

# 66.70 Estructura del Computador

## **Memoria**

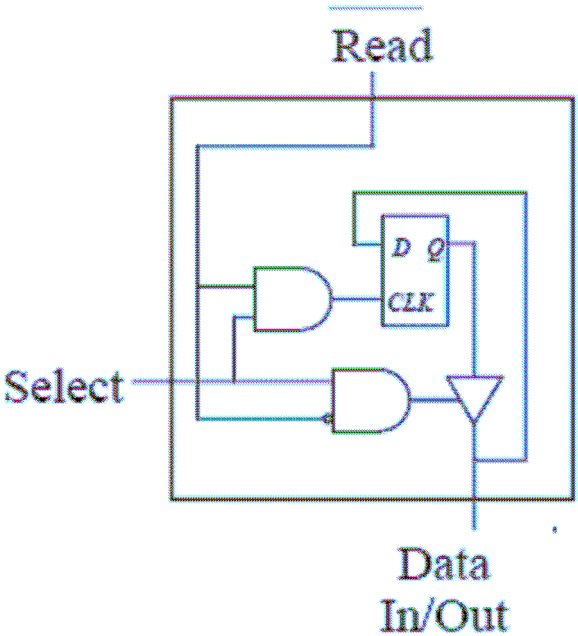
- Memoria volátil y no volátil
- RAM
- ROM
- Disco rígido
- Pendrive
- CD-ROM, DVD-ROM
- Cinta

# Clasificación por su modo de acceso

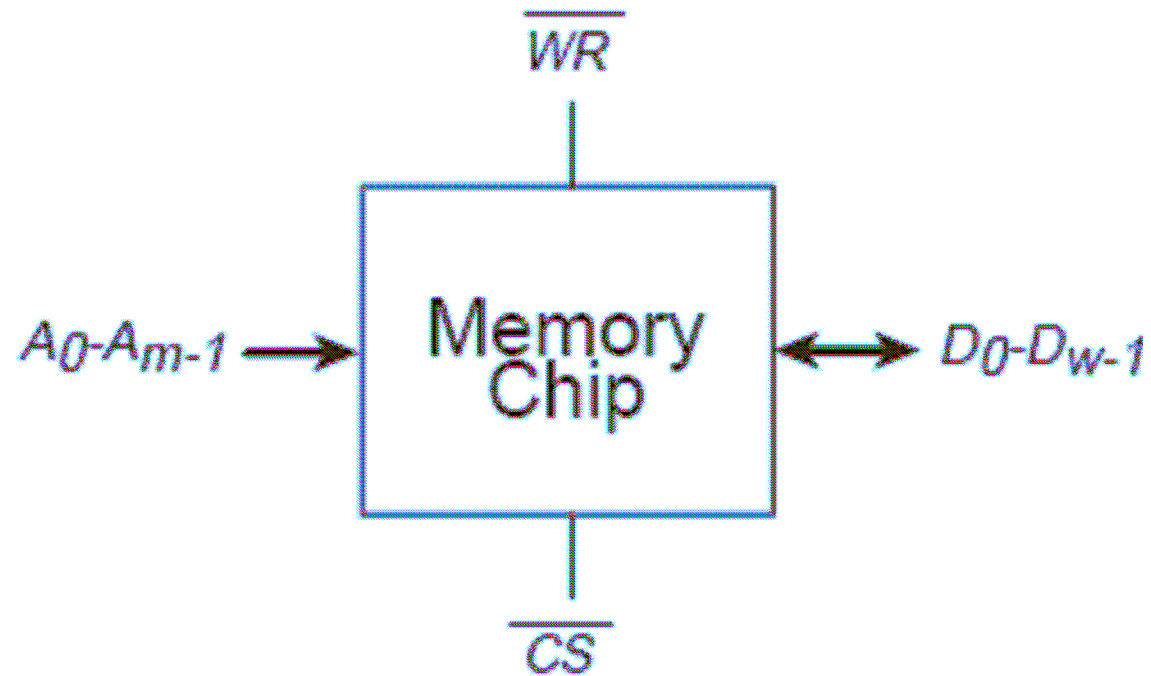
- RAM: M. de Acceso Aleatorio  
El tiempo y procedimiento para el acceso es independiente de la dirección accedida
  - Lectura/Escritura (RWM, en gral. referida como RAM)
  - ROM: Sólo Lectura
- CAM: M. Direccionable por Contenido (o Asociativa)
- SAM: M. de Acceso Secuencial  
Cinta, Reg. de desplazamiento serie-serie, Stack,
- DAM: M. de Acceso Directo o Semi-Aleatorio  
Disco rígido: acc. aleatorio a la pista y secuencial dentro de esta

# Memoria de lectura/escritura

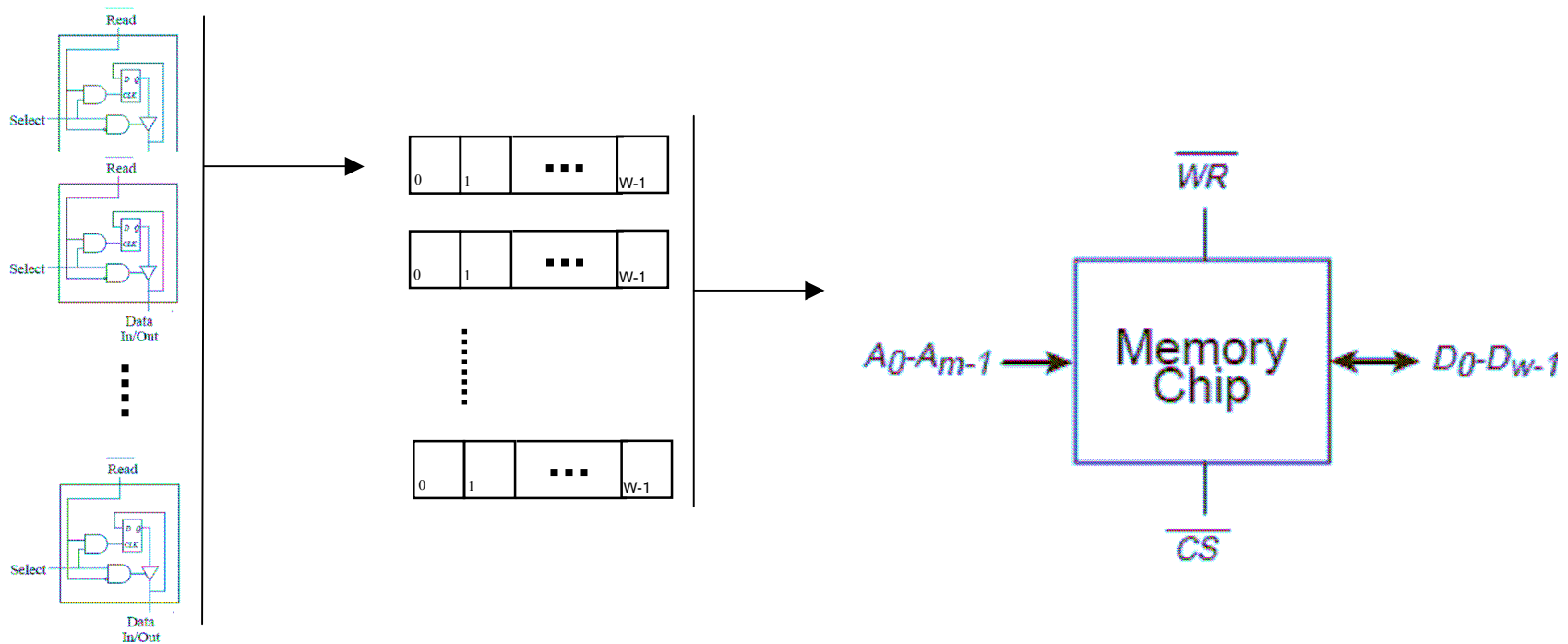
## Celda de 1 bit

	<p>→ Memoria Estática</p> <p>→ Memoria Dinámica</p> <p><i>lect./escri.</i></p> <p><i>refresh</i></p>
Bloque <b><u>funcional</u></b>	Formas de implementación

# Memoria de lectura/escritura de $2^m$ posiciones de $w$ bits



# Memoria de $2^m$ posiciones de $w$ bits a partir de celdas de 1 bit



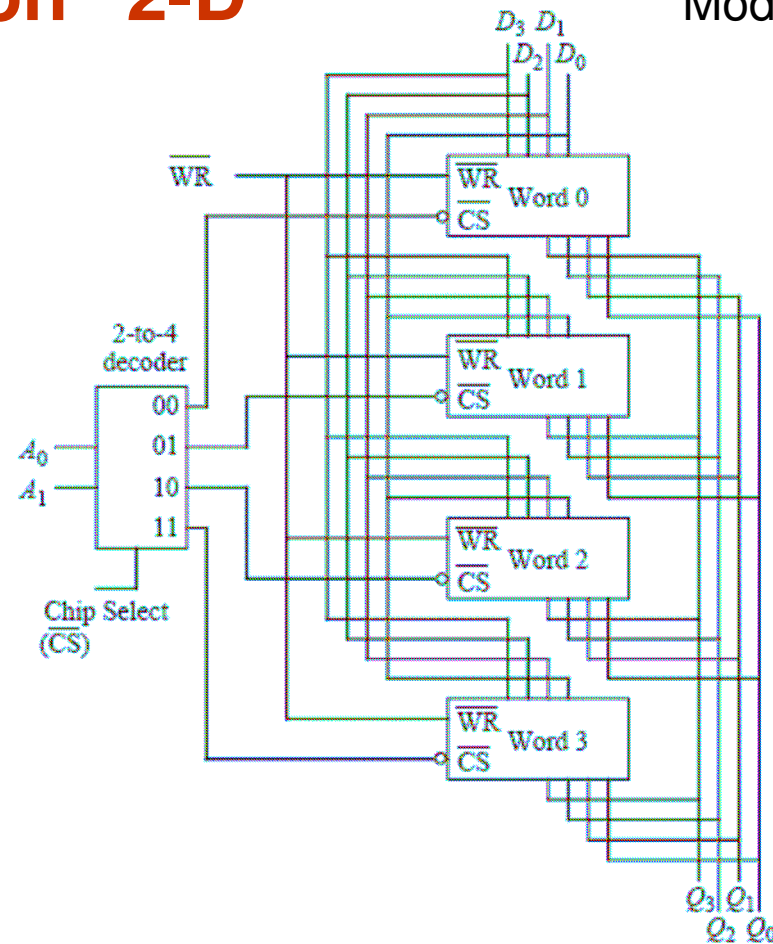
- Se ve como  $2^m$  registros
- Direcciona un registro por vez
- Selecciona lectura/escritura

Chip RAM

# Direccionando cada registro

## Organización “2-D”

Módulo de 4 posiciones de 4 bits

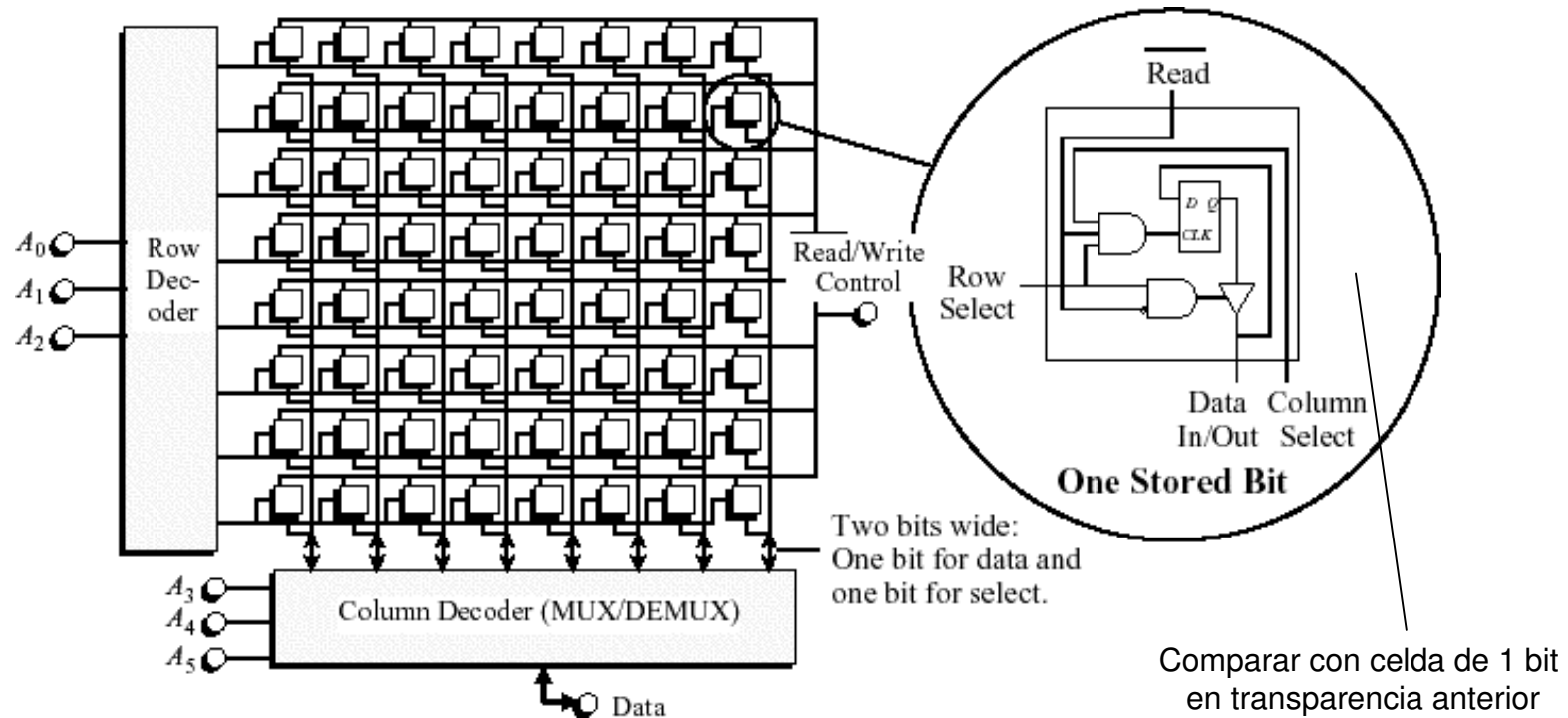


¿Qué limitaciones aparecen con este tipo de organización?  
p.e. con 128M x 8 bits

# Direccionando cada registro

## Organización “2-1/2 D”

Módulo de 64 posiciones de **1 bit**



- $n$  bits de address  $\Rightarrow n/2$  bits en filas y  $n/2$  bits en columnas
  - Direccionar 1 bit  $\Rightarrow$  (a) Seleccionar fila (b) Seleccionar columna
  - Sólo  $n/2$  bits de Address multiplexados en el tiempo y fila latcheada
- ✓ Menor número de pines  
✓ Menor tiempo de acceso de palabras sucesivas (decodifica sólo columna)



---

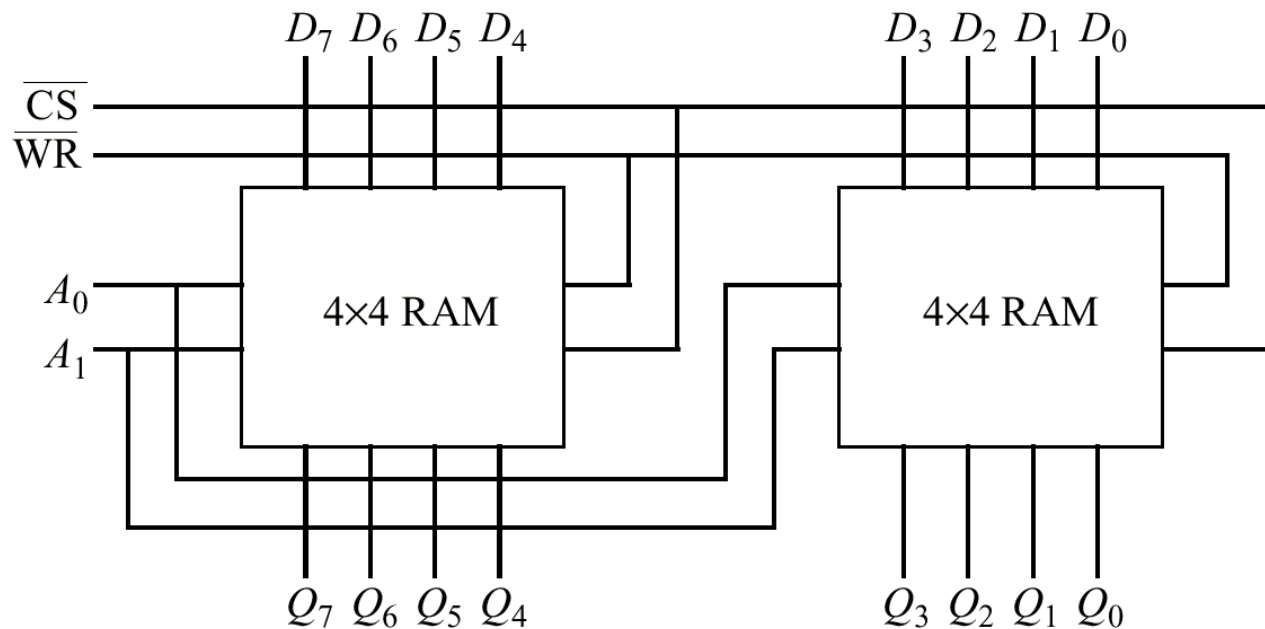
# Memorias “grandes” a partir de memorias “pequeñas”

## Casos:

- Cantidad de bits por palabra
  - Memoria de 256 Mbytes con 8 módulos de 256 Mbits
- Cantidad de palabras direccionables
  - Memoria 128 Mbits con 4 módulos de 32 Mbits
  - Memoria 512 Mbytes con 8 módulos de 64 Mbytes
- Ambos
  - Memoria de 256 Mbytes con 32 módulos de 64 Mbits

# Construcción de memoria de $4 \times 8 \text{ bits}$ con módulos $4 \times 4 \text{ bits}$

- Cantidad de bits por palabra



Los dos chips son simultáneamente:

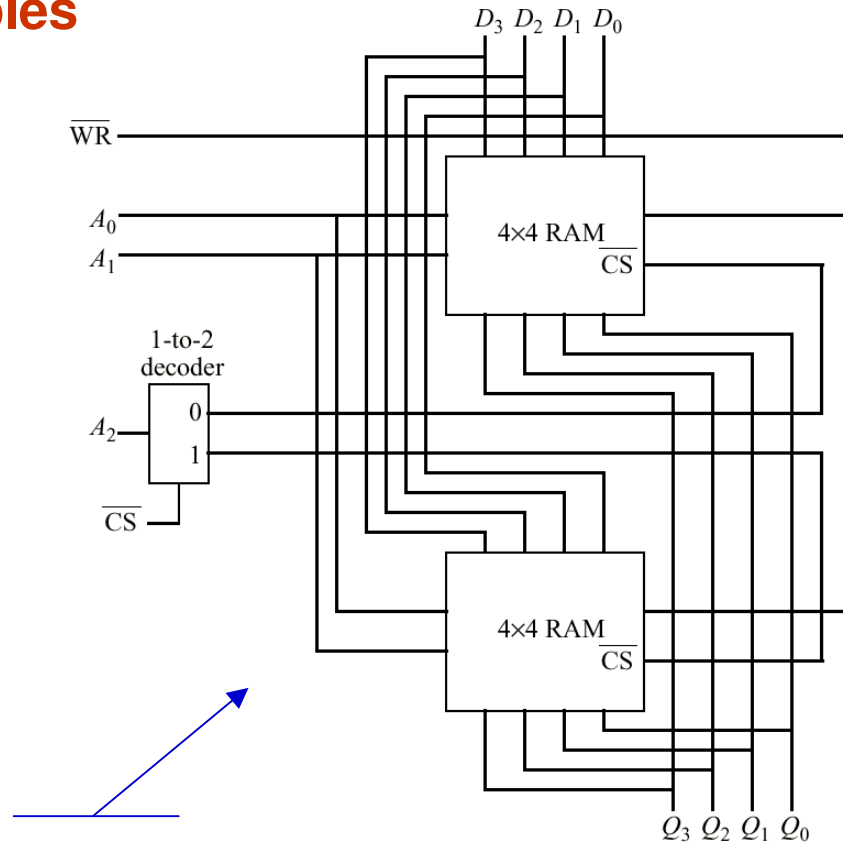
- Seleccionados con el Chip Select
- Direccionados internamente con las mismas líneas de Address

# Construcción de memorias de $8 \times 4 \text{ bits}$ a partir de módulos de $4 \times 4 \text{ bits}$

- **Cantidad de palabras direccionables**

- ✓ Bits adicionales para poder direccionar mayor rango de memoria
- ✓ Lógica combinacional para elegir el chip correspondiente

- ❖ **Decoders vs. Compuertas**



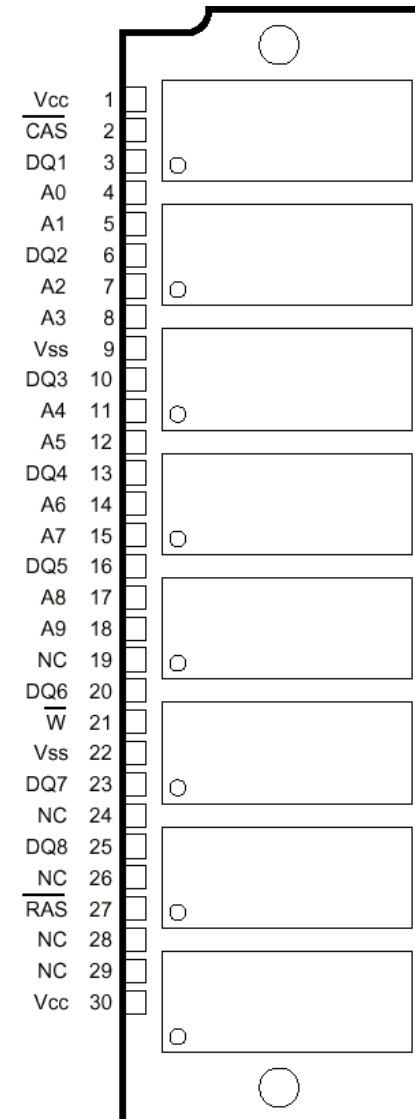
- Decoder selecciona uno u otro chip por medio del CS
- Los bits de direcc. menos significativos direccionan internamente el chip seleccionado

# Módulos de memoria comerciales

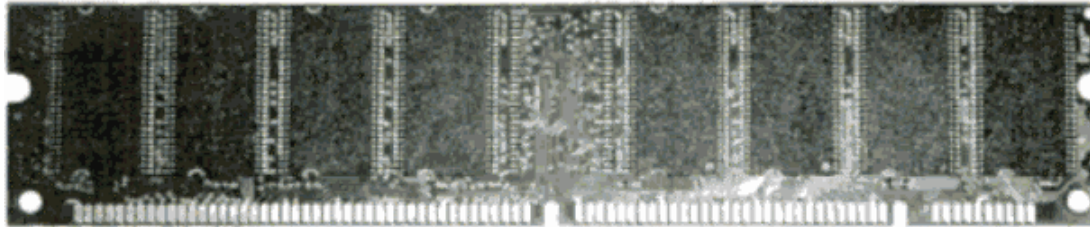
PIN NOMENCLATURE	
A0-A9	Address Inputs
$\overline{\text{CAS}}$	Column-Address Strobe
DQ1-DQ8	Data In/Data Out
NC	No Connection
$\overline{\text{RAS}}$	Row-Address Strobe
$V_{\text{CC}}$	5-V Supply
$V_{\text{SS}}$	Ground
$\overline{\text{W}}$	Write Enable

- ✓ 8 chips de 1 Mbit
  - ✓ DRAM
  - ✓ SIMM 30 pines
  - ✓  $V_{\text{CC}}$  3.3 - 2.5 - 1.8 V (menor  $V_{\text{CC}}$ , mayor velocidad-menor consumo)
- Señales RAS y CAS
  - La memoria no puede ser leída durante el refresh

¿Cómo direcciona cada posición de memoria?



# Módulos de memoria comerciales



- ✓ DRAM
- ✓ DIMM 168 pines
- ✓ 16 chips de 16 Mbytes
- ✓ Organizada en palabras de 64 bits
- ✓  $DQ_0 \dots DQ_{63}$ : 8 bytes leídos en paralelo
- ✓  $A_0 \dots A_{12}$ : direccionamiento
- ✓ WE: habilita escritura
- ✓ 16 pines con  $V_{SS}$  y 16 pines con  $V_{DD}$

PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
1	$V_{SS}$	43	$V_{SS}$	85	$V_{SS}$	127	$V_{SS}$
2	DQ0	44	OE2	86	DQ32	128	NC
3	DQ1	45	RAS2	87	DQ33	129	RAS3
4	DQ2	46	CAS2	88	DQ34	130	CAS6
5	DQ3	47	CAS3	89	DQ35	131	CAS7
6	$V_{DD}$	48	WE2	90	$V_{DD}$	132	NC
7	DQ4	49	$V_{DD}$	91	DQ36	133	$V_{DD}$
8	DQ5	50	NC	92	DQ37	134	NC
9	DQ6	51	NC	93	DQ38	135	NC
10	DQ7	52	NC	94	DQ39	136	NC
11	DQ8	53	NC	95	DQ40	137	NC
12	$V_{SS}$	54	$V_{SS}$	96	$V_{SS}$	138	$V_{SS}$
13	DQ9	55	DQ16	97	DQ41	139	DQ48
14	DQ10	56	DQ17	98	DQ42	140	DQ49
15	DQ11	57	DQ18	99	DQ43	141	DQ50
16	DQ12	58	DQ19	100	DQ44	142	DQ51
17	DQ13	59	$V_{DD}$	101	DQ45	143	$V_{DD}$
18	$V_{DD}$	60	DQ20	102	$V_{DD}$	144	DQ52
19	DQ14	61	NC	103	DQ46	145	NC
20	DQ15	62	NC	104	DQ47	146	NC
21	NC	63	NC	105	NC	147	NC
22	NC	64	$V_{SS}$	106	NC	148	$V_{SS}$
23	$V_{SS}$	65	DQ21	107	$V_{SS}$	149	DQ53
24	NC	66	DQ22	108	NC	150	DQ54
25	NC	67	DQ23	109	NC	151	DQ55
26	$V_{DD}$	68	$V_{SS}$	110	$V_{DD}$	152	$V_{SS}$
27	WE0	69	DQ24	111	NC	153	DQ56
28	CAS0	70	DQ25	112	CAS4	154	DQ57
29	CAS1	71	DQ26	113	CAS5	155	DQ58
30	RAS0	72	DQ27	114	RAS1	156	DQ59
31	OE0	73	$V_{DD}$	115	NC	157	$V_{DD}$
32	$V_{SS}$	74	DQ28	116	$V_{SS}$	158	DQ60
33	A0	75	DQ29	117	A1	159	DQ61
34	A2	76	DQ30	118	A3	160	DQ62
35	A4	77	DQ31	119	A5	161	DQ63
36	A6	78	$V_{SS}$	120	A7	162	$V_{SS}$
37	A8	79	NC	121	A9	163	NC
38	A10	80	NC	122	A11	164	NC
39	A12	81	NC	123	NC	165	SA0
40	$V_{DD}$	82	SDA	124	$V_{DD}$	166	SA1
41	NC	83	SCL	125	NC	167	SA2
42	NC	84	$V_{DD}$	126	NC	168	$V_{DD}$

# Reduciendo los tiempos de acceso

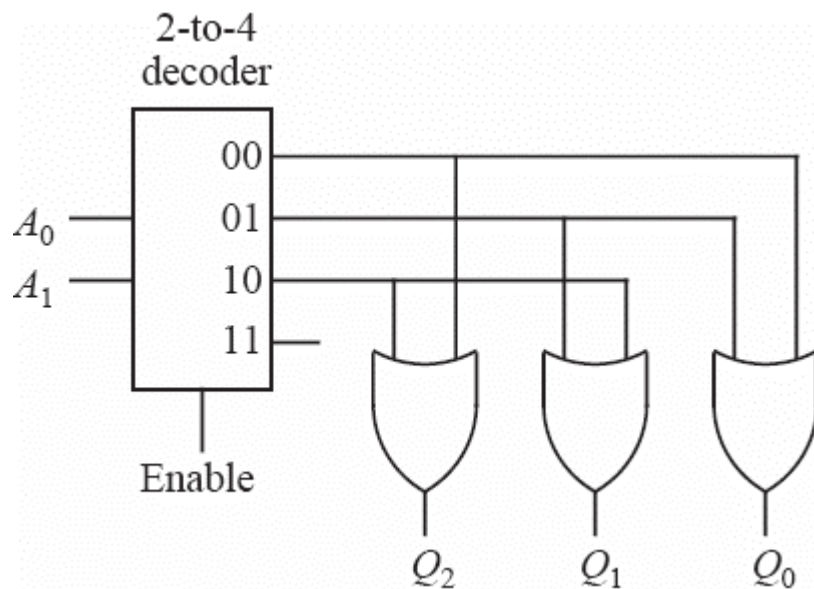
- ✓ Mayor velocidad del clock (*bus de memoria síncronico*)
  - Problemas a resolver:
    - Mayor consumo de potencia
    - Proclive a errores de almacenamiento de bits
- ✓ Mayor cantidad de bits leídos en paralelo (*ancho del bus*)
  - Problemas a resolver
    - Espacio ocupado por la mayor cantidad de pines
- ✓ Bancos entrelazados
  - Accede a un banco mientras en el otro se refresca la información
  - Enmascara el tiempo de refresco
  - Mejora rendimiento si pos. sucesivas están en bancos diferentes

---

La RAM dinámica es relativamente similar en todas las tecnologías, las principales diferencias están en su conexionado, direccionamiento y mejoras con circuitos adicionales on-chip

# Memoria de sólo-lectura (ROM)

- Sin Flip-Flops ni capacitores
- Es un circuito combinacional

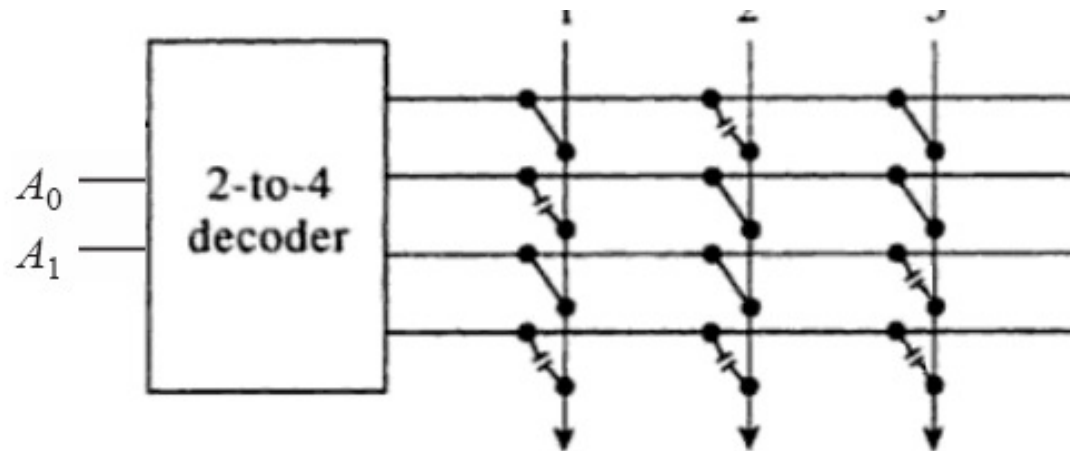


Location	Stored word
00	101
01	011
10	110
11	000

ROM de 4x3bits

# Memoria de sólo-lectura (ROM)



- Sin Flip-Flops ni capacitores
- Es un circuito combinacional



Location	Stored word
00	101
01	011
10	110
11	000

ROM de 4x3bits

Programable

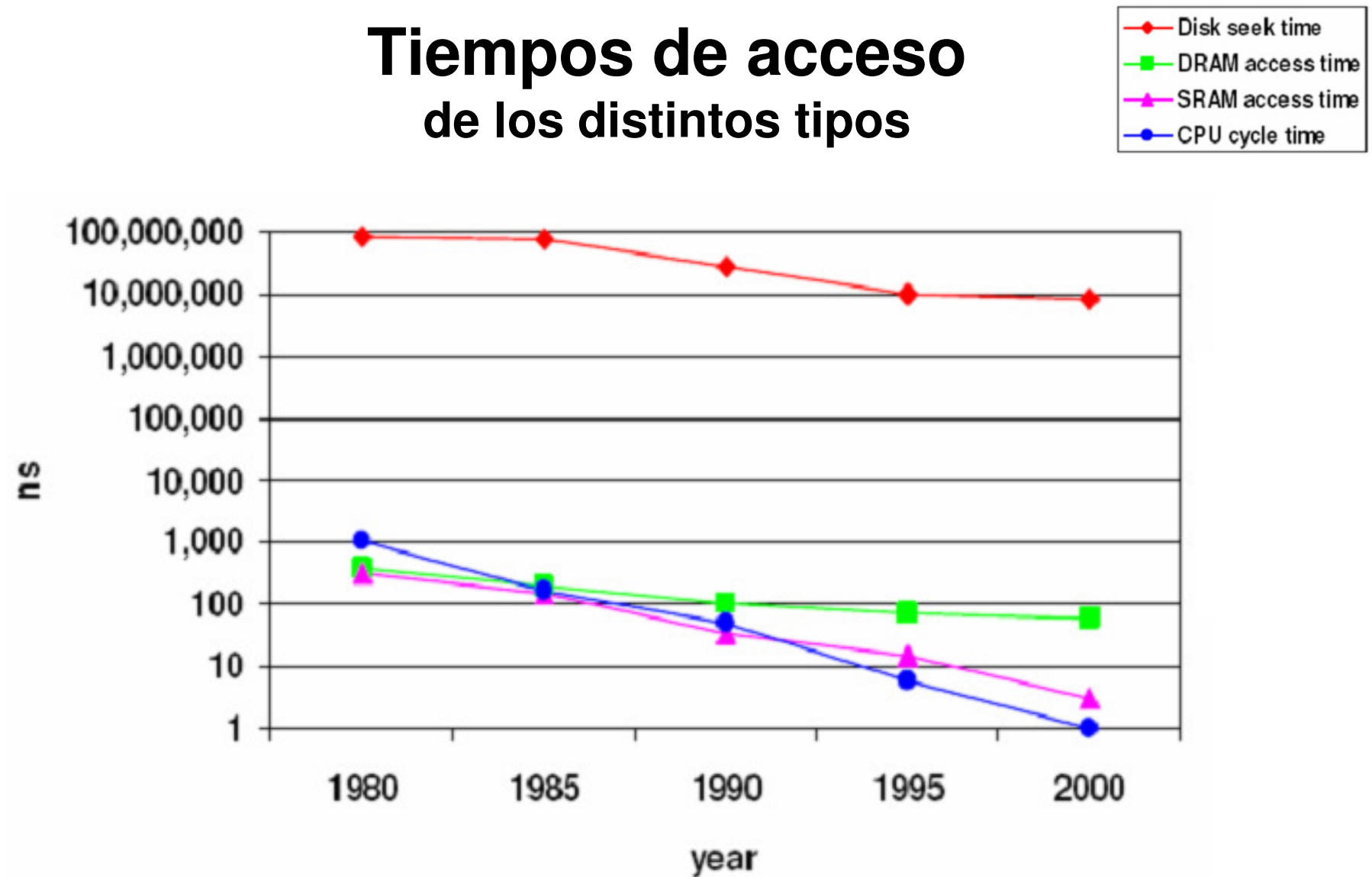
abierto   
corto 



# Tipos de memoria no volátil

- **ROM** (Grabadas en fabrica, sólo grandes cantidades)
- **PROM** (Programables por el usuario con dispositivo especial,  
sólo 1 escritura)
- **EPROM** (Regrabable, borrado por UV)
- **EEPROM** (Regrabable, borrado por potencial eléctrico)

## Tiempos de acceso de los distintos tipos



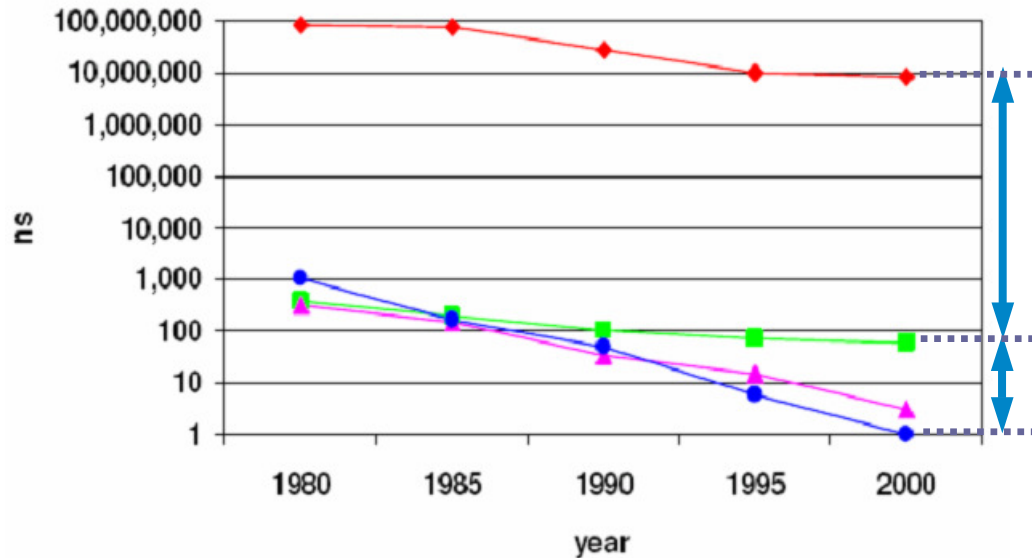
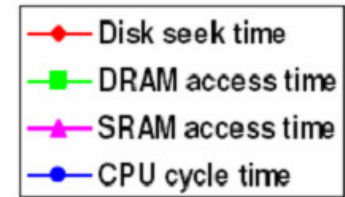
## Capacidad vs. Velocidad

- **Menor** tiempo de acceso => **mayor** costo por bit
- **Mayor** capacidad => **menor** costo por bit
- **Mayor** capacidad => **mayor** tiempo de acceso

una computadora ...

...requiere de mucha memoria y de memoria muy rápida

# Tiempos de acceso



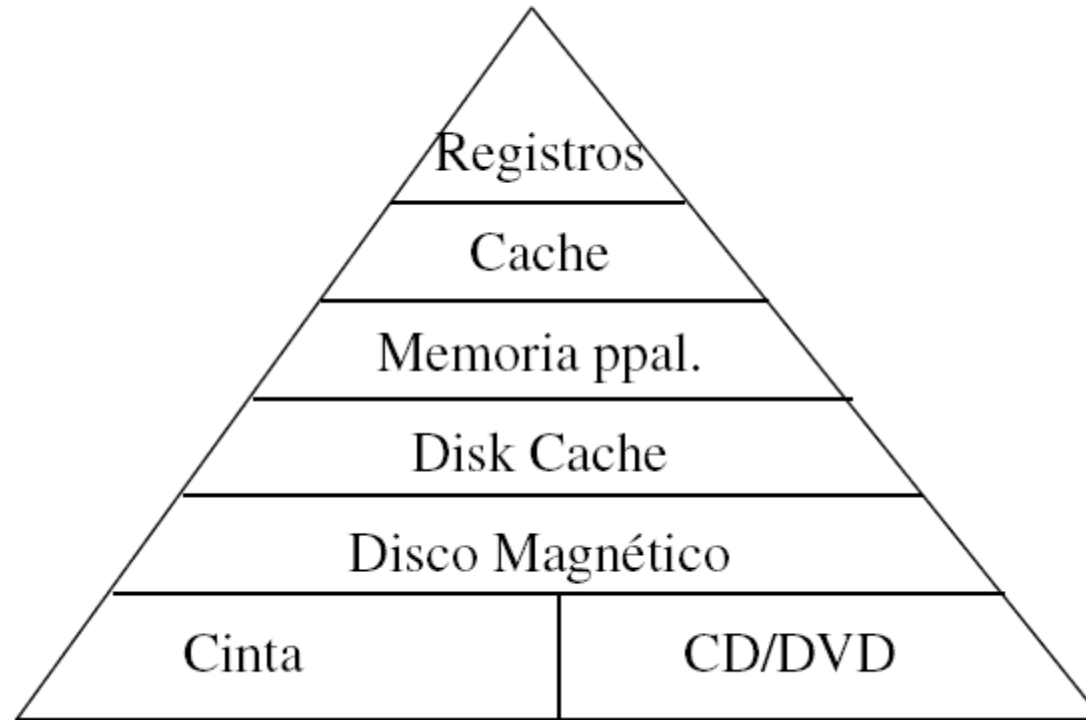
El S.O. busca compensar esta gran diferencia y usa el HD para aumentar la memoria disponible

~~El procesador debe esperar a la RAM~~

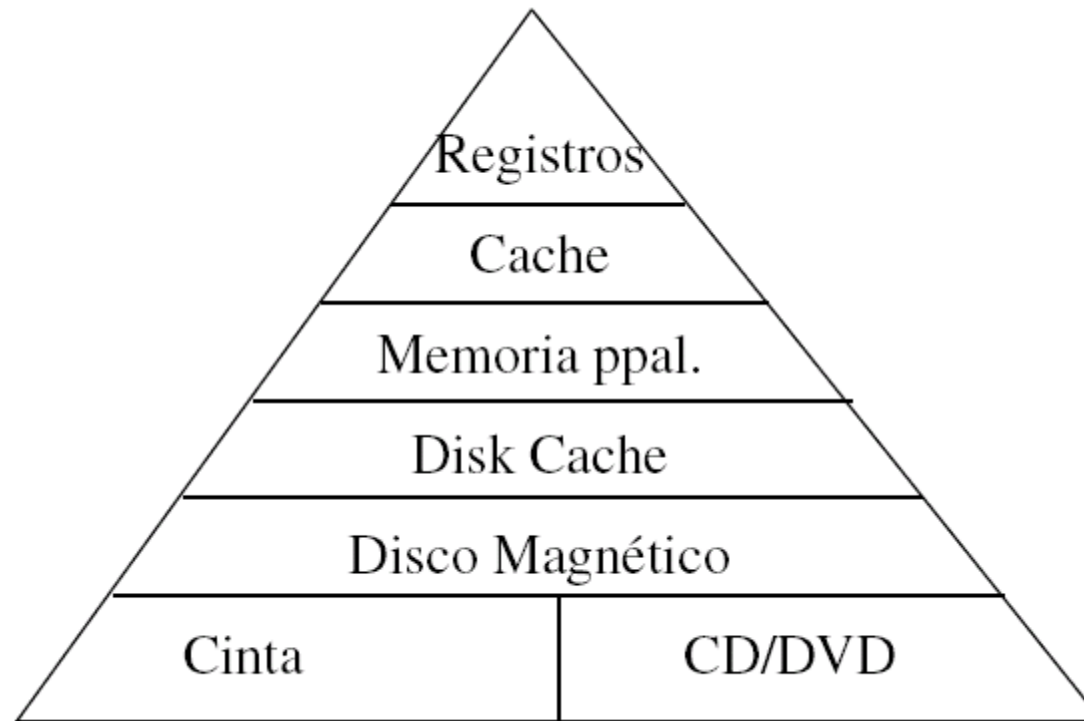
Organizar el funcionamiento de los distintos tipos de memoria (SRAM, DRAM, HD)

Rendimiento del sistema: como si tuviera mucha memoria y sólo memoria rápida

# Organización en jerarquías

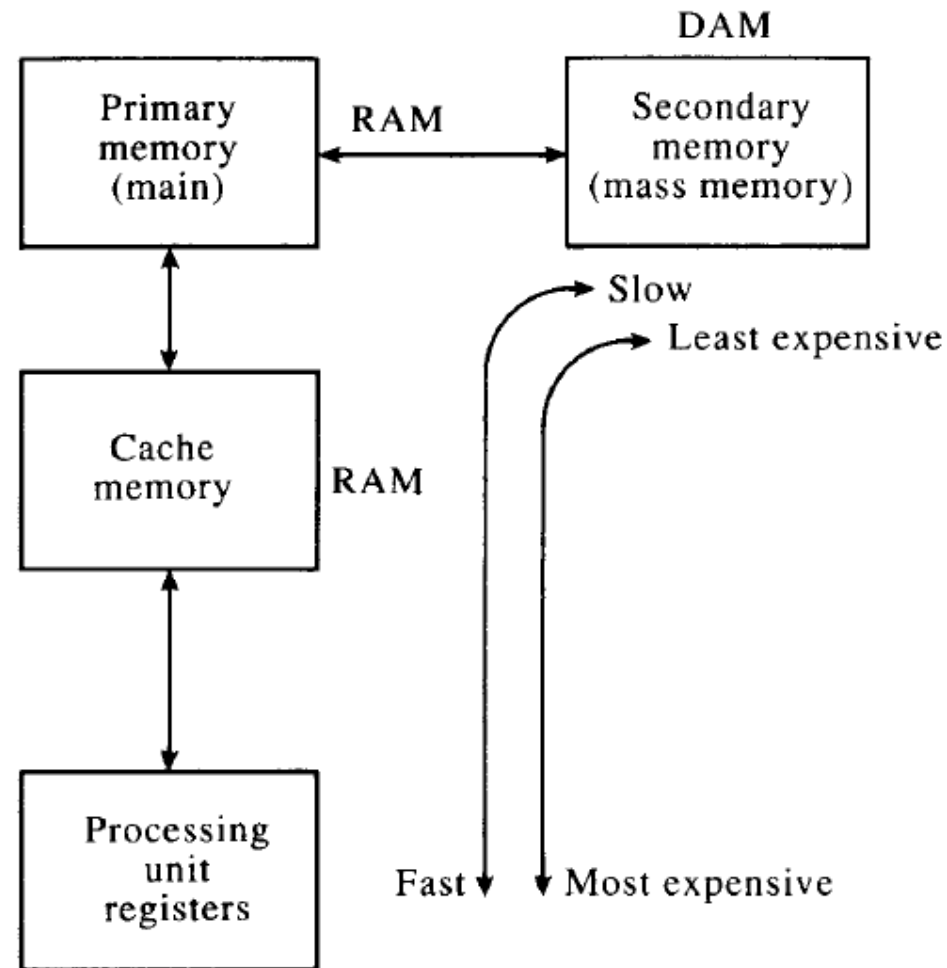


# Organización en jerarquías



Memory type	Access time	Cost/MB	Typical amount used	Typical cost
Registers	0.5 ns	High	2 KB	–
Cache	5–20 ns	\$80	2 MB	\$160
Main memory	40–80ns	\$0.40	512 MB	\$205
Disk memory	5 ms	\$0.005	40 GB	\$200

# Organización en jerarquías



---

# El porqué de la Memoria Cache

90% del tiempo de ejecución corresponde al 10% del código

## **“Principio de localidad”**

- Localidad temporal  
Si accedo a una dirección, en poco tiempo volveré a accederla
- Localidad espacial  
Si accedo a una dirección, las direcciones cercanas tienen mayor probabilidad de ser accedidas



---

## El porqué de la Memoria Cache

90% del tiempo de ejecución corresponde al 10% del código

### “Principio de localidad”

- Localidad temporal —————  
Si accedo a una dirección, en poco tiempo volveré a accederla
- Localidad espacial —————  
Si accedo a una dirección, las direcciones cercanas tienen mayor probabilidad de ser accedidas

Datos almacenados en posiciones contiguas

Iteraciones, procedimientos recursivos

# Memoria Cache

**Evita el cuello de botella producido por la marcada diferencia entre la velocidad del CPU y la velocidad de memoria principal**

**Físicamente el cache es:**

- Memoria muy rápida
- de poca capacidad
- “Cercana” al CPU

**Cómo funciona:**

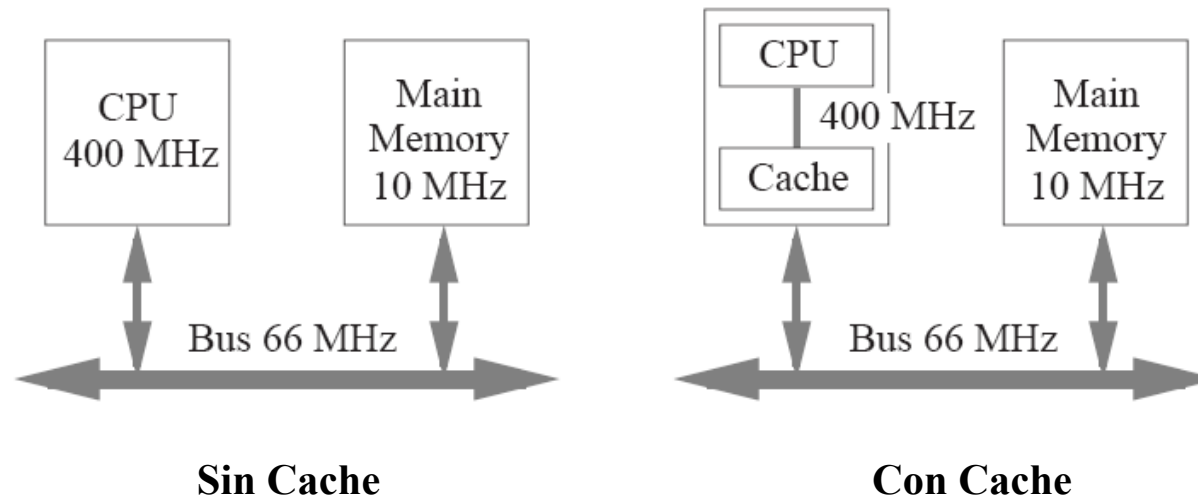
- Memoria dividida en bloques
- Al acceder un dato de mem. principal: bajo bloque completo al cache
- En el próximo acceso verifico si la posición buscada esta en cache,  
si no cargo otro bloque

**Agrega pasos al proceso de lectura/escritura pero:**

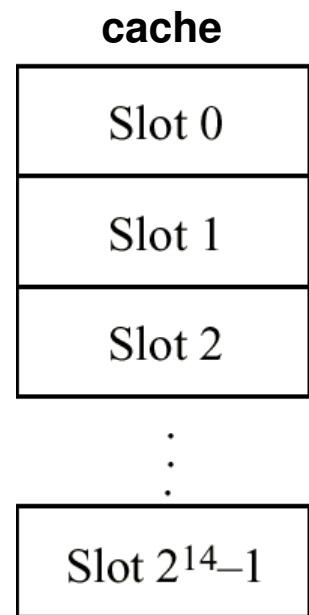
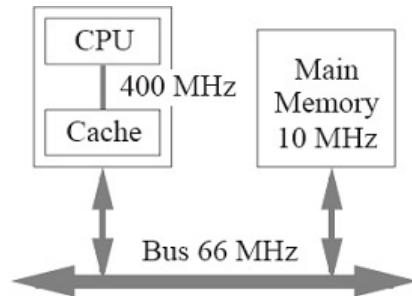
- *Cache + Principio de localidad => Aumenta el rendimiento*

# Memoria Cache

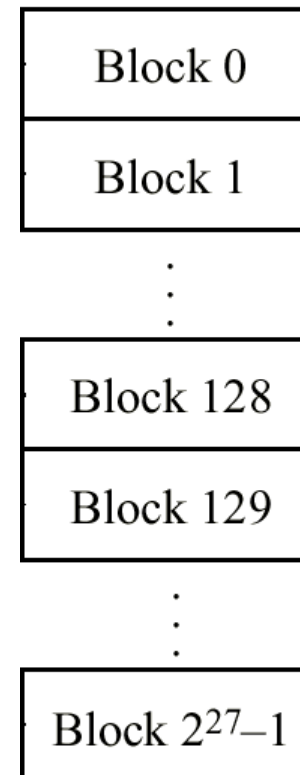
- ✓ El cache es invisible al programador
- ✓ Porqué la mem. cache es más rápida que la mem. princ.?
  - Construida con electrónica más rápida (SRAM)
    - es más cara, ocupa más espacio y disipa más potencia
    - pero es escasa
  - Por ser escasa su árbol de decodificación es pequeño
  - Su cercanía al CPU es física y lógica, **no** se comunica por un bus compartido



# Mapeo de Memoria Cache

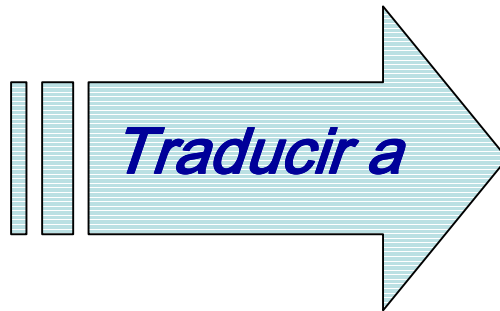


## Memoria principal



# Mapeo de Memoria Cache

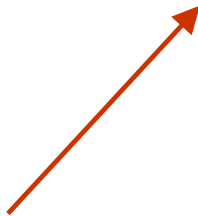
*Dirección  
del dato en  
memoria  
principal*



*Dirección dentro  
de la memoria  
cache*

- Mapeo asociativo
- Mapeo directo
- Mapeo asociativo por conjuntos

HARDWARE



# Midiendo la performance del cache

- Tasa de aciertos

$$\text{Hit ratio} = \frac{\text{No. times referenced words are in cache}}{\text{Total number of memory accesses}}$$

- Tiempo de acceso efectivo

$$\text{Eff. access time} = \frac{(\# \text{ hits})(\text{Time per hit}) + (\# \text{ misses})(\text{Time per miss})}{\text{Total number of memory access}}$$

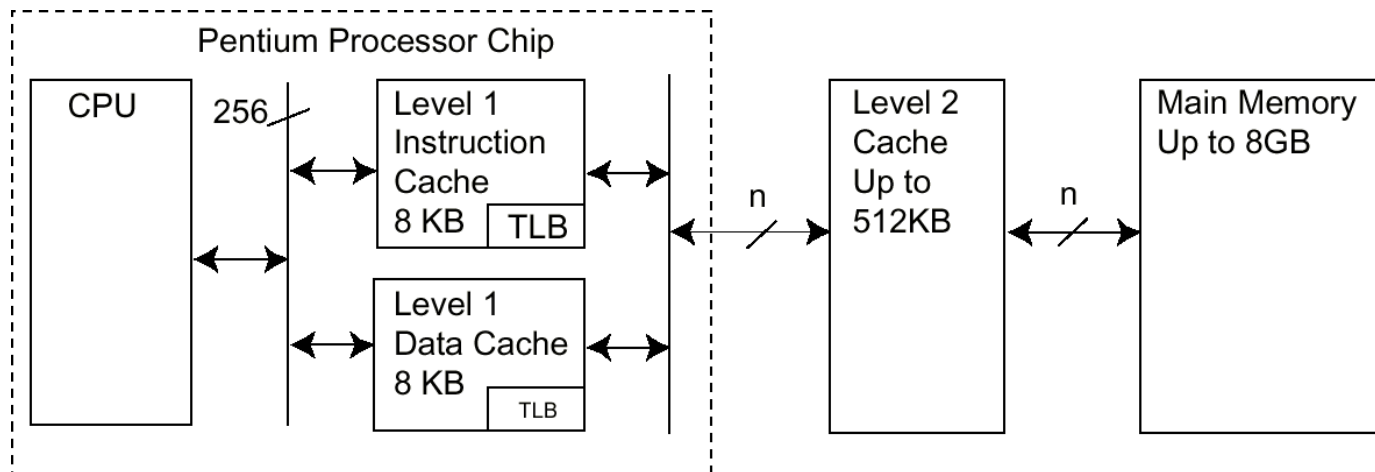
# Estructuras del Cache

## ✓ Cache especializado

- Cache de datos
- Cache de instrucciones

## ✓ Cache multinivel

- Cache más grandes son más **lentos**
  - Cache grandes, mayor **índ. de aciertos**
- | => varios niveles



# Historia del cache en procesadores Intel

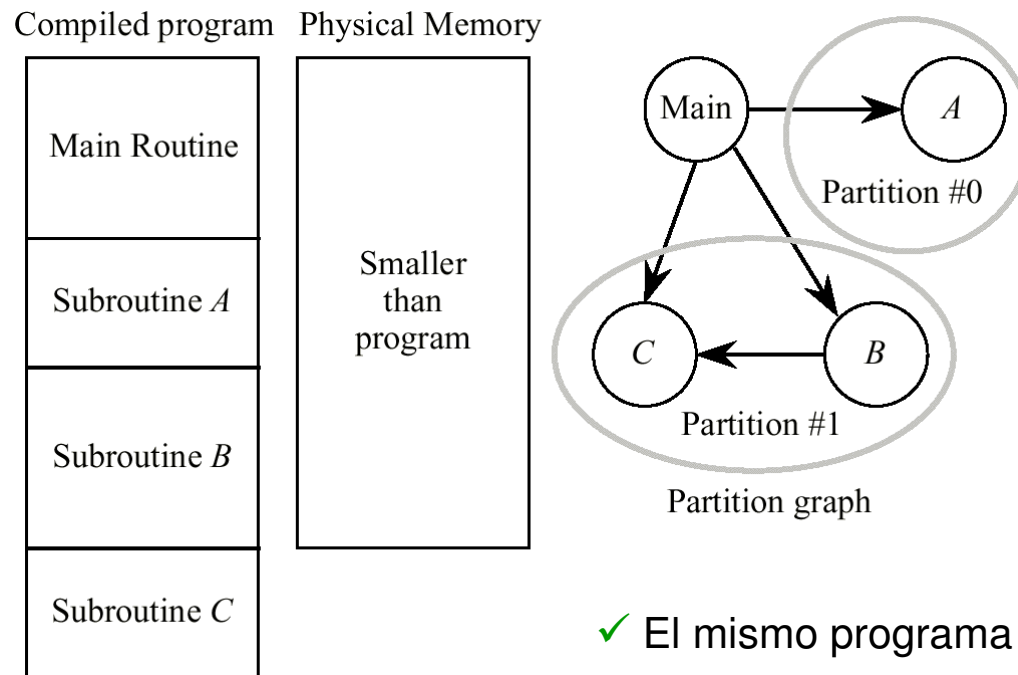
- **80386** – sin caché on-chip.
- **80486** – 8kB de caché unificada. Bloques de 16 bytes, asociativa de cuatro vías.
- **Pentium** – 2 cachés on-chip. 8kB para datos y 8kB para instrucciones.
- **Pentium II** – caché L2
- **Pentium 4**  
Caché L1: 8kBytes (4k+4k), bloques de 64 bytes, asociativa de 4 vías.  
Caché L2 256kB, líneas de 128 Bytes, asociativa de 8 vías.  
Caché L3
- **Core 2 duo**  
Cache L1 (32k + 32 k) L2 6 Mb
- **Core i7**  
Cache L1 (32k + 32 k) L2 256k L3 8Mb



# Memoria virtual

*“Cuando la RAM no alcanza...disco rígido”*

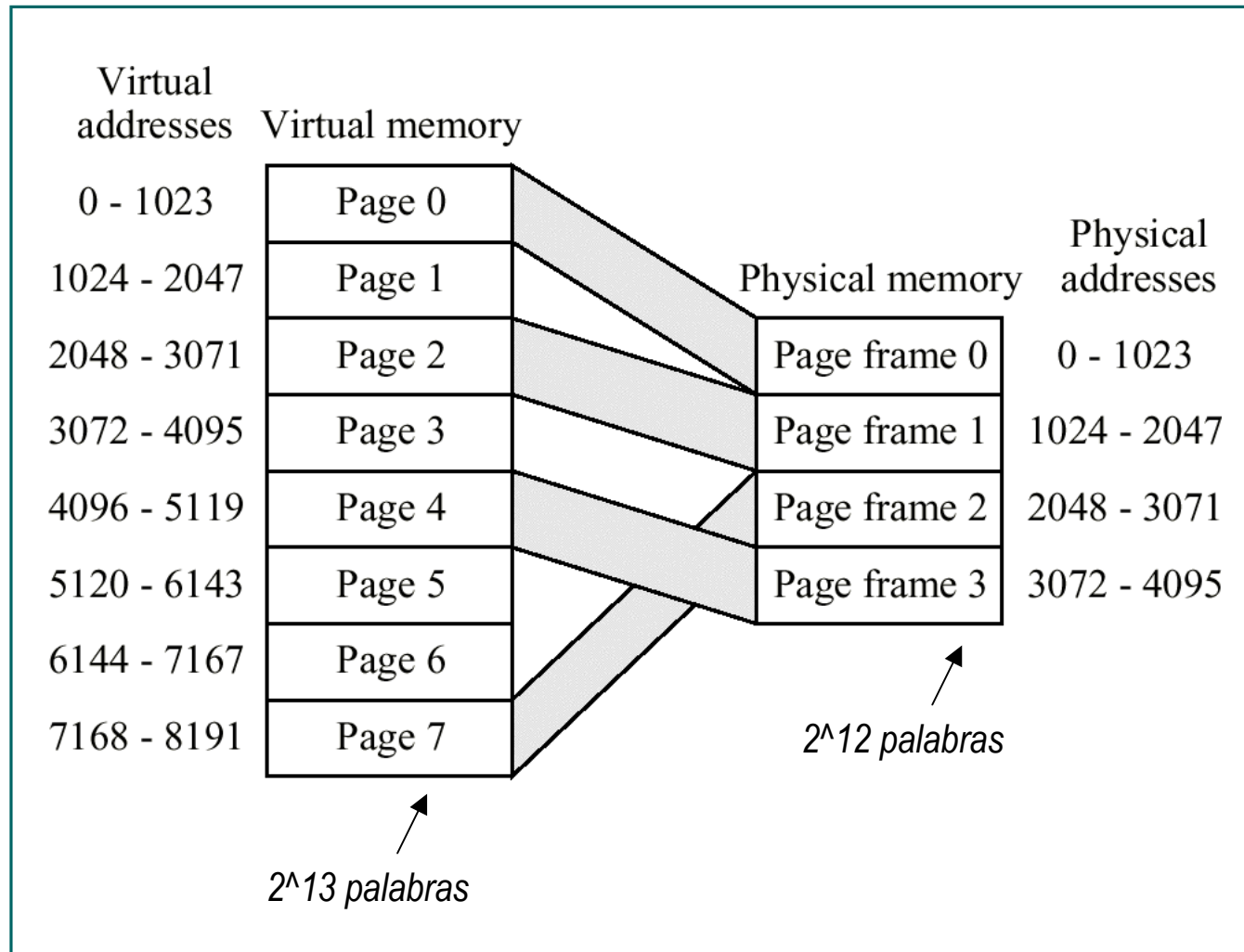
- El concepto de memoria virtual comenzó con **Overlay**



- ✓ El mismo programa administra el uso de overlay
- ✓ **Actualmente en desuso**

# Memoria paginada

Mapeo de memoria virtual a memoria física



Administrado  
por la MMU  
(on-chip)

?

Page #	Present bit	Disk address	Page frame
0	1	01001011100	00
1	0	11101110010	xx
2	1	10110010111	01
3	0	00001001111	xx
4	1	01011100101	11
5	0	10100111001	xx
6	0	00110101100	xx
7	1	01010001011	10

Tabla de páginas

# Memoria paginada

Mejorar performance

Con Memoria virtual y sólo tabla de páginas:

- Acceso a memoria = Acceso a Tabla de Página + Acceso al dato
- Aún accediendo directamente a RAM es más lento!



Solución: **Translation Lookaside Buffer (TLB)**

Pequeña **memoria asociativa** (en CPU) guarda las traducciones más recientes de memoria virtual a memoria física

=> Disminuye tiempo de acceso

Valid	Virtual page number	Physical page number
1	0 1 0 0 1	1 1 0 0
1	1 0 1 1 1	1 0 0 1
0	- - - - -	- - - -
0	- - - - -	- - - -
1	0 1 1 1 0	0 0 0 0
0	- - - - -	- - - -
1	0 0 1 1 0	0 1 1 1
0	- - - - -	- - - -

TLB de 8 entradas

Memoria virtual 32 palabras

Memoria física 16 palabras

---

# Memoria asociativa o “Direccionable por contenido” (CAM)

*Comparación con memoria RAM:*

Address	Value
0000A000	0F0F0000
0000A004	186734F1
0000A008	0F000000
0000A00C	FE681022
0000A010	3152467C
0000A014	C3450917
0000A018	00392B11
0000A01C	10034561

← 32 bits → ← 32 bits →

Random access memory

**RAM**

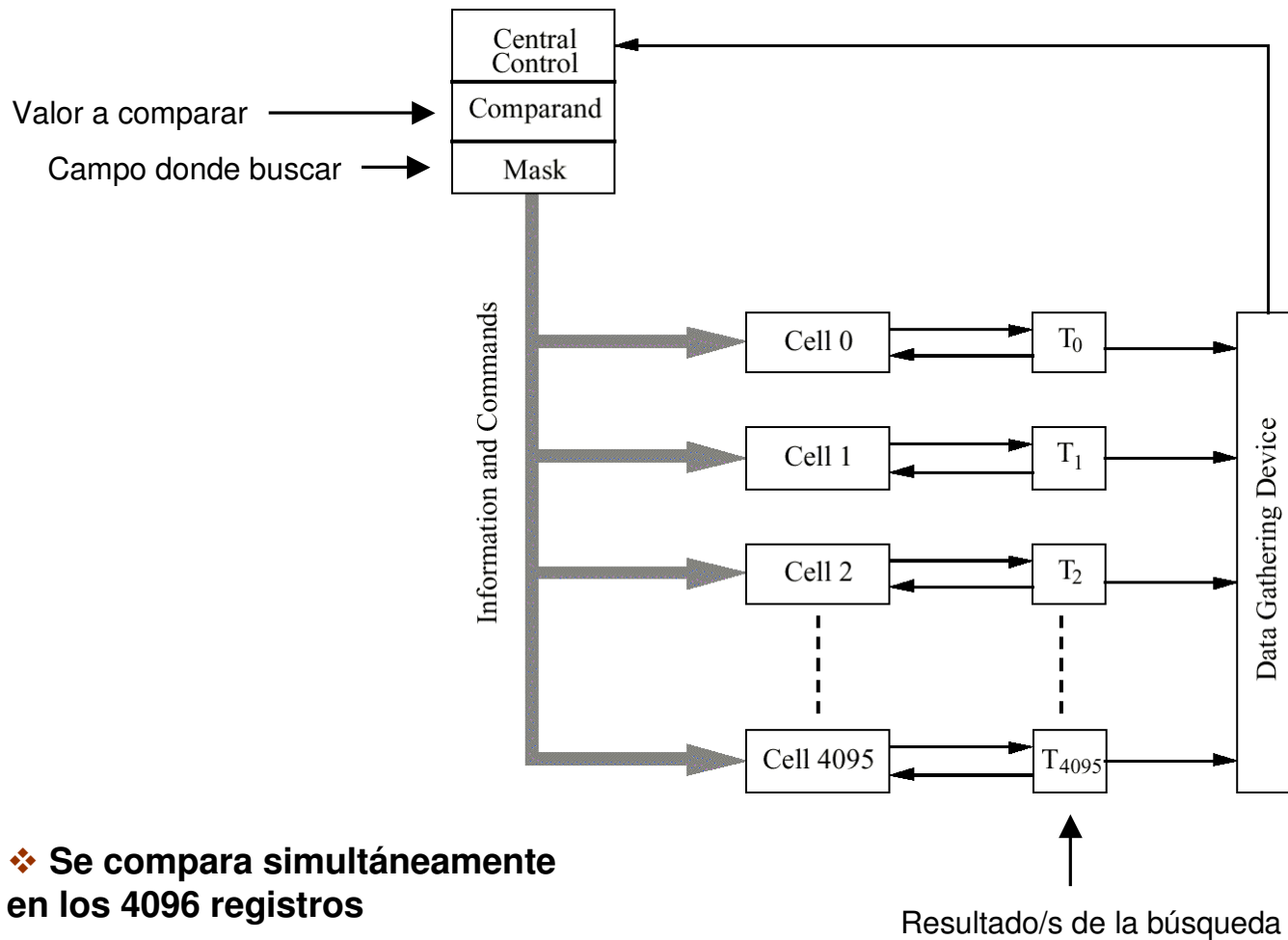
Field1	Field2	Field3
000	A	9E
011	0	F0
149	7	01
091	4	00
000	E	FE
749	C	6E
000	0	50
575	1	84

← 12 bits → ← 4 bits → ← 8 bits →

Content addressable memory

**CAM**

# Memoria asociativa o “Direccional por contenido” (CAM)



# Memoria asociativa o “Direccional por contenido” (CAM)

- Es difícil implementar un diseño eficiente
- Muy costosa
- Muy utilizadas pero en pocos campos de aplicación:
  - *Tag memory en memoria cache*
  - *Translation Lookaside Buffer en memoria paginada*
  - *Routers*
  - *Compresión de datos*

