Pipelining

Introducción

Definición

- Técnica de implementación.
- Consiste en ejecutar las instrucciones traslapadas.
- La siguiente instrucción puede comenzar antes de que la instrucción actual termine de ejecutarse.

Ejemplo

- Considerar la tarea de lavar y secar ropa.
- Si hay más de una carga de ropa, hay dos formas de lavar:
  1. Versión secuencial: lavar y secar una carga de ropa a la vez.
  2. Versión con pipeline: separar la tarea en pasos y hacer distintos pasos en paralelo.

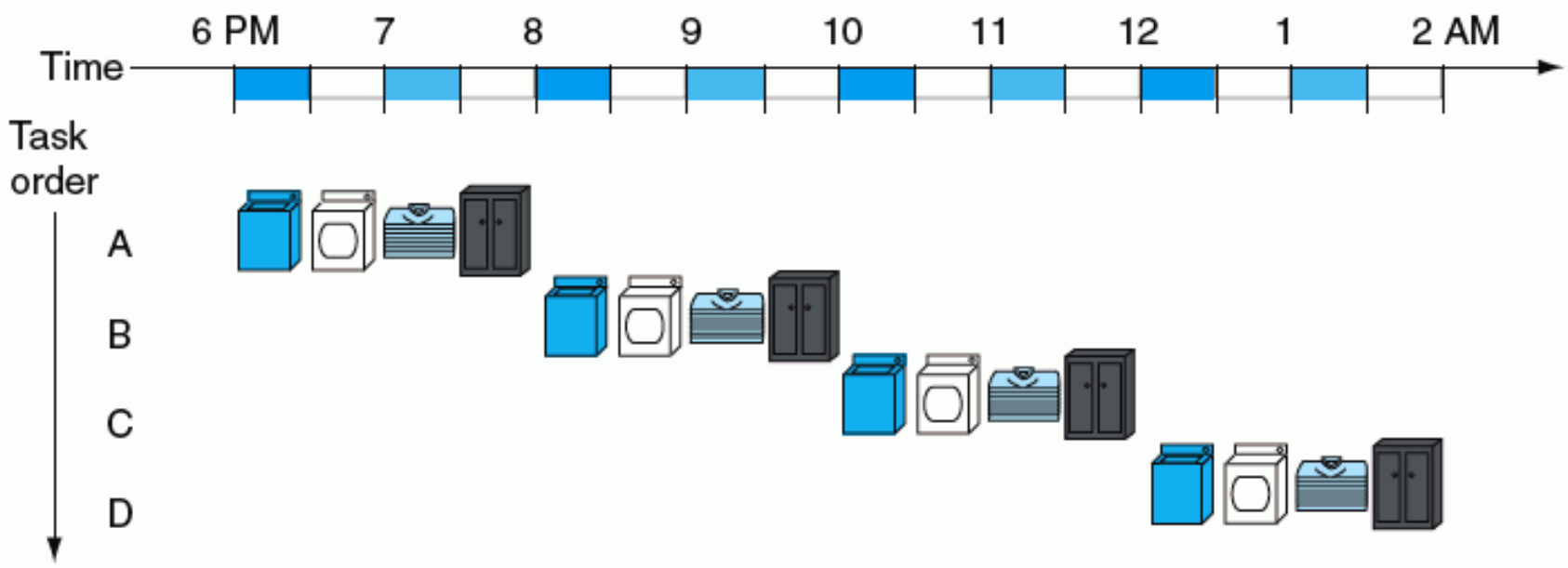
Ejemplo

- Pasos para lavar y secar una carga de ropa:

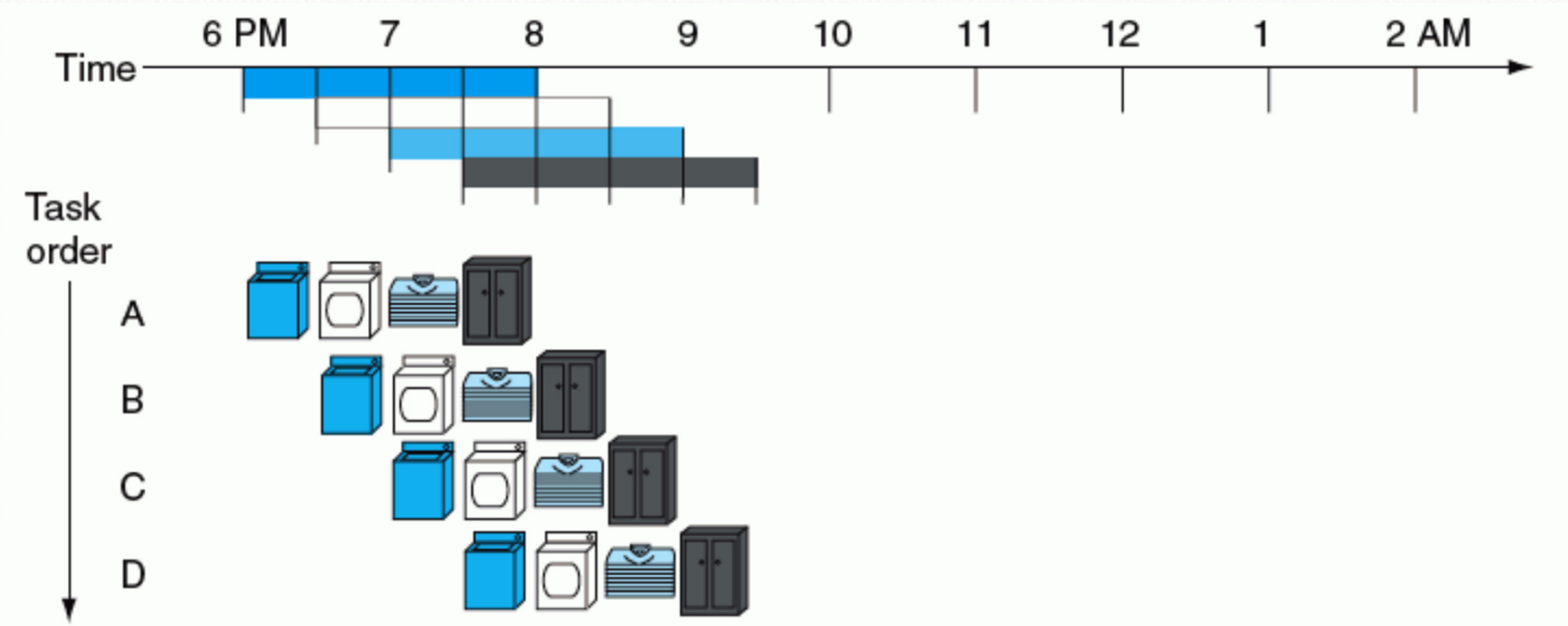
1. Poner una carga de ropa sucia en la lavadora.
2. Al terminar, poner la ropa húmeda en la secadora.
3. Al terminar, poner la ropa seca en una mesa y doblarla.
4. Al terminar, guardar la ropa doblada.

- Suponer que hay cuatro cargas de ropa y que cada paso tarda media hora (30 minutos).

Versión secuencial



Versión con pipeline



Conclusión

- Para 1 carga de ropa:
  - Versión secuencial: 2 horas.
  - Versión con pipeline: 2 horas.
  - Speedup: 2 / 2 = 1.0.
- Para 4 cargas de ropa:
  - Versión secuencial: 8 horas.
  - Versión con pipeline: 3.5 horas.
  - Speed-up: 8 / 3.5 = 2.29.

Conclusión

- Para 20 cargas de ropa:
  - Versión secuencial: 40 horas.
  - Versión con pipeline: 11.5 horas.
  - Speed-up: 40 / 11.5 = 3.48.
- El speed-up está limitado por el número de *etapas* del pipeline.
- El speed-up depende del factor de utilización del pipeline.

Conclusión

- Para tener un pipeline se necesitan recursos para cada etapa.

Pipelining en MIPS

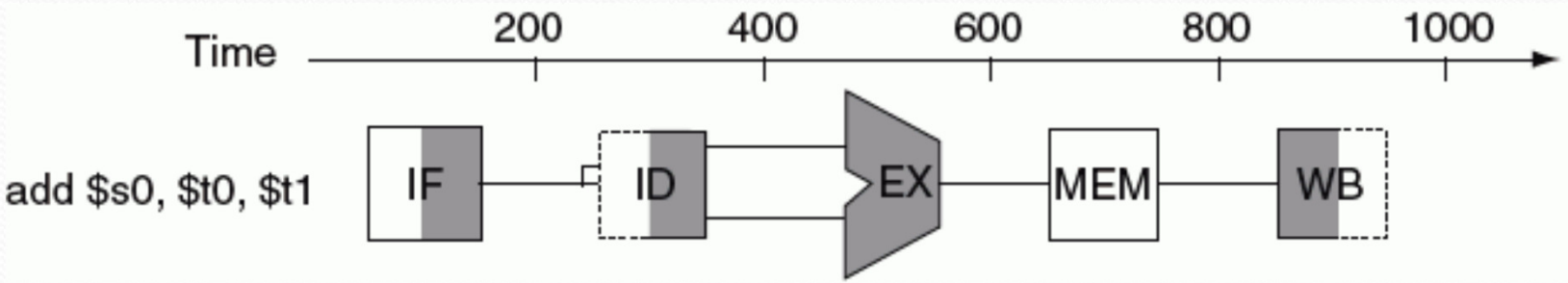
- Las instrucciones de MIPS tienen 5 etapas:

1. Obtener (fetch) una instrucción de la memoria.
2. Leer los registros mientras se decodifica la instrucción.
3. Ejecutar la operación o calcular una dirección.
4. Accesar un operando en la memoria de datos.
5. Escribir el resultado en un registro.

Pipelining en MIPS

- IF (instruction fetch): obtiene la instrucción de la memoria.
- ID (instruction decoding): decodifica la instrucción y lee los operandos.
- EX (execute): ejecuta la instrucción, o si es lw/sw calcula la dirección de la memoria.
- MEM (memory access): lee/escribe la memoria. Es una no-op para instrucciones tipo-R.
- WB (write back): escribe el resultado en el registro.

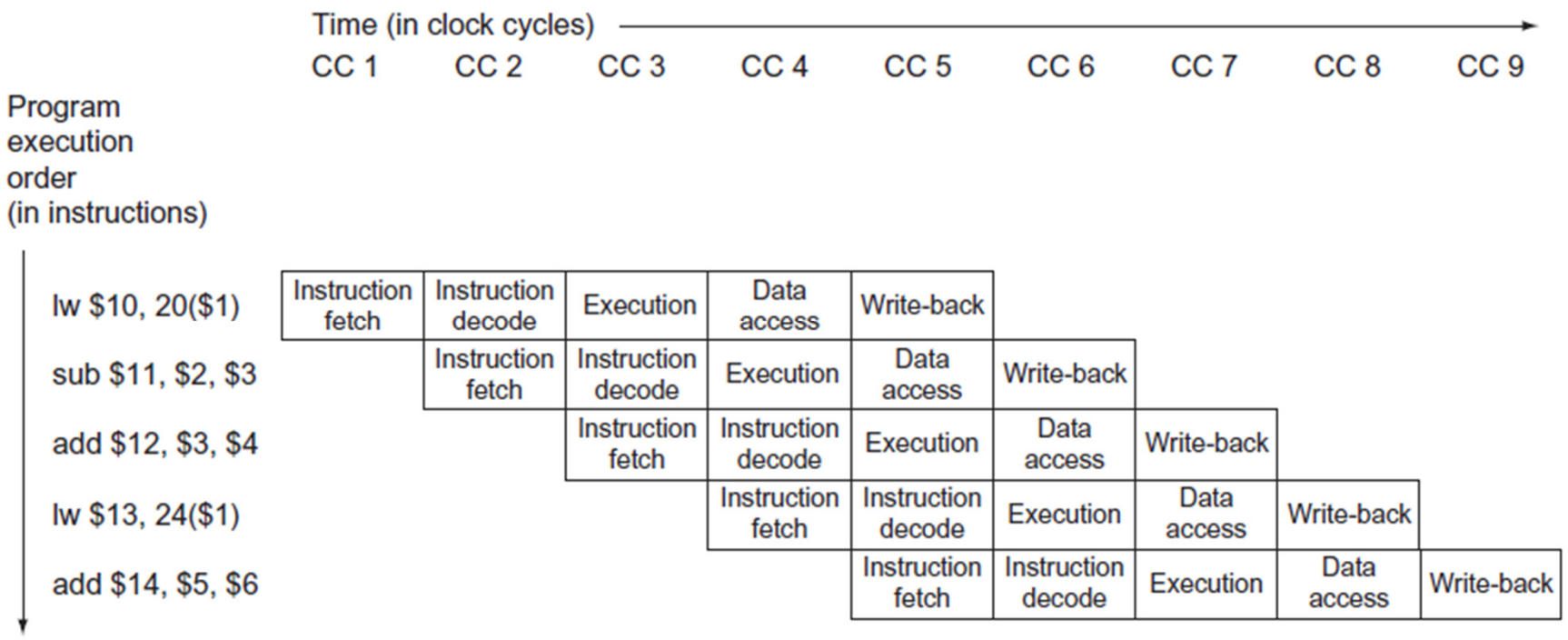
Ejemplo



- Cada etapa se hace en un ciclo.
- El cuadro completo representa la memoria de instrucciones.
- El cuadro punteado representa el banco de registros.
- EX tiene el símbolo de la ALU.
- MEM está en blanco porque *add* no usa la memoria de datos.

Ejemplo

- Idealmente en cada ciclo entra una instrucción.



- **Fuente: COD5, p. 299.**

Pipelining en MIPS

- Características de MIPS que ayudan a implementar cada etapa del pipeline en un ciclo:

1. Las instrucciones tienen la misma longitud. Etapas IF e ID.
2. Solo 3 formatos de instrucción. Etapa ID.
3. Las instrucciones *lw* y *sw* son las únicas que accesan la memoria. Etapas EX y MEM.
4. Los operandos están alineados en la memoria. Etapa MEM.

Comparación

- Comparar la implementación MIPS de un ciclo contra la implementación de un pipeline.
- Se van a comparar una secuencia de 3 instrucciones *lw*.
- Versión de un ciclo: todas las instrucciones tardan un ciclo de reloj.
- Versión con pipeline: cada etapa tarda un ciclo.

Comparación

- Suponer los siguientes tiempos:
  - 200 ps por acceso a memoria.
  - 200 ps por operación de la ALU.
  - 100 ps por leer o escribir en el banco de registros.

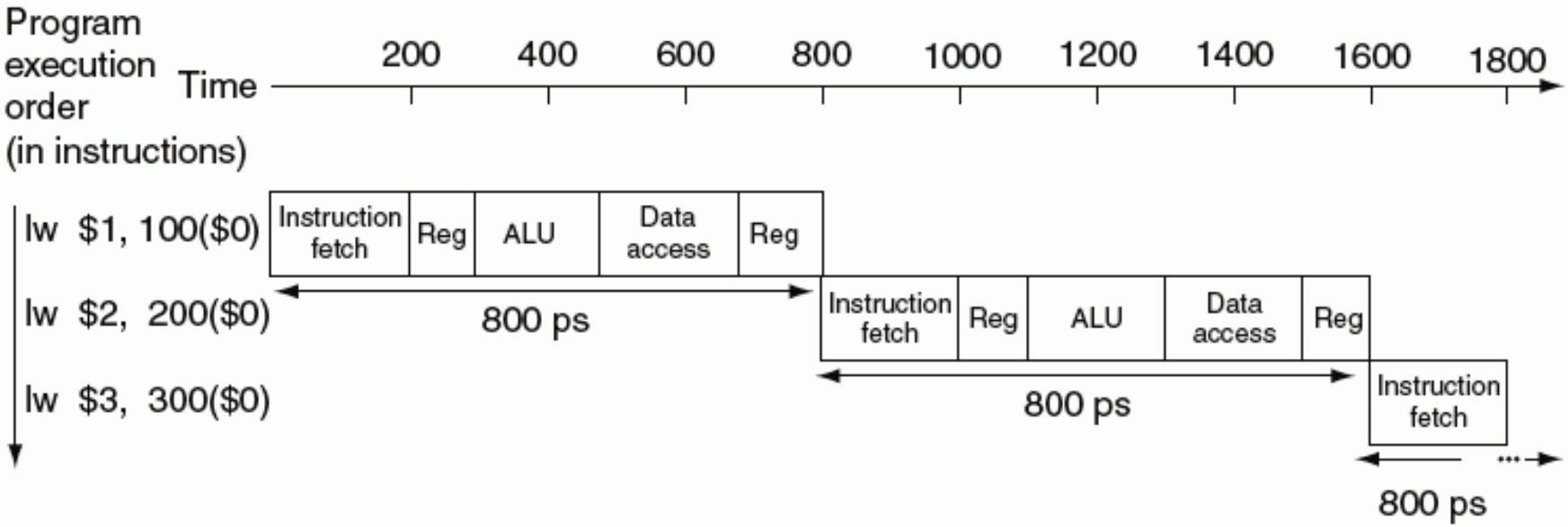
Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (`lw`)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (`sw`)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (`add`, `sub`, `and`, `or`, `slt`)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (`beq`)	200 ps	100 ps	200 ps			500 ps

Comparación

- En la versión de un ciclo, la duración del ciclo de reloj se acopla a la instrucción mas lenta.
- El periodo de reloj es de 800 ps, aunque haya instrucciones que tarden 500 ps.

Versión de un ciclo

- Las 3 instrucciones tardan 800 x 3 = 2,400 ps.

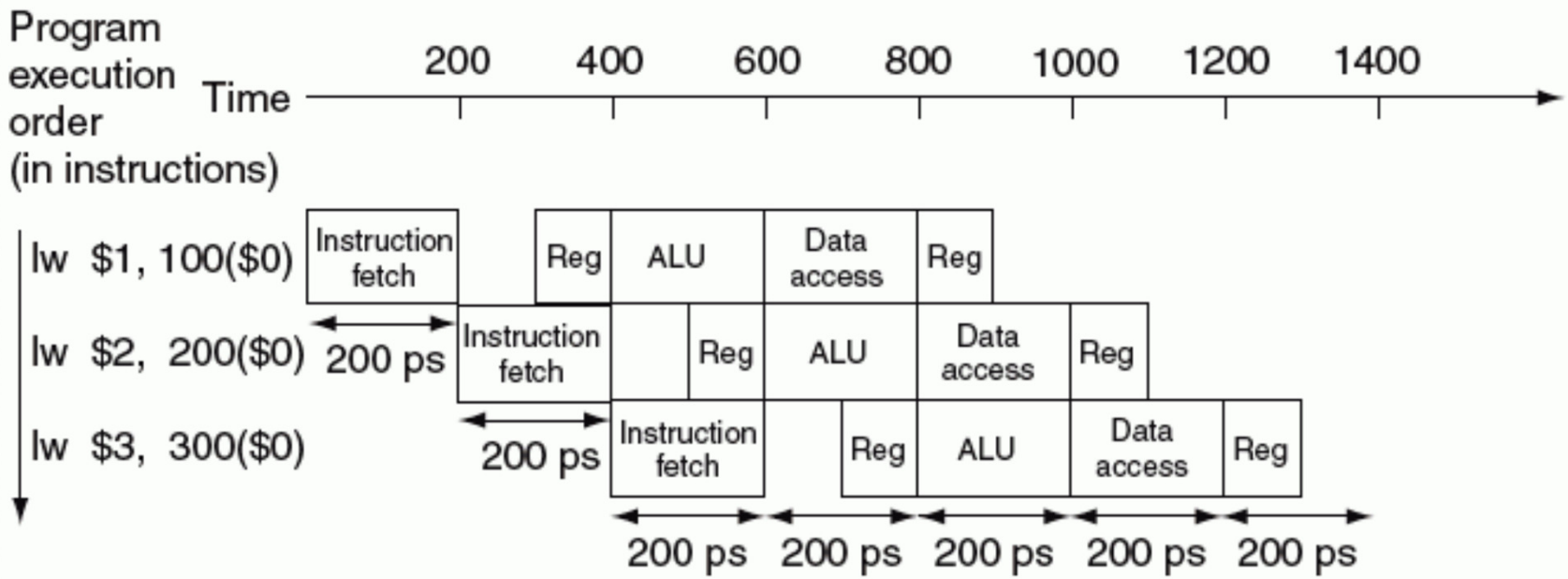


Versión con pipeline

- Cada etapa se realiza en un ciclo.
- La duración del ciclo de reloj se acopla a la etapa más lenta.
- El periodo de reloj es de 200 ps, aunque haya etapas que tarden 100 ps.

Versión con pipeline

- Las 3 instrucciones tardan 1,400 ps.



Conclusiones

- Speedup = 2400 / 1400 = 1.71.
- La versión con pipeline es 1.71 veces más rápida que la versión secuencial de un ciclo.

Speedup

- El speedup depende de la utilización del pipeline.
- En el ejemplo, con 3 instrucciones el speedup es de 1.71.
- Si se ejecutan un millón de instrucciones:

$$\text{Speedup} = \frac{800{,}000{,}000}{200{,}000,\ 800} = 3.99998$$

Speedup

- El speedup máximo que se obtiene al usar un pipeline es:

$$\text{Speedup}_{\max} = \frac{\text{Ciclo}_{\text{secuencial}}}{\text{Ciclo}_{\text{pipeline}}}$$

- En el ejemplo:

$$\text{Speedup}_{\max} = \frac{800}{200} = 4$$

Speedup

- **Pipeline balanceado**. Es un pipeline donde todas las etapas duran lo mismo.
- En un pipeline balanceado el speedup máximo es:

$$\text{Speedup}_{\max} = \text{Num. de etapas}$$

- Un pipeline de 5 etapas puede ser hasta 5 veces más rápido que la versión sin pipeline (secuencial).
- El pipeline del ejemplo es no balanceado (unas etapas duran 100 ps y otras 200 ps).
- Por eso el speedup está limitado a 4.

Conclusión

- El uso de un pipeline mejora el rendimiento incrementando el throughput sin decrementar el tiempo de ejecución de una instrucción individual.