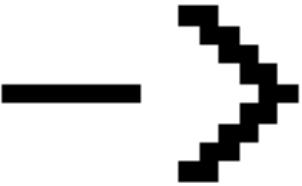


Organización del computador

Memoria

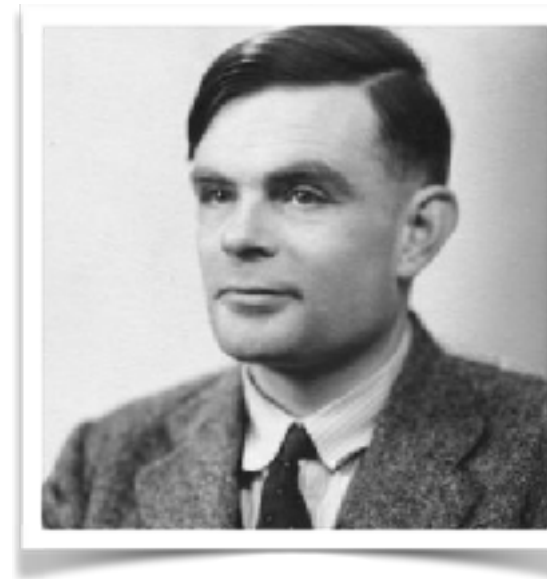
Jerarquía de máquina



Nivel 6	Usuario	Programa ejecutables
Nivel 5	Lenguaje de alto nivel	C++, Java, Python, etc.
Nivel 4	Lenguaje ensamblador	Assembly code
Nivel 3	Software del sistema	Sistema operativo, bibliotecas, etc.
Nivel 2	Lenguaje de máquina	Instruction Set Architecture (ISA)
Nivel 1	Unidad de control	Microcódigo / hardware
Nivel 0	Lógica digital	Circuitos, compuertas, memorias

- ➤ Cada nivel funciona como una máquina abstracta que oculta la capa anterior
- ➤ Cada nivel es capaz de resolver determinado tipo de problemas a partir de comprender un tipo de instrucciones específico
- ➤ La capa inferior es utilizada como servicio

Von Newman / Turing



- * Los programas y los datos se almacenan en la misma memoria sobre la que se puede leer y escribir
- * La operación de la máquina depende del estado de la memoria
- * El contenido de la memoria es accedido a partir de su posición
- * La ejecución es secuencial (a menos que se indique lo contrario)

Arquitectura de von Neumann

—> 3 componentes principales:

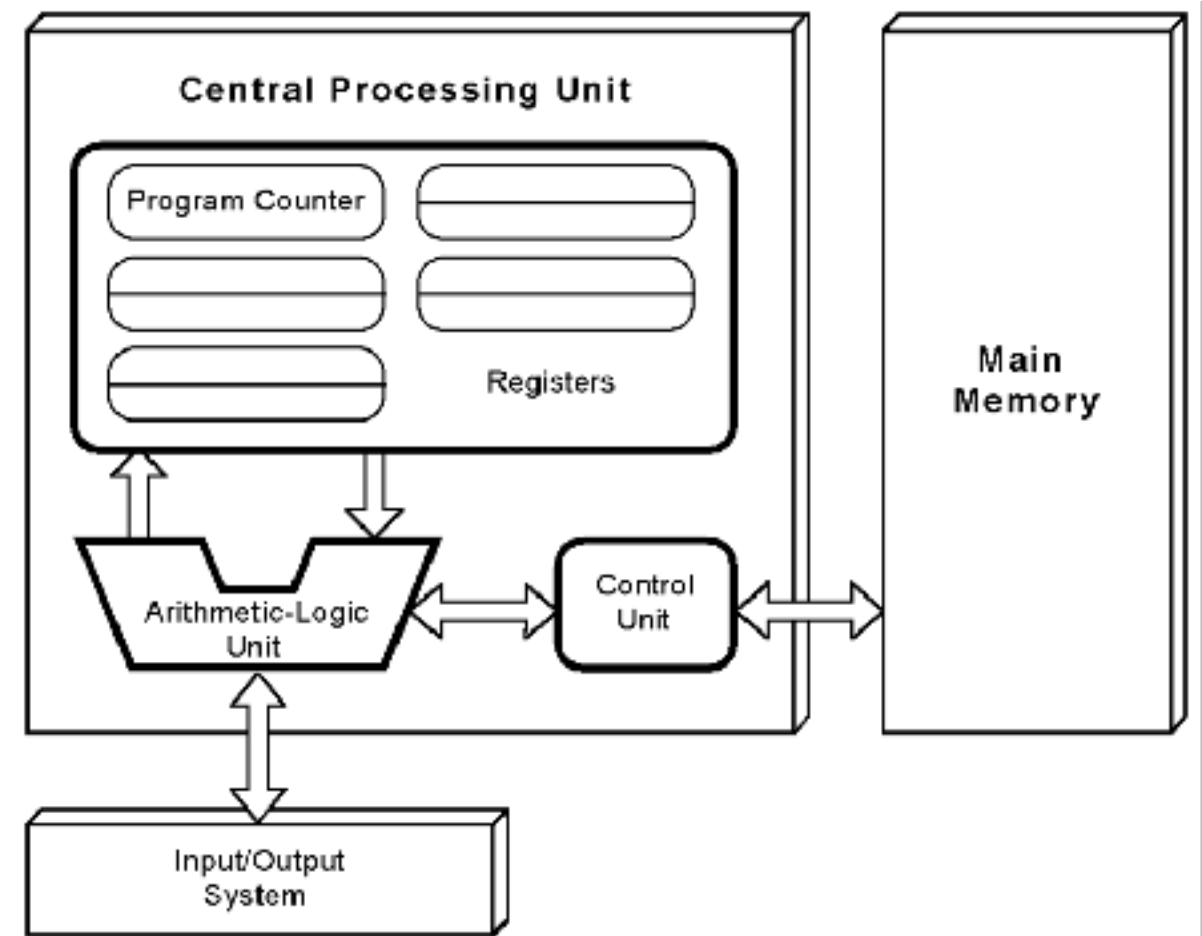
- **CPU:** Unidad de control, Unidad Aritmética lógica, Registros
- **Memoria:** Almacenamiento de programas y datos
- **Sistema** de Entrada y Salida

—> Procesamiento secuencial de instrucciones

—> Datos almacenados en sistema binario

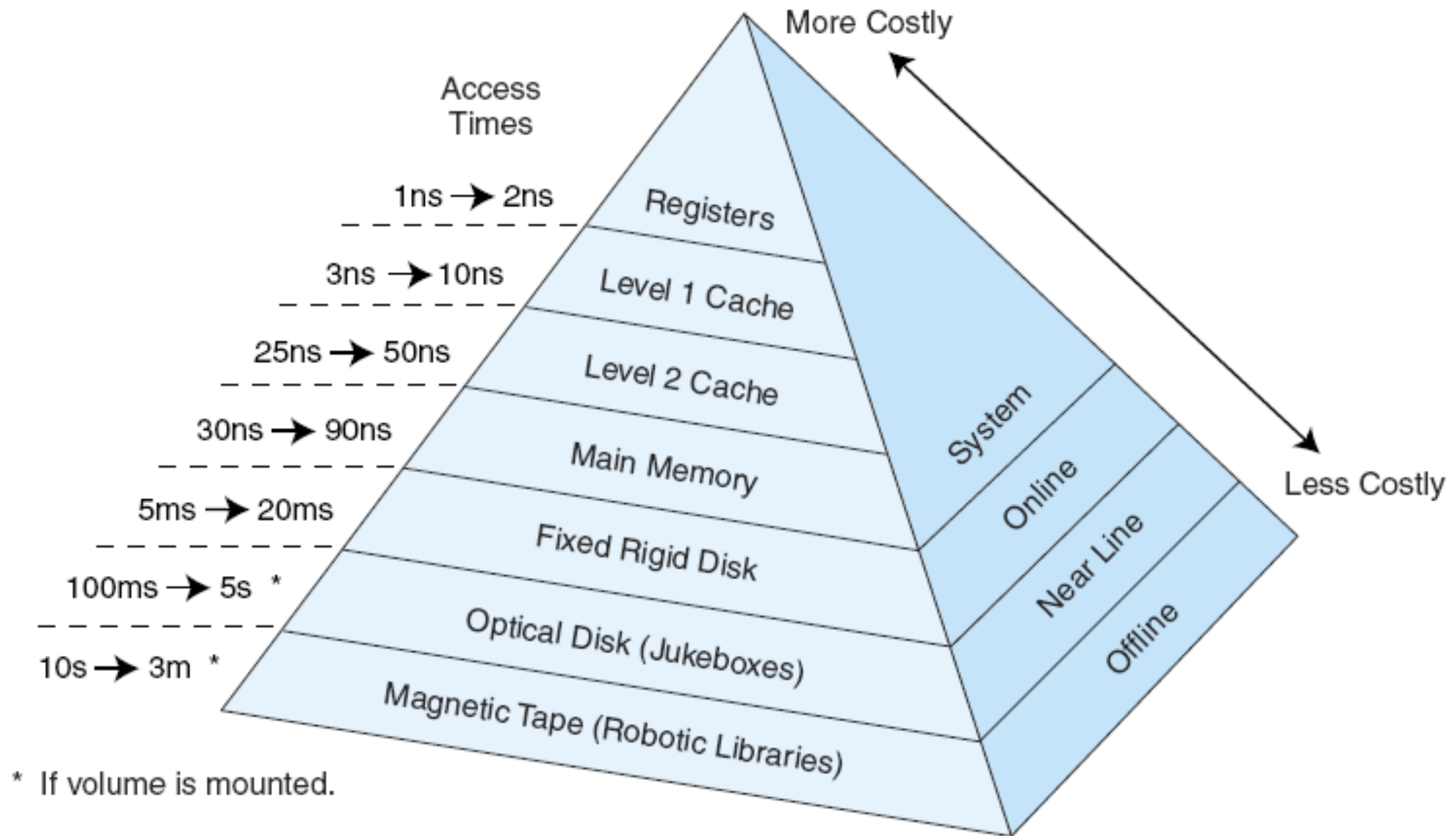
—> Sistema de interconexión de componentes:

- Conecta la Unidad de Control con la Memoria mediante un camino único
- La unicidad del camino fuerza la alternación entre ciclos de lectura / escritura y ejecución
- Esta alternación se llama cuello de botella de von Neumann (von Neumann bottleneck)^[*]

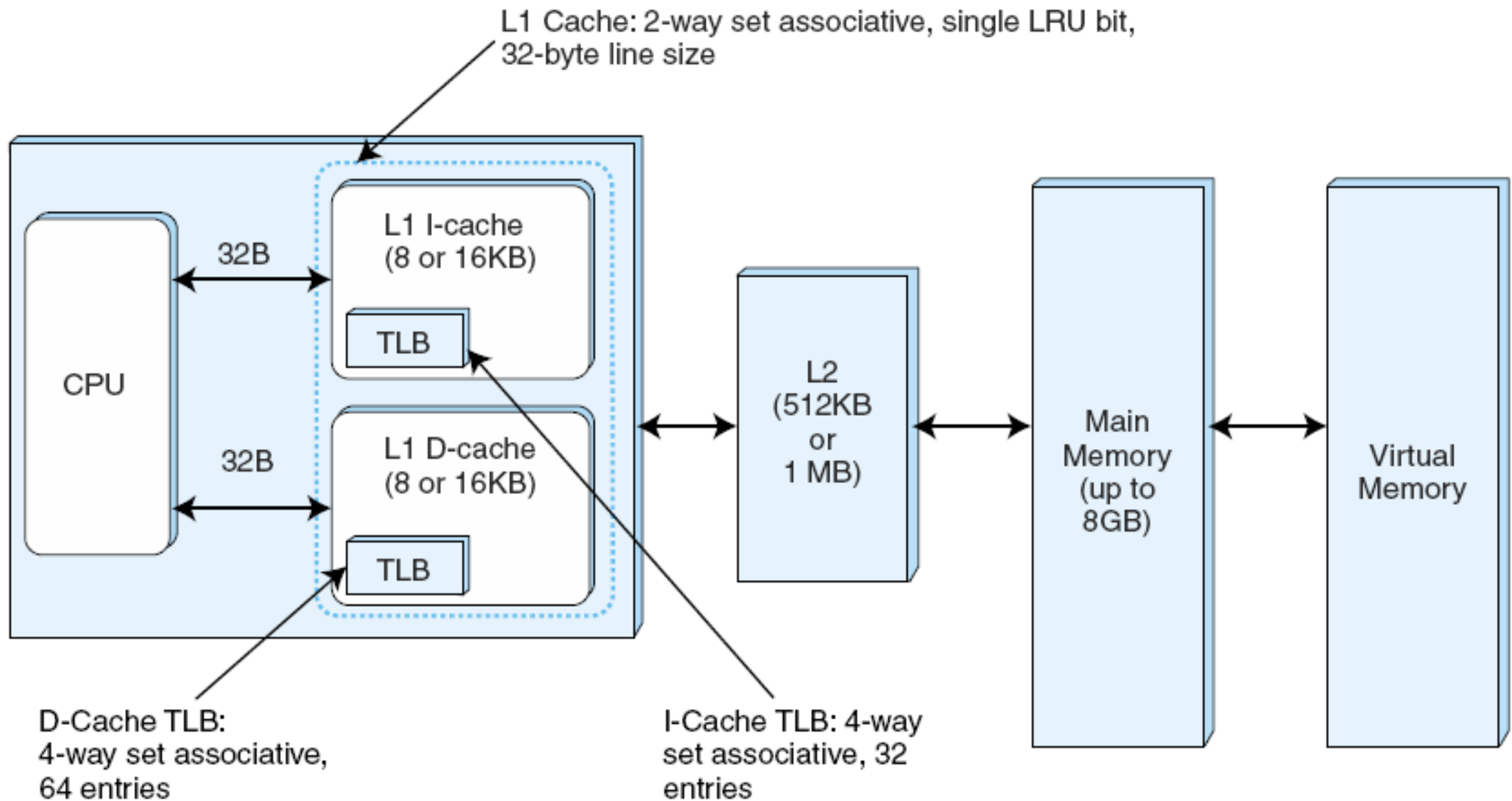


[*] El término “cuello de botella de von Neumann” fue acuñado por John Backus en su conferencia de la concesión del Premio Turing ACM de 1977.

Jerarquía de memoria



Jerarquía de memoria



Métricas de análisis de una memoria

- ➤ **Capacidad de almacenamiento:** cantidad de (múltiplos de) bytes que es capaz de almacenar
- ➤ **Tiempo de acceso:** tiempo requerido en (múltiplos de) segundos necesario para leer/escribir en una posición
- ➤ **Velocidad de transferencia:** tiempo requerido en (múltiplos de) segundos necesario para transferir (múltiplos de) bytes
- ➤ **Consumo de energía**
- ➤ **Tamaño físico**
- ➤ **Costo total / costo por (múltiplo de) bytes**

Tipos de memoria

- ➤ **Read Only Memory (ROM):**

- ➤ Programmable ROM

- ➤ Erasable Programmable ROM

- ➤ Electronically Erased Programmable ROM

- ➤ **Random Access Memory:**

- ➤ Static RAM

- ➤ Dynamic RAM y Synchronic Dynamic RAM

- ➤ Non-volatile RAM

Memoria OTP^[*] EPROM

Features

- Fast Read Access Time – 90 ns
- Dual Voltage Range Operation
 - Low Voltage Power Supply Range, 3.0V to 3.6V or Standard 5V \pm 10% Supply Range
- Compatible with JEDEC Standard AT27C512R
- Low Power CMOS Operation
 - 20 μ A Max (Less than 1 μ A Typical) Standby for V_{CC} = 3.6V
 - 29 mW Max Active at 5 MHz for V_{CC} = 3.6V
- JEDEC Standard Packages
 - 32-lead PLCC
 - 28-lead SOIC
 - 28-lead TSOP
- High Reliability CMOS Technology
 - 2,000V ESD Protection
 - 200 mA Latchup Immunity
- Rapid Programming Algorithm – 100 μ s/Byte (Typical)
- CMOS and TTL Compatible Inputs and Outputs
 - JEDEC Standard for LVTTL
- Integrated Product Identification Code
- Industrial Temperature Range
- Green (Pb/Halide-free) Packaging Option



**512K (64K x 8)
Low Voltage
OTP EPROM**

AT27LV512A

[*] OTP significa One-Time Programmable pues las EPROM normalmente se borran a través del uso de luz ultravioleta que pasa por una ventana de cuarzo en la parte superior del chip pero resultan muy costosas de fabricar.



Memoria OTP^[*] EPROM

Diagrama de bloques

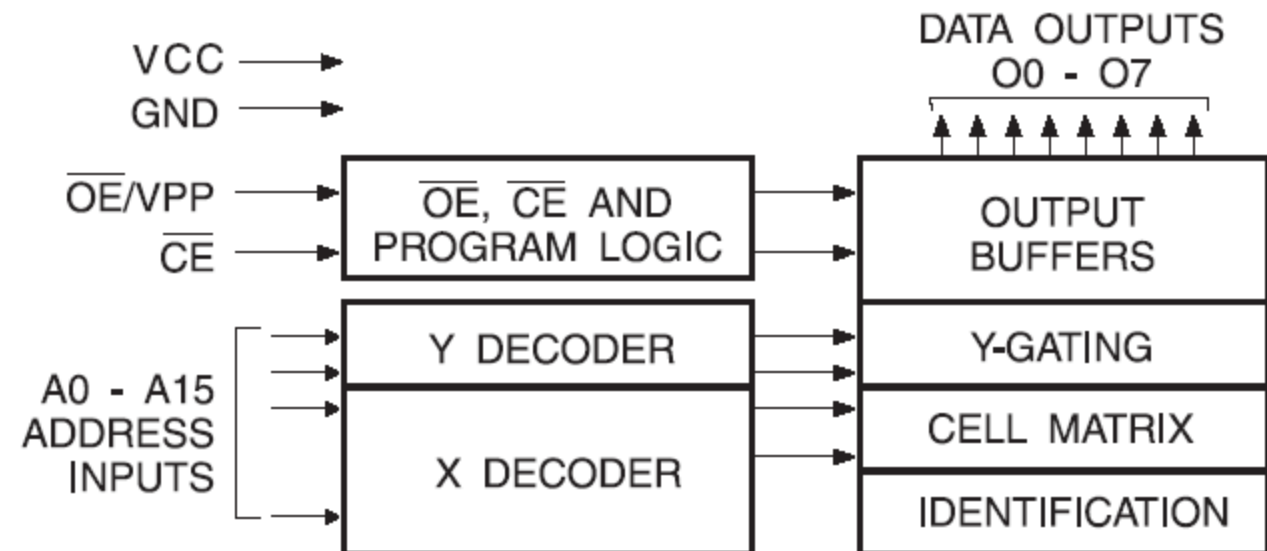
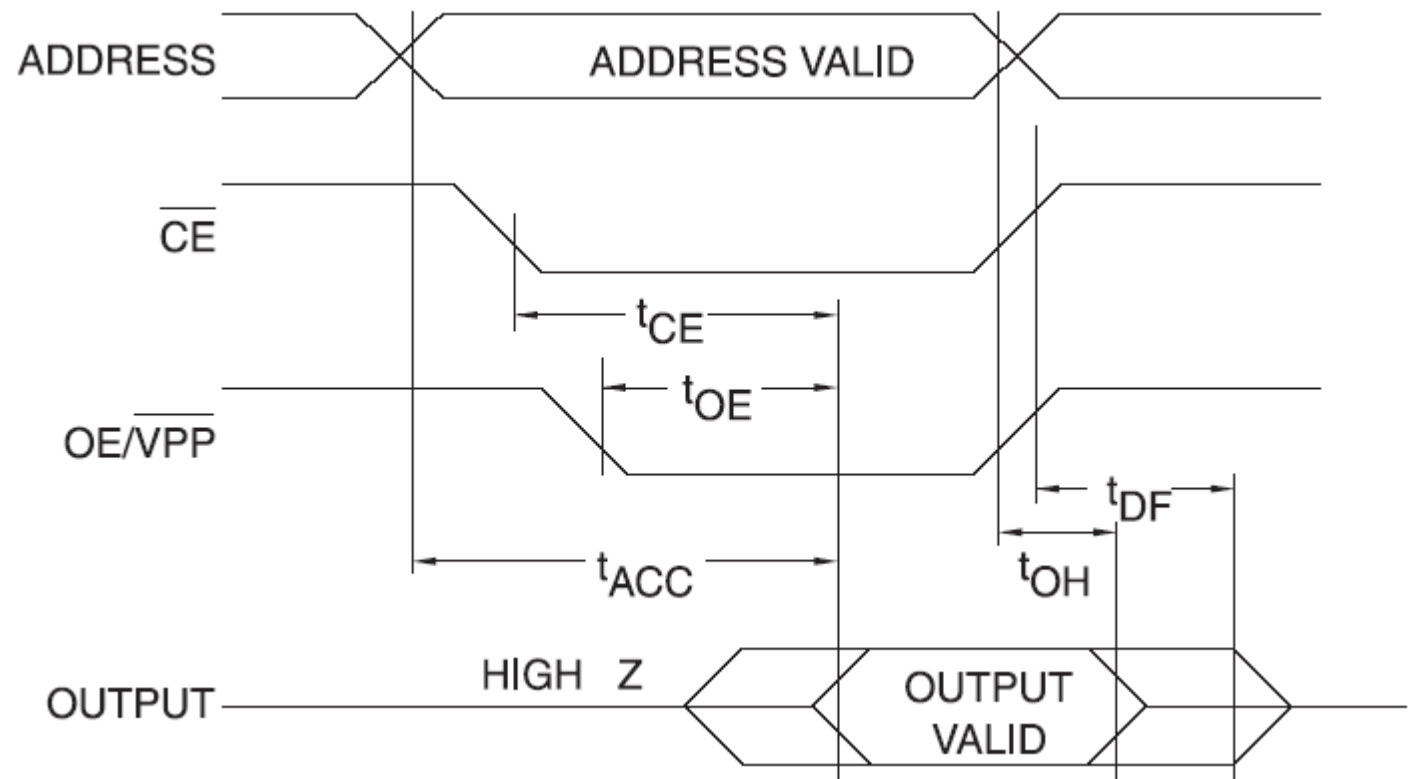


Diagrama de tiempos



Memoria EEPROM

Features

- 2.7V to 3.6V Supply
 - Full Read and Write Operation
- Low Power Dissipation
 - 8 mA Active Current
 - 50 μ A CMOS Standby Current
- Read Access Time - 300 ns
- Byte Write - 3 ms
- Direct Microprocessor Control
 - DATA Polling
 - READY/BUSY Open Drain Output
- High Reliability CMOS Technology
 - Endurance: 100,000 Cycles
 - Data Retention: 10 Years
- JEDEC Approved Byte-Wide Pinout
- Commercial and Industrial Temperature Ranges

Description

The AT28BV64 is a low-voltage, low-power Electrically Erasable and Programmable Read Only Memory specifically designed for battery powered applications. Its 64K of memory is organized 8,192 words by 8 bits. Manufactured with Atmel's advanced nonvolatile CMOS technology, the device offers access times to 200 ns with power dissipation less than 30 mW. When the device is deselected the standby current is less than 50 μ A.

(continued)



64K (8K x 8)
Battery-Voltage[™]
Parallel
EEPROMs

AT28BV64

Memoria SRAM

DATA SHEET

NEC

MOS INTEGRATED CIRCUIT

μ PD43256B

**256K-BIT CMOS STATIC RAM
32K-WORD BY 8-BIT**

Description

The μ PD43256B is a high speed, low power, and 262,144 bits (32,768 words by 8 bits) CMOS static RAM.

Battery backup is available. And A and B versions are wide voltage operations.

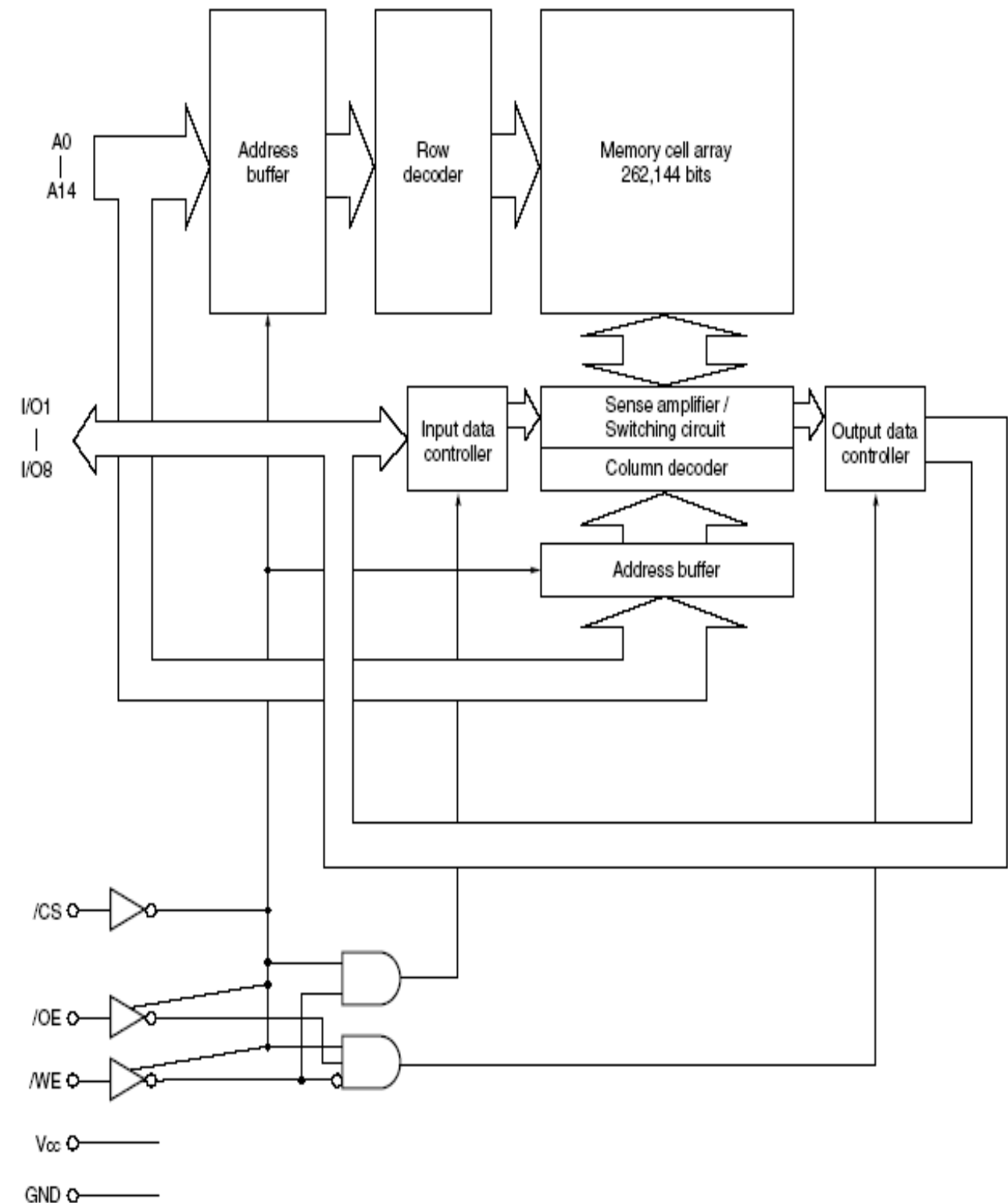
The μ PD43256B is packed in 28-pin PLASTIC DIP, 28-pin PLASTIC SOP and 28-pin PLASTIC TSOP (I) (8 x 13.4 mm).

Features

- 32,768 words by 8 bits organization
- Fast access time: 70, 85, 100, 120, 150 ns (MAX.)
- Low voltage operation (A version: $V_{CC} = 3.0$ to 5.5 V, B version: $V_{CC} = 2.7$ to 5.5 V)
- Low V_{CC} data retention: 2.0 V (MIN.)
- /OE input for easy application

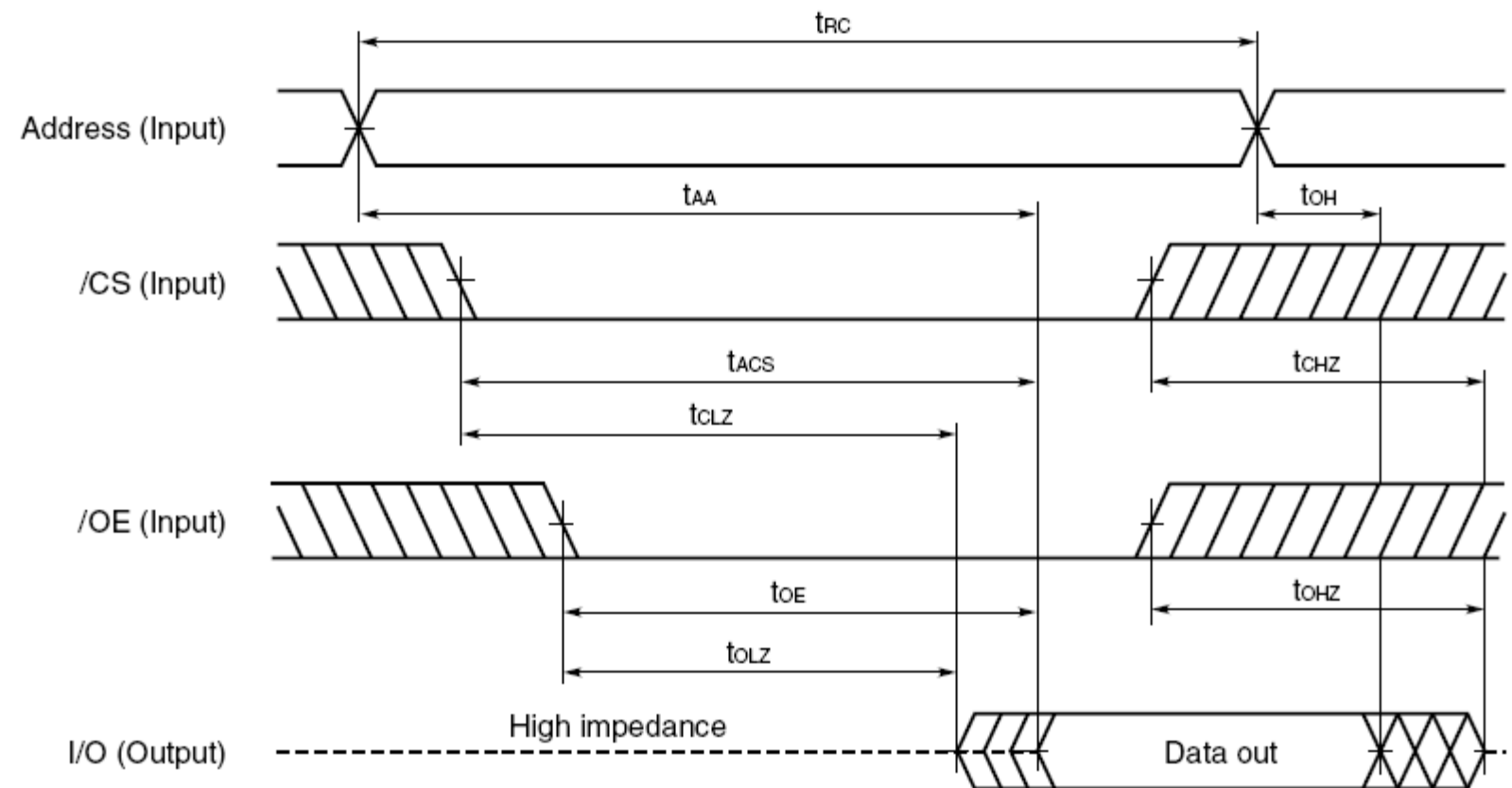
Memoria SRAM

Diagrama de bloques



Memoria SRAM

Read Cycle Timing Chart



Remark In read cycle, /WE should be fixed to high level.

Diagrama de tiempos

Memoria SRAM

Write Cycle Timing Chart 2 (/CS Controlled)

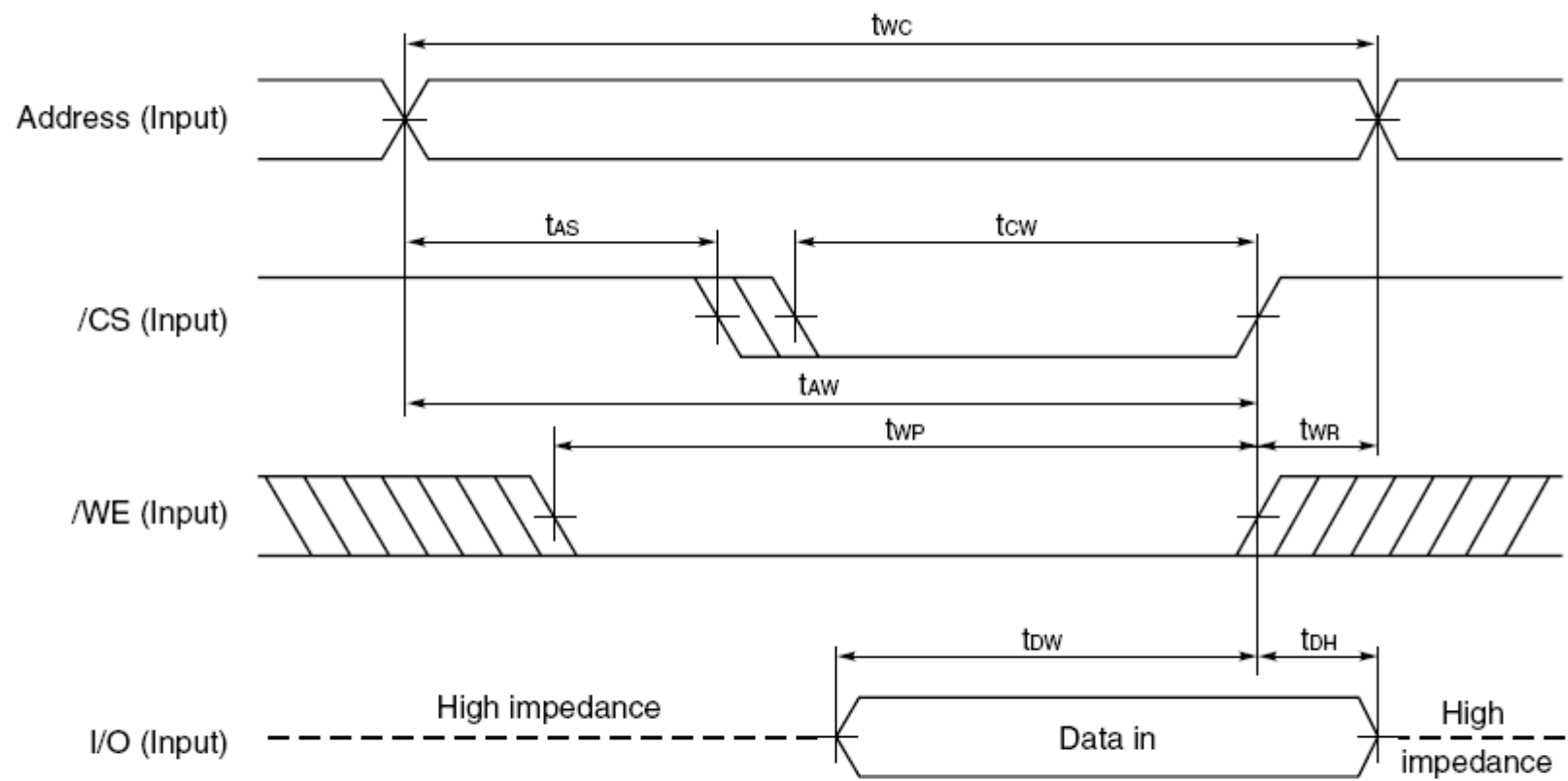


Diagrama de tiempos

- Cautions
1. /CS or /WE should be fixed to high level during address transition.
 2. When I/O pins are in the output state, therefore the input signals must not be applied to the output.

Memoria Flash

Features

- Single Supply Voltage Range, 2.7V to 3.6V
- Single Supply for Read and Write
- Fast Read Access Time – 90 ns
- Internal Program Control and Timer
- 8K Bytes Boot Block with Lockout
- Fast Erase Cycle Time – 10 Seconds
- Byte-by-byte Programming – 30 μ s/Byte Typical
- Hardware Data Protection
- $\overline{\text{DATA}}$ Polling for End of Program Detection
- Low Power Dissipation
 - 25 mA Active Current
 - 50 μ A CMOS Standby Current
- Typical 10,000 Write Cycles
- Green (Pb/Halide-free) Packaging Option

1. Description

The AT49BV512 is a 3-volt only, 512K Flash memories organized as 65,536 words of 8 bits each. Manufactured with Atmel's advanced nonvolatile CMOS technology, the devices offer access times to 90 ns with power dissipation of just 90 mW over the commercial temperature range. When the devices are deselected, the CMOS standby current is less than 50 μ A.

To allow for simple in-system reprogrammability, the AT49BV512 does not require high input voltages for programming. Three-volt only commands determine the read and programming operation of the device. Reading data out of the device is similar to reading from an EPROM. Reprogramming the AT49BV512 is performed by erasing the entire 1 megabit of memory and then programming on a byte-by-byte basis. The typical byte programming time is a fast 30 μ s. The end of a program cycle can be optionally detected by the $\overline{\text{DATA}}$ polling feature. Once the end of a byte program cycle has been detected, a new access for a read or program can begin. The typical number of program and erase cycles is in excess of 10,000 cycles.



**512K (64K x 8)
Single 2.7-volt
Battery-Voltage
Flash Memory**

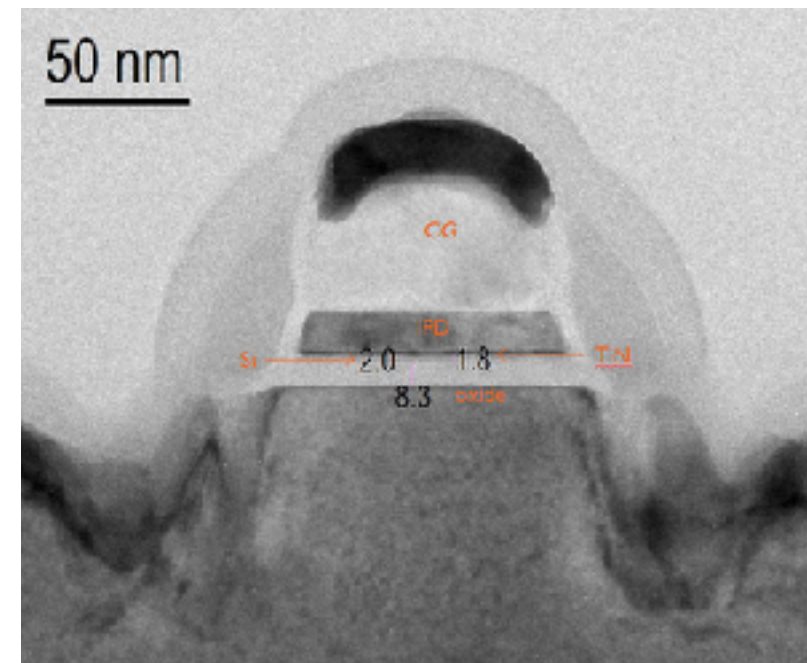
AT49BV512

Tecnología de integración

**Influenza virion
aprox. 100 nm**



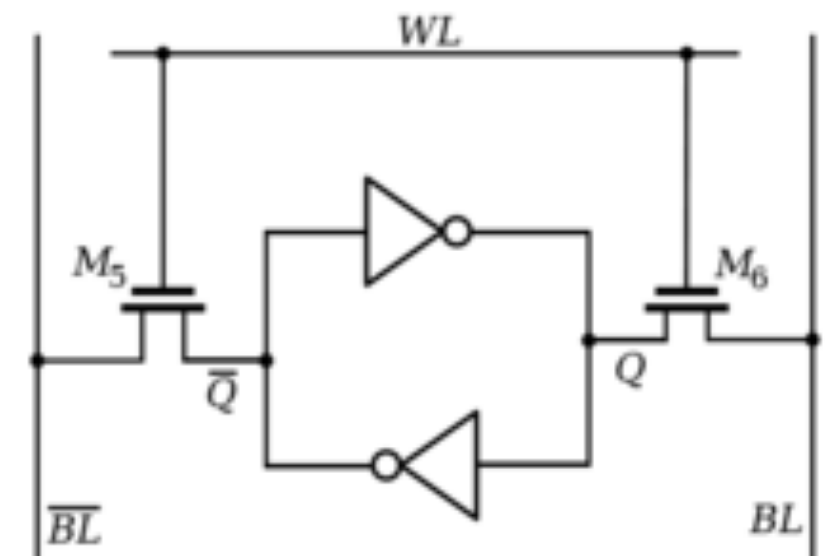
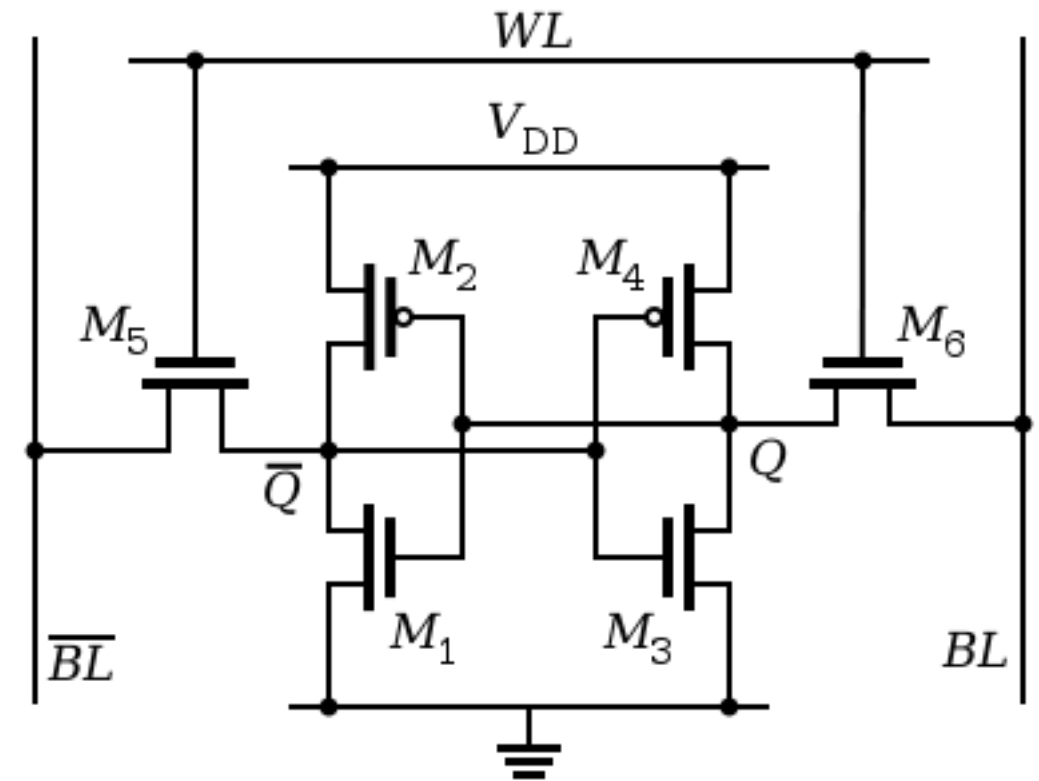
**Transistor
tecnología de 65 nm**



- ➤ La tecnología de 65 nm ha sido usada desde el Intel Pentium IV hasta la actualidad que está siendo reemplazado por tecnología de 20/22 nm, 14/16 nm y 10 nm.

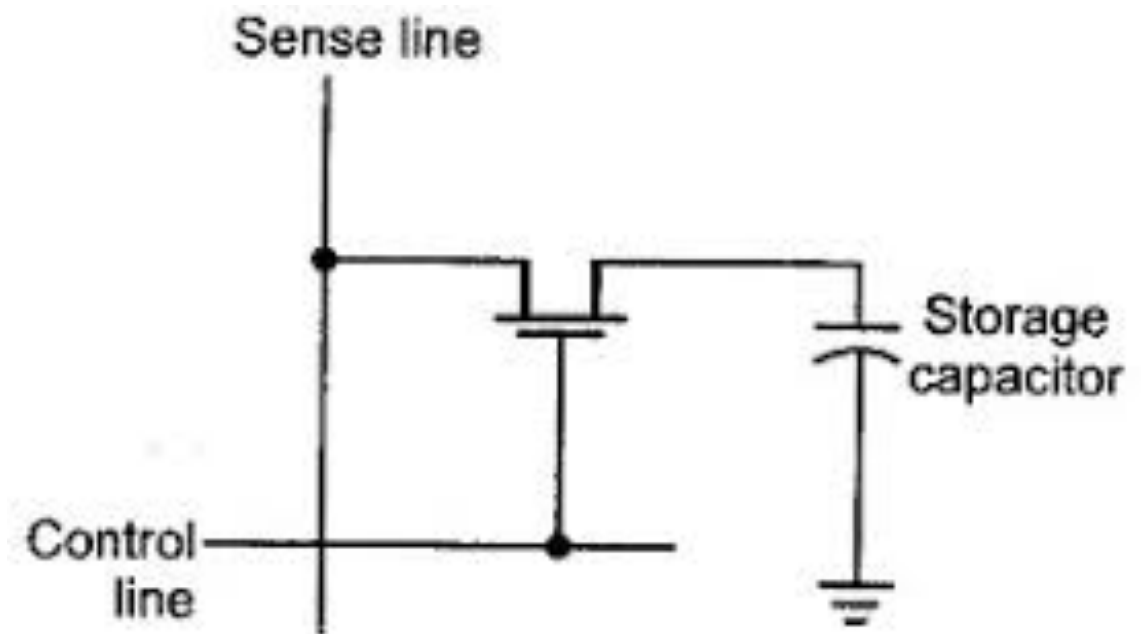
Memoria SRAM

- ➤ La información se almacena en un biestable como vimos en circuitos secuenciales
- ➤ Una celda de memoria utiliza 6 transistores, 3 de los cuales consumen cantidad máxima de energía lo que produce un gran consumo
- ➤ La lectura es directa y no destructiva lo que las hace muy rápidas



Memoria DRAM

- ➤ La información se almacena como una carga en un capacitor
- ➤ La lectura es indirecta pues se lee una carga de un capacitor y destructiva pues este se descarga
- ➤ Las cargas deben ser refrescadas periódicamente y cuando son leídas por lo que resultan muy lentas
- ➤ Tienen consumo mínimo ya que el transistor solo se activa para realizar la lectura



Problema

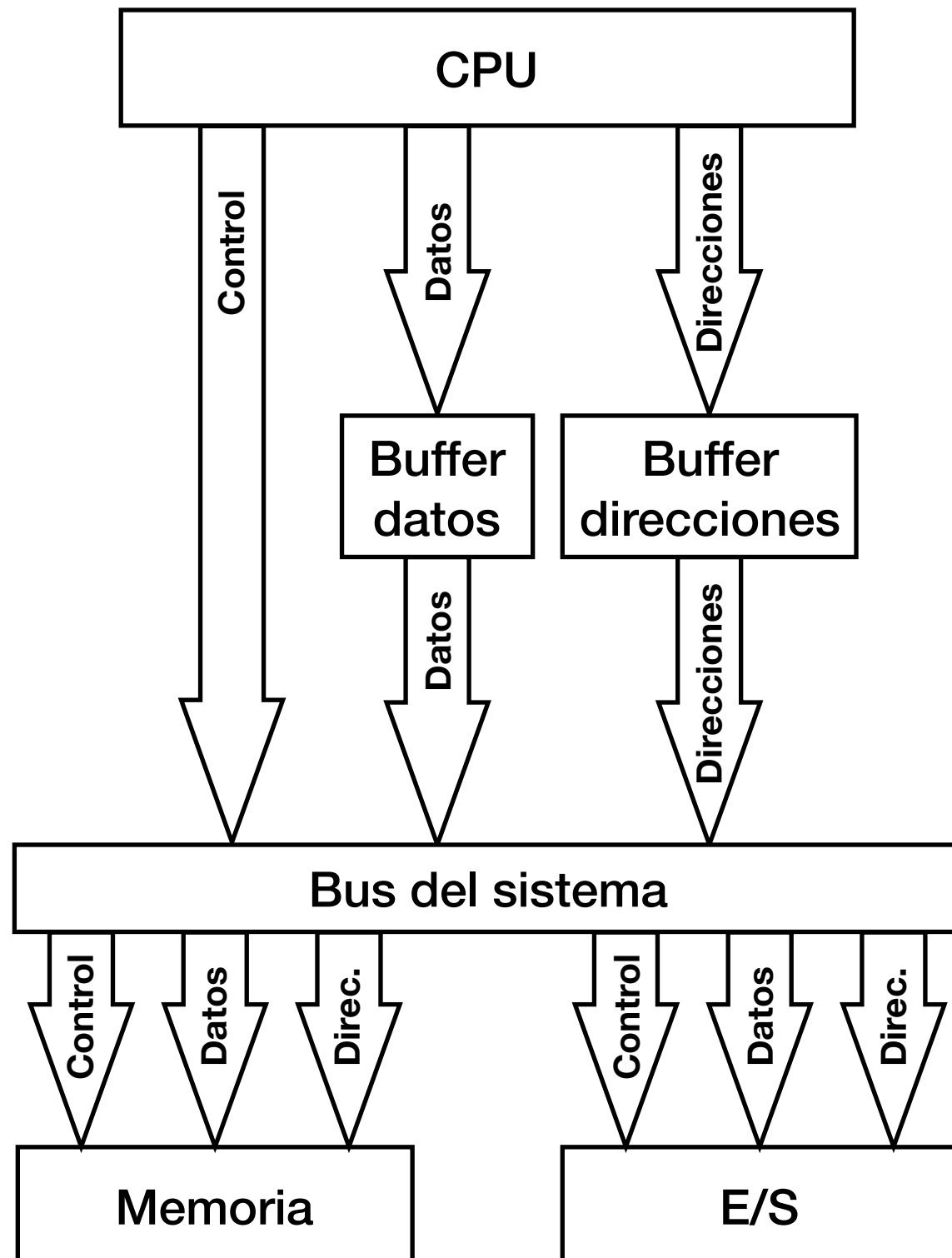
SRAM

- > Alto consumo relativo.
- > Capacidad de almacenamiento comparativamente baja.
- > Costo por bit alto.
- > Tiempo de acceso bajo (es mas rápida).
- > Si construimos el banco de memoria utilizando RAM estática, el costo y el consumo de la computadora son altos.

DRAM

- > Consumo mínimo.
- > Capacidad de almacenamiento comparativamente alta.
- > Costo por bit bajo.
- > Tiempo de acceso alto (lento), debido al circuito de regeneración de carga.
- > Si construimos el banco de memoria utilizando RAM dinámica, no aprovechamos la velocidad del procesador.

Estructura de bus clásica



- ➤ La estructura de conexión de una unidad de proceso está (típicamente) conectada al resto de las componentes del sistema a través de un bus que consta de 3 tipos de señales (Control, Datos y Direcciones)
- ➤ Una memoria lenta hará que el procesador, de mayor velocidad, tenga que recurrir a ciclos de espera
- ➤ Resulta inviable la utilización de SRAM para la construcción de la memoria del sistema

Solución

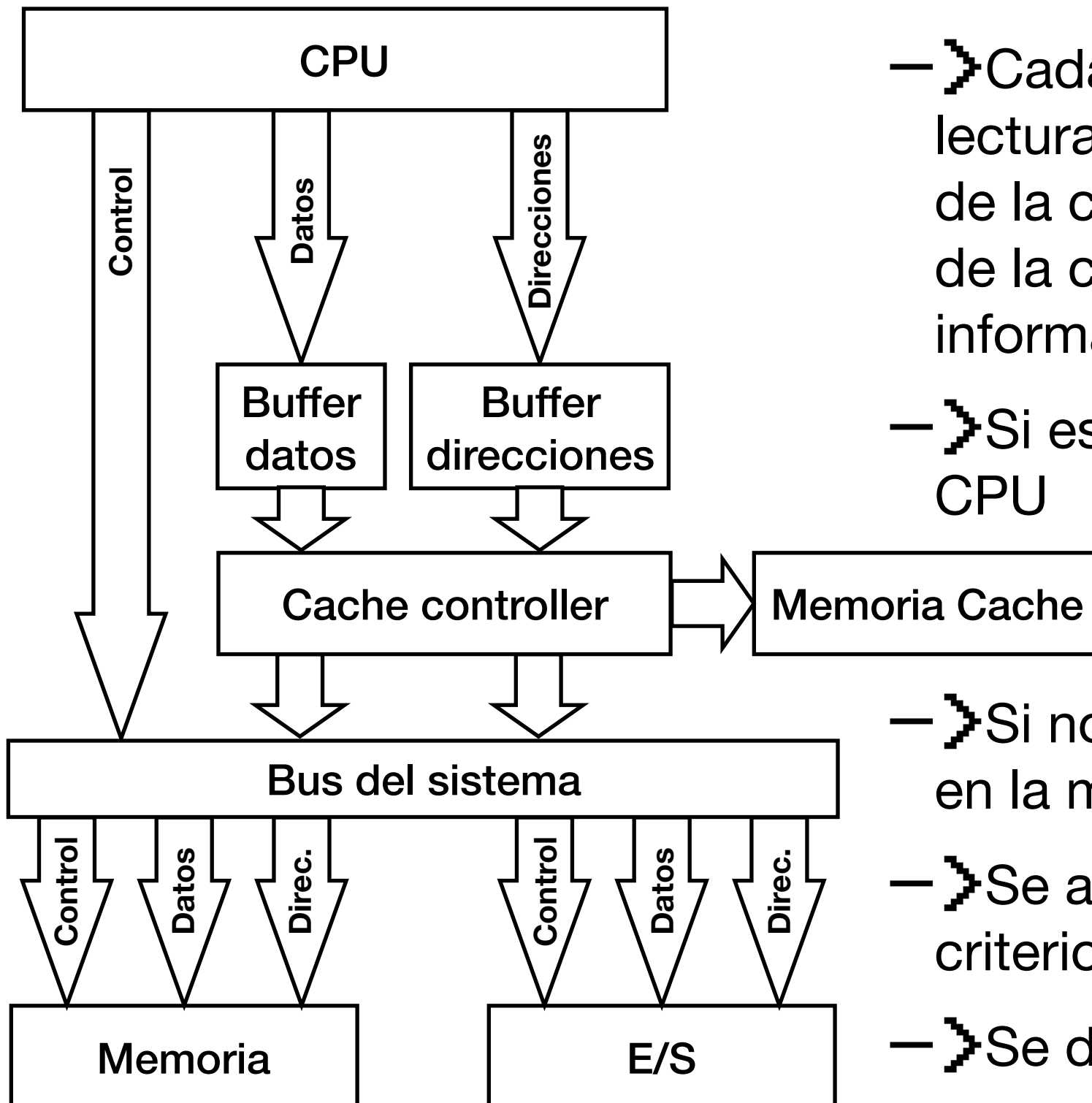
Memoria Cache

- > Siendo la **memoria principal** es un gran volumen de DRAM
- > Se implementa un banco de SRAM de muy alta velocidad denominado **cache**, que contiene una copia de (algunos de) los datos e instrucciones que están en memoria principal
- > El arte consiste en que esta copia esté disponible justo cuando el procesador la necesita permitiéndole acceder a esos ítems sin recurrir a wait states
- > Requiere de hardware adicional que asegure que este pequeño banco de memoria cache contenga datos e instrucciones consistentes con los que está almacenado en la memoria principal

Cache

- > El tamaño de la memoria cache debe ser:
 - > Suficientemente grande para que el procesador sea capaz de resolver la mayor cantidad posible de accesos en esta memoria sin recurrir a la memoria principal, asegurando una alta performance
 - > Suficientemente pequeña para no afectar el consumo ni el costo del sistema
- > Se dice que se logra un ***hit*** cuando se accede a un ítem (dato o código) y éste se encuentra en la memoria cache.
- > En caso contrario, se dice que el resultado del acceso es un ***miss***
- > Se define ***hit rate*** = $\text{hit} / \text{\#accesos}$
- > Se espera un ***hit rate*** lo mas alto posible

Cache read



- ➤ Cada vez que la CPU inicia una lectura en la memoria, el controlador de la cache que busca en el directorio de la cache para determinar si la información está presente allí
- ➤ Si está (**hit**) se devuelve el item a la CPU
- ➤ Si no está (**miss**) se accede al item en la memoria principal
- ➤ Se actualiza la cache según el criterio utilizado
- ➤ Se devuelve el item al procesador

Principios de la Cache

- ➤ Uno de los principales problemas del uso de memorias Cache reside en la elección de la información a ser almacenada en ella
- ➤ **Principio de vecindad espacial:** si un ítem es utilizado es probable que los ítems cercanos en la memoria sean utilizados en el futuro cercano (por ejemplo, las posiciones de un arreglo)
- ➤ **Principio de vecindad temporal:** si un ítem es utilizado es probable que sea utilizado nuevamente en el futuro cercano (por ejemplo, variables utilizadas como contadores en un ciclo)

Principios de la Cache

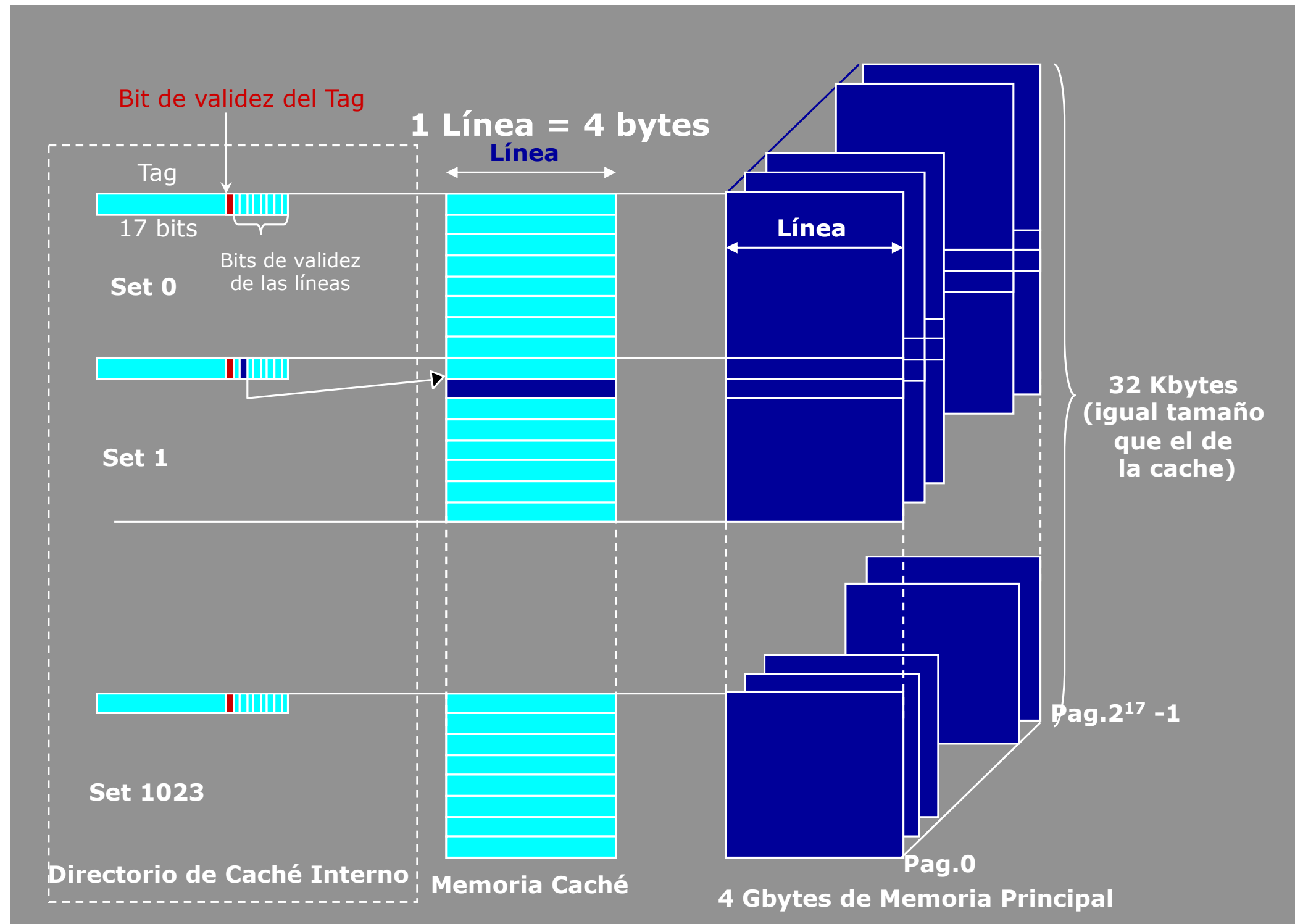
— ➤ Ejemplo:

```
for (i = 0 ; i < 256 ; i++ ) {  
    suma = 0.0f;  
    for (j = 0 ; (j <= i && j < 256) ; j++)  
        suma += v0[i-j] * v1[j];  
    fAux[i] = suma;  
}
```

- ➤ La utilización de las variables suma, i, j responden al principio de vecindad temporal
- ➤ La utilización de los arreglos fAux, v0, v1 responden al principio de vecindad espacial

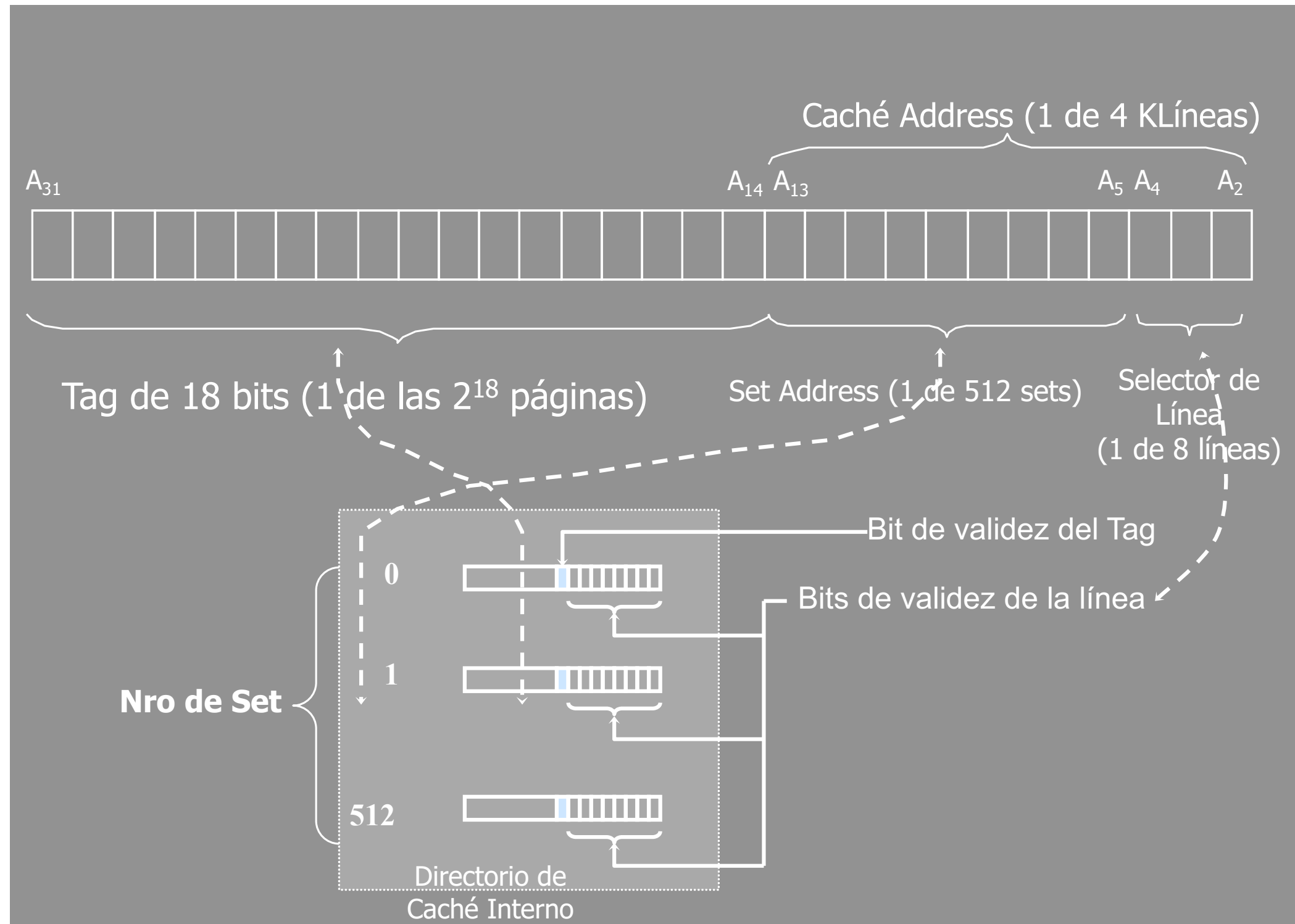
Organización de la Cache

Mapeo directo



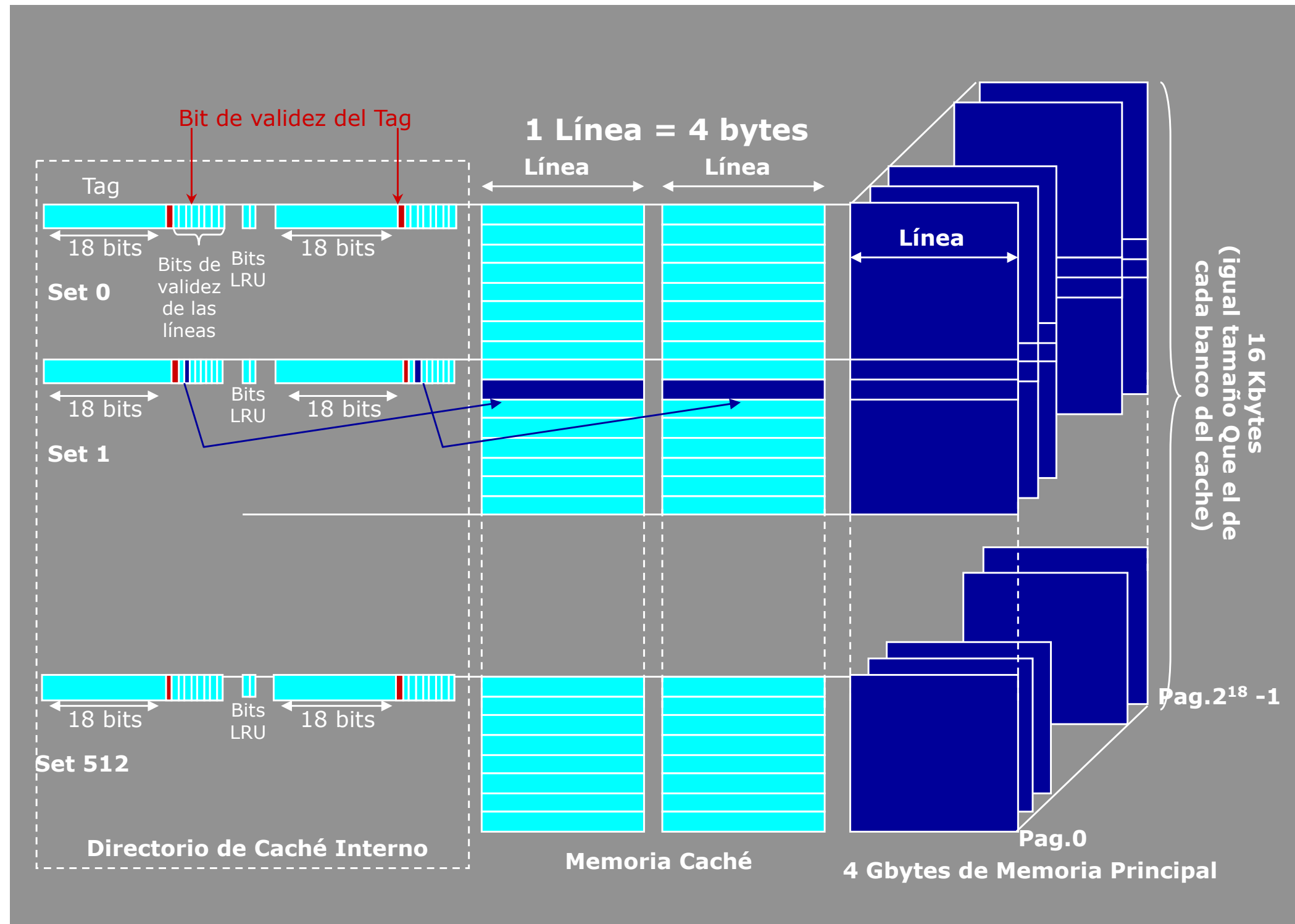
Organización de la Cache

Mapeo directo



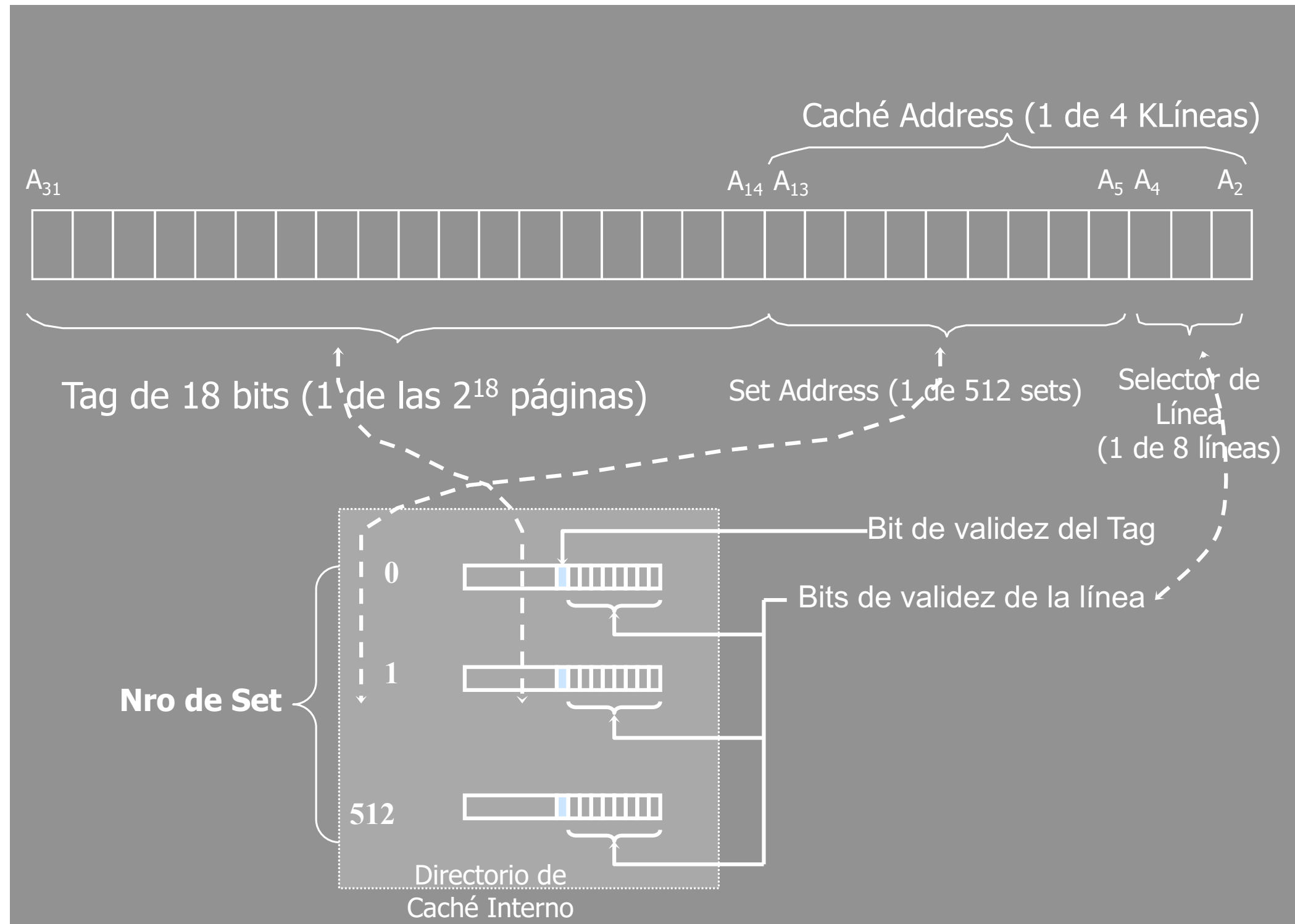
Organización de la Cache

Mapeo asociativo



Organización de la Cache

Mapeo directo



Política de reemplazo de contenido de la Cache

- ➤ Las políticas de reemplazo de contenidos en la Cache suelen tener una correlación con los principios que hacen que el contenido haya sido colocado en ella
 - ➤ **Least recently used:** se corresponde con el principio de vecindad temporal y prioriza aquellos contenidos que han sido utilizados recientemente
 - ➤ **Least frequently used:** se corresponde con el principio de vecindad temporal y prioriza aquellos contenidos que han sido más utilizados
 - ➤ **FIFO (first in - first out):** se purgan los contenidos más antiguos

Coherencia del contenido de la Cache

- ➤ Si una variable está en la **memoria cache** debe estar alojada en alguna dirección de la **memoria principal**
- ➤ Cuando se modifica un valor hay varios modos de actuar:
 - ➤ **Write through**: el procesador escribe en la memoria y el controlador de la memoria cache refresca el dato actualizado
 - ➤ **Write through buffered**: el procesador escribe en la cache, y el controlador de la memoria cache luego actualiza la copia en la memoria principal
 - ➤ **Copy back**: Se marcan las líneas de la memoria cache cuando el procesador escribe en ellas. Luego en el momento de eliminar esa línea de la memoria cache el controlador deberá actualizar la memoria principal

Cache multinivel

