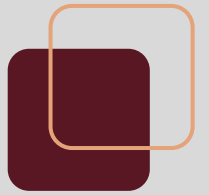
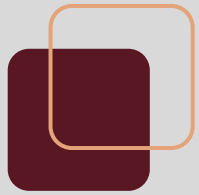


Aspectos avanzados de arquitectura de computadoras
Pipeline

Facultad de Ingeniería - Universidad de la República

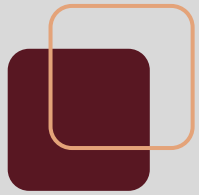
Curso 2017

Objetivo



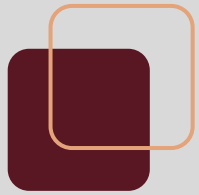
- Mejorar el rendimiento
  - ¿Incrementar frecuencia de reloj?
  - ¿Ancho de los registros?
  - ¿Ancho del bus de datos?
  - ¿Ancho del bus de direcciones?

Pipeline



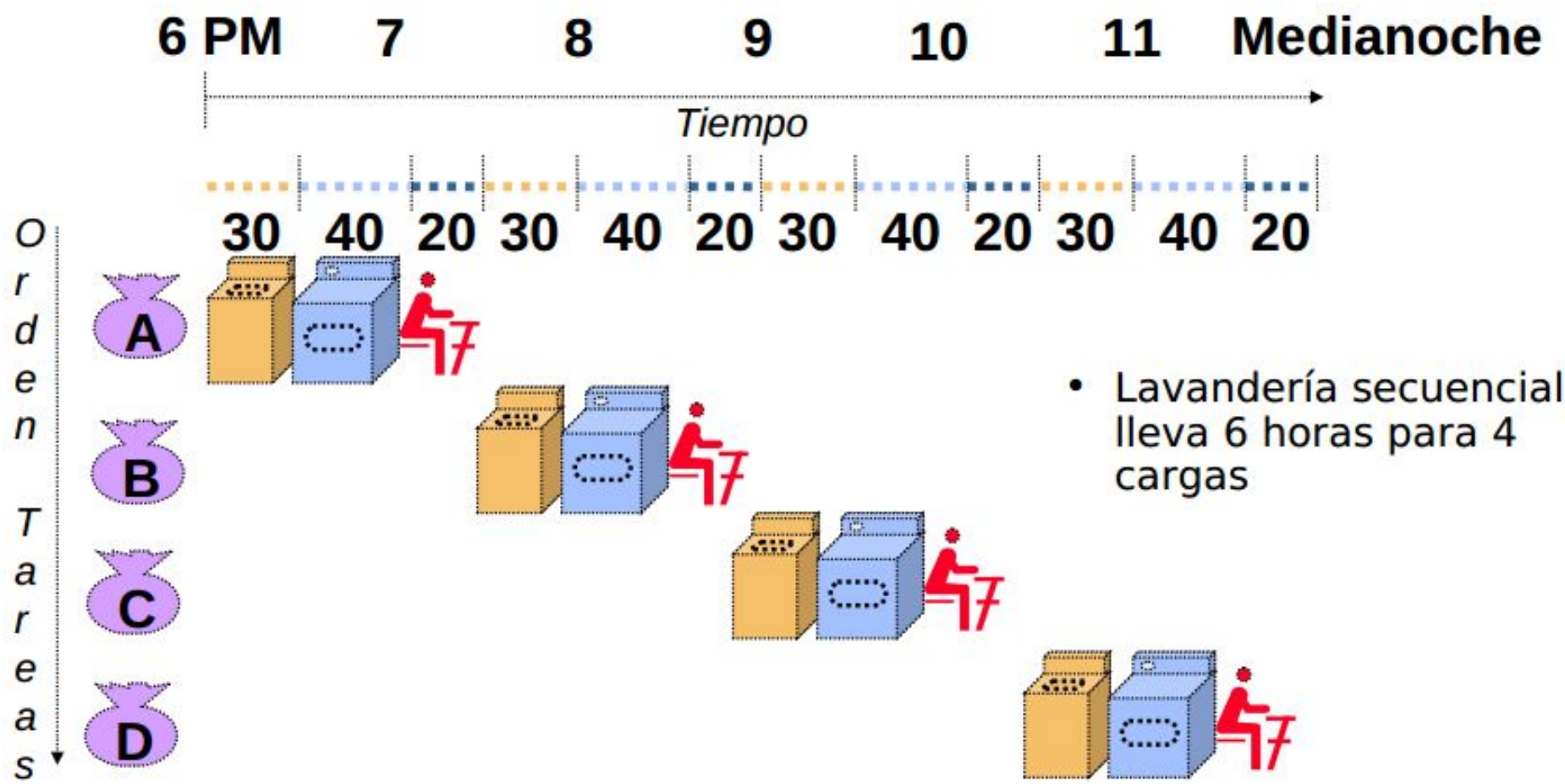
- Concepto:
  - Para una tarea ‘larga’, que se debe repetir, y que se puede dividir en etapas: paralelizar varias ejecuciones de la tarea, solapando las etapas que puedan realizarse al mismo tiempo.
  - Responde a la idea de ‘línea de producción’ utilizada en las fábricas!

Ejemplo Lavandería (1/4)



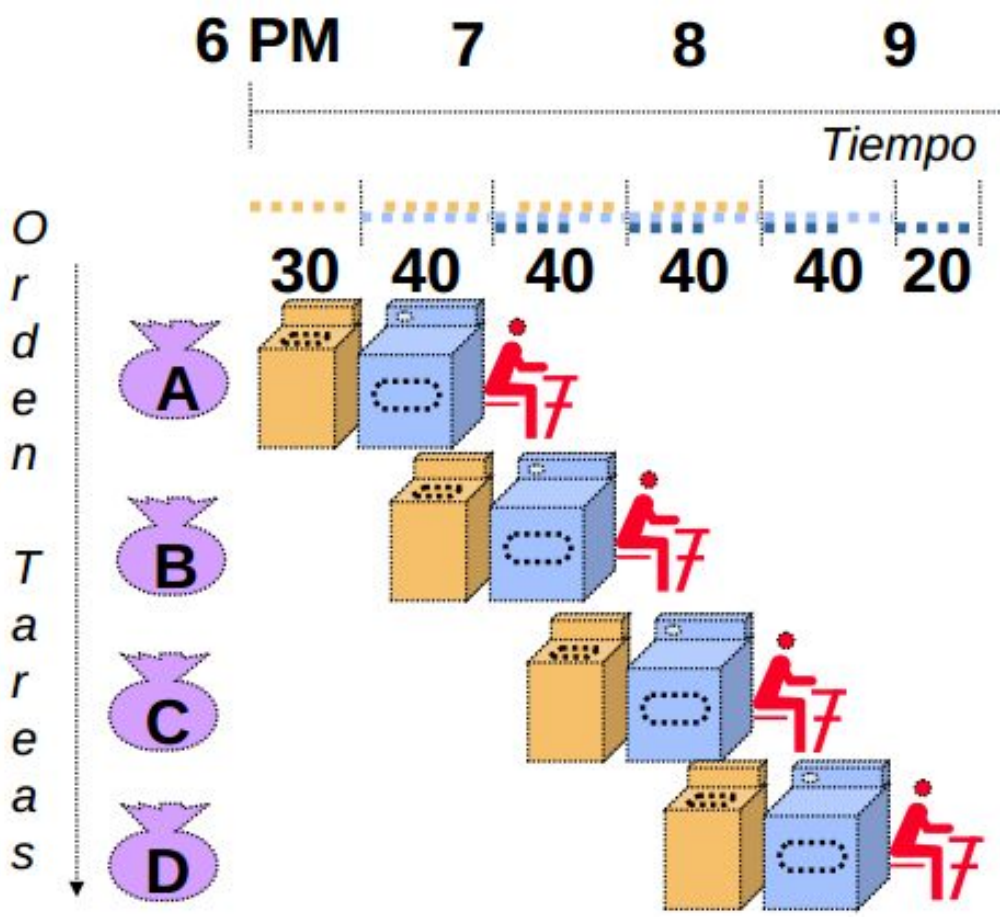
- Supongamos el proceso de lavado de ropa en una lavandería con los siguientes pasos:
  - Lavado: 30 minutos
  - Secado: 40 minutos
  - Doblado de ropa: 20 minutos

Ejemplo Lavandería (2/4)
Lavandería Secuencial



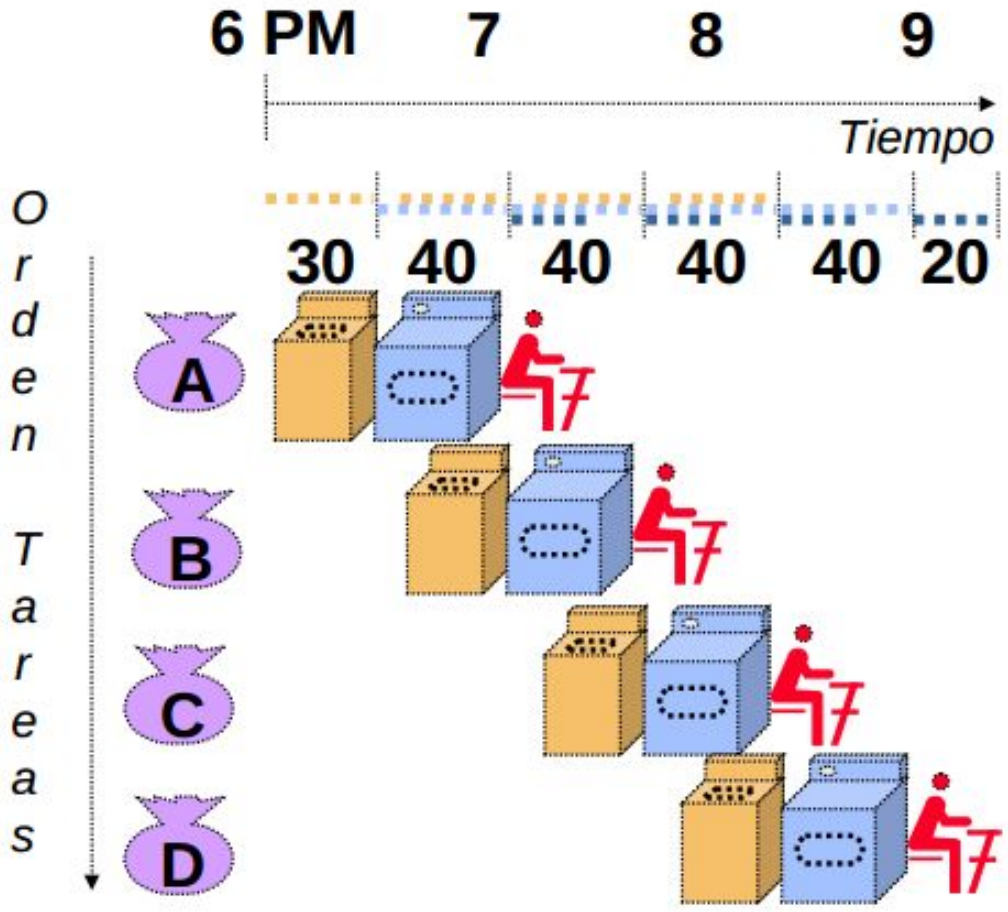
- Lavandería secuencial lleva 6 horas para 4 cargas

Ejemplo Lavandería (3/4)
Lavandería en Pipeline



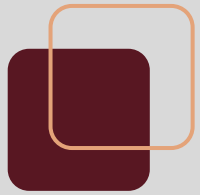
- Lavandería en Pipeline lleva 3.5 horas para 4 cargas!

Ejemplo Lavandería (4/4)
Lavandería en Pipeline

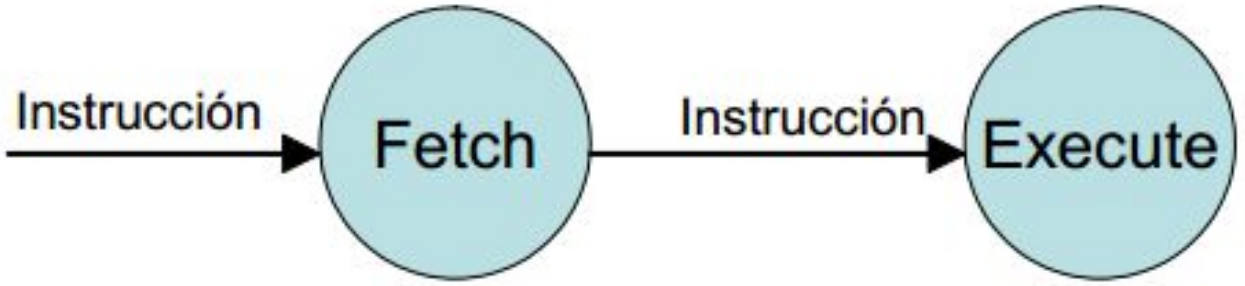


- Pipelining no mejora la latencia de cada tarea, sino el throughput de toda la carga de trabajo
- Velocidad del Pipeline limitada por el paso más lento
- Aceleración potencial = Cantidad de pasos del pipeline
- Tiempo para "llenar" y "vaciar" el pipeline reduce la aceleración

Implementación: Pipeline de dos etapas (1/2)

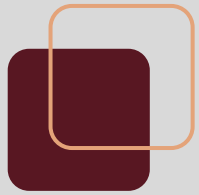


- Supongamos un ciclo de instrucción de dos etapas: Fetch y Execute



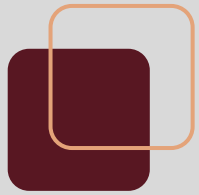
- ¿Cómo aplica la idea de pipeline? ¿Es paralelizable la ejecución?

Implementación: Pipeline de dos etapas (2/2)



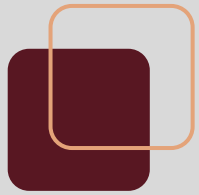
- Para poder aplicar la técnica de pipeline, las etapas deben acceder a recursos diferentes:
  - ¿Comparten recursos?
- Este pipeline mejora la *tasa de ejecución de instrucciones* (throughput) pero no lo duplica. ¿Por qué?

Pipeline MIPS (1/5)



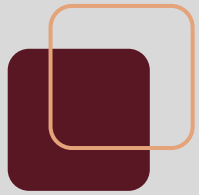
- En el pipeline anterior, las etapas duraban como mínimo la mitad del ciclo completo de instrucción.
- A continuación veremos el Pipeline de un procesador que implementa la arquitectura MIPS 32
- MIPS 32 es una arquitectura de 32 bits, RISC y de tres vías.

Pipeline MIPS (2/5)



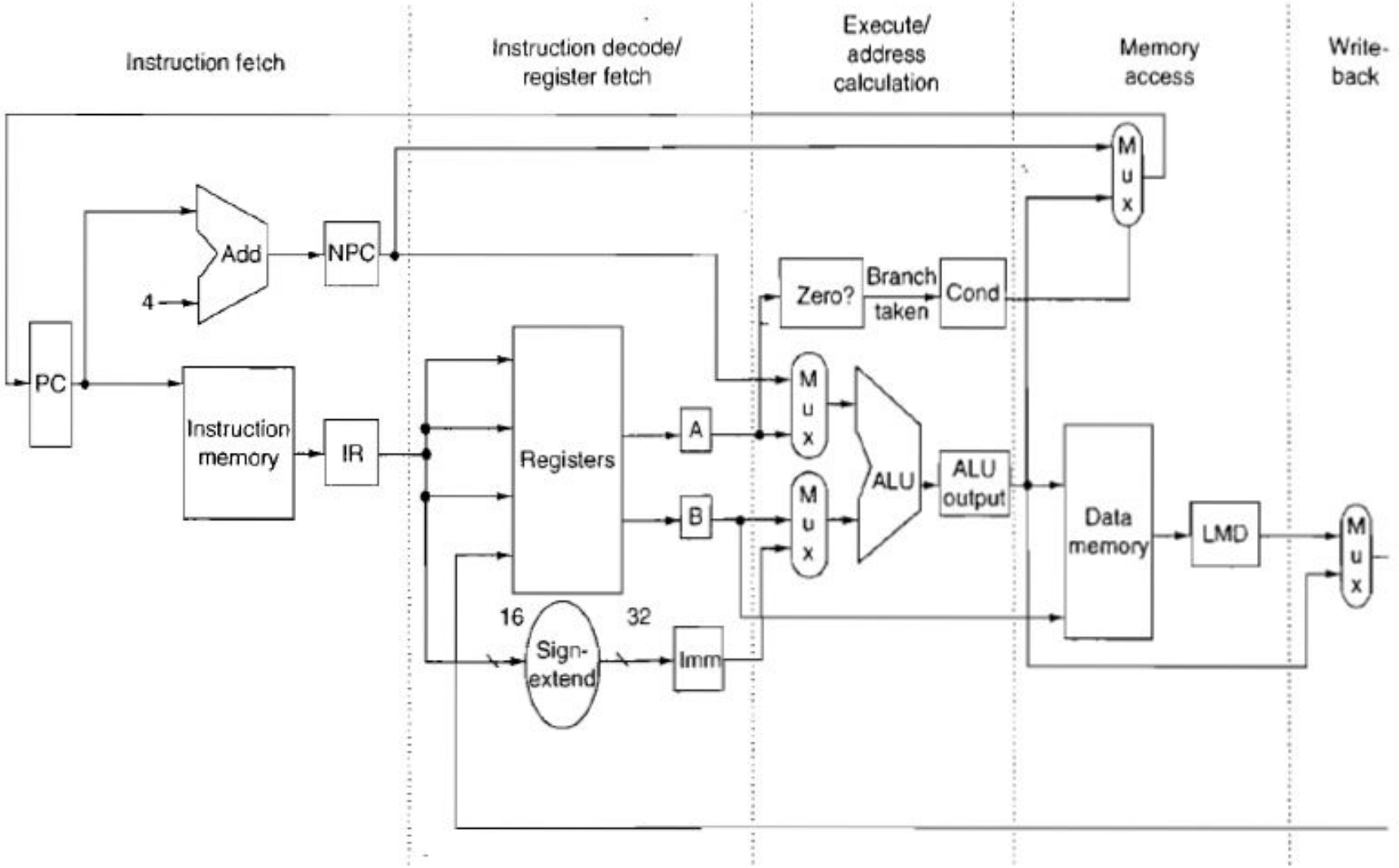
- Ciclo de instrucción:
  - Fetch: la instrucción es leída de memoria y se incrementa el PC (Program Counter)
  - Decode / Register Fetch: la instrucción es decodificada y las correspondientes señales de control activadas. Los operandos en registros o inmediatos son apropiadamente movidos a registros de trabajo de la ALU.

Pipeline MIPS (3/5)

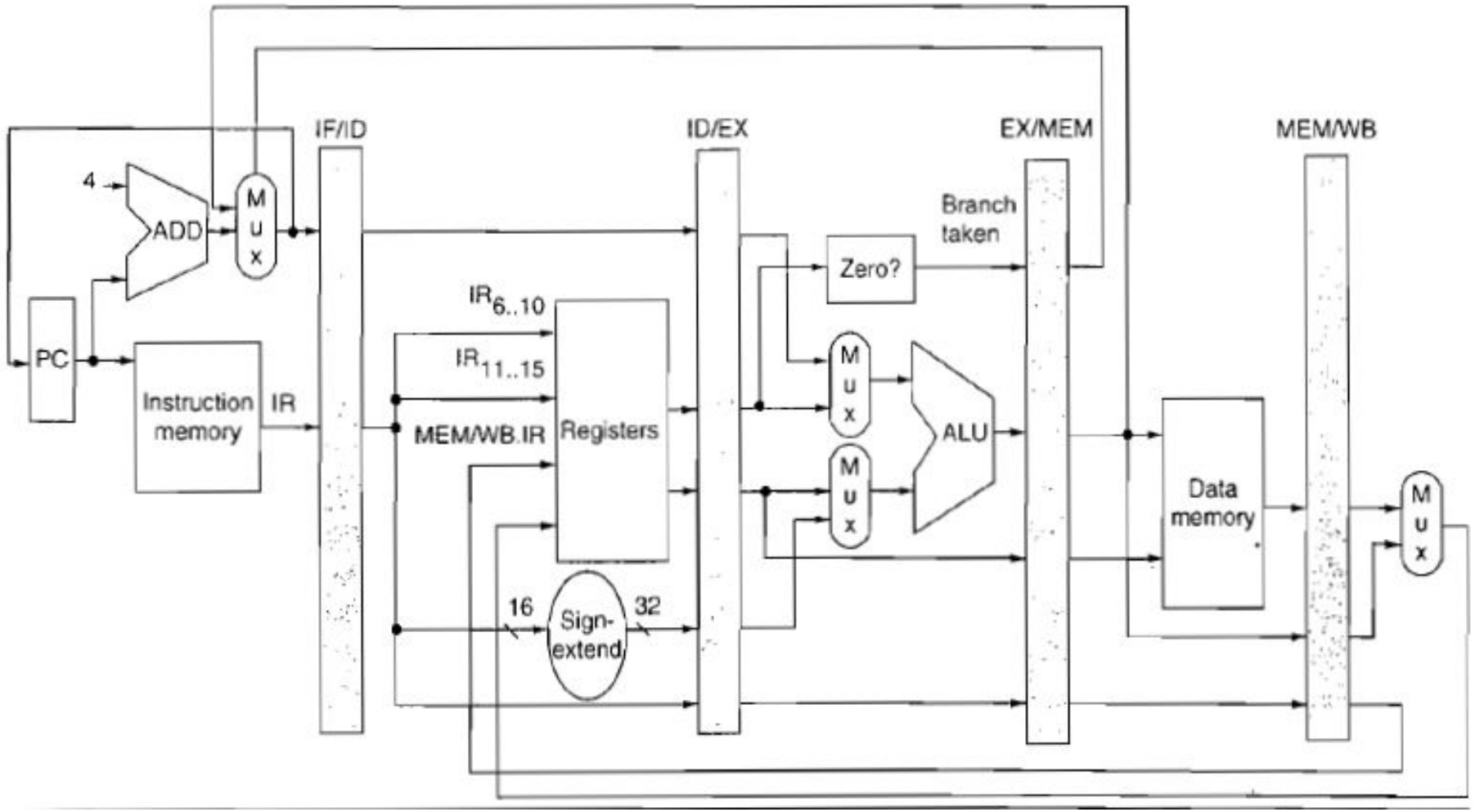


- Ciclo de instrucción:
  - Execute: se ejecuta la operación. Si la instrucción es de memoria (load/store) se calcula la dirección de memoria a acceder. En caso de salto condicional, la condición es verificada y la dirección a saltar calculada.
  - Memory Read/Write: si la operación es de memoria se lee o escribe la misma.
  - Write Back: se transfiere el resultado de la ejecución desde el registro auxiliar hacia el registro destino.

Pipeline MIPS (4/5)
CPU SIN Pipeline



Pipeline MIPS (5/5)
CPU CON Pipeline

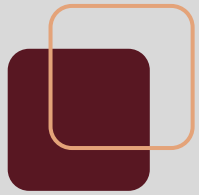


Diagramas de tiempos

- La ejecución del pipeline se puede expresar con un diagrama de tiempo vs instrucciones.

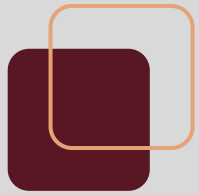
	Clock number								
Instruction number	1	2	3	4	5	6	7	8	9
Instruction *i*	IF	ID	EX	MEM	WB				
Instruction *i* + 1		IF	ID	EX	MEM	WB			
Instruction *i* + 2			IF	ID	EX	MEM	WB		
Instruction *i* + 3				IF	ID	EX	MEM	WB	
Instruction *i* + 4					IF	ID	EX	MEM	WB

Pipeline Hazards (1/13)



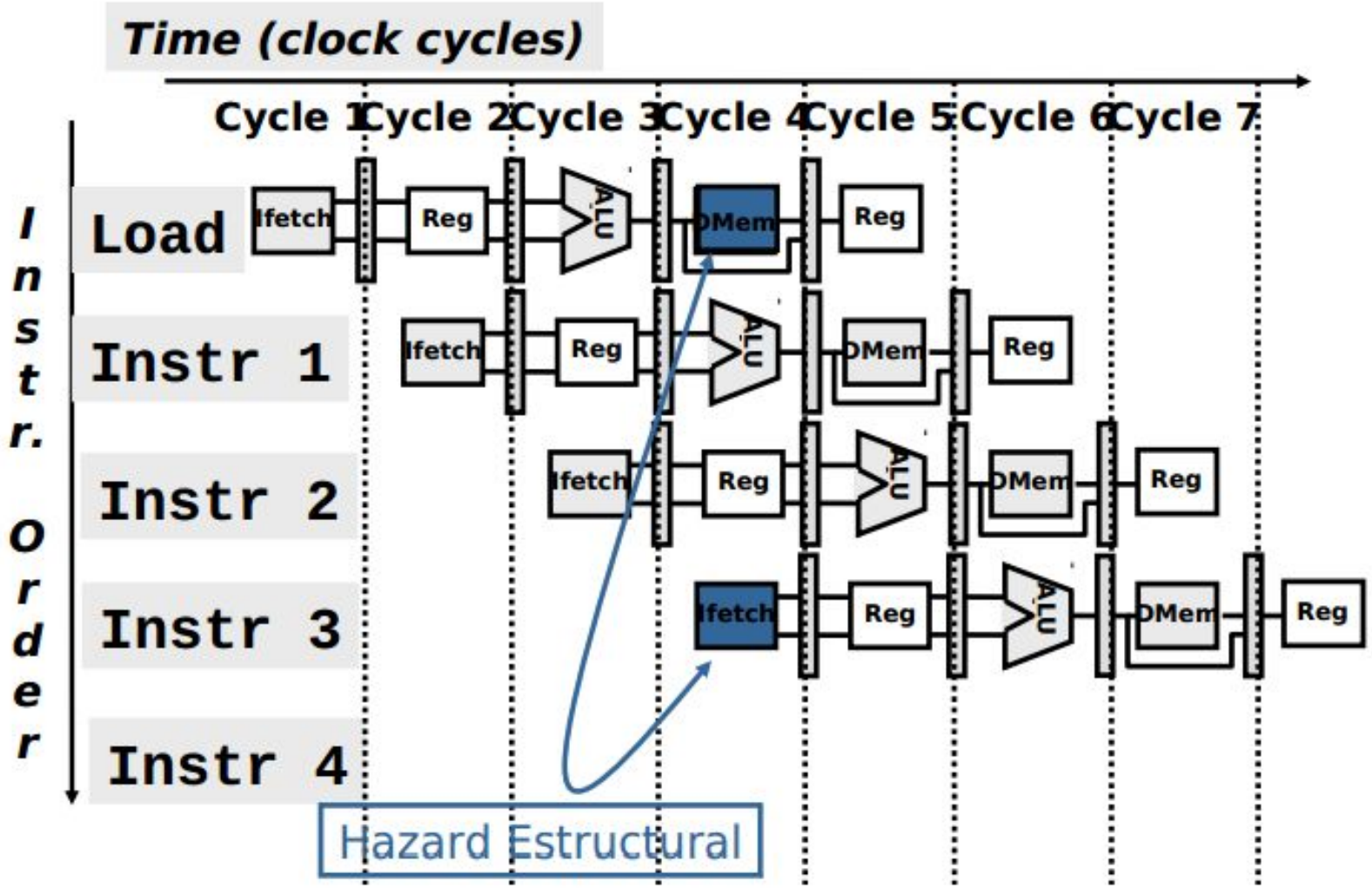
- En el diagrama anterior se asume que todas las etapas pueden progresar de forma ideal, pero como se mostrará a continuación, la aplicación de la técnica de Pipeline introduce complejidades que a menudo impiden la ejecución ideal.
- Cuando se detiene el Pipeline se dice que se está en presencia de un *hazard*.

Pipeline Hazards (2/13)

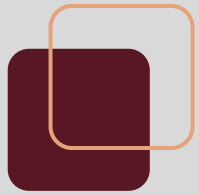


- Tipos de Hazards:
  - Hazards estructurales: se dan cuando hay un conflicto de hardware para alguna combinación de instrucciones.
  - Hazards de datos: se dan cuando por alguna dependencia de datos en las instrucciones y el uso de pipeline, se altera el flujo de datos del programa.
  - Hazards de control: son causados por instrucciones de saltos u otras modificaciones del PC.

Pipeline Hazards (3/13)
Ejemplo Hazard Estructural



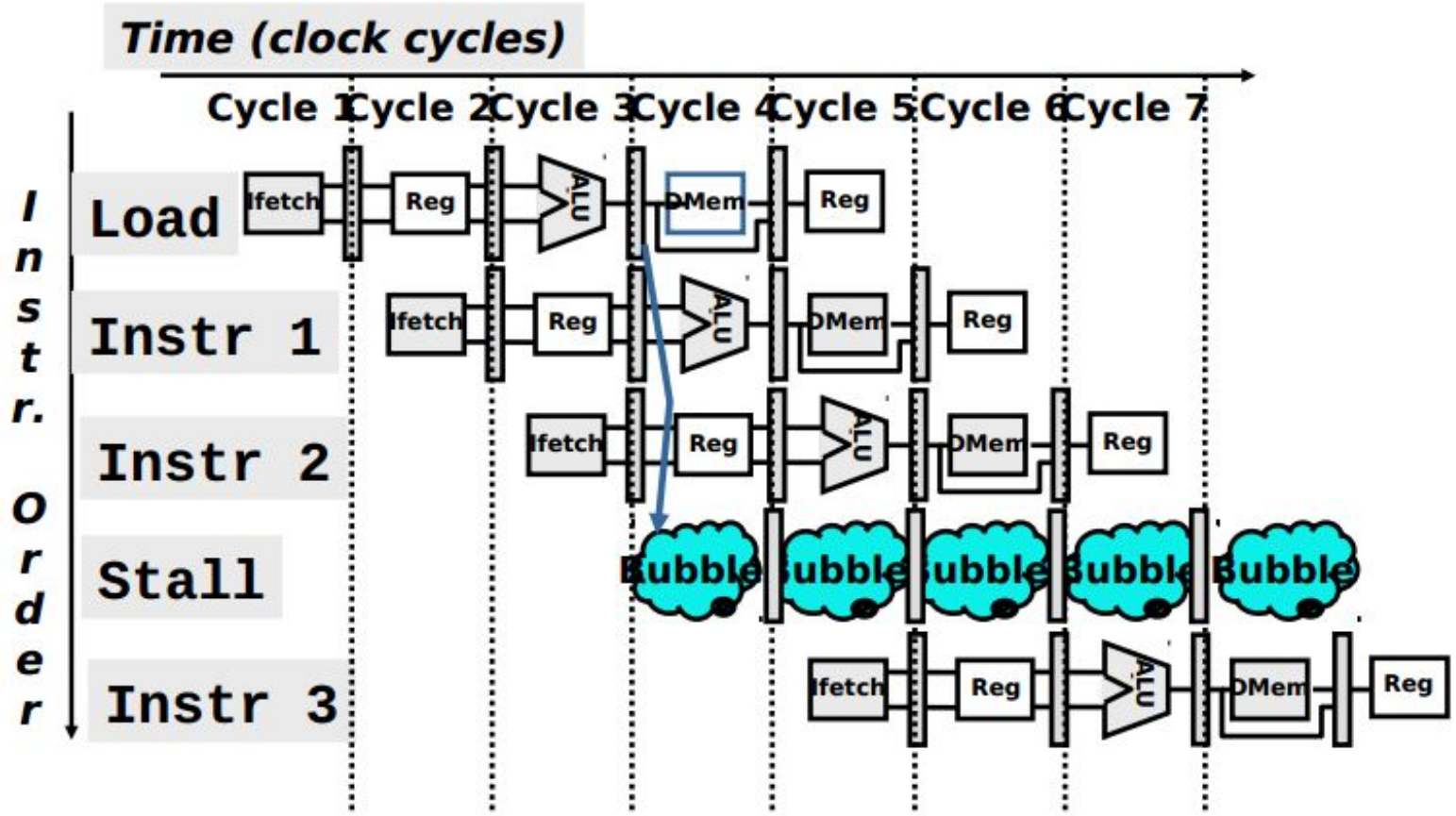
Pipeline Hazards (4/13)
Resolución Hazard Estructural



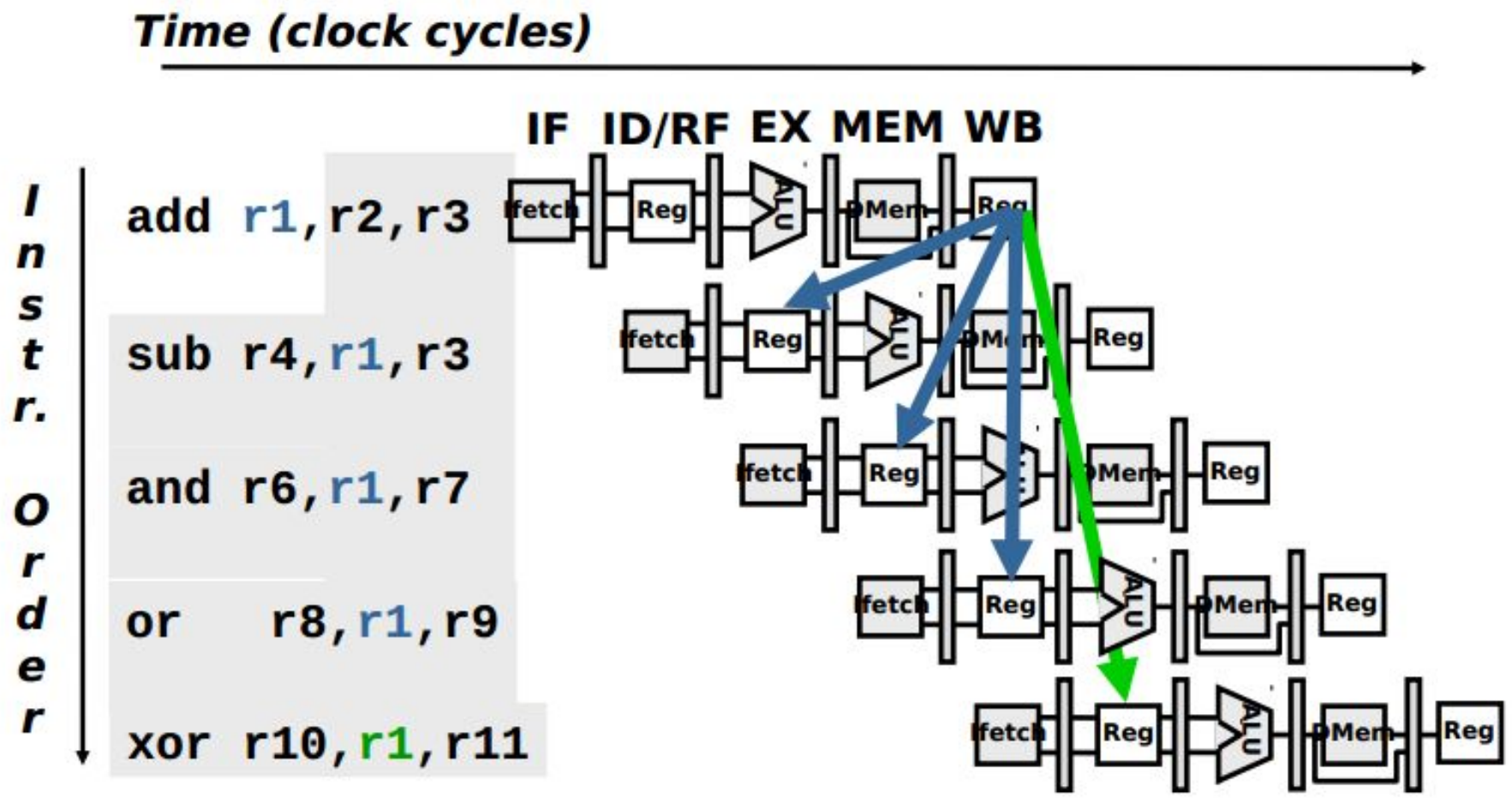
- Un hazard estructural se produce cuando dos instrucciones quieren acceder al mismo recurso de hardware al mismo tiempo.
- Dos opciones para solucionar:
  - Esperar… Se debe generar una señal de stall
  - Agregar más hardware!

Pipeline Hazards (5/13)
Resolución Hazard Estructural

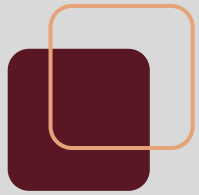
- Esperar



Pipeline Hazards (6/13)
Hazard de Datos



Pipeline Hazards (7/13)
Tipos de Hazard de Datos



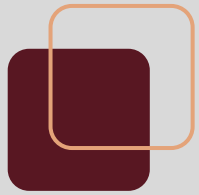
- Read after Write (RaW): Se produce si una instrucción posterior lee un resultado anterior que aún no fue calculado.

  I: add r1,r2,r3

  J: sub r4,r1,r3

  - Denominada 'true data dependency'

Pipeline Hazards (8/13)
Tipos de Hazard de Datos



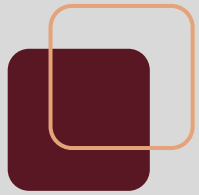
- Write after Read (WaR): Se da cuando una instrucción posterior intenta escribir un dato antes que una instrucción anterior lo haya leído.

  I: sub r4,r1,r3

  J: add r1,r2,r3

  - Denominada 'anti-dependencia'.
  - Se produce por reusar el registro r1

Pipeline Hazards (9/13)
Tipos de Hazard de Datos



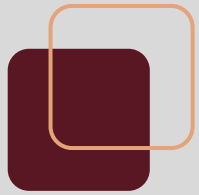
- Write after Write (WaW): Se produce cuando una instrucción posterior escribe un dato antes que una instrucción anterior lo pueda escribir.

  I: sub r1,r4,r3

  J: add r1,r2,r3

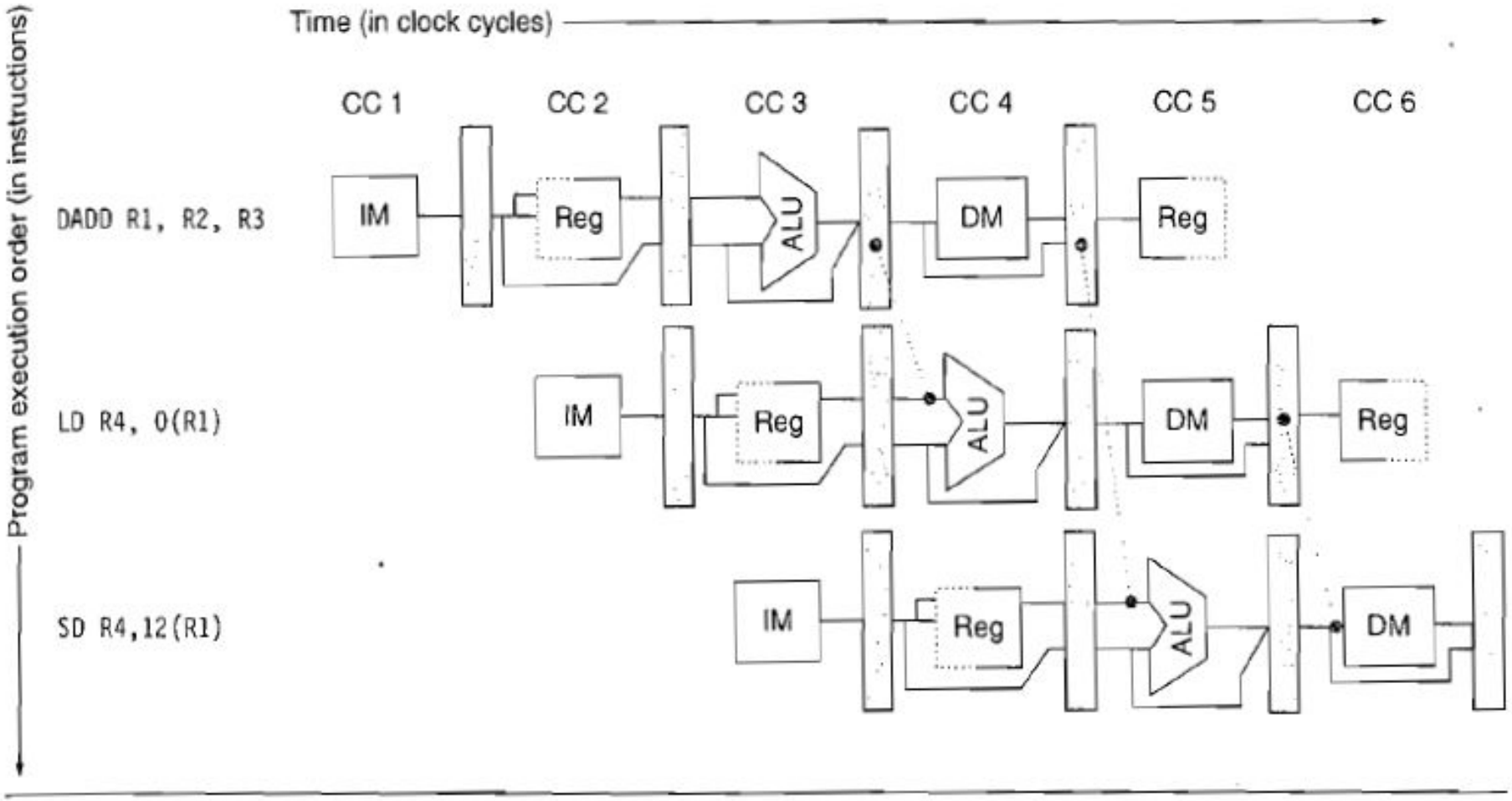
  - Denominada 'dependencia de salida'
  - Se produce por reusar el registro r1.

Pipeline Hazards (10/13)
Resolución de Hazards de Datos

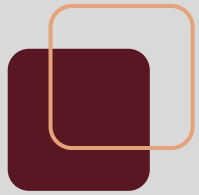


- Opciones:
  - Esperar… Se debe generar una señal de stall o tener ayuda del compilador.
  - Agregar conexiones al camino de datos para solucionar el problema (*forwarding*).

Pipeline Hazards (11/13)
Forwarding



Pipeline Hazards (12/13)
Hazards de Control

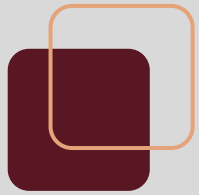


- Se puede dar al ejecutar un salto:

Untaken branch instruction	IF	ID	EX	MEM	WB				
Instruction $i + 1$		IF	ID	EX	MEM	WB			
Instruction $i + 2$			IF	ID	EX	MEM	WB		
Instruction $i + 3$				IF	ID	EX	MEM	WB	
Instruction $i + 4$					IF	ID	EX	MEM	WB

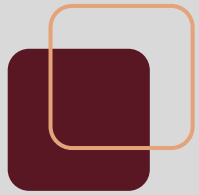
Taken branch instruction	IF	ID	EX	MEM	WB				
Instruction $i + 1$		IF	**idle**	**idle**	**idle**	**idle**			
Branch target			IF	ID	EX	MEM	WB		
Branch target + 1				IF	ID	EX	MEM	WB	
Branch target + 2					IF	ID	EX	MEM	WB

Pipeline Hazards (13/13)
Solución de Hazards de Control



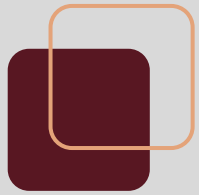
- Prefetch del destino: cargar al mismo tiempo la siguiente instrucción y el destino del salto.
- Múltiples Flujos: consiste en mejorar el paso anterior hasta la etapa donde se decide el salto.
- Salto demorado: Ejecutar sí o sí la instrucción que está después del salto.
- Predicción de saltos.

Predicción de Saltos (1/5)



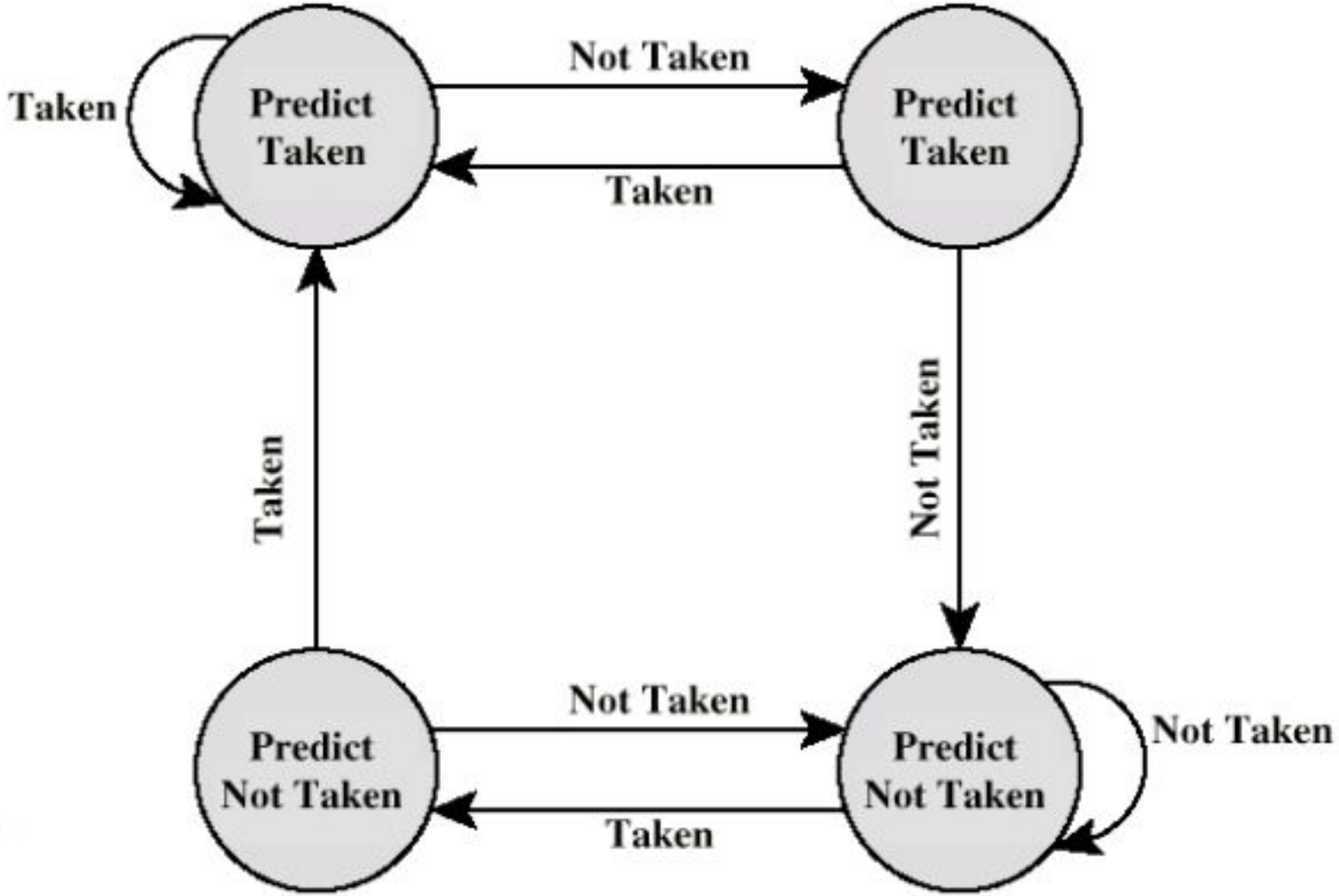
- Los saltos provocan hazards de control únicamente si las instrucciones que se cargaron posteriormente al salto no deben ser ejecutadas.
- Para evitar esto, los procesadores con pipeline incluyen predictores de saltos, que intentan adivinar cuál será el resultado del salto para cargar las instrucciones que se deben ejecutar.

Predicción de Saltos (2/5)

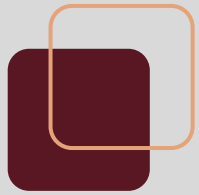


- Predictores:
  - Predecir que nunca se toma el salto (always not taken)
  - Predecir que siempre se toma el salto (always taken)
  - Predicción basada en el código de operación
  - Conmutar taken/not taken

Predicción de Saltos (3/5)
Conmutar Taken / Not Taken

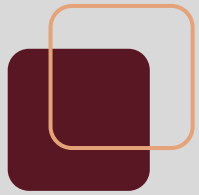


Predicción de Saltos (4/5)
Branch History Table (BHT)



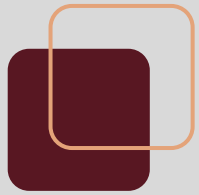
- Consiste en mantener una pequeña memoria en el CPU, la cual se indiza con la dirección de la instrucción. Cada entrada mantiene un predictor en particular (típicamente uno de dos bits como en el caso anterior).
- Mejora el caso anterior, ya que disminuye la probabilidad de que dos saltos interfieran en las predicciones.

Predicción de Saltos (5/5)



- Otros predictores:
  - Branch Target Buffer
  - Predictores de Torneo
- Más adelante en el curso!

Fin



¿Preguntas?