Memorias semiconductoras

Conceptos básicos:

- Una memoria está formada por una *matriz de celdas* en las que se almacena información (1 bit en cada celda)

- Las memorias están diseñadas para manejar *información* en grupos de bits denominados *palabras* (*words*).

- En cada fila de la matriz de celdas se guarda 1 *palabra* (*word*).

- En función de la memoria que se considere, las palabras pueden ser de 1 bit, 4 bits, 8 bits , 9 bits o un múltiplo entero de ocho.

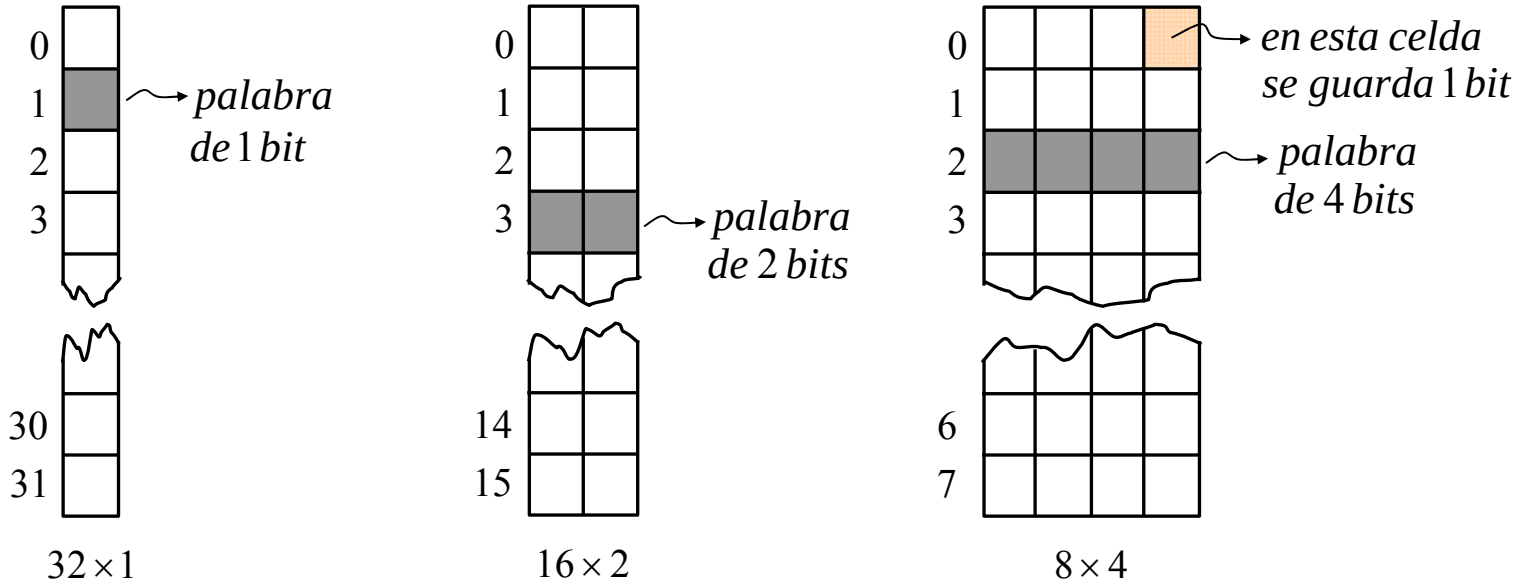
El tablero es el Mundo, las piezas son los fenómenos del Universo, las reglas del juego corresponden a lo que llamamos leyes de la Naturaleza. El jugador del otro lado está oculto, sabemos que juega limpio, genuina y pacientemente. Sin embargo, también sabemos que, a nuestra costa, nunca perdona un error u otorga la más mínima concesión a la ignorancia.

Thomas Henry Huxley

- Capacidad de una memoria: indica el número total de bits que puede guardar

Los fabricantes acostumbran a indicar la capacidad de las memorias como el producto del *número de palabras* que pueden guardar por el *número de bits* que tiene cada *palabra*. Así, por ejemplo, una memoria ROM 512×4 guarda 512 palabras de 4 bits (512 *nibbles*) o, si se prefiere, 2048 bits.

Ejemplos de diferentes organizaciones de una memoria de 32 bits de capacidad



Por cuestiones de optimización de los circuitos, el número de palabras que puede guardar una memoria siempre va a ser una potencia entera de la base 2. Esto ha hecho que se hayan adoptado las siguientes unidades para indicar la capacidad de las memorias:

Electrónica Digital	*Física*
$1k_{(kilo)} \equiv 2^{10} = 1024_{10}$	$1k \equiv 10^{3}$
$1M_{(mega)} \equiv 2^{20} = 1048576_{10}$	$1M \equiv 10^{6}$
$1G_{(giga)} \equiv 2^{30} = 1073741824_{10}$	$1G \equiv 10^{9}$
$1T_{(tera)} \equiv 2^{40} = 2^{20} \times 2^{20}$	$1T \equiv 10^{12}$

Para evitar la confusión con los sufijos *k* (kilo, 10^3), *M* (mega, 10^6), *G* (giga, 10^9), . . . que se utilizan en otros campos de la Ingeniería, en Física, en Química, etc., en enero de 1999 el IEC (*International Electrotechnical Commission*) publicó la norma IEC 60027-2 en la que se establecen nuevos sufijos para representar potencias enteras de 2. Dichos sufijos fueron adoptados en 2002 por el IEEE en la norma IEEE Standard 1541-2002. En éstas normas se recomienda utilizar los siguientes sufijos:

$$1ki_{(kibi,kilobinary)} \equiv 2^{10} = 1024_{10}$$

$$1Mi_{(mebi,megabinary)} \equiv 2^{20} = 1048576_{10}$$

$$1Gi_{(gibi,gigabinary)} \equiv 2^{30} = 1073741824_{10}$$

$$1Ti_{(tebi,terabinary)} \equiv 2^{40} = 2^{20} \times 2^{20}$$

Antes (y ahora):

Cantidades de bits: *kilobit, Megabit, Gigabit, Terabit, Petabit, Exabit, Zetabit, Yottabit,…*

Cantidades de bytes: *kilobyte, Megabyte, Gigabyte, Terabyte, Petabyte, Exabyte, Zetabyte, Yottabyte,…*

De acuerdo a la norma actual:

Cantidades de bits: *kibibit, Mebibit, Gibibit, Tebibit, Pebibit, Exbibit, Zebibit, Yobibit,…*

Cantidades de bytes: *kibibyte, Mebibyte, Gibibyte, Tebibyte, Pebibyte, Exbibyte, Zebibyte, Yobibyte,…*

Ejemplo:

$$4ki = 4 \cdot 1ki = 2^2 \cdot 1 \cdot 2^{10} = 2^{12} = 4096_{10}$$

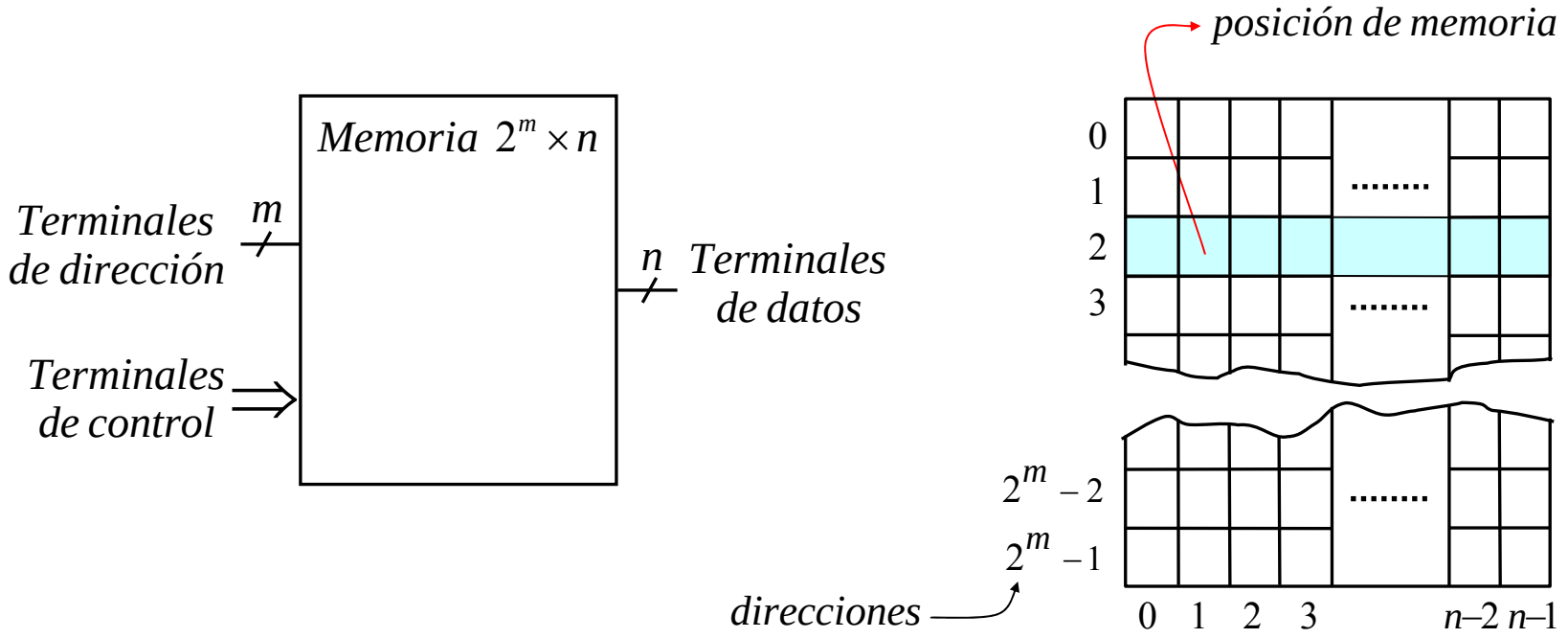
Nota: hay que tener presente que la terminología *Ki*, *Mi*, *Gi*, etc. no está muy difundida. En la mayoría de los documentos técnicos aún se utilizan los sufijos indicados en la diapositiva 4.

Pregunta: *¿Si la terminología ki, Mi, etc. está poco difundida, cómo sabemos cuando k representa* 10^3 *y cuando representa* 2^{10}?

Pregunta: *¿Cuál es la diferencia entre* $1kib$ *y* $1kiB$?

Pasatiempo: *¿Cuál es la diferencia entre* $1K$ *y* $1k$?

- Terminales básicos de una memoria de pequeña capacidad



_ La memoria tiene 2^m posiciones (filas). Cada posición tiene asociada una dirección (número binario). La posición (fila) en la que se *lee* o se escribe (*guarda*) un dato se determina decodificando la dirección presente en los terminales de dirección (bus de direcciones).

_ En cada posición (dirección) de la memoria se guardan *n* bits (dato).

- Operación de lectura: con una operación de lectura se leen los bits guardados en una posición de la memoria (≡ se lee el dato guardado en una fila de la matriz de celdas)

Para realizar una *operación de lectura* se hace lo siguiente:

1º: En los *terminales de dirección* se pone el valor correspondiente a la posición de memoria (número asociado a la fila de la matriz de celdas) en la que está guardado el dato (conjunto de bits) que se quiere conocer (leer).

2º: Se configuran los *terminales de control* para realizar una *operación de lectura* y un tiempo después, en los *terminales de datos* aparecen los bits guardados en la posición de memoria indicada en los *terminales de dirección*

Nota: el número y el tipo de *terminales de control* depende de la memoria que se considere. En general, los terminales de control permiten: establecer la operación a realizar (*lectura* o *escritura*), conectar eléctricamente los terminales de datos al *bus de datos*, inhibir las operaciones de lectura y escritura, proteger los datos guardados por hardware, poner la memoria en *standby* (≡ modo de bajo consumo), etc.

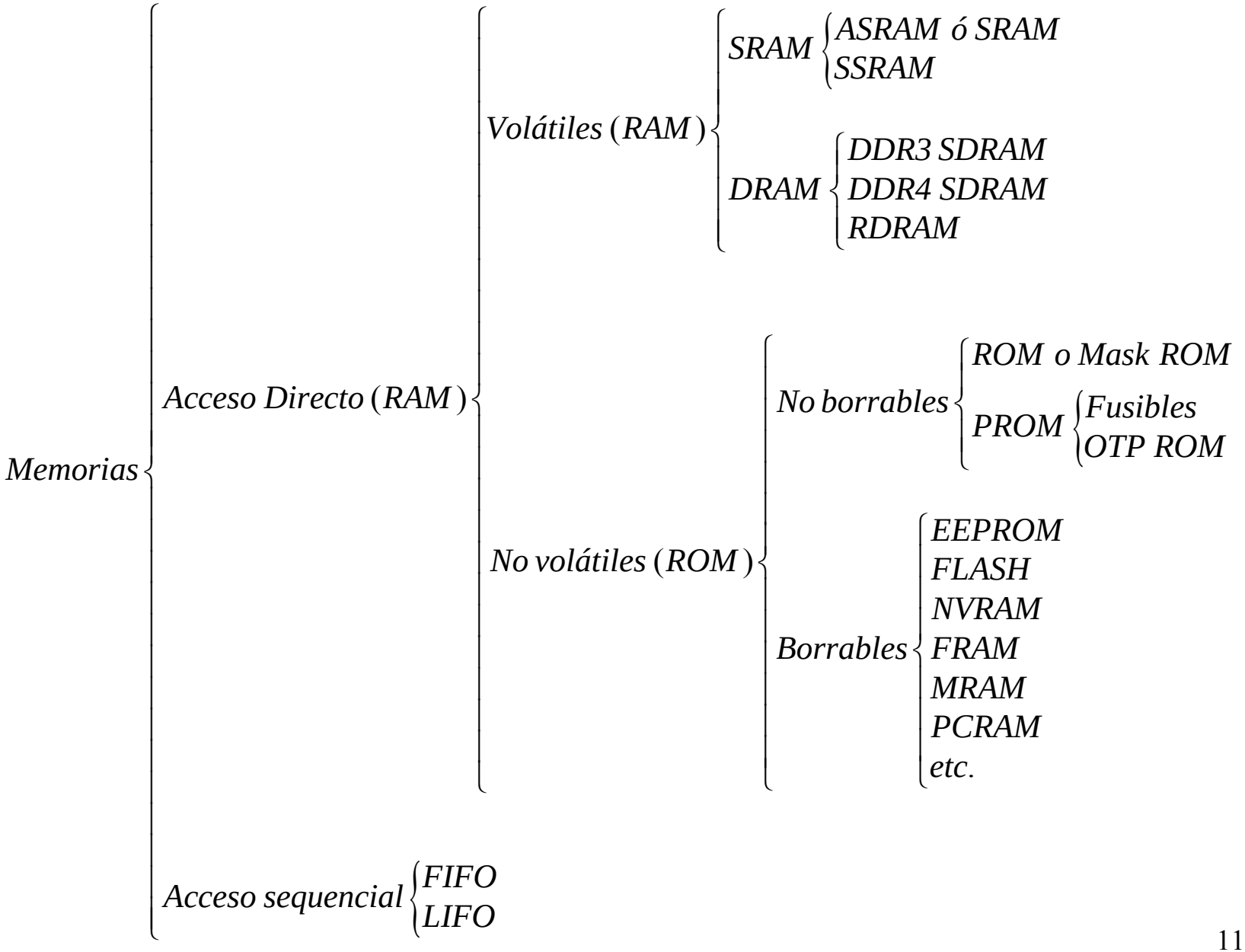
- Operación de escritura: en una memoria de capacidad $2^m \times n$, con una operación de escritura se guardan n bits en una posición de memoria (≡ se guarda 1 bit en cada una de las celdas de una determinada fila de la matriz de celdas).

Para realizar una operación de escritura se hace lo siguiente:

1º: En los *terminales de dirección* se pone la dirección correspondiente a la posición de memoria (≡ fila de la matriz de celdas) en la que se quiere guardar (escribir) un dato de n bits.

2º: En los *terminales de datos* se pone el dato de n bits a guardar en la posición de memoria (fila) seleccionada por el valor aplicado a los terminales de dirección.

3º: Se configuran los *terminales de control* para realizar una operación de *escritura* y un tiempo después, en la posición de memoria indicada queda guardado el dato presente en los terminales de datos de la memoria.


Memorias
Acceso Directo (RAM)
Volátiles (RAM)
SRAM
ASRAM ó SRAM
SSRAM
DRAM
DDR3 SDRAM
DDR4 SDRAM
RDRAM
No volátiles (ROM)
No borrables
ROM o Mask ROM
PROM
Fusibles
OTP ROM
Borrables
EEPROM
FLASH
NVRAM
FRAM
MRAM
PCRAM
etc.
Acceso sequencial
FIFO
LIFO

SRAM (*static RAM*):

_ Se utilizan *biestables* para implementar las celdas.

_ El circuito asociado a cada celda tiene un tamaño (volumen) elevado $\Rightarrow$ se pueden integrar 'pocas' celdas (biestables) en un chip $\Rightarrow$ sólo se pueden construir memorias (chips) de baja capacidad.

_ $t_{\text{operación lectura}} \approx t_{\text{operación escritura}}$

_ Son las memorias más rápidas $\Rightarrow$ menor $t_{\text{operación lectura}}$ y $t_{\text{operación escritura}}$.

_ Se utilizan para implementar las memorias *cachés* en los PCs.

_ Son volátiles

DRAM (*dynamic RAM*):

_ Se utilizan *condensadores* para implementar las celdas.

_ El circuito asociado a cada celda tiene un tamaño (volumen) muy pequeño $\Rightarrow$ se pueden integrar muchos condensadores ($\equiv$ celdas) en un chip $\Rightarrow$ se pueden fabricar memorias (chips) con una capacidad elevada.

_ Los condensadores se descargan con el paso del tiempo $\Rightarrow$ es necesario reestablecer (refrescar) su carga eléctrica cada pocos milisegundos para que no se pierda la información guardada. Esto hace que sean memorias bastante más lentas que las SRAM, debido a que durante las operaciones de refresco no se pueden realizar operaciones ni de lectura ni de escritura

_ $t_{\text{operación lectura}} \approx t_{\text{operación escritura}}$

_ Son memorias más lentas que las SRAM.

_ Se utilizan para implementar la *memoria principal* en los PCs.

no

DDR4 (≡ Double Data Rate type 4)

_ Es una memoria SDRAM (memoria RAM, dinámica, síncrona).

_ Es la generación más reciente de las memorias DDR (≡ Double Data Rate)

_ Duplica la velocidad de la generación DDR3

_ Tiene una tasa de transferencia de datos de: 3.2GT/s (*GigaTransfers/second*)

Es decir, la señal de reloj es de $3{,}2 \cdot 10^9/2$ = 1.6GHz (utiliza una frecuencia de reloj 16 veces más rápida que la primera generación de memorias DDR)

Nota: *data transfer rate*:

Bits transferred/second = Channel width (bits/transfer) × transfers/second

_ La primera generación de memorias DDR se fabricó con una escala de integración de 130nm y operaba con tensiones de 2.5 voltios. Las memorias DDR4 utilizan una escala de 14nm y operan con tensiones de 1 voltio

_ Tienen un menor consumo que las DDR3

FLASH:

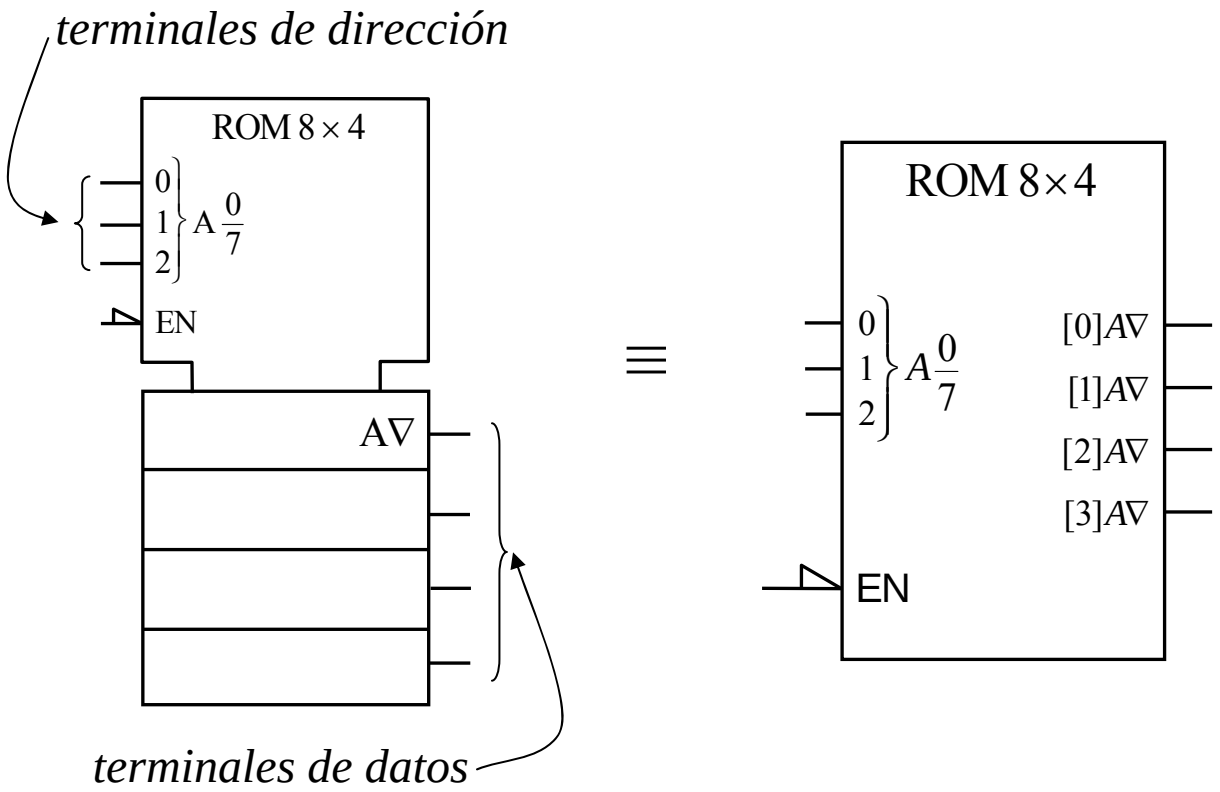
_ Son memorias no volátiles.

_ $t_{\text{operación lectura}} <<< t_{\text{operación escritura}}$

_ Aunque a día de hoy se comercializan chips que contienen memorias Flash con una elevada capacidad, sus tiempos de acceso (de lectura y sobre todo de escritura) impiden que se puedan usar para sustituir a las memorias SRAM o DRAM. Lo cual no quiere decir que las memorias Flash no tengan su campo de aplicación comercial.

Nota: el término "*endurance*" se refiere al número de ciclos de borrado/escritura que puede soportar un bloque de una memoria Flash sin estropearse (typ: 10^6)

Nota: 'el *dorado*' es una memoria *no volátil*, con la *capacidad* de una memoria DRAM y con el *tiempo de acceso* de una memoria SRAM.



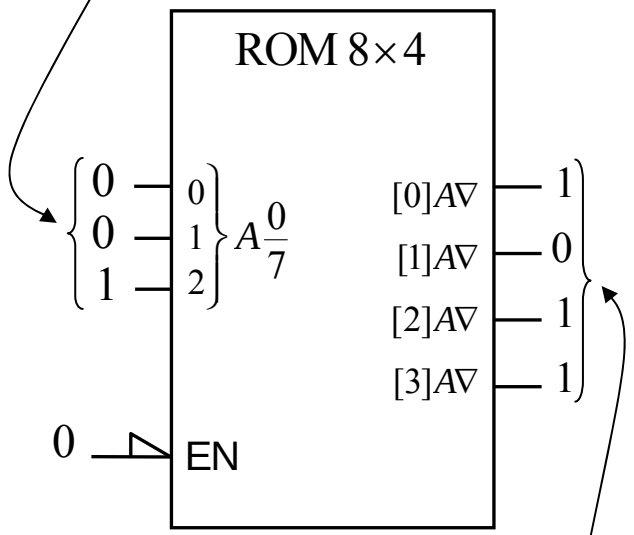
Nota: en simbología normalizada IE[3] Std. 91-1984:

_ **A** indica dirección

_ **EN** indica permitir, habilitar, activar, etc.

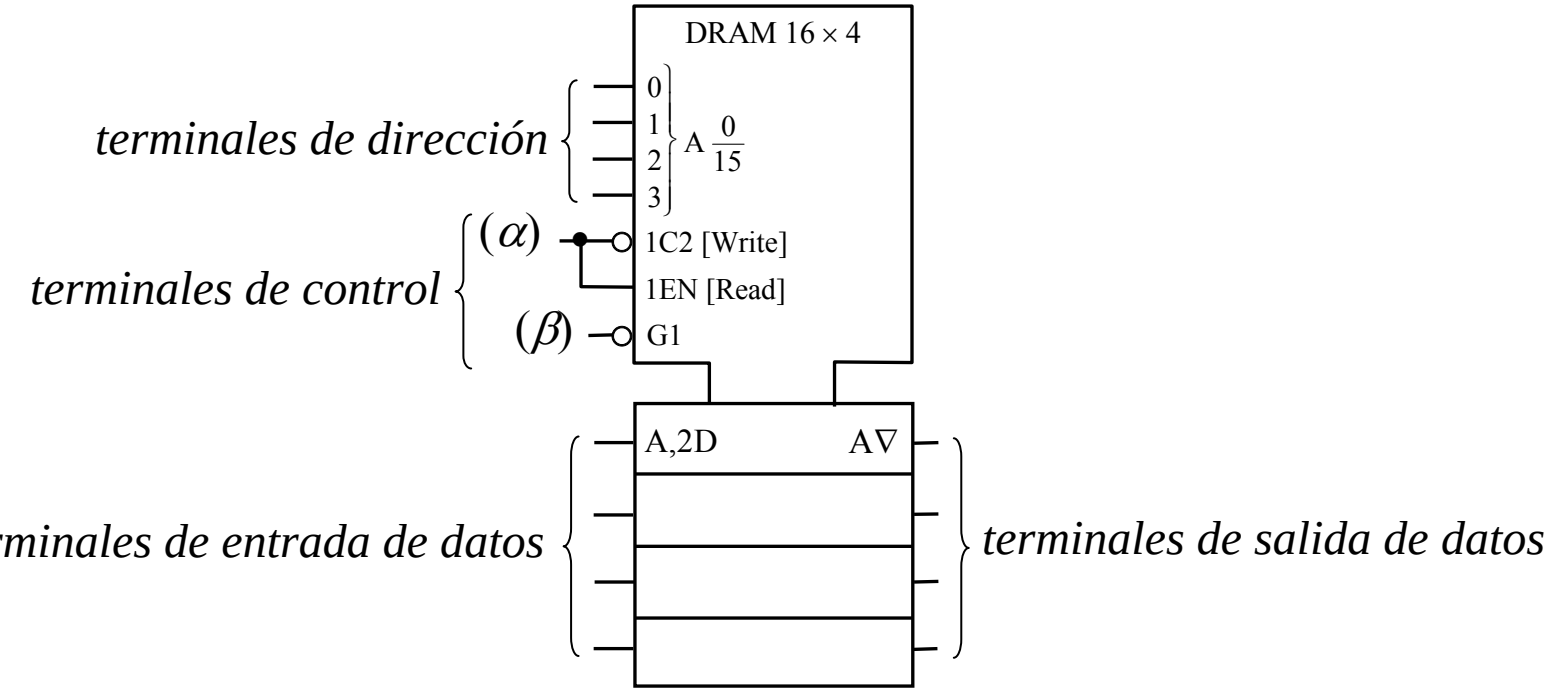
_ ∇ indica terminal con tercer estado (≡ estado de alta impedancia)

se indica (*selecciona*) *la dirección* 100_2



direcciones			*contenido memoria*			
0	0	0	0	1	1	0
0	0	1	1	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	1	0	0
1	1	0	0	1	0	1
1	1	1	0	1	1	1

valor guardado en la posición de memoria 100_2

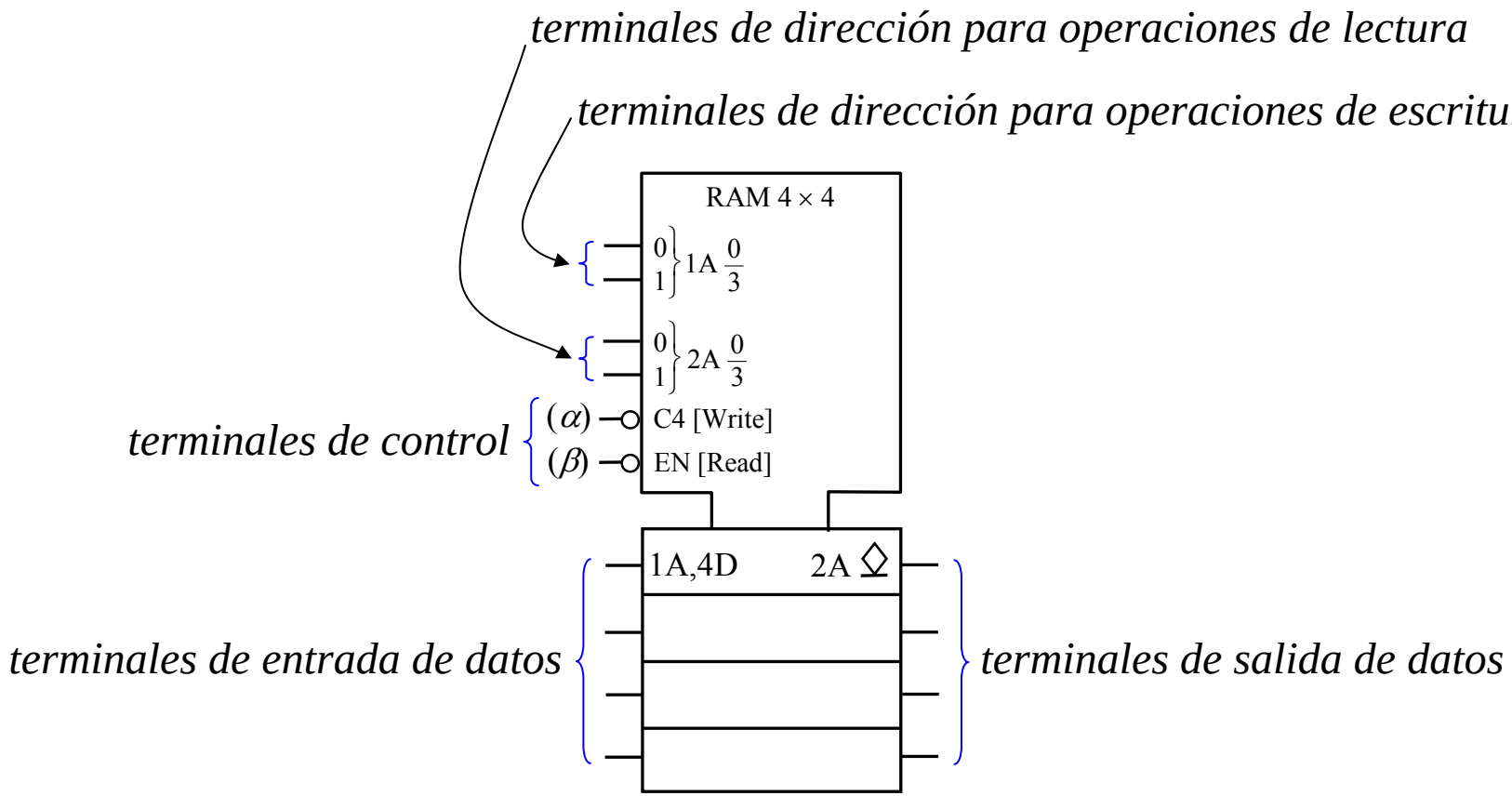


Nota: (β, α) = 0 0 $\rightarrow$ se realiza una operación de escritura (los terminales de salida de datos están en tercer estado)

(β, α) = 0 1 $\rightarrow$ se realiza una operación de lectura

(β, α) = 1 x $\rightarrow$ no se permite realizar operaciones de lectura ni de escritura.

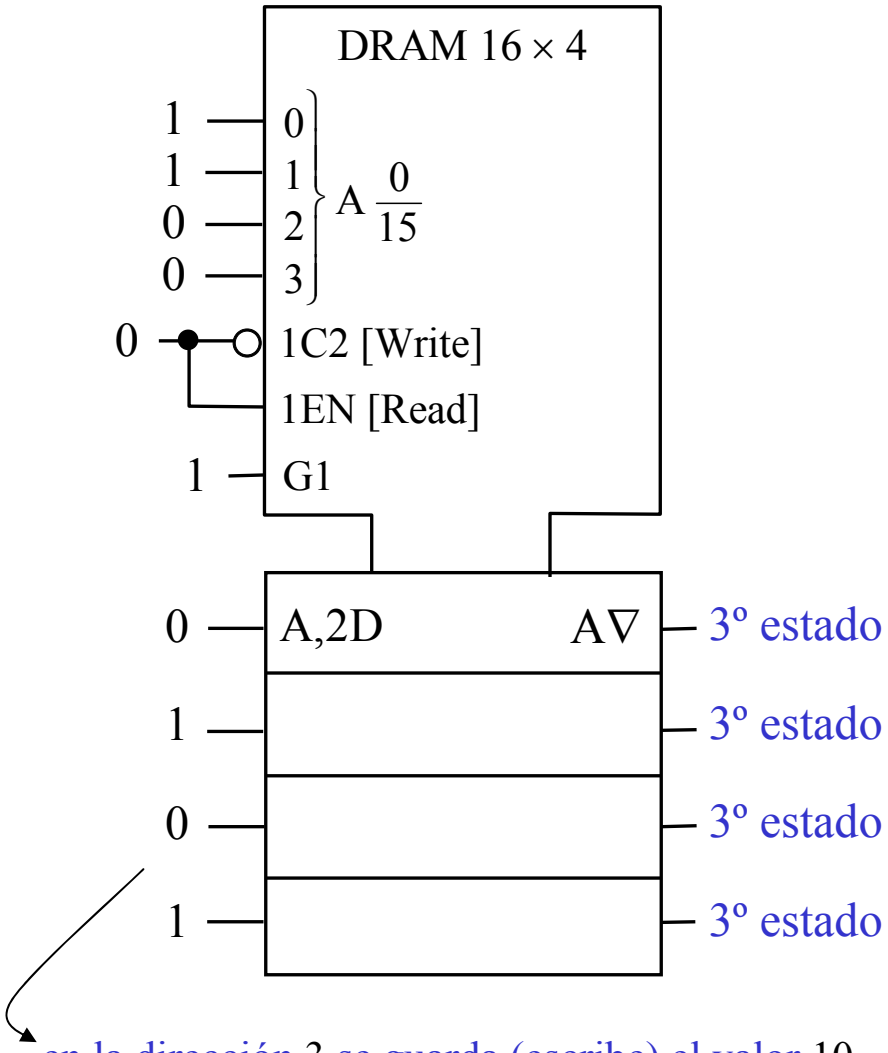
Nota: *Esta memoria permite realizar al mismo tiempo una operación de lectura y una operación de escritura, siempre que no se realicen en la misma dirección.*



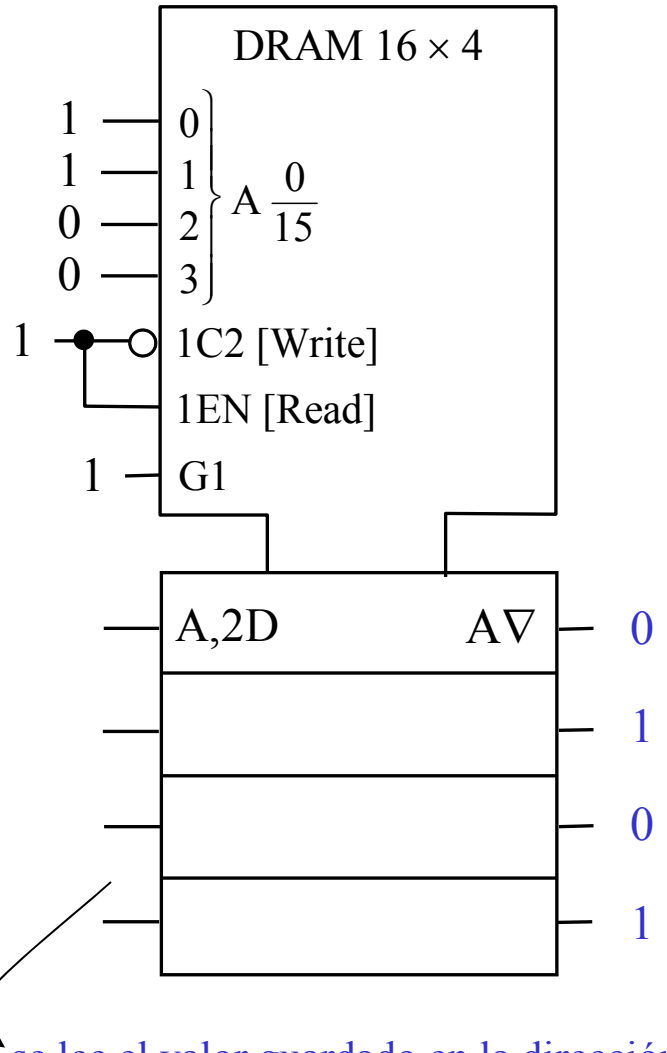
Nota: $(\alpha) = 0 \rightarrow$ se realiza una operación de escritura

$(\beta) = 0 \rightarrow$ se realiza una operación de lectura

Ejemplos:

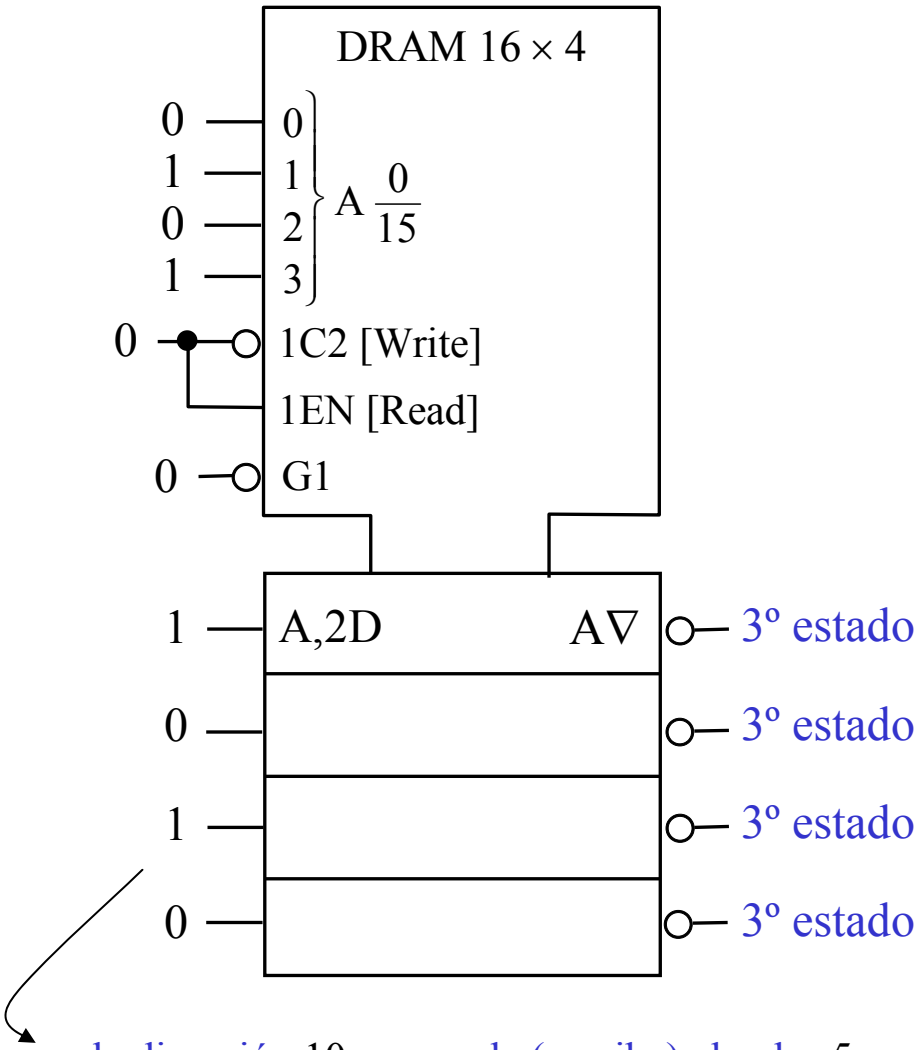


en la dirección 3 se guarda (escribe) el valor 10

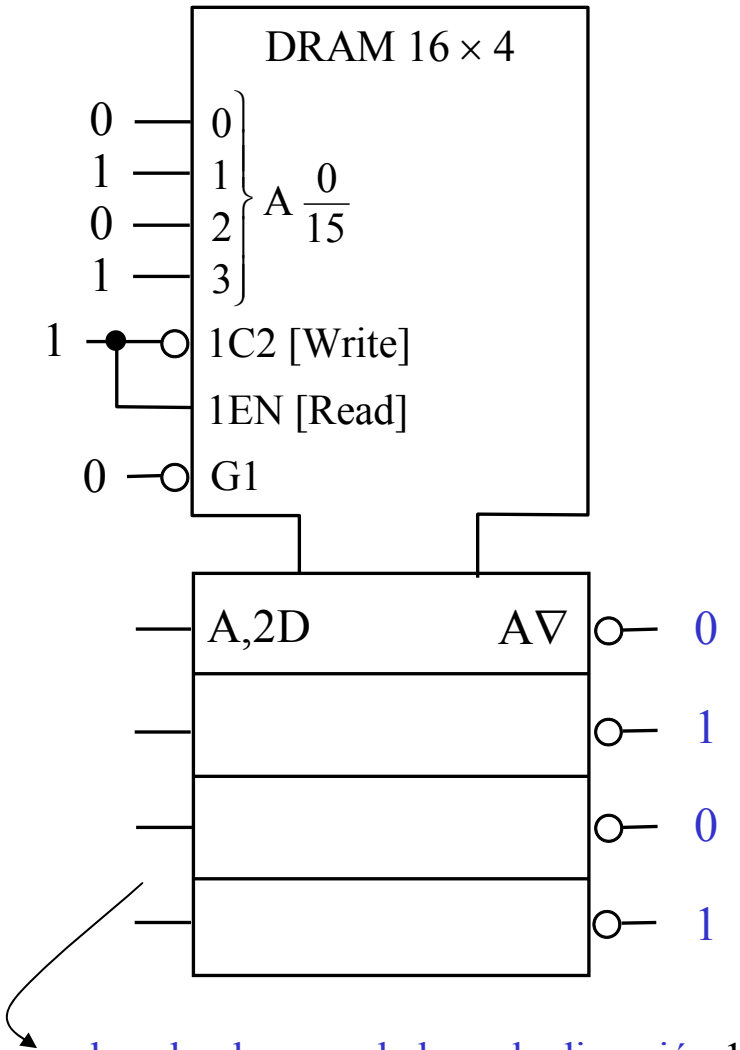


se lee el valor guardado en la dirección 3

Ejemplos:

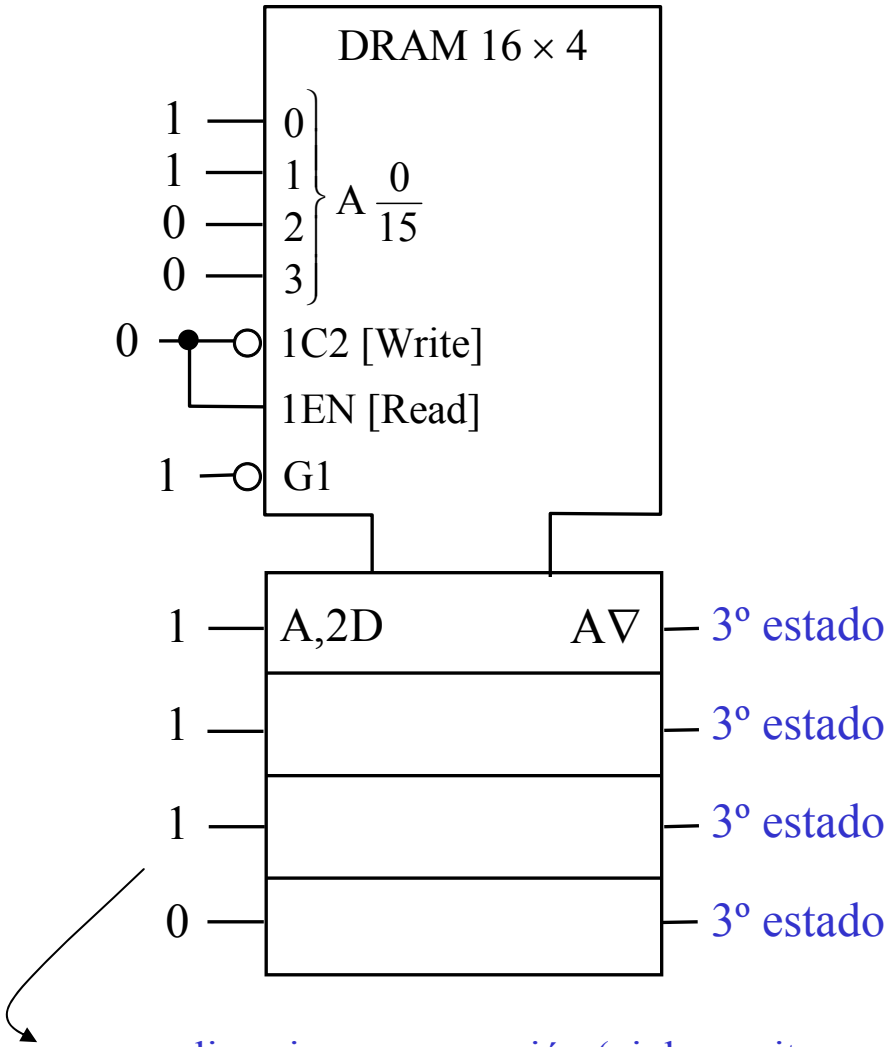


en la dirección 10 se guarda (escribe) el valor 5

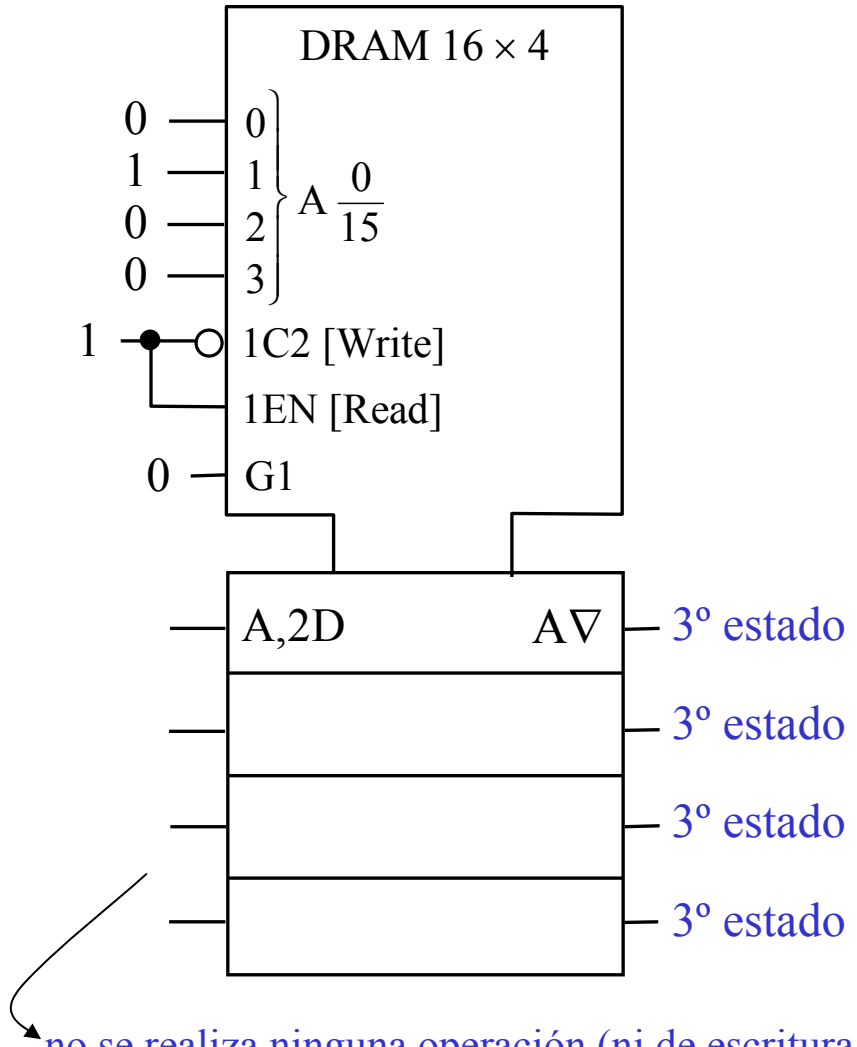


se lee el valor guardado en la dirección 10

Ejemplos:



no se realiza ninguna operación (ni de escritura ni de lectura)



no se realiza ninguna operación (ni de escritura ni de lectura)

Aplicaciones de las memorias:

- ✓ Guardar datos temporalmente (volátiles) o de forma permanente (no volátiles)
- ✓ Implementar funciones
- ✓ Generar señales complejas
- ✓ Hacer de *interface* de dispositivos que operan a diferente velocidad.
- ✓ etc.

Ejemplo de aplicación: Implementación de funciones

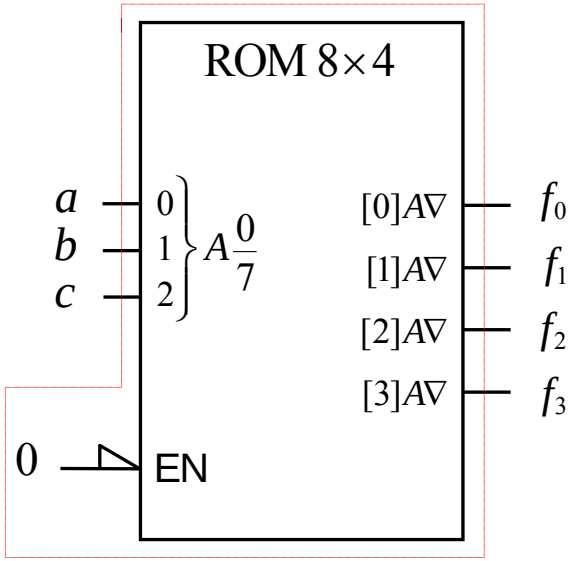
$$f_0(c,b,a) = \sum_3(1,2,4)$$

$$f_1(c,b,a) = \sum_3(0,4,5,6,7)$$

$$f_2(c,b,a) = \sum_3(0,2,3,7)$$

$$f_3(c,b,a) = \sum_3(1,4,6,7)$$

Nota: antes de poner en funcionamiento la memoria hay que guardar en ella la información adecuada. En los terminales de dirección se aplican las señales (*c*,*b*,*a*) presentes en las entradas del circuito a construir. La memoria proporciona en sus terminales de datos los valores que guarda en la dirección que indiquen en cada momento los valores de las entradas *c*, *b* y *a* (los valores proporcionados deben corresponder a los valores de las funciones f_3, f_2, f_1 y f_0 a implementar).



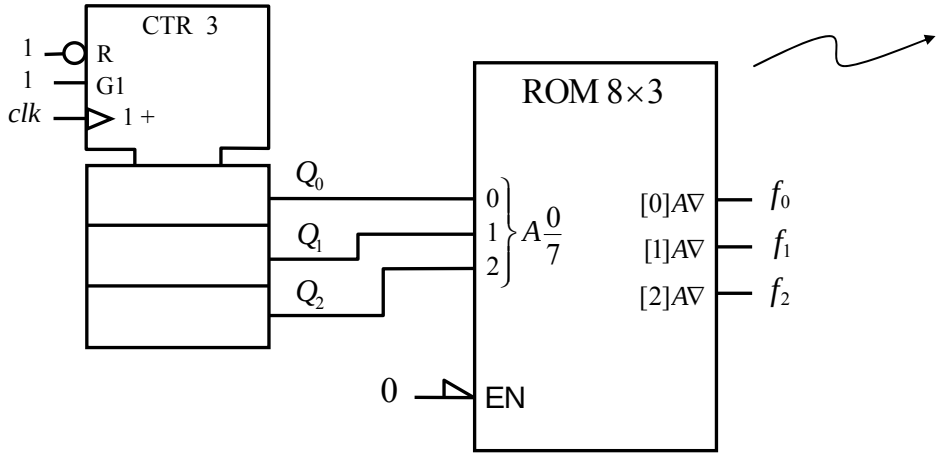
direcciones *contenido memoria*

c	*b*	*a*	f_0	f_1	f_2	f_3
0	0	0	0	1	1	0
0	0	1	1	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	1	0	0
1	1	0	0	1	0	1
1	1	1	0	1	1	1

$f_2(0,1,0)$

contenido de la memoria (valores guardados en la memoria)

Aplicación: Generar señales complejas (periódicas)



dirección	contenido
000	001
001	100
010	010
011	101
100	011
101	000
110	100
111	110