Memoria:
caches y memoria virtual

Arquitectura de Computadores – IIC2343

2021-2 Yadran Eterovic

Las CPUs siempre han sido más rápidas que las memorias

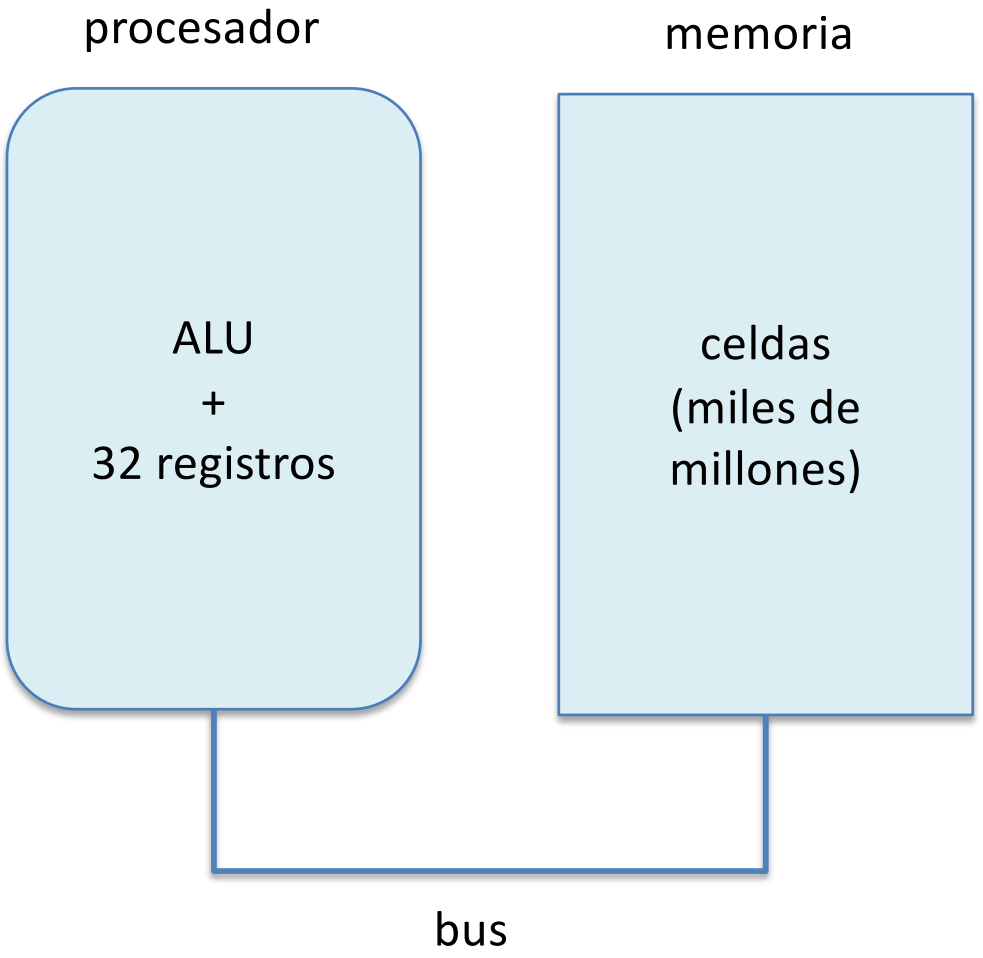
… y la diferencia sólo ha ido aumentando con el tiempo

Parte de la solución es colocar la memoria en el chip de la CPU:

- ya que el acceso a la memoria a través del bus es muy lento

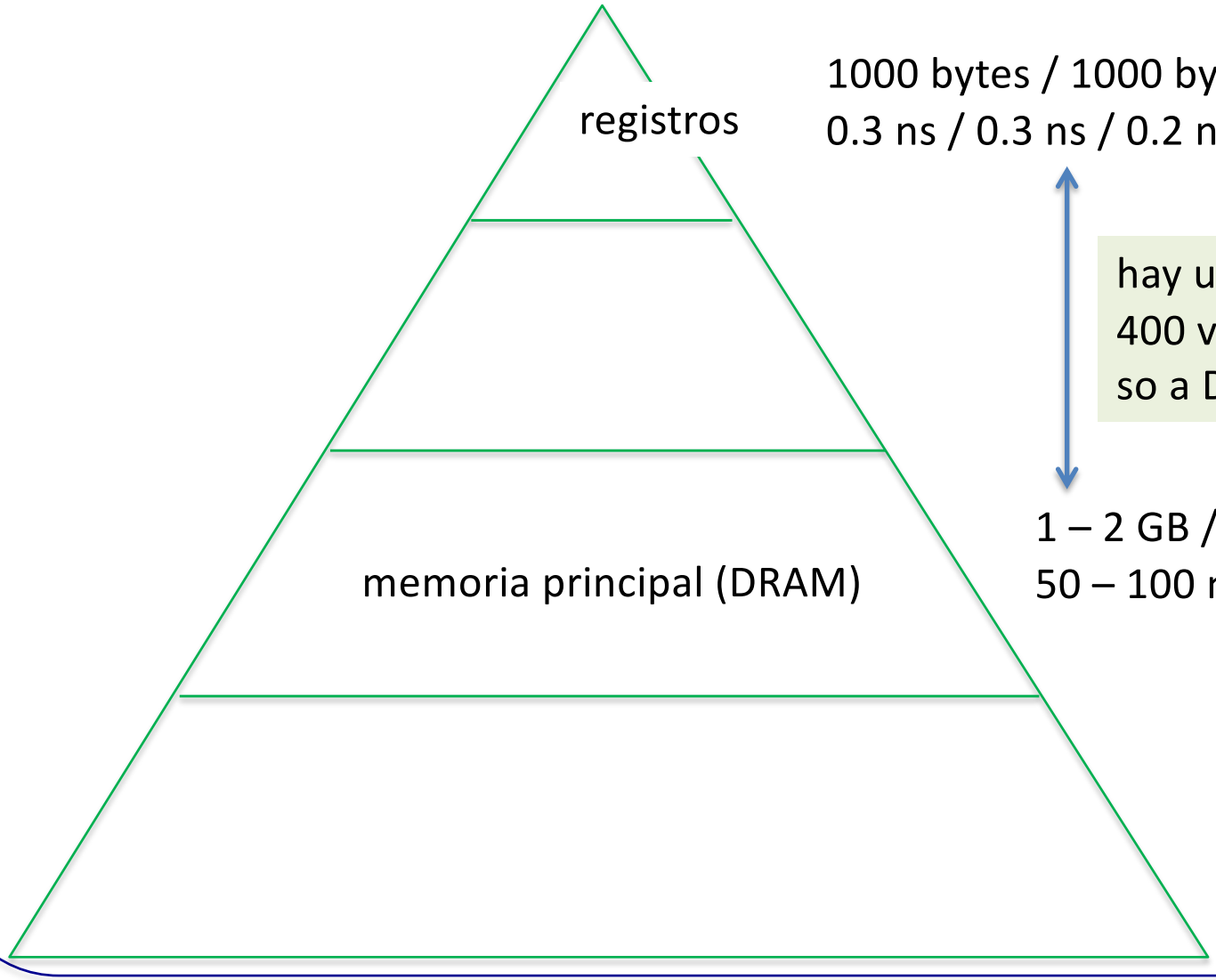
… pero esto significa CPUs más grandes:

- hay límites económicos y prácticos para el tamaño del chip de la CPU



Tamaños y tiempos de acceso típicos (≈ 2019) para los registros y la memoria principal en tres tipos de computadores:

teléfono celular / laptop / servidor



1000 bytes / 1000 bytes / 4000 bytes
0.3 ns / 0.3 ns / 0.2 ns

hay un factor de entre 200 y 400 veces en tiempos de acceso a DRAM frente a registros

1 – 2 GB / 4 – 16 GB / 32 – 256 GB
50 – 100 ns (tres casos)

Los programadores siempre hemos querido tener cantidades ilimitadas de memoria rápida

Pero la elección pareciera ser entre poca memoria rápida

… o mucha memoria lenta

Excepto que hay técnicas para combinarlas de modo de obtener **simultáneamente** la velocidad de una memoria rápida

… y la capacidad de una memoria grande

… a un precio razonable:

- estudiaremos cómo crear la ilusión de tener cantidades ilimitadas de memoria rápida

El **principio de localidad** subyace a la forma en que funcionan los programas en la práctica:

> **los programas accesan una porción relativamente pequeña de su espacio de direcciones* en un momento cualquiera del tiempo**
>
> … es decir, un programa no accesa todo su código o todos sus datos al mismo tiempo con igual probabilidad

**espacio de direcciones*: todas las direcciones de memoria, tanto de las instrucciones del programa como de los datos, a las que el programa podría hacer referencia durante su ejecución

Esta localidad se da naturalmente en los programas:

- *loops*, cuyas instrucciones y datos son usados repetidamente —localidad temporal
- ejecución secuencial de instrucciones —localidad espacial
- accesos secuenciales a los elementos de un arreglo —localidad espacial
- ver ejs. en las próximas diaps.

Como vamos a ver, esto hace posible que la mayoría de los accesos a memoria sean rápidos

… y al mismo tiempo tengamos una memoria grande

Localidad temporal

Si se hace referencia a una instrucción o un dato, entonces probablemente se hará referencia a esa misma instrucción o dato pronto

Dirección	Label	Instrucción/Dato
	CODE:	
0x00	start:	MOV CL, [var1]
0x01	while:	MOV AL,[res]
0x02		ADD AL,[**var2**]
0x03		MOV [res],AL
0x04		SUB CL,1
0x05		CMP CL,0
0x06		JNE while
	DATA:	
0x07	var1	3
0x08	**var2**	**2**
0x09	res	0

Localidad espacial

Si se hace referencia a un ítem, probablemente pronto se hará referencia a ítemes cuyas direcciones están (numéricamente) cerca

Dirección	Label	Instrucción/Dato
	CODE:	
0x00	start:	MOV SI, 0
0x01		MOV AX, 0
0x02		MOV BX, arreglo
0x03		MOV CL, [n]
0x04	while:	CMP SI, CX
0x05		JGE end
0x06		MOV DX, **[BX + SI]**
0x07		ADD AL, DL
0x08		INC SI
0x09		JMP while
0x0A	end:	DIV CL
0x0B		MOV [prom], AL
	DATA:	
0x0C	**arreglo**	**6**
0x0D		**7**
0x0E		**4**
0x0F		**5**
0x10		**3**
0x11	n	5
0x12	prom	0

Aprovechamos este principio de localidad para implementar una **jerarquía de memorias**:

- múltiples niveles de memoria con diferentes velocidades y tamaños

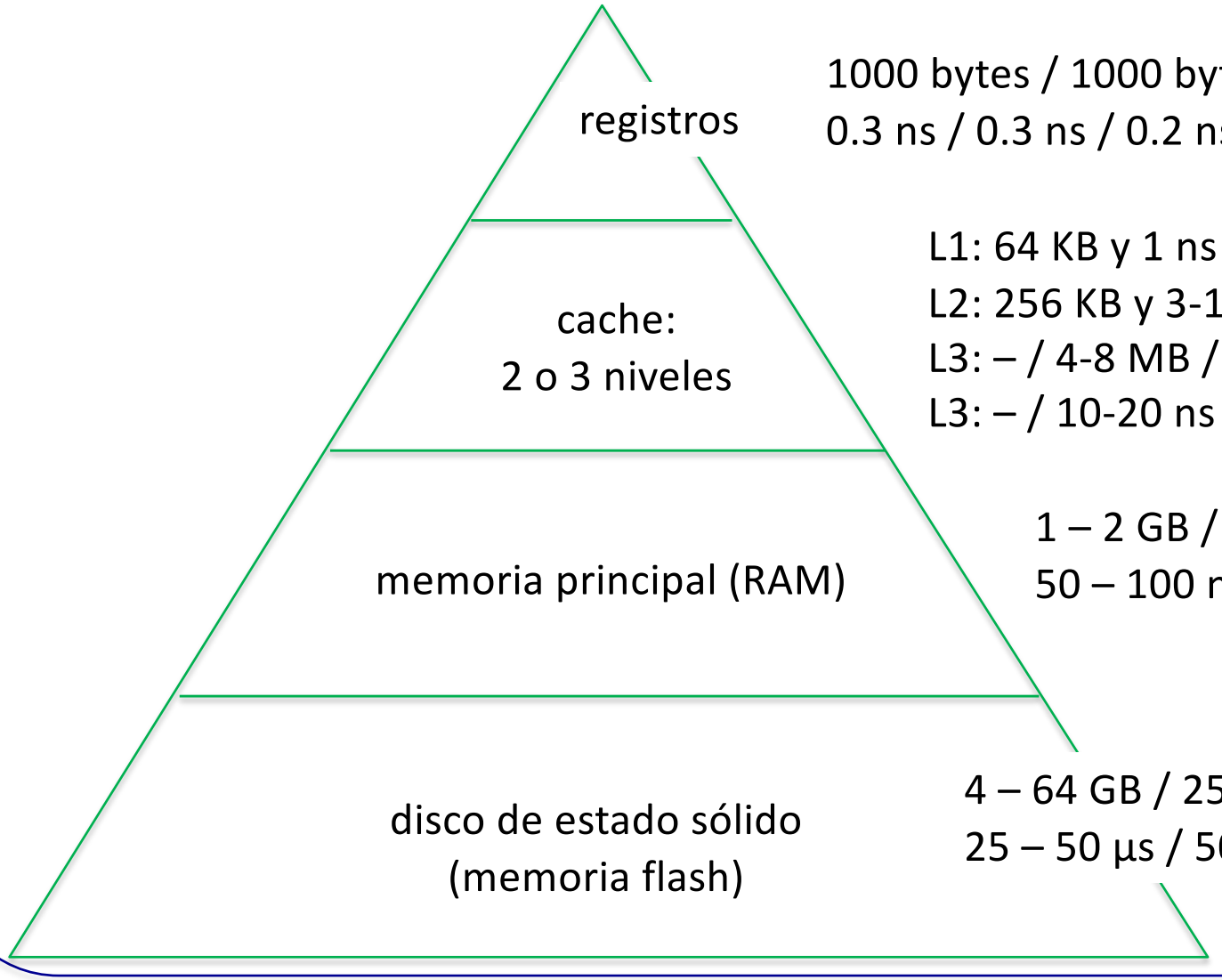
Las memorias más rápidas están más cerca del procesador, son más caras por bit que las memorias más lentas, y por lo tanto son más pequeñas

El propósito es ofrecer al programador tanta memoria como esté disponible en la tecnología más barata

… pero a la velocidad de acceso de la memoria más rápida

Tamaños y tiempos de acceso típicos (≈ 2019) para los cuatro niveles de la jerarquía en tres tipos de computadores:

teléfono celular / laptop / servidor



1000 bytes / 1000 bytes / 4000 bytes
0.3 ns / 0.3 ns / 0.2 ns

L1: 64 KB y 1 ns (tres casos)
L2: 256 KB y 3-10 ns (tres casos)
L3: – / 4-8 MB / 16-64 MB
L3: – / 10-20 ns / 10-20 ns

1 – 2 GB / 4 – 16 GB / 32 – 256 GB
50 – 100 ns (tres casos)

4 – 64 GB / 256 GB – 1 TB / 1 – 16 TB
25 – 50 µs / 50 – 100 µs / 100 – 200 µs

El contenido de las memorias también está jerarquizado (no sólo los tamaños y tiempos de acceso)

El contenido de un nivel más cerca del procesador **es un subconjunto** del contenido de cualquier nivel que está más lejos:

- … y el total del contenido necesario para ejecutar un programa —tanto las instrucciones como los datos— está en el nivel más lejano

Como vemos, a medida que nos alejamos del procesador, los accesos a los distintos niveles de memoria toman progresivamente más tiempo