

# **Tema 10 - Arquitectura de microprocesadores actuales**

Sistemas Electrónicos para Automatización  
Grado en Ingeniería Electrónica, Robótica y  
Mecatrónica

Hipólito Guzmán Miranda

## Contenido

- Revisión de arquitecturas clásicas
- Concepto de pipeline
- Memorias caché

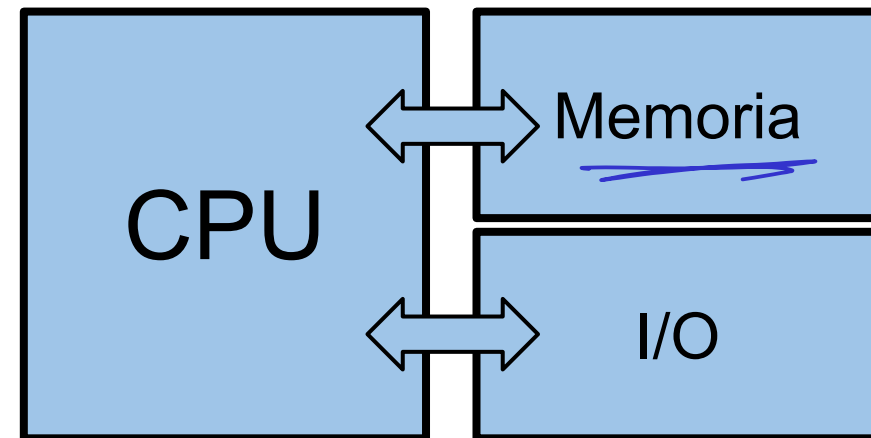
## Contenido

- **Revisión de arquitecturas clásicas**
- Concepto de pipeline
- Memorias caché

## Revisión arquitecturas clásicas

La separación entre la CPU y la memoria hace que el computador sea programable

- CPU (Unidad Central de Proceso)
- Memoria
- Perif. Entrada/Salida



## Von Neumann vs Harvard

### Von Neumann

La misma memoria contiene datos e instrucciones

Único bus  
datos/direcciones  
entre CPU y memoria

Von Neumann bottleneck!

### Harvard

Memorias separadas para datos e instrucciones

Dos buses  
datos/direcciones  
entre CPU y

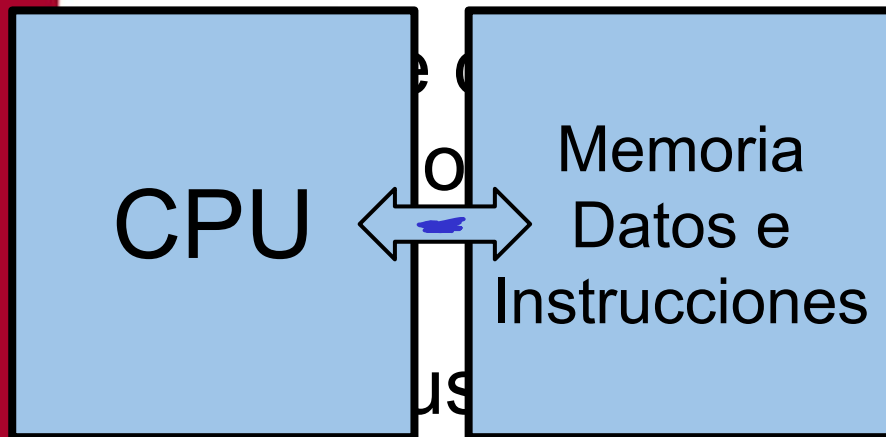
Mayor uso de recursos (\$)



## Von Neumann vs Harvard

### Von Neumann

La misma memoria

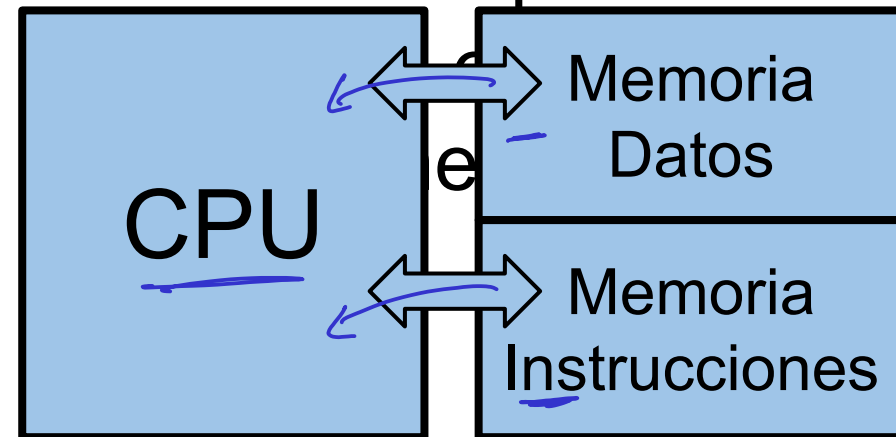


datos/direcciones  
entre CPU y memoria

Von Neumann bottleneck!

### Harvard

Memorias separadas



datos/direcciones  
entre CPU y

Mayor uso de recursos (\$)

## RISC vs CISC

ISA :  
Instruction Set  
Architecture

### Reduced Instruction Set Computer

Instrucciones compactas y  
uniformes

Facilita el pipelining

Mayor uso de memoria

Permiten mejores  
optimizaciones al  
compilador

opcode :  $b_3 \ b_2 \ b_1 \ b_0$

### Complex Instruction Set Computer

opcode ~ 16 bits



Instrucciones anchas

Múltiples modos de  
direccionamiento

Uso de memoria más  
eficiente (alta densidad de  
código)

Posible necesidad de  
optimizaciones manuales  
(sist. empotrados)

## Algunos ejemplos

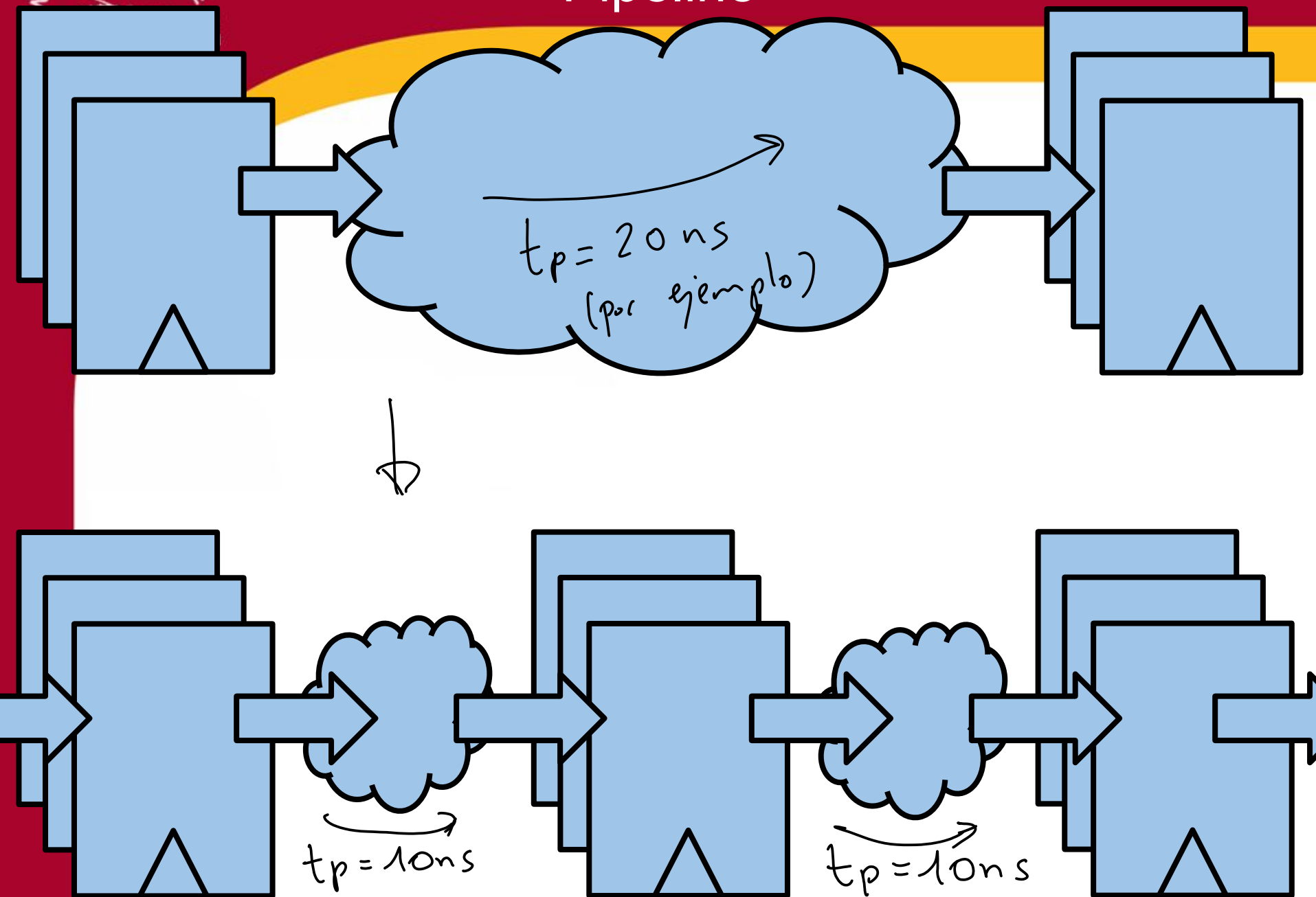
	Von Neumann	Harvard
<b>RISC</b>	<u>MSP430</u> ARM7	<u>MicroBlaze</u> Leon4 ARM9
<b>CISC</b>	<u>Pentium</u>	8051 SHARC <u>DSP</u> <u>DSPs en general</u>

Digital Signal Processor



## Contenido

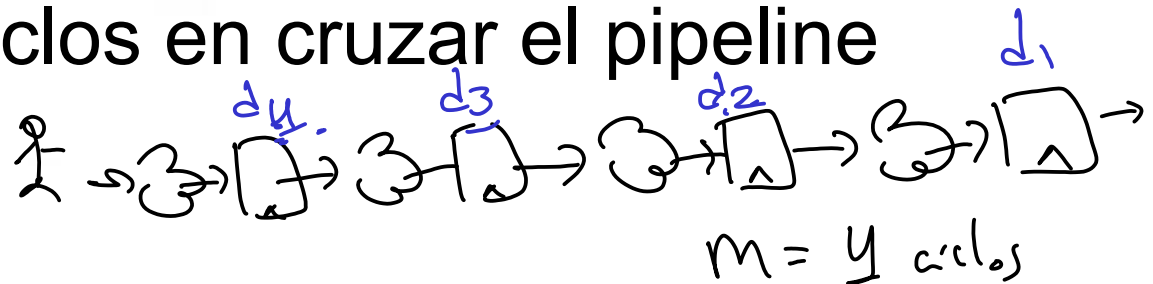
- Revisión de arquitecturas clásicas
- **Concepto de pipeline**
- Memorias caché



## Latencia y Throughput

**Latencia:** tiempo entre que entra el dato y sale procesado

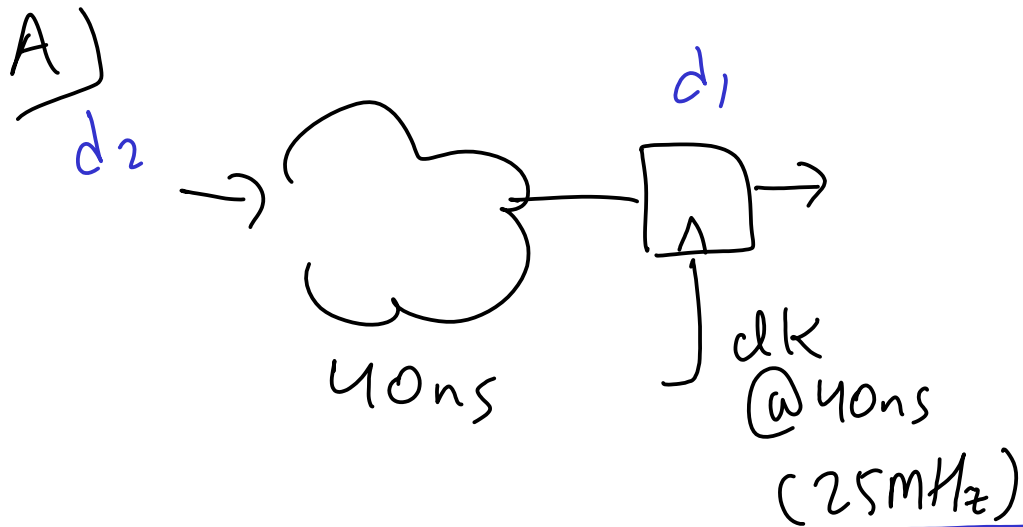
(tarda  $M$  ciclos en cruzar el pipeline completo)



**Throughput:** cada cuántos ciclos sale un dato nuevo

(un dato cada  $N$  ciclos)

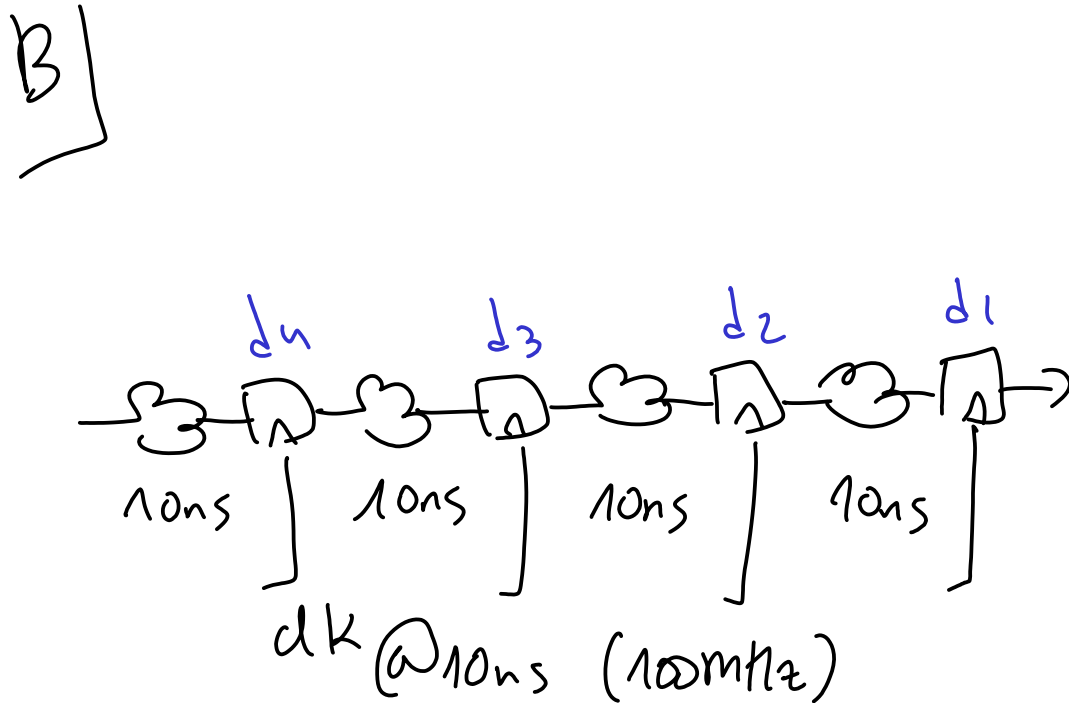
$$N = 1 \text{ ciclo}$$



Latencia = 1 ciclo  
( $40\text{ns}$ )

Throughput = 1 ciclo  
( $40\text{ns}$ )

100 datos en  $4000\text{ns}$

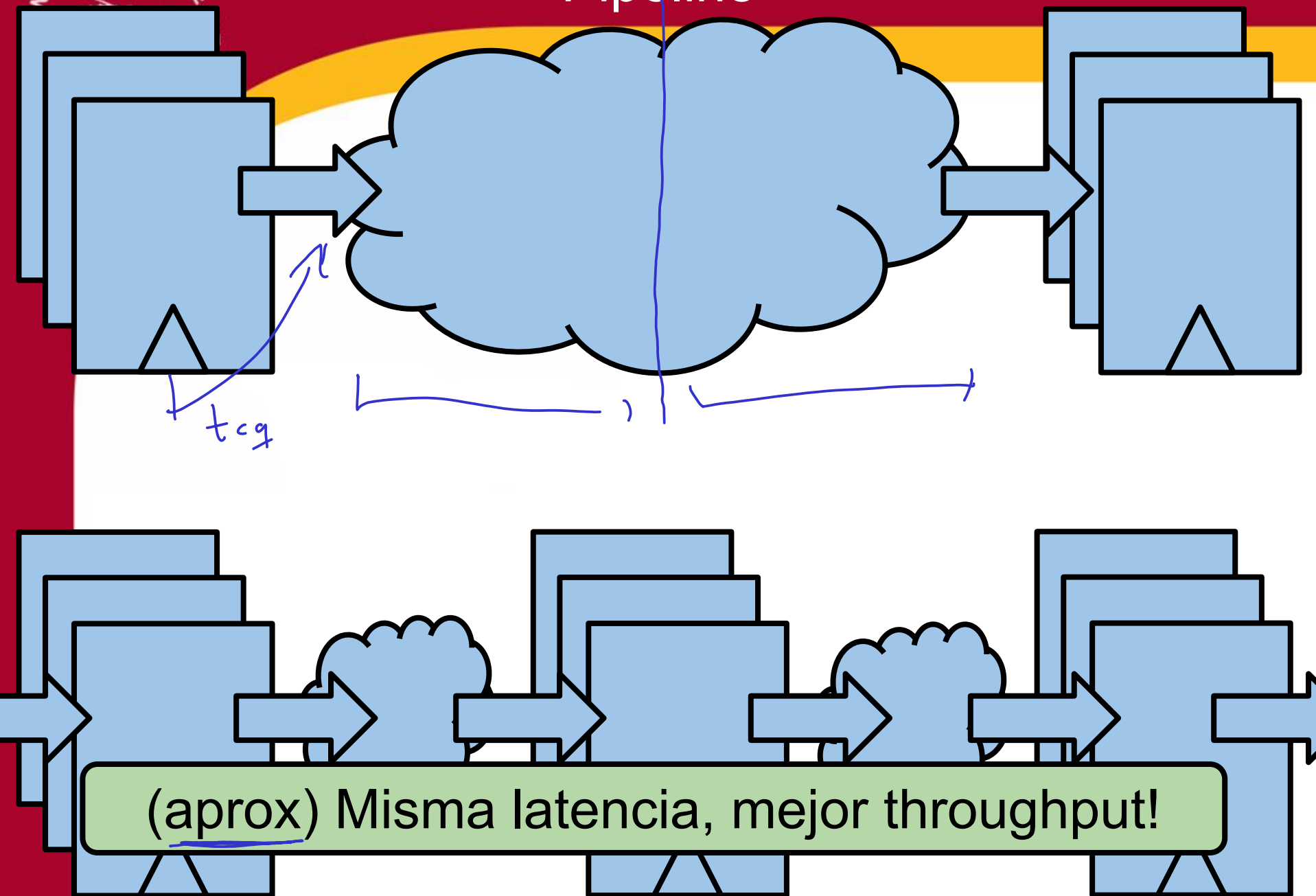


Latencia = 4 ciclos  
( $40\text{ns}$ )

Throughput = 1 ciclo  
( $10\text{ns}$ )

100 datos en  $\underline{40} + 99 \cdot 10 =$   
 $= 1030\text{ns}$

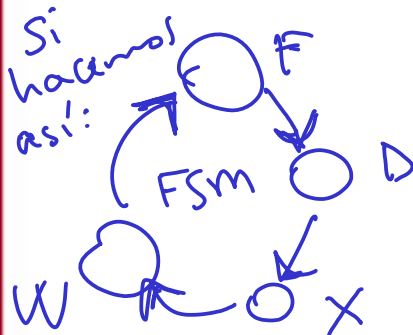
# Pipeline



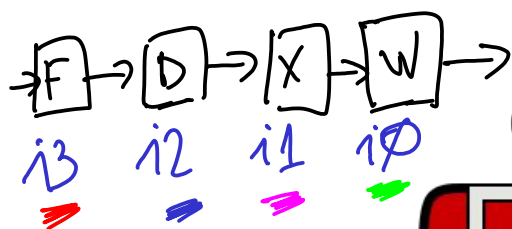
## Técnica muy utilizada en microprocesadores

Si para cada instrucción tenemos que hacer:

- Fetch leer instr. de la mem. de instrucciones
- Decode Decodificar la instrucción
- Execute Ejecutar la instrucción
- Write-back escribir el resultado



→ tardarías 4 ciclos por instrucción



Clock cycle

0 1 2 3 4 5 6 7 8

Waiting instructions

Pipeline

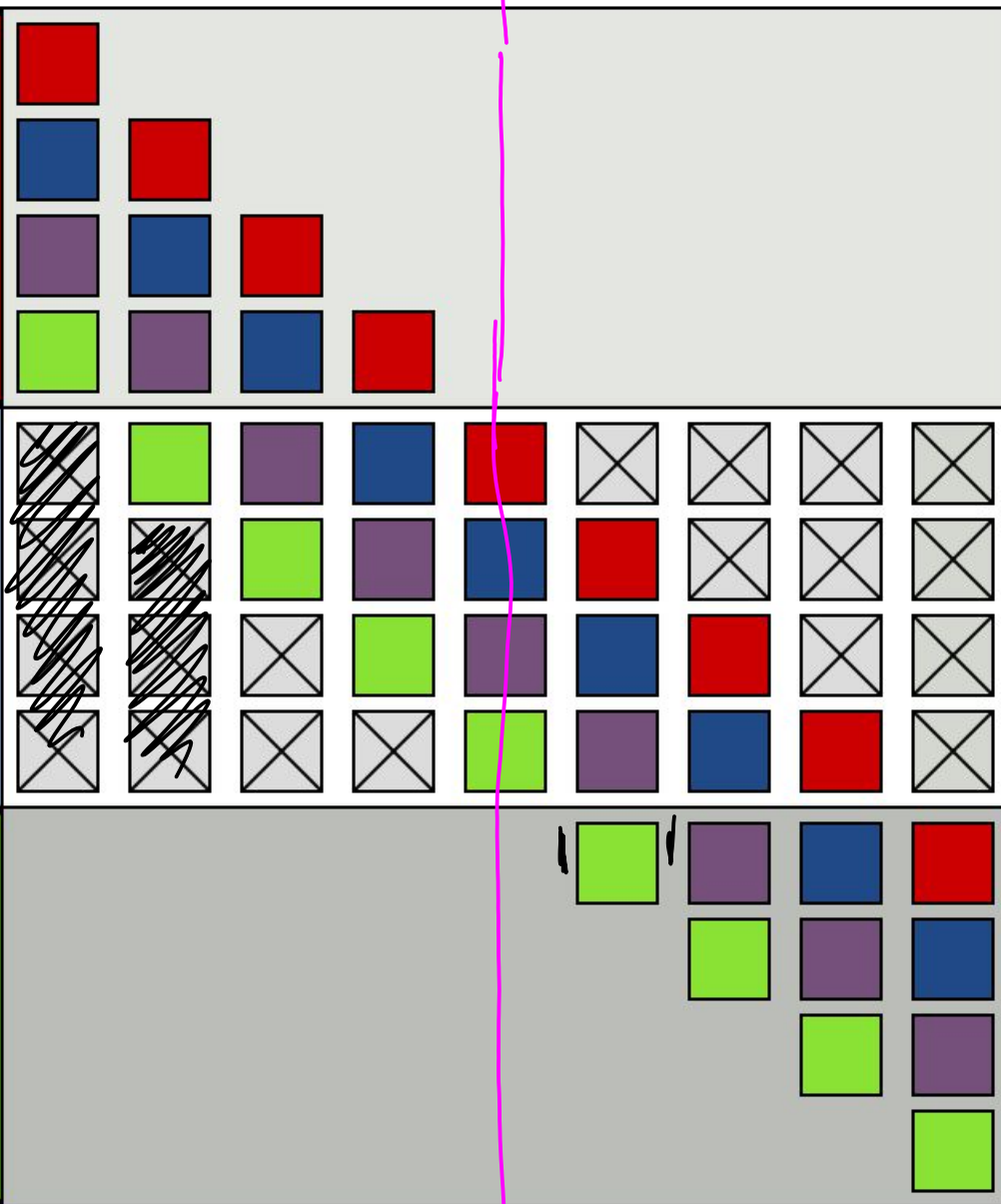
Stage 1: Fetch

Stage 2: Decode

Stage 3: Execute

Stage 4: Write-back

Completed instructions



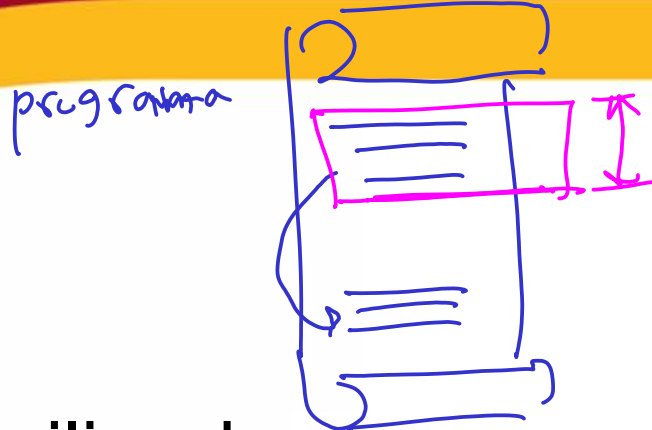
Si el pipeline está lleno,  
1 ciclo / instrucción



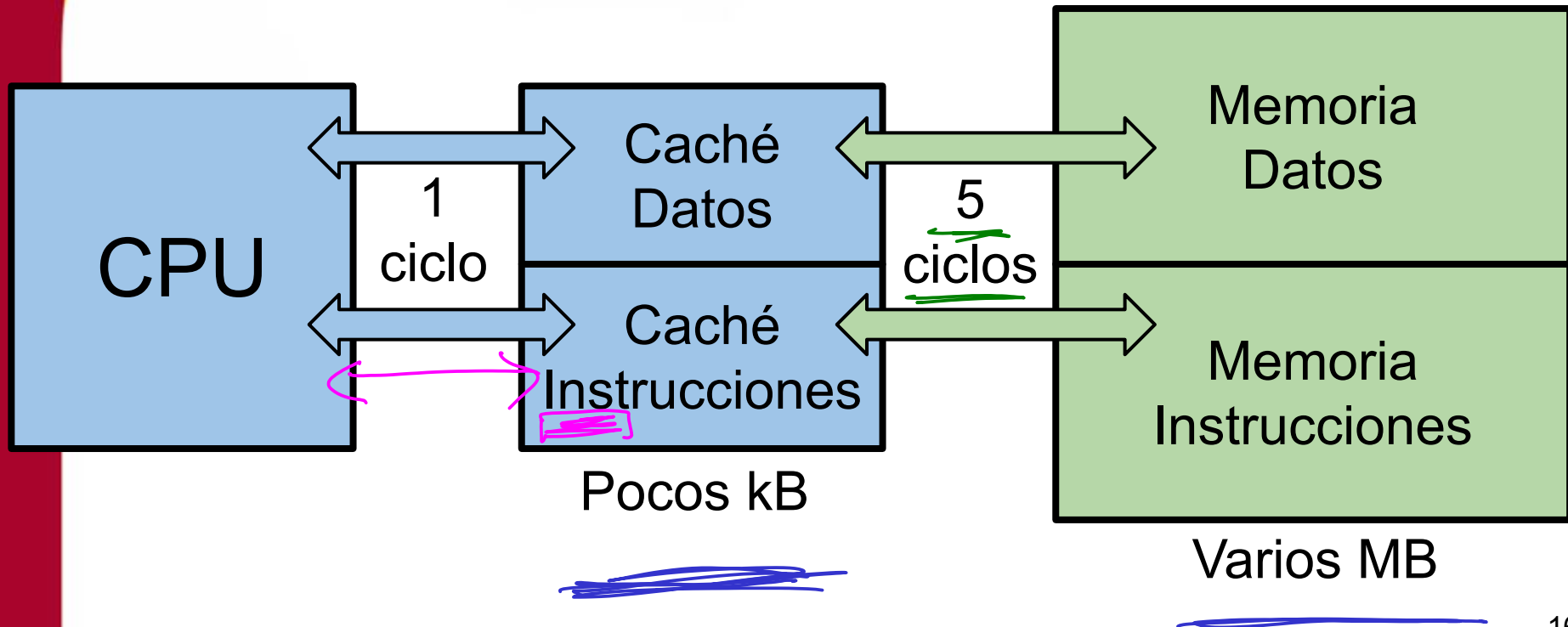
## Contenido

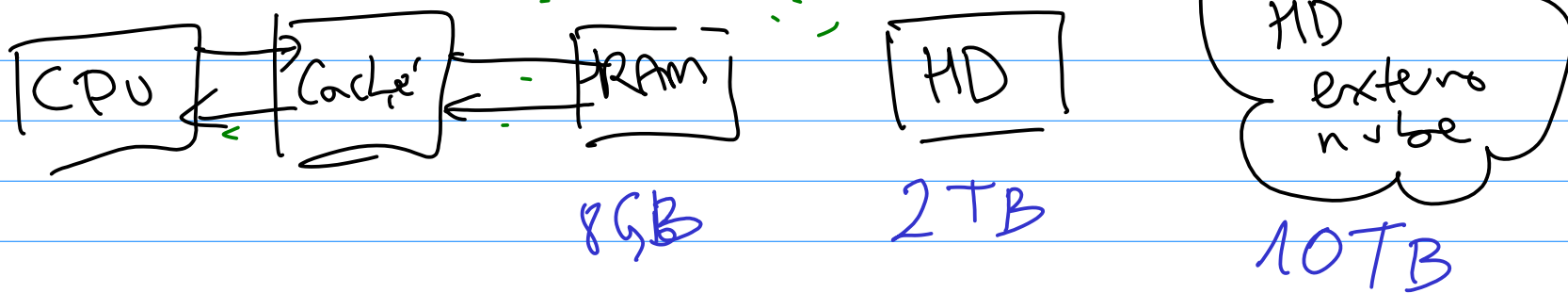
- Revisión de arquitecturas clásicas
- Concepto de pipeline
- **Memorias caché**

# Memorias caché



Memoria auxiliar de acceso más rápido





Tradeoff capacidad / velocidad

## Funcionamiento

La caché tiene menos capacidad que la memoria externa, pero es más rápida.

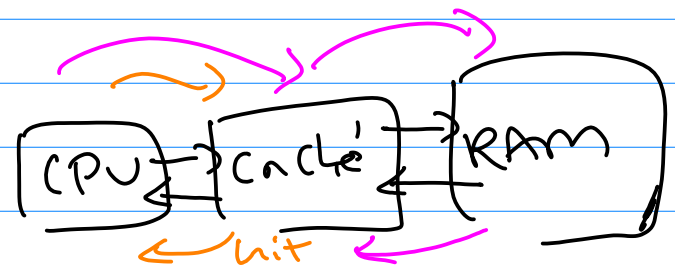
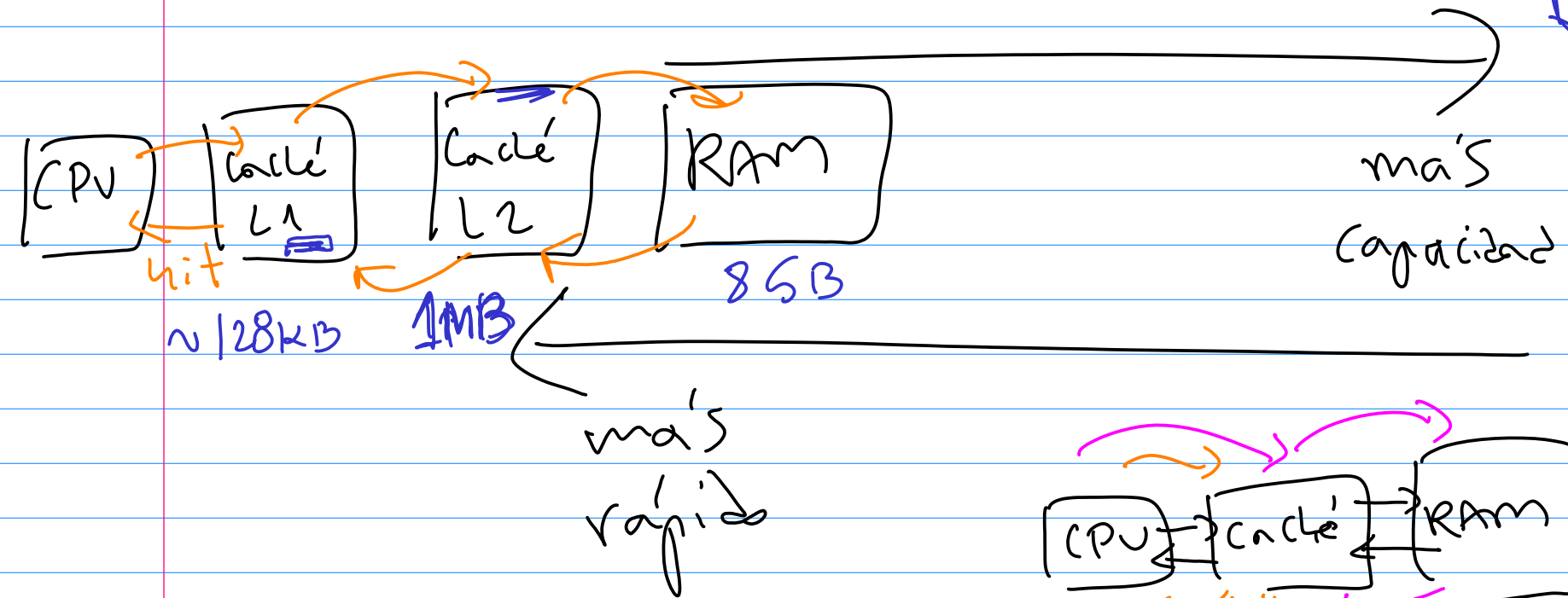
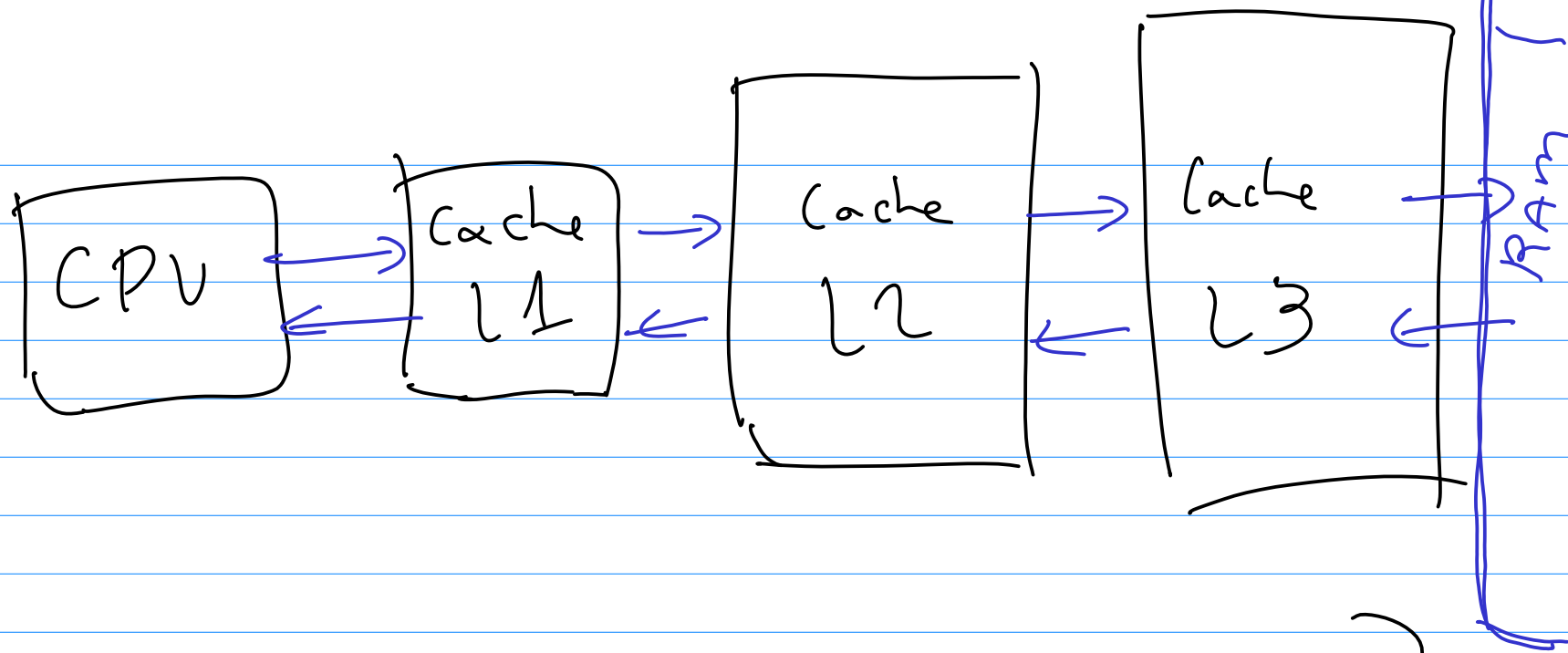
La caché guarda 'páginas' de memoria completas

Es 'transparente' para el procesador: el micro habla con la caché y la caché lee la memoria externa si necesita datos

## Funcionamiento

**Cache hit:** la dirección (address) solicitada por el microprocesador está en alguna de las páginas almacenadas en la caché (más rápido).

**Cache miss:** la dirección solicitada no está almacenada en la caché. La caché debe realizar el acceso externo (más tiempo).



NP

Cache'

morado  
naranja

0x00FFB000  
0x00FFB0FD

