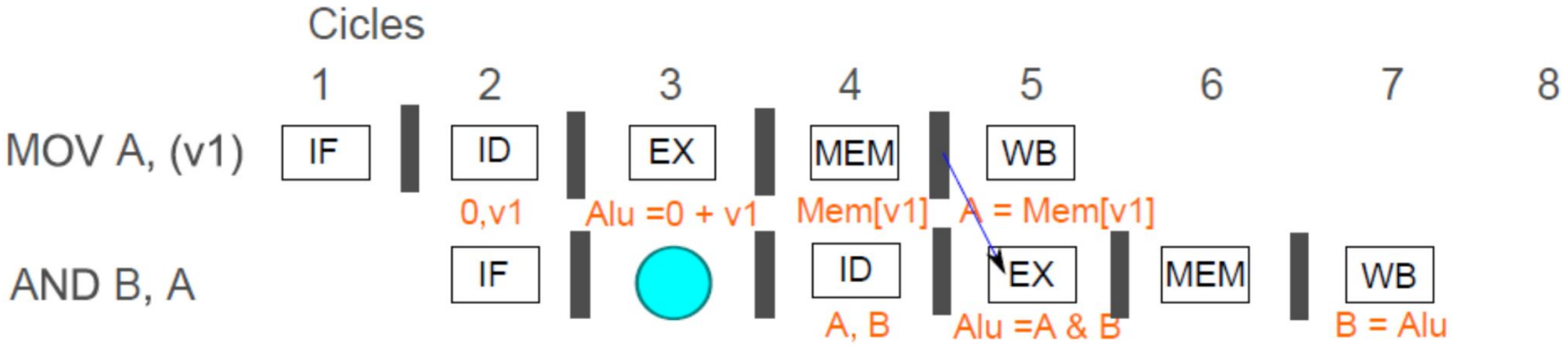
Computador básico con pipeline: Hazards de datos

- **Forwarding** llega muy tarde, **no** es una solución factible

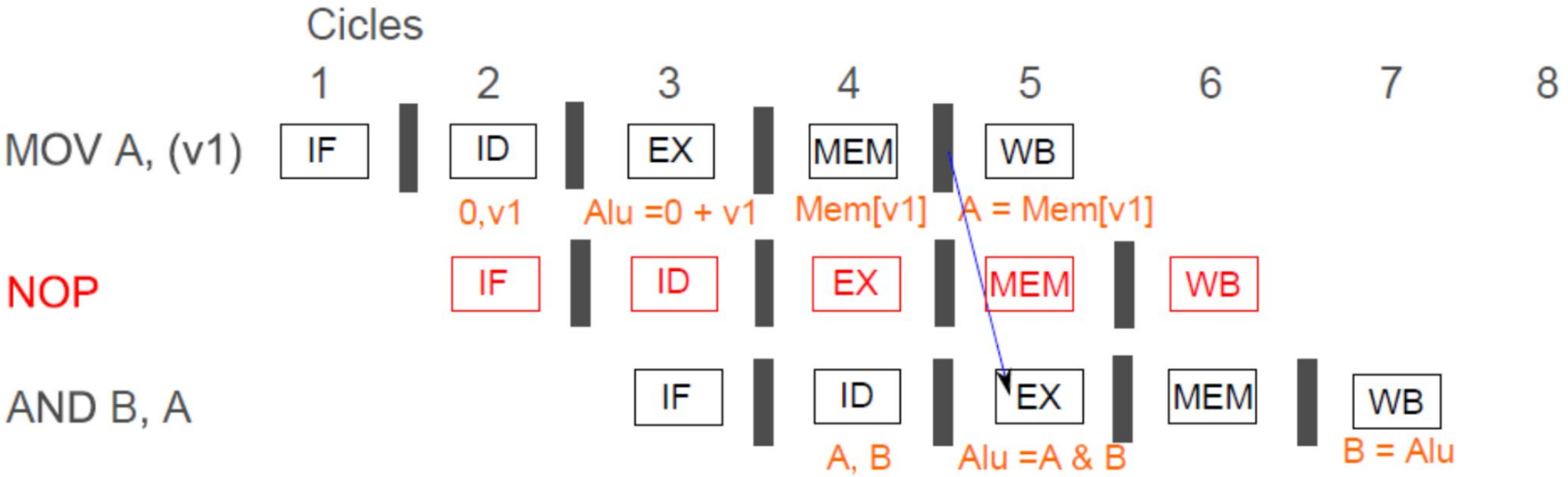


Computador básico con pipeline: Hazards de datos

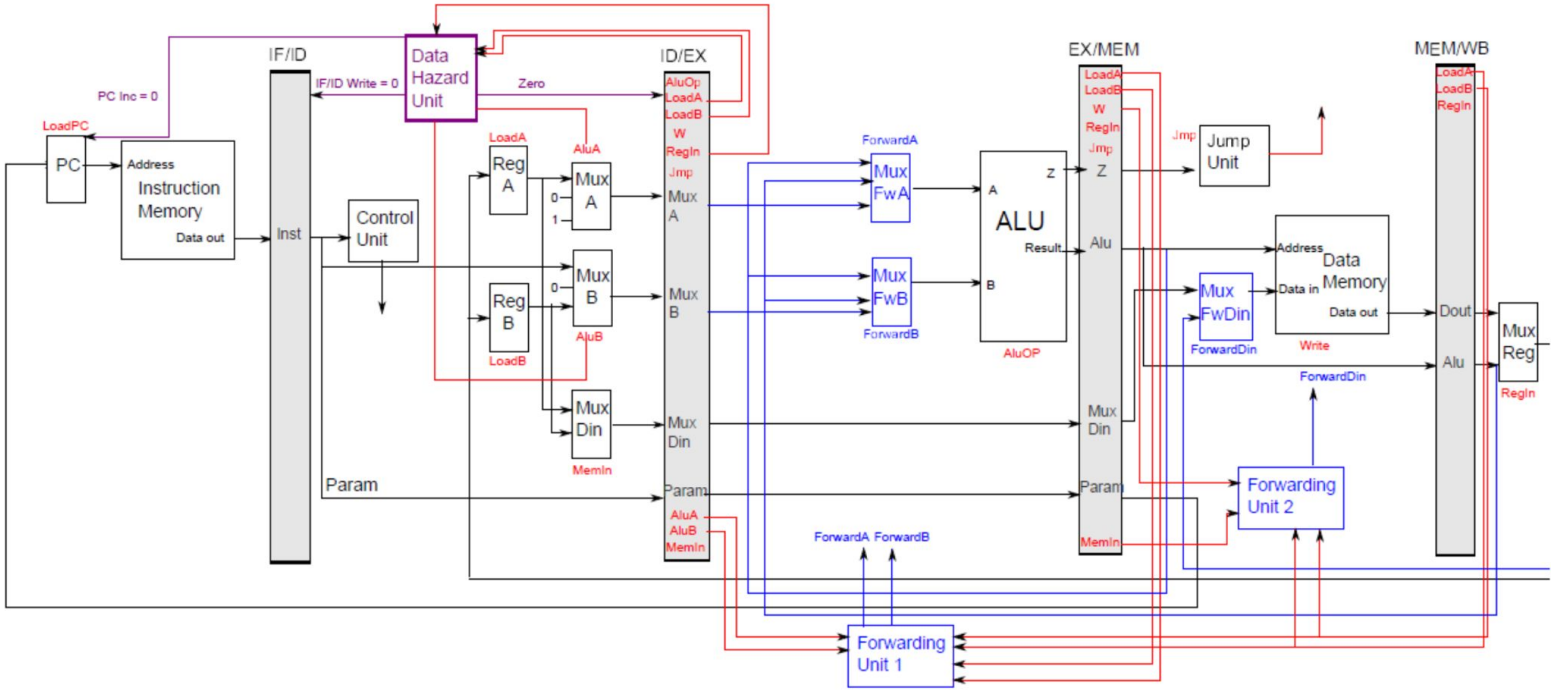
- Necesitamos que el procesador espere, lo que se conoce como **stalling**



Computador básico con pipeline: Stalling por software



Computador básico con pipeline: Stalling por hardware



Computador básico con pipeline: Data Hazard Unit

- Agregaremos una nueva unidad llamada **Data Hazard Unit** que detectará el caso anterior de la siguiente forma:
- **if (ID/EX.LoadA == 1 and ID/EX.RegIn == Dout and ID.AluA == A)**

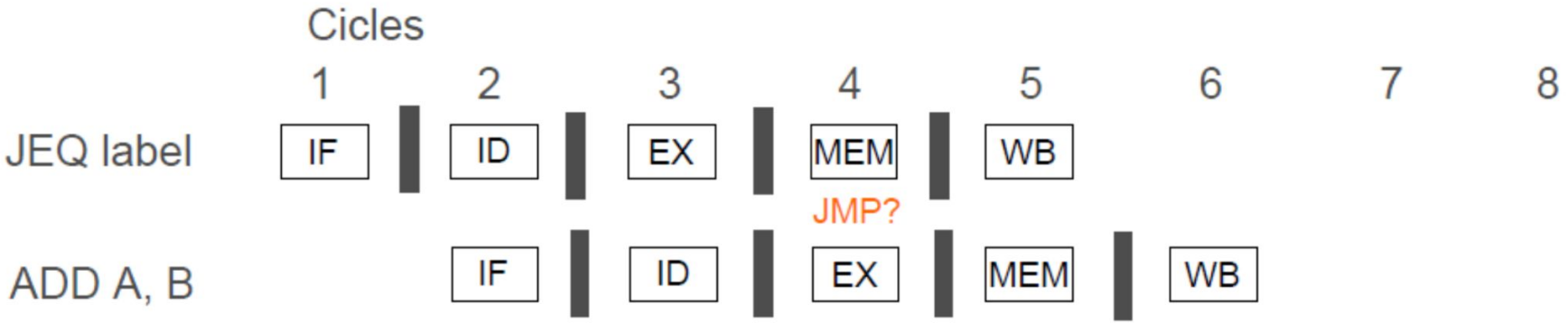
  or (ID/EX.LoadB == 1 and ID/EX.RegIn == Dout and ID.AluB == B)

- **Insertar Burbuja**

¿Dudas?

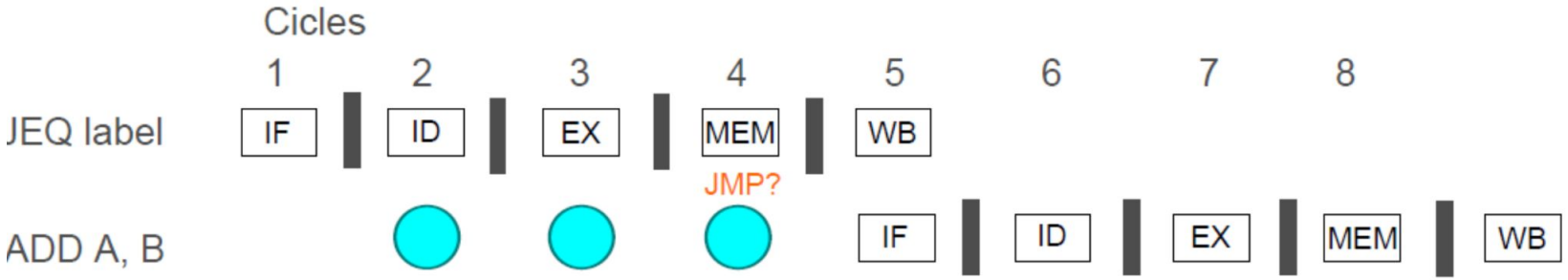
Computador básico con pipeline: Hazards de control

- Estos ocurren cuando existen saltos



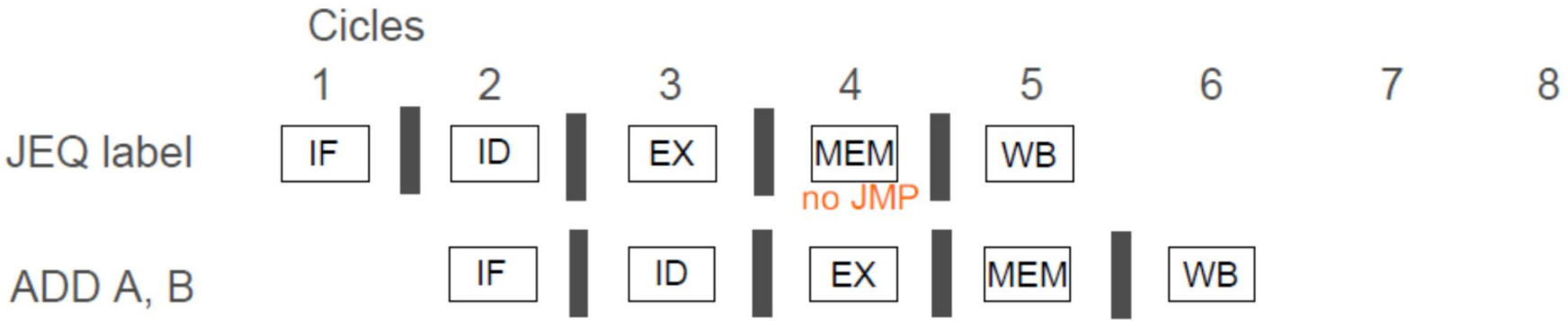
Computador básico con pipeline: Hazards de control

- Solución simple stalling



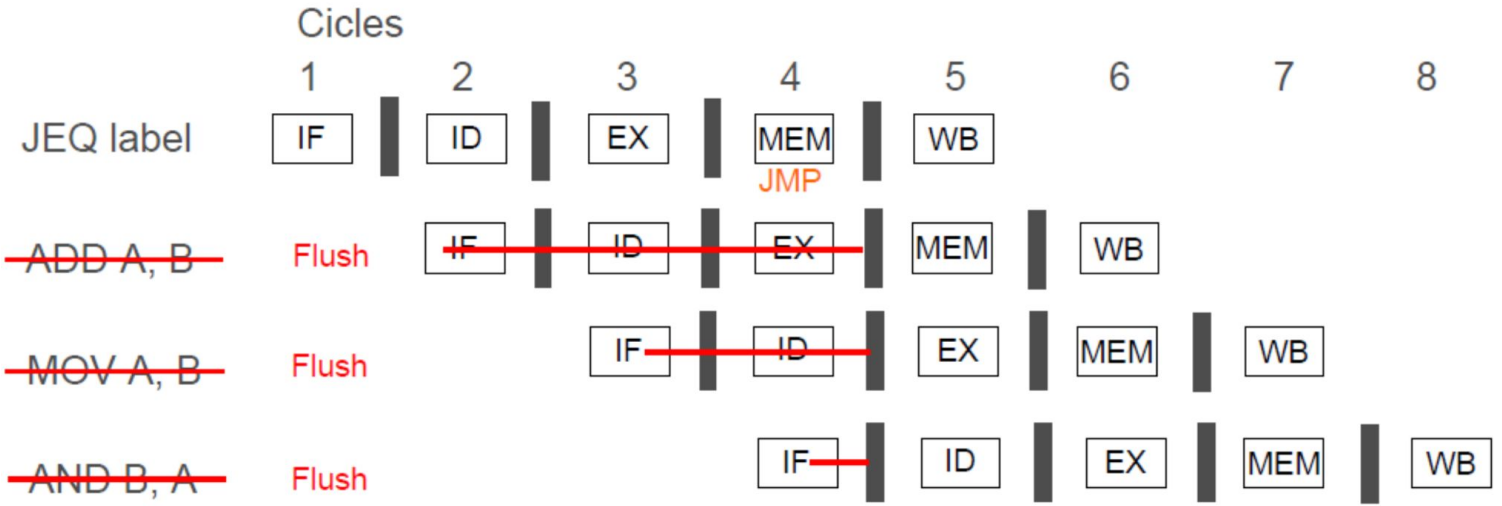
Computador básico con pipeline: Hazards de control

- En la práctica usamos predicción de salto

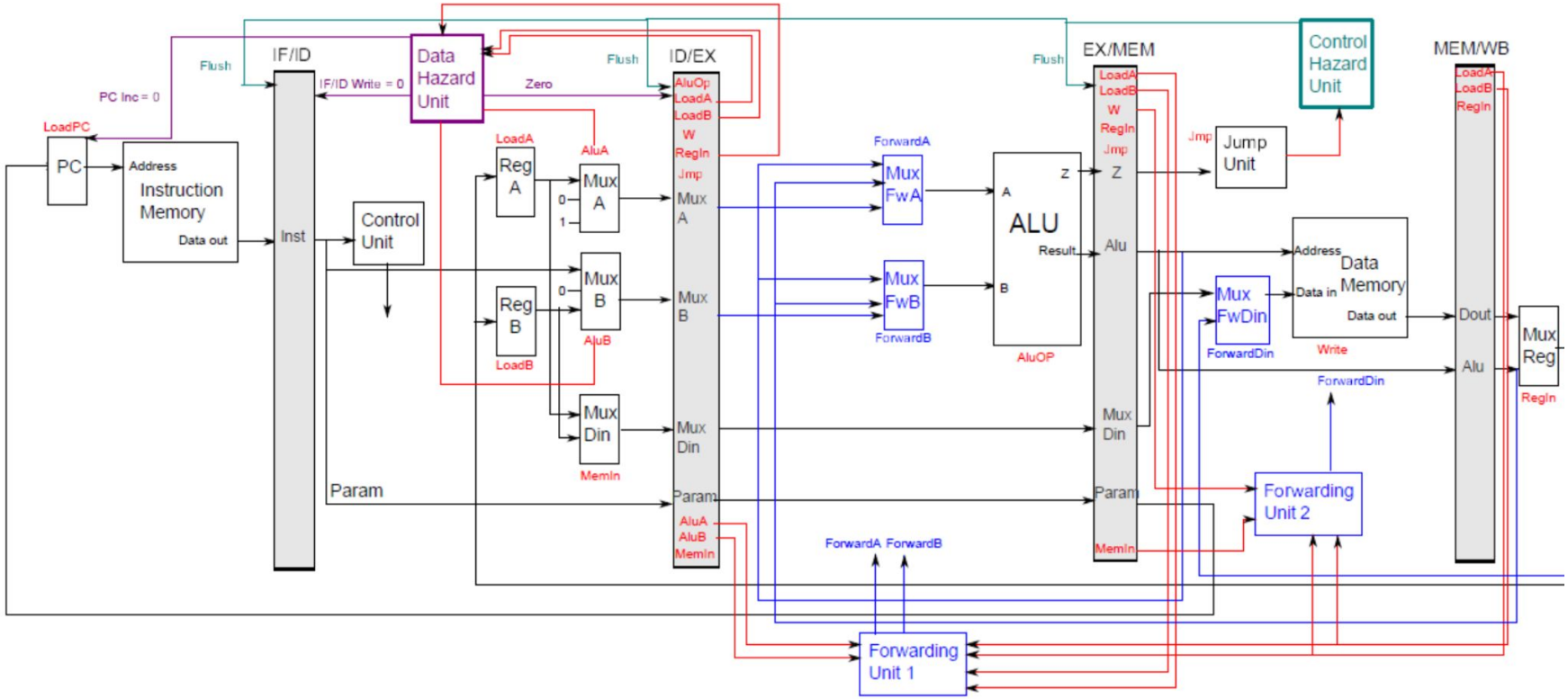


Computador básico con pipeline: Hazards de control

- En la práctica usamos predicción de salto



Computador básico con pipeline: Hazards de control



¿Dudas?

Computador básico con pipeline: Hazards estructurales

- Estos ocurren cuando ocurren cuando dos etapas necesitan usar la misma unidad funcional
- **Ejemplo:**

  Etapas IF y MEM en una arquitectura Von Neumann. Como en este tipo de arquitectura la memoria de datos e instrucciones se encuentran en una única unidad, estas dos etapas requieren acceder a ella en un mismo instante de tiempo.

- ¿Cómo se podría resolver este caso?

Computador básico con pipeline: Hazards estructurales

- **Solución 1:** Agregar stalling de forma intercalada a cada etapa para que haya un solo acceso por ciclo.

  Problema: Pérdida de 2 ciclos por instrucción, siempre.

- **Solución 2:** Agregar otra unidad de memoria.

  Problema: Aumento significativo de costo, cambio estructural de la arquitectura.

Computador básico con pipeline: Hazards estructurales

Solución real:

Hacer uso de una **caché split.** ¿Por qué esto resuelve el problema?

- Las **caché split** tienen la particularidad de permitir dos accesos de memoria concurrentes: uno para la sección de datos y otra para la sección de instrucciones
- Así, el **pipeline** trata los accesos de memoria como una arquitectura **Harvard**, aunque sea **Von Neumann**

¿Dudas?

Arquitectura de Computadores: Mejoras y extensiones

En lo que hemos visto hasta ahora

- Una máquina programable que ejecuta programas
- Interacción con dispositivos de entrada y salida

Lo que nos queda ver **mejoras:**

- Mejora los accesos a memoria ✓
- Mejora en lo que respecta a la ejecución de más de un programa ✓
- Mejora en lo que respecta a la ejecución de instrucciones en la CPU ✓

Clase 23 - Paralelismo Parte 2

IIC2343 - Arquitectura de Computadores

Profesor:
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl