



Universidad Nacional  
de Entre Ríos

Tecnicatura universitaria en desarrollo web

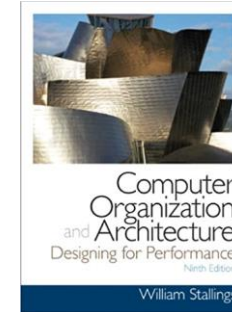
# Segmentación de instrucciones

Semana 8 – Arquitectura de computadoras

# Esta presentación esta basada en el libro de:

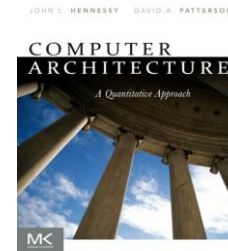
❑ William Stallings, Computer Organization and Architecture, 9th Edition, 2017.

- Capítulo 16: "INSTRUCTION-LEVEL PARALLELISM"



❑ Hennessy-Patterson - Computer Architecture A Quantitative Approach (5th edition)

- Capítulo 3: "ILP Instruction-Level Parallelism"

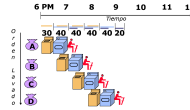


Archivos presentación y ejemplos se alojan en:



<https://github.com/ruiz-jose/tudw-arq.git>

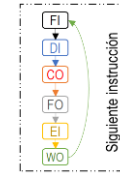
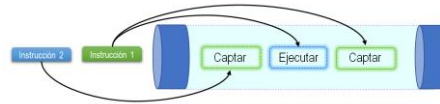
- Técnica de segmentación



- Ciclo de instrucción



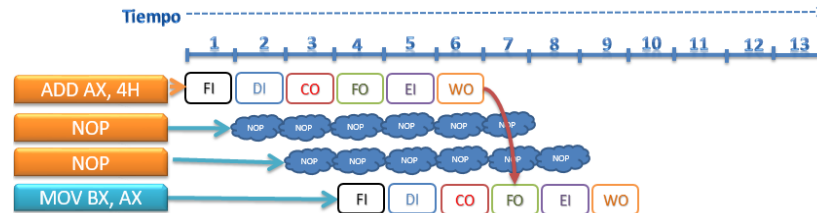
- Segmentación de instrucciones



- Riesgos



- Posibles soluciones

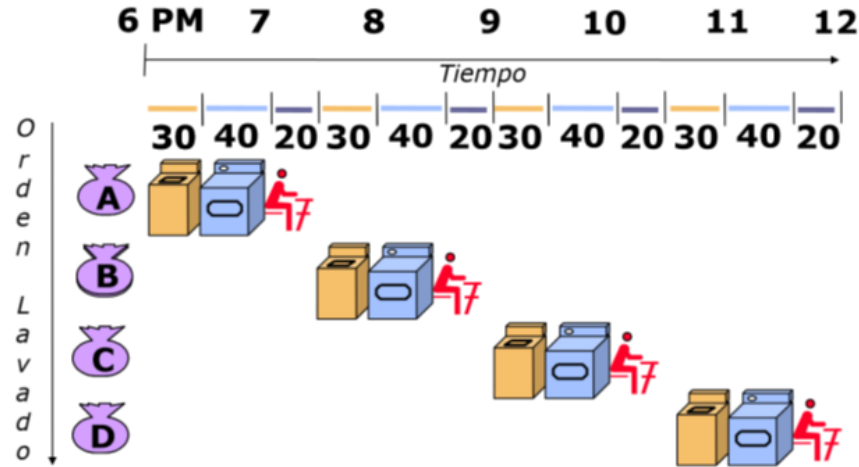



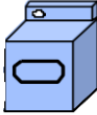

## Objetivo

**Comprender los conceptos de segmentación de instrucciones y los principales factores que limitan su rendimiento.**

La técnica de segmentación se utiliza en la cadena de montaje en una fábrica.  
**Por ejemplo una lavandería.**

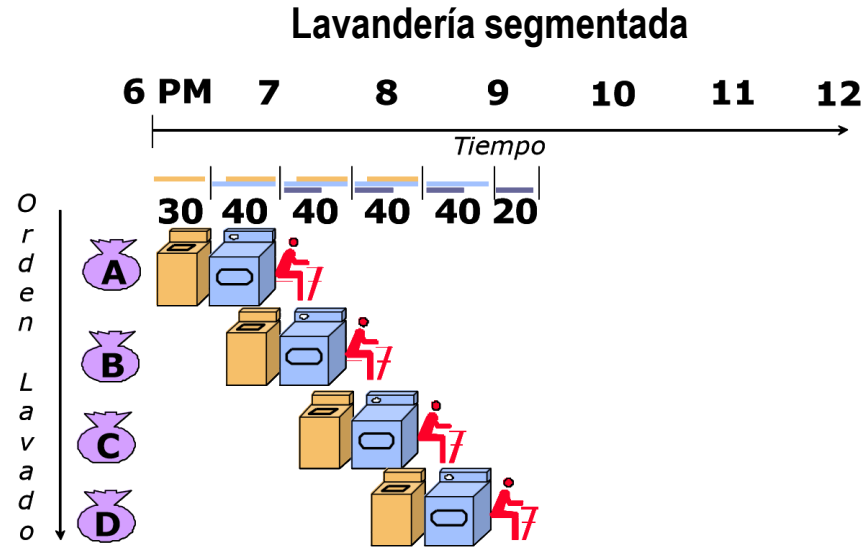
### Lavandería secuencial



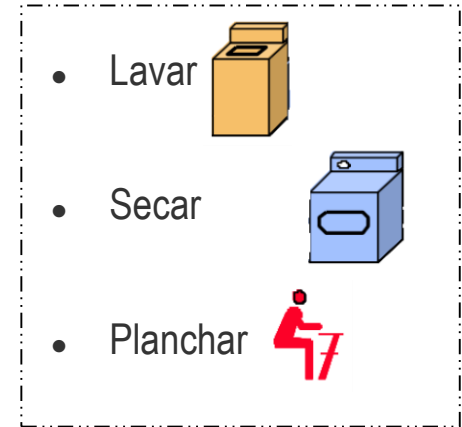
- Lavar 
- Secar 
- Planchar 

**90 minutos cada carga, total (4 cargas): 6 horas.**

En el proceso de lavado de la lavandería se puede trabajar sobre cada carga en varias etapas simultáneamente.



**Total (4 cargas): 3,5 horas.**



**En la lavandería:**  
**¿Se reduce el tiempo de lavado de cada carga?**



## Lavandería secuencial

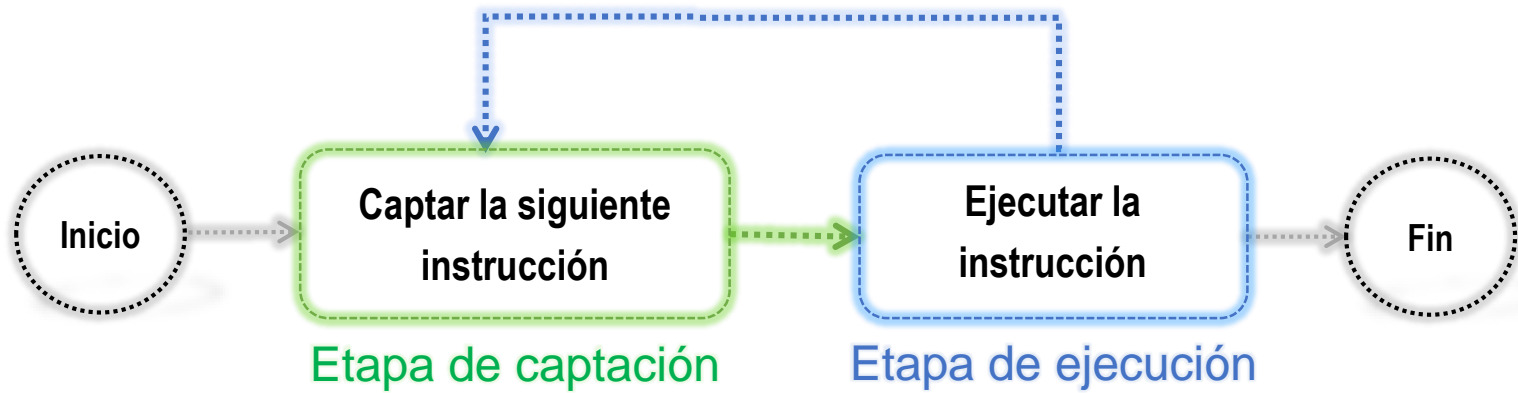


6 horas = 4 cargas

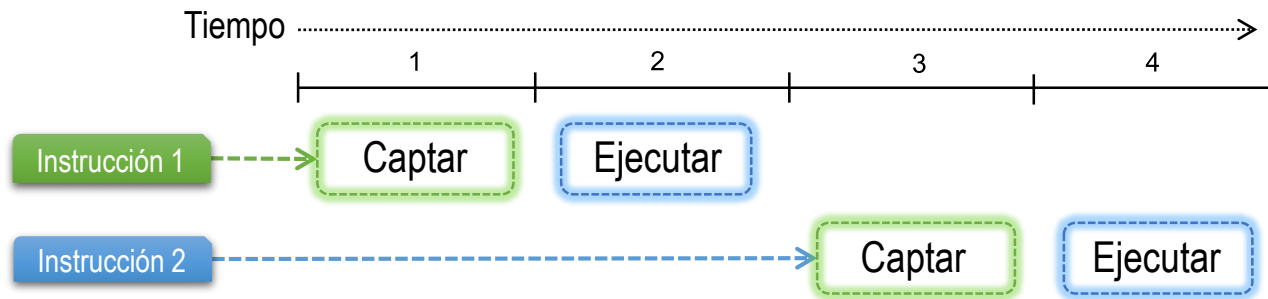
## Lavandería segmentada



6 horas  $\approx$  aproximadamente 7 cargas

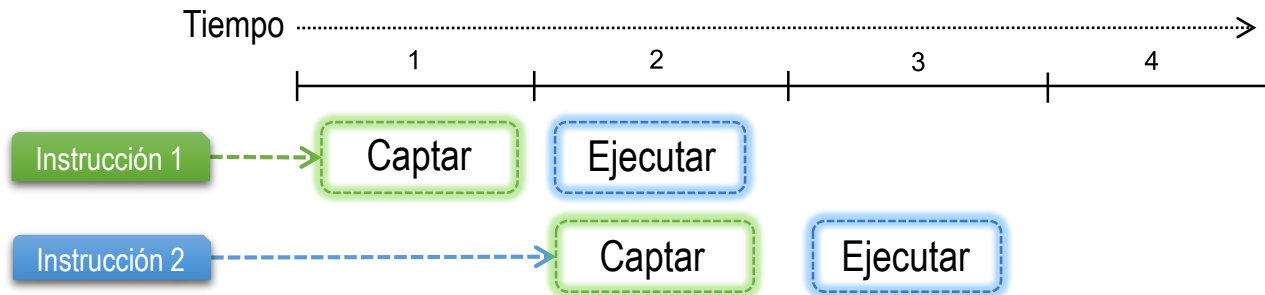


## ❑ Ejecución secuencial



Pero hay periodos en la ejecución de una instrucción en los que no se accede a memoria principal, entonces este tiempo podría utilizarse para captar la siguiente instrucción en paralelo con la ejecución de la actual.

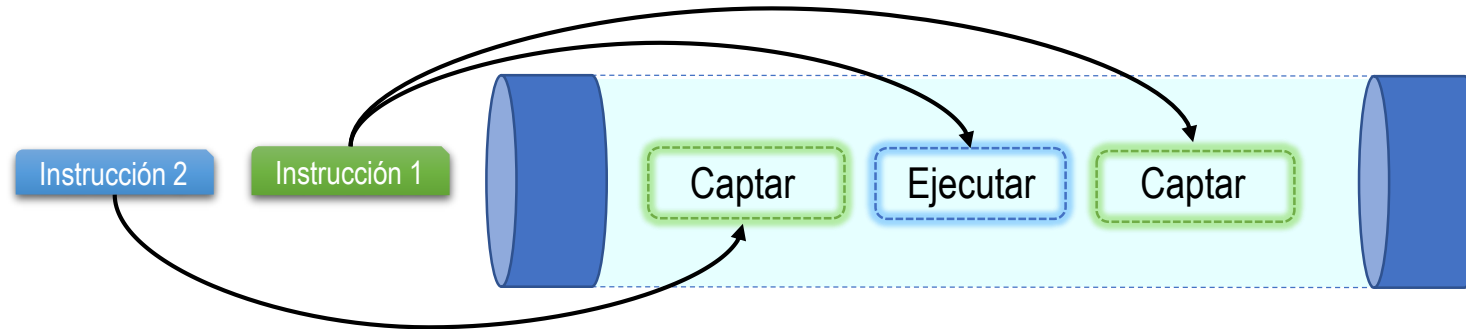
## ❑ Ejecución segmentada



¿Qué es la segmentación de instrucciones?

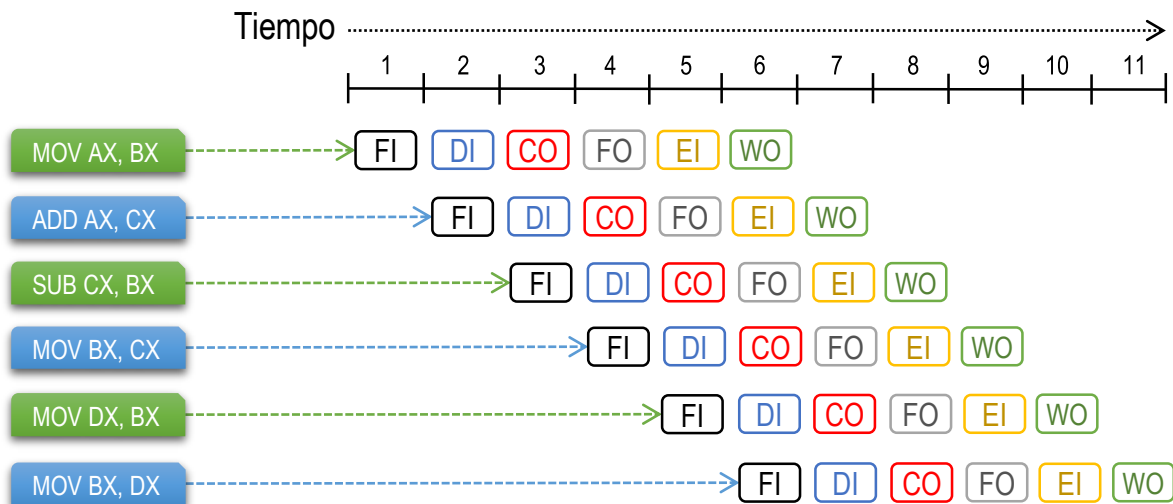
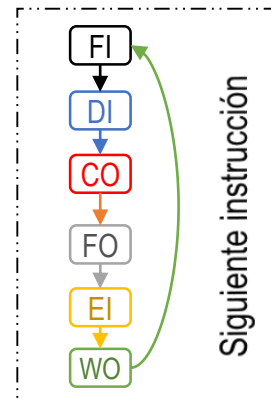
Es una técnica que consiste en descomponer el ciclo de instrucción en etapas que permitan una ejecución simultánea.

También se denomina **segmentación de cauce o Pipelining**: porque al igual que en una tubería (pipeline), en un extremo se aceptan entradas nuevas antes de que las anteriores sean salidas en el otro extremo.



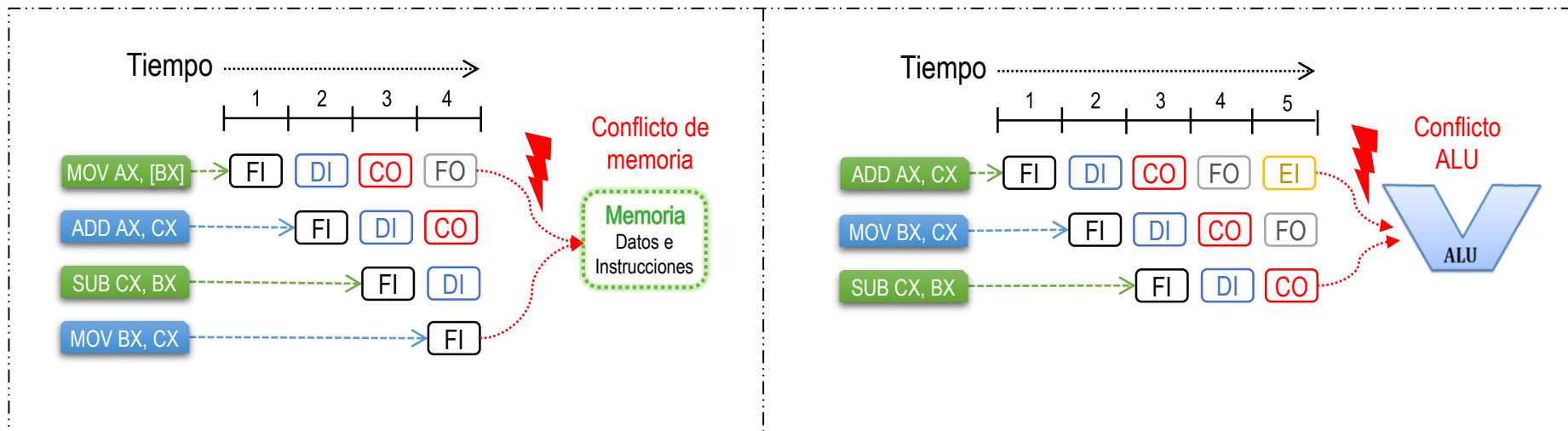


- 1 Captar Instrucción (FI - **Fetch Instruction**)
- 2 Decodificar Instrucción (DI - **Decode Instruction**)
- 3 Calcular Operandos (CO - **Calculate Operands**)
- 4 Captar Operandos (FO - **Fetch Operands**)
- 5 Ejecutar instrucción (EI - **Execute Instruction**)
- 6 Escribir Operando (WO - **Write Operand**)



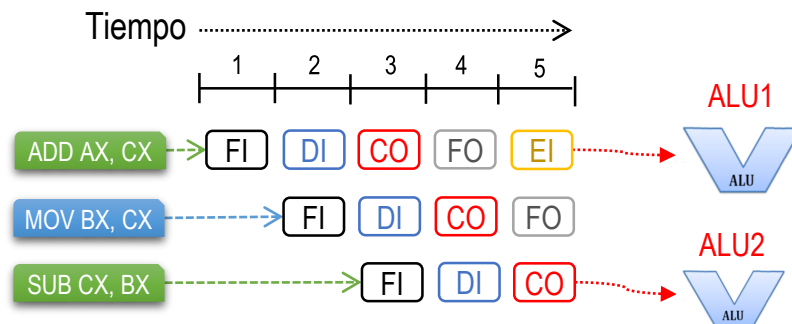
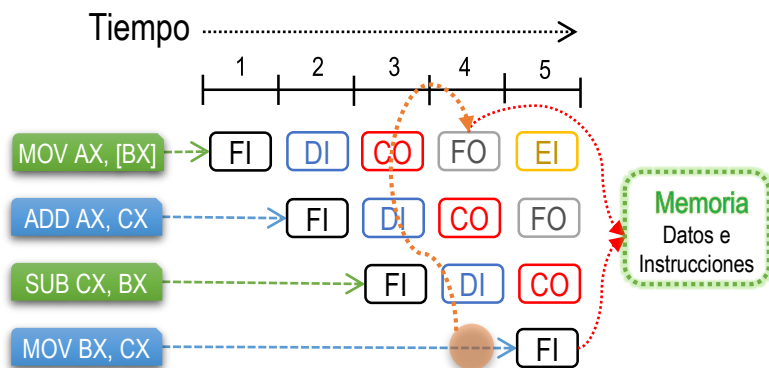


Ocurre cuando dos o más instrucciones que están en el cauce necesitan utilizar el mismo recurso hardware en el mismo ciclo.



1 Insertar un ciclo ocioso por hardware.

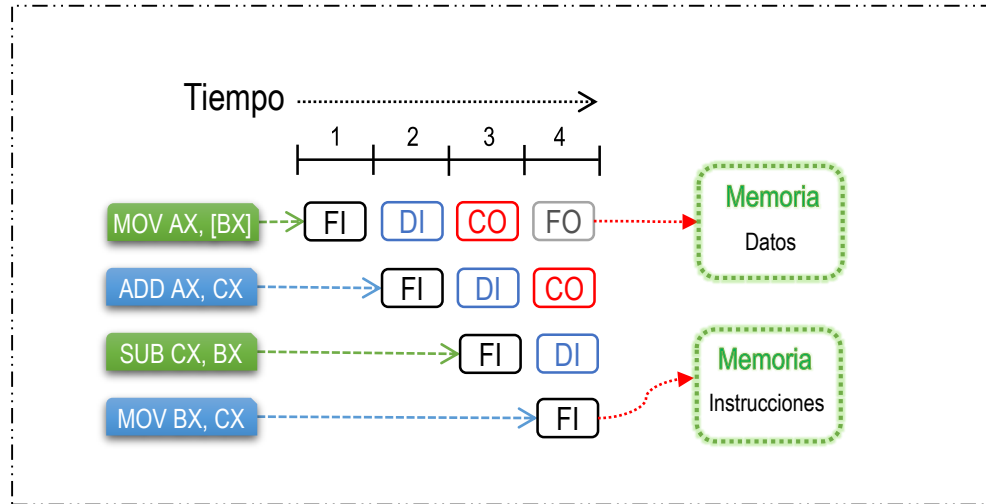
2 Duplicación de hardware: dos ALU.



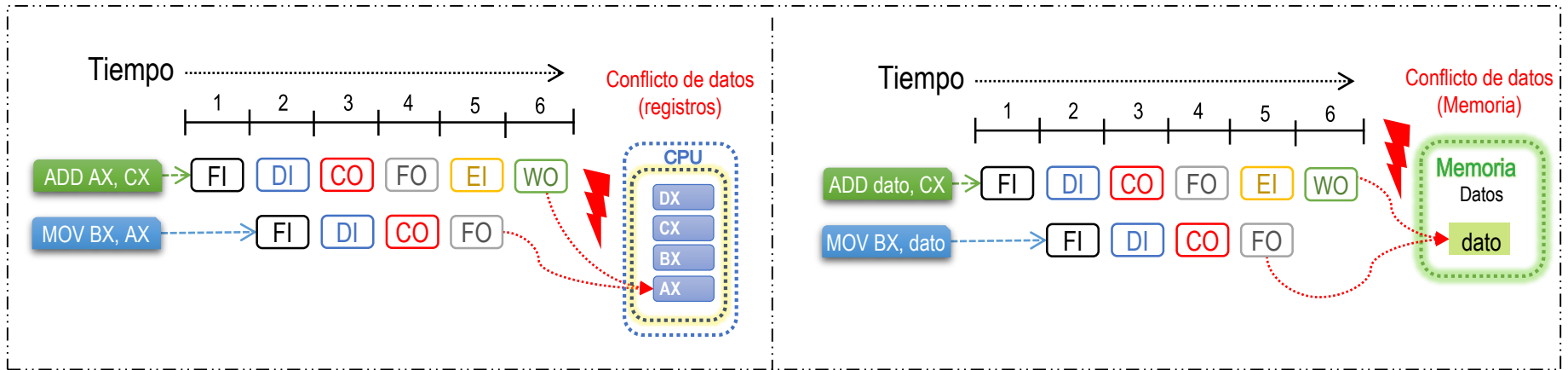


3

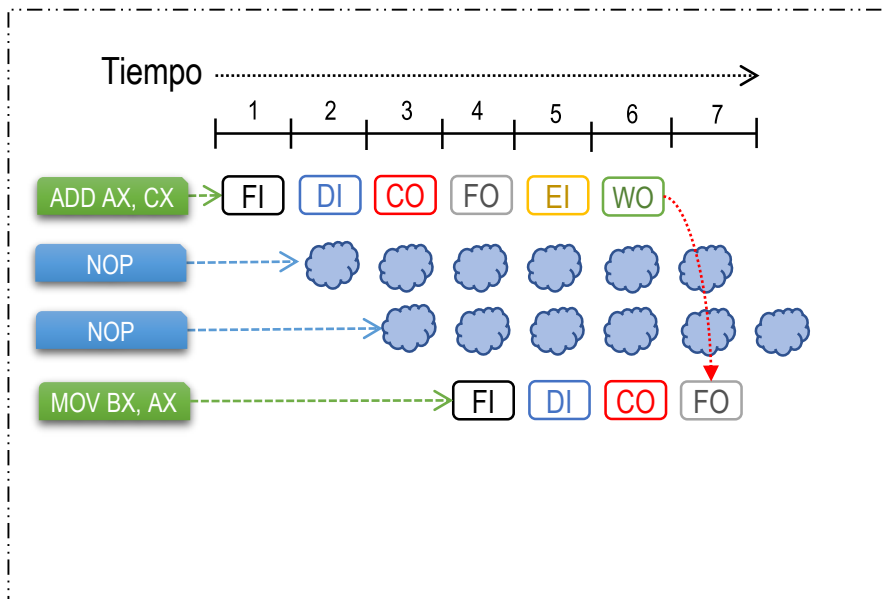
Separar memoria en datos e instrucciones (Arq. Harvard).



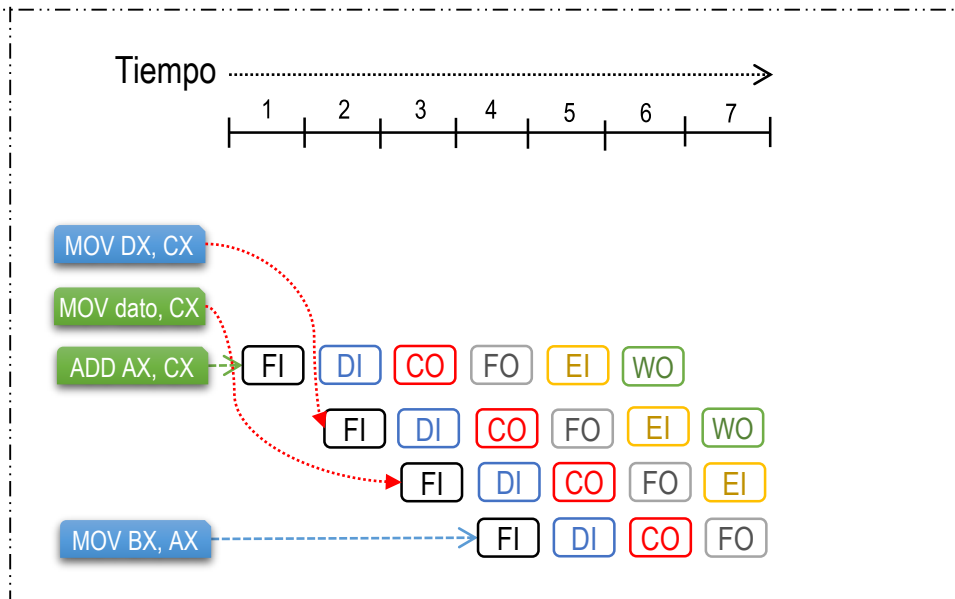
Ocurren cuando dos instrucciones utilizan el mismo dato en determinada etapa del cauce.



1 El compilador inserta **ciclos ociosos** (burbujas), instrucción **NOP**.

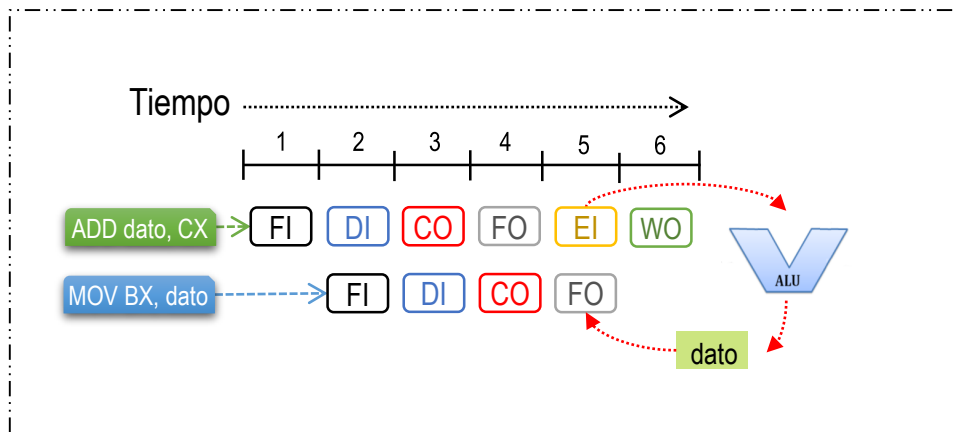


2 **Reordenación de código:** se separa lo máximo posible las instrucciones con dependencia de datos.

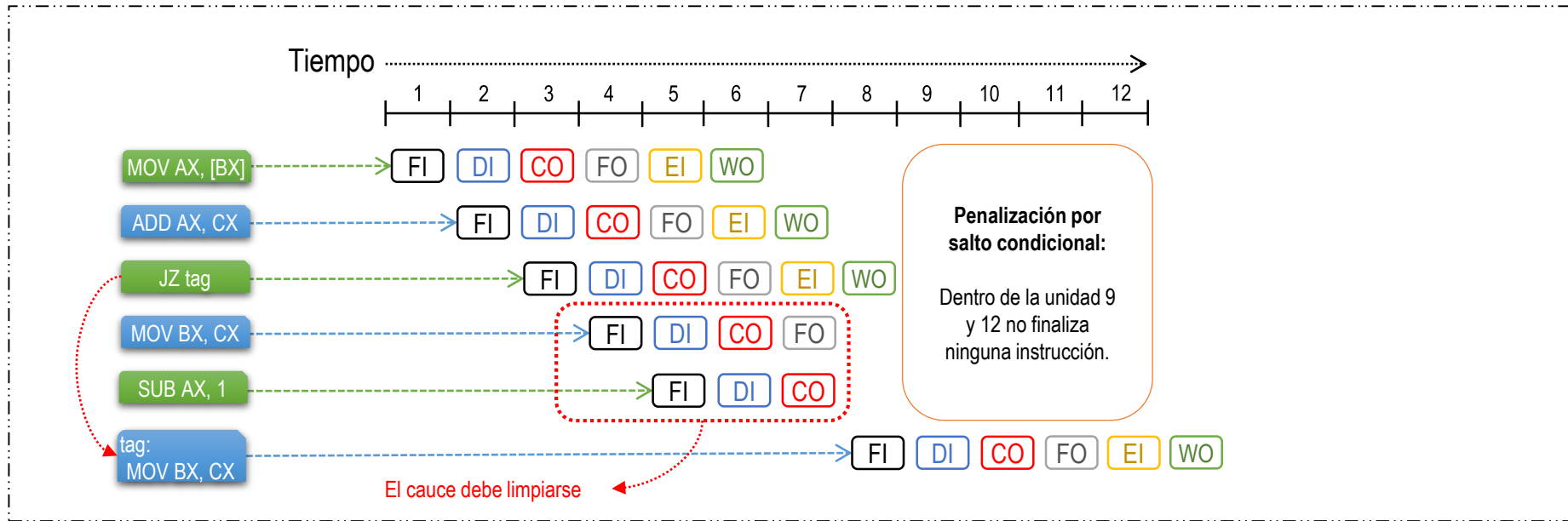


3

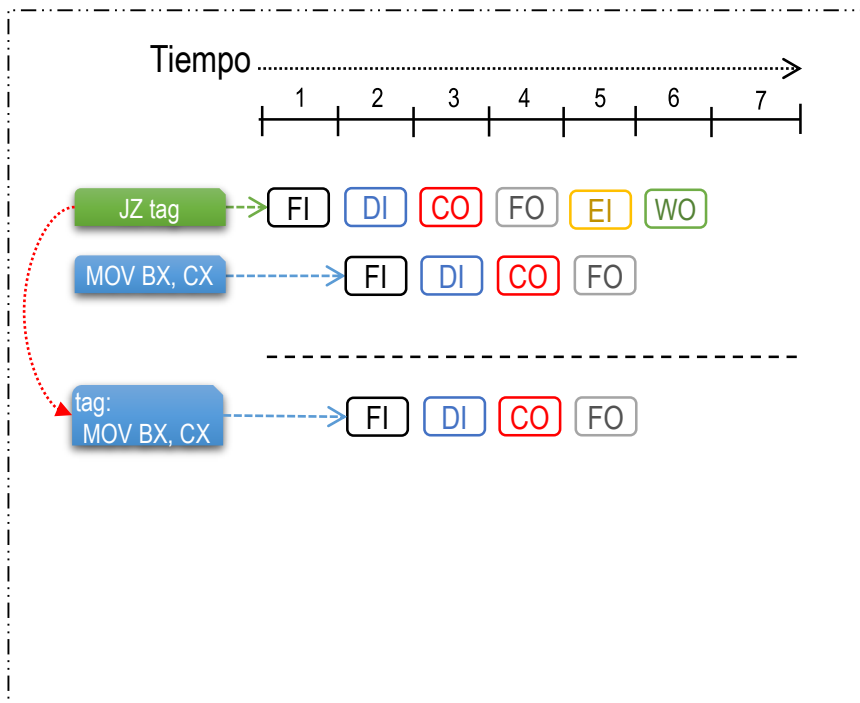
**Adelantamiento de operandos (forwarding):** consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando (se hace por hardware)



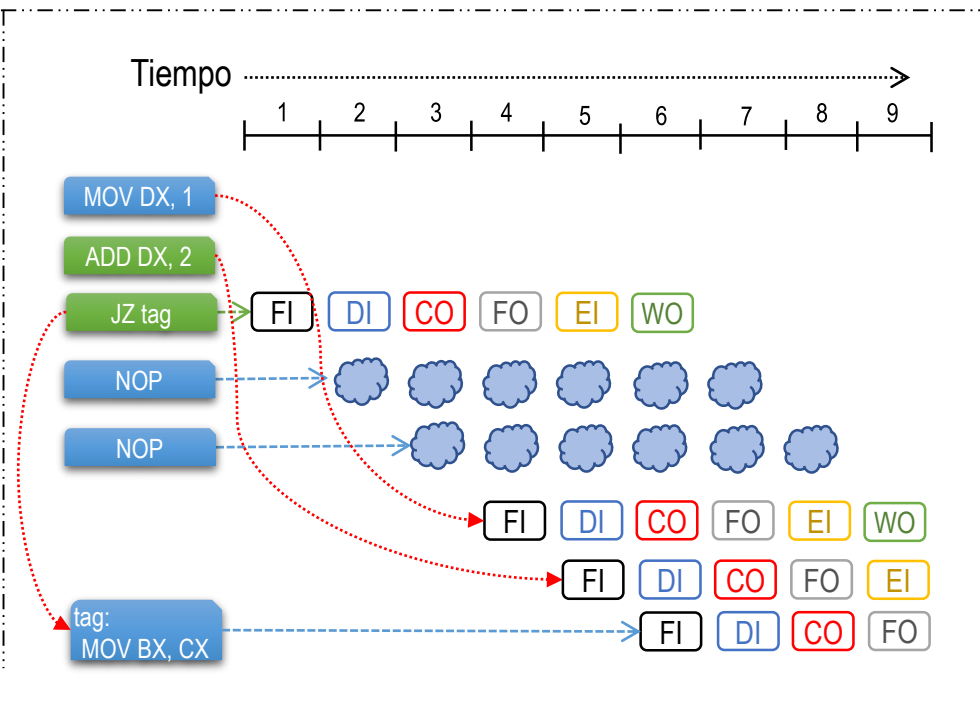
El problema se presenta cuando la ejecución de una instrucción depende de cómo se ejecute otra. **Ejemplo instrucción de salto condicional.**



- 1 Flujo múltiple:** se implementa duplicando las partes iniciales del cauce y se captan los dos caminos (siguiente instrucción e instrucción destino del salto).



- 2 Salto retardado:** en el período de penalización de una instrucción de salto, el compilador trata de situar instrucciones útiles (que no dependan del salto) en ese periodo. Si no es posible, se utilizan instrucciones NOP (requiere reordenamiento de código).



❖ **Estáticas:** no dependen de la historia de la ejecución.

- Asume que **nunca** se salta.
- Asume que **siempre** se salta.
- Predecir según el **código** de operación.

❖ **Dinámicas:** dependen de la historia de la ejecución, registran la historia de saltos de un programa.

- **Tabla de historia de saltos:** posee una cache asociada y por cada instrucción de salto se almacena en la cache dirección de la instrucción de salto, dirección destino de la instrucción y el historial del salto (si salto o no).

Dirección instrucción salto

Dirección instrucción destino

Estado

# Preguntas?