

Tecnologías y sistema de memoria

Sistemas con Microprocesadores

Mg.Ing. Esteban Volentini (evolentini@herrera.unt.edu.ar)

<http://www.microprocesadores.unt.edu.ar/procesadores>

Cronograma

Actividad	Inicio	Descripción	Completa
Presentación	24/08	Presentación y reglamento	✓
Tema 1	24/08	Estructura de las computadoras	✓
Tema 2	26/08	Proyecto con un microcontrolador	✓
Tema 3	31/08	Descripción funcional del ISA RV32IM	✓
Tema 4	07/09	Programación en lenguaje ensamblador	✓
Tema 5	18/09	Diseño del camino de datos de un procesador	✓
Tema 6	30/09	Diseño de la unidad de control del procesador	✓
Tema 7	05/10	Tecnologías y sistema de memoria	←
Tema 8	09/10	Dispositivos de entrada/salida	
Tema 9	19/10	Entradas/salidas digitales (GPIO)	
Tema 10	28/10	Generalidades de las interrupciones	
Tema 11	04/10	Interrupciones y excepciones en RISC-V	
Tema 12	13/10	Manejo de señales con temporizadores	
Tema 13	18/11	Transferencias de datos	
Proyectos	30/11	Seminarios de proyectos	

Arquitectura de una computadora



- ▶ La memoria almacena instrucciones y datos
- ▶ El procesador ejecuta las instrucciones para transformar los datos
- ▶ La entrada/salida permite la comunicación con el mundo exterior

Memoria principal y secundaria

- ▶ La memoria principal esta gestionada en forma directa por el procesador
 - ▶ El tiempo de acceso es el mismo para cualquier celda, independiente de la secuencia previa de acceso (Random)
 - ▶ Se utiliza para almacenar las instrucciones y los datos
- ▶ La memoria secundaria se gestiona como un periférico de entrada/salida
 - ▶ El tiempo de acceso puede variar en función de la secuencia previa de lecturas y/o escrituras
 - ▶ Se utiliza para almacenar bloques de datos que se transfieren desde y hacia la memoria principal cuando es necesario

La memoria principal

- ▶ Funciona como una gran tabla unidimensional
- ▶ La dirección de memoria es el número de fila, es decir el índice en la tabla
- ▶ Cada fila puede almacenar un valor de tamaño fijo (n bits)
- ▶ La celda básica utilizada para almacenar cada bit determina las características de la memoria

0	n bits
1	n bits
2	n bits
3	n bits
4	n bits
5	n bits
6	n bits

...

Memorias de lectura y escritura

- ▶ Son memorias donde la lectura y la escritura demoran la misma cantidad de tiempo
- ▶ SRAM: Static Random Access Memory
 - ▶ La celda de almacenamiento es un flip-flop
 - ▶ El contenido se mantiene indefinidamente con V_{cc}
 - ▶ Baja densidad, alto consumo, cara, rápida
- ▶ DRAM: Dynamic Random Access Memory
 - ▶ La celda básica de almacenamiento es un capacitor
 - ▶ Necesita ser “refrescada” regularmente
 - ▶ Alta densidad, bajo consumo, barato, lento

Memorias de solo lectura

- ▶ Son memorias donde no se puede escribir, o la escritura requiere un proceso diferente que demora más tiempo que la lectura
- ▶ ROM: Read Only Memory
 - ▶ La celda de almacenamiento es una conexión eléctrica
 - ▶ El valor almacenado se define al fabricar el chip de memoria
- ▶ PROM: Programmable Read-Only Memory
 - ▶ La celda básica de almacenamiento es un diodo
 - ▶ Se puede programar una sola vez quemando ciertos diodos
- ▶ EPROM: Erasable Programmable Read-Only Memory
 - ▶ La celda básica es un “pozo de energía potencia”
 - ▶ Se puede programar eléctricamente y borrar con luz UltraVioleta
- ▶ EEPROM: Electrically Erasable Programmable Read-Only Memory
 - ▶ La celda básica es un “pozo de energía potencia”
 - ▶ Se puede programar y borrar eléctricamente una cantidad limitada de veces

Diagrama Lógico RAM

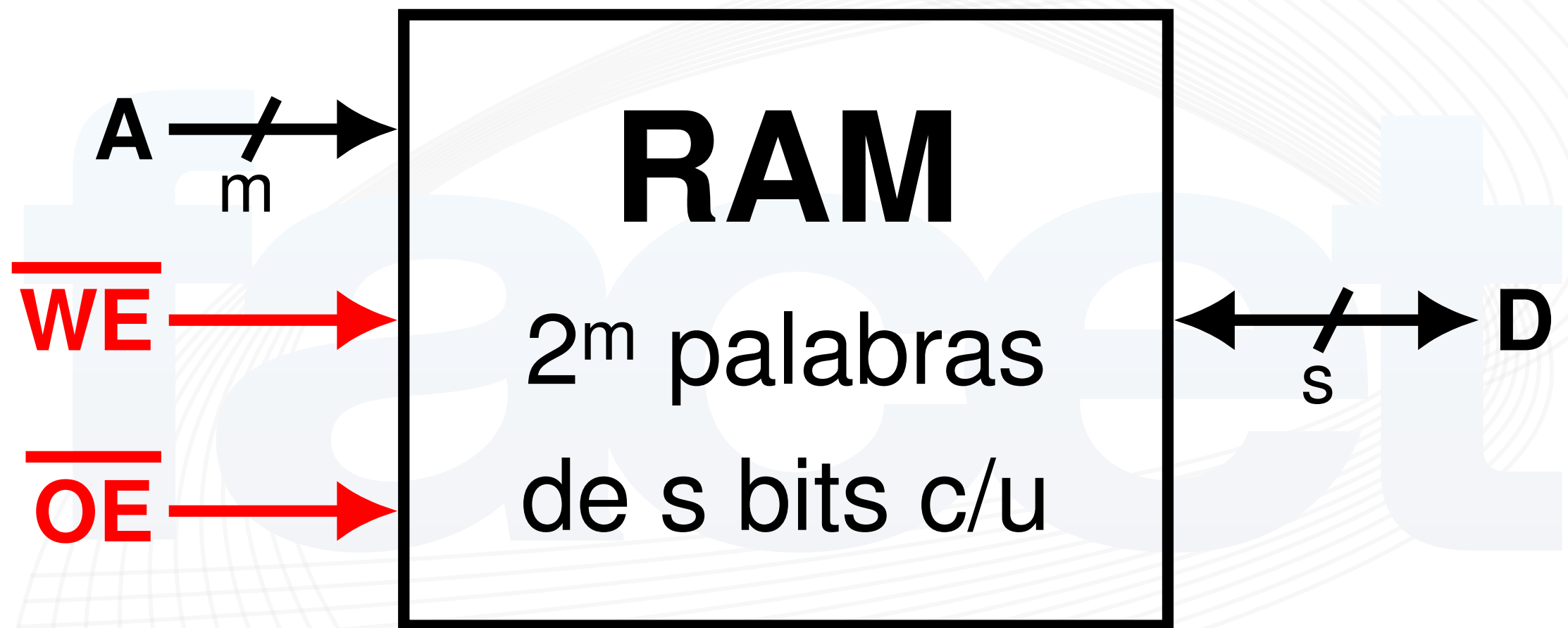
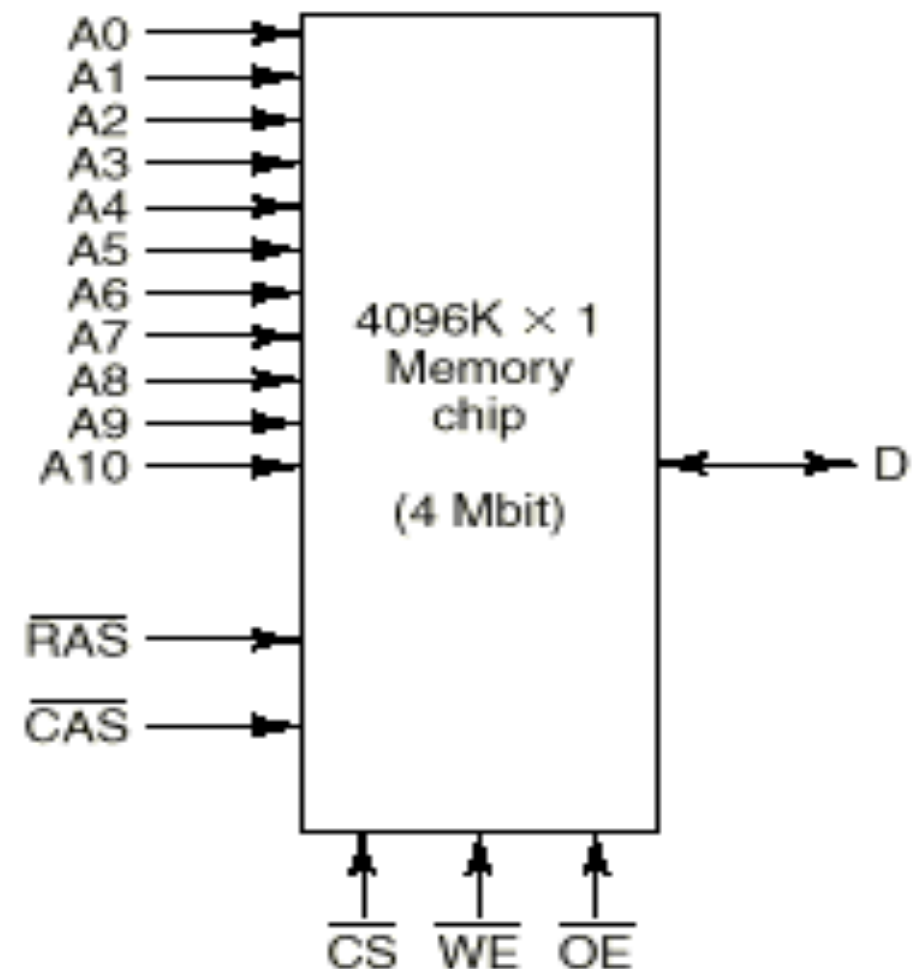
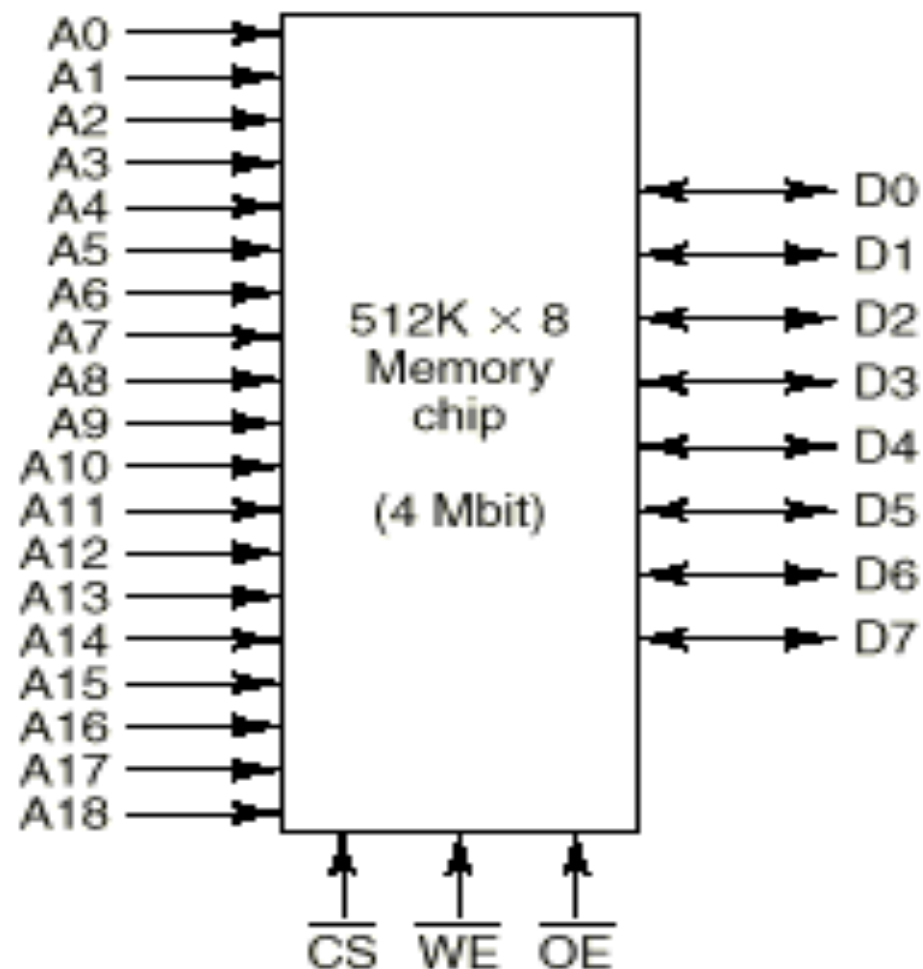


Diagrama Lógico RAM

- ▶ **Señales de control:** Write Enable y Output Enable
 - ▶ Generalmente se activan en “Low” ($\overline{WE}$, $\overline{OE}$)
 - ▶ $\overline{OE}$ permite ahorrar terminales, unificando Din y Dout
- ▶ **WRITE:** $\overline{WE}$ se activa (0), $\overline{OE}$ se desactiva (1)
 - ▶ D funciona como entrada de datos
- ▶ **READ:** $\overline{WE}$ se desactiva (1), $\overline{OE}$ se activa (0)
 - ▶ D es la salida de datos
- ▶ **INDEFINIDO:** Si tanto $\overline{WE}$ como $\overline{OE}$ se activan:
 - ▶ Resultado no definido. ¡¡No hacerlo!!

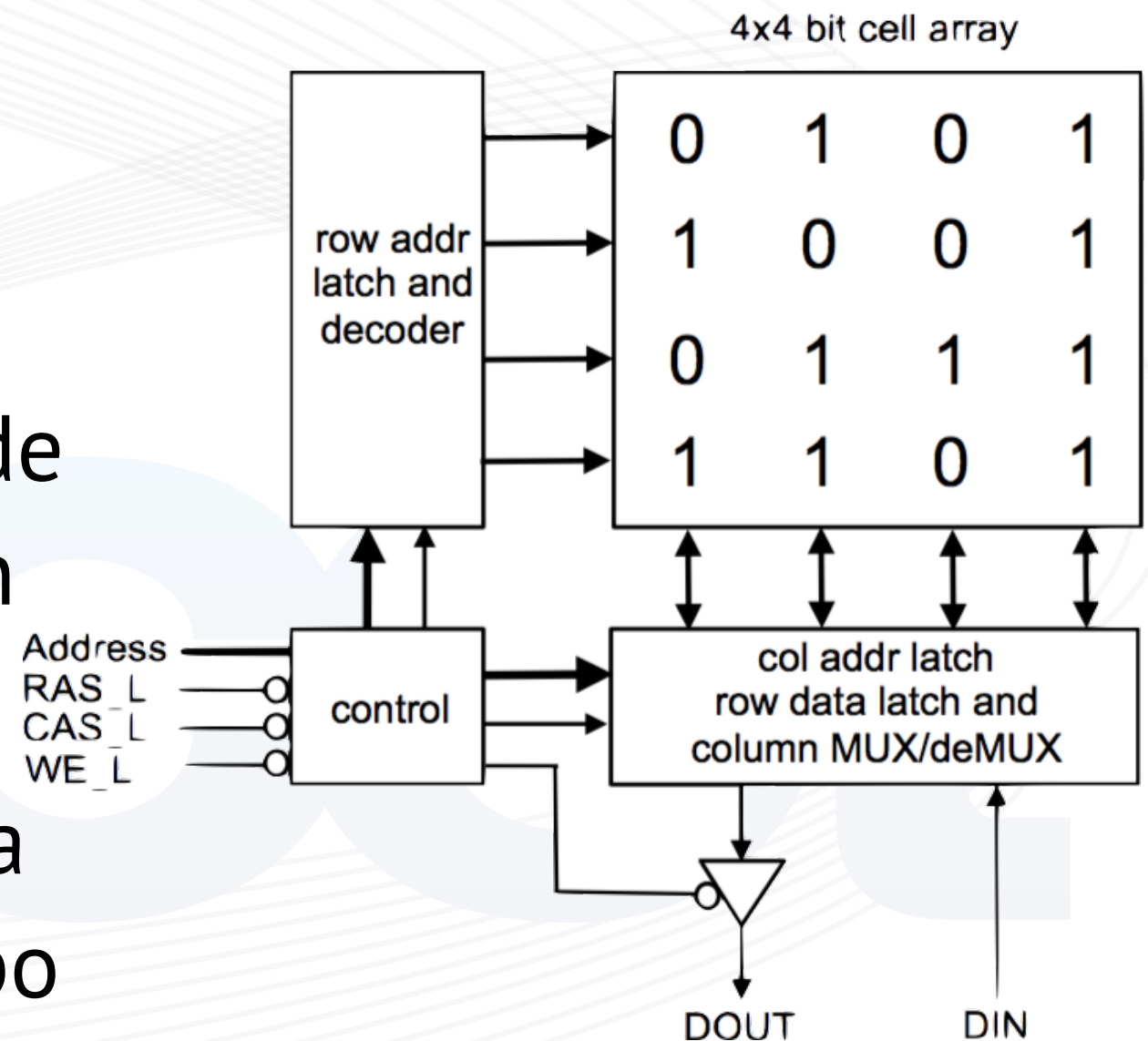
Memorias SRAM y DRAM

- ¿Cómo se puede verificar el tamaño en



Direcciones partidas en DRAM

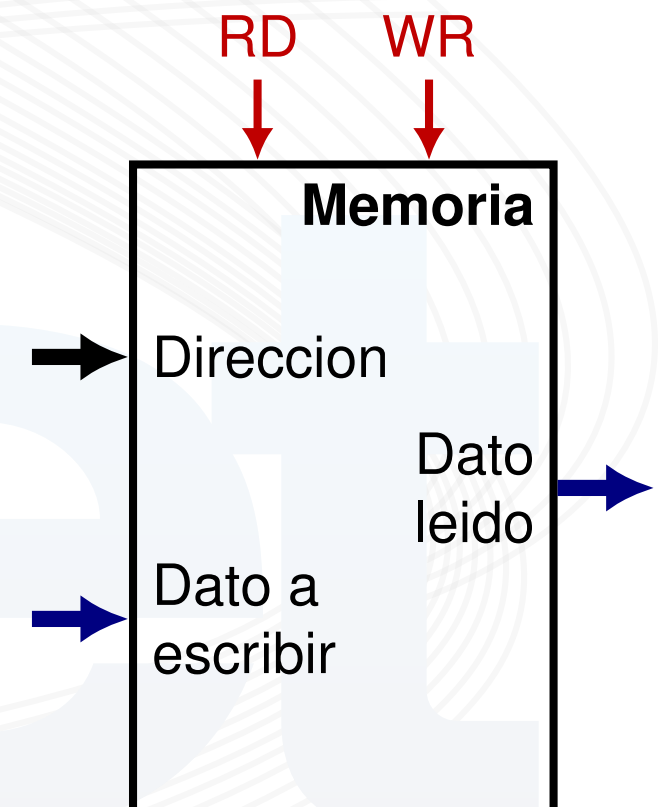
- ▶ Las memorias se construyen en forma de cuadrados
- ▶ Los tiempos de acceso de la celda DRAM permiten partir la dirección
- ▶ Usar esta técnica en una SRAM afectaría el tiempo de acceso total de la memoria



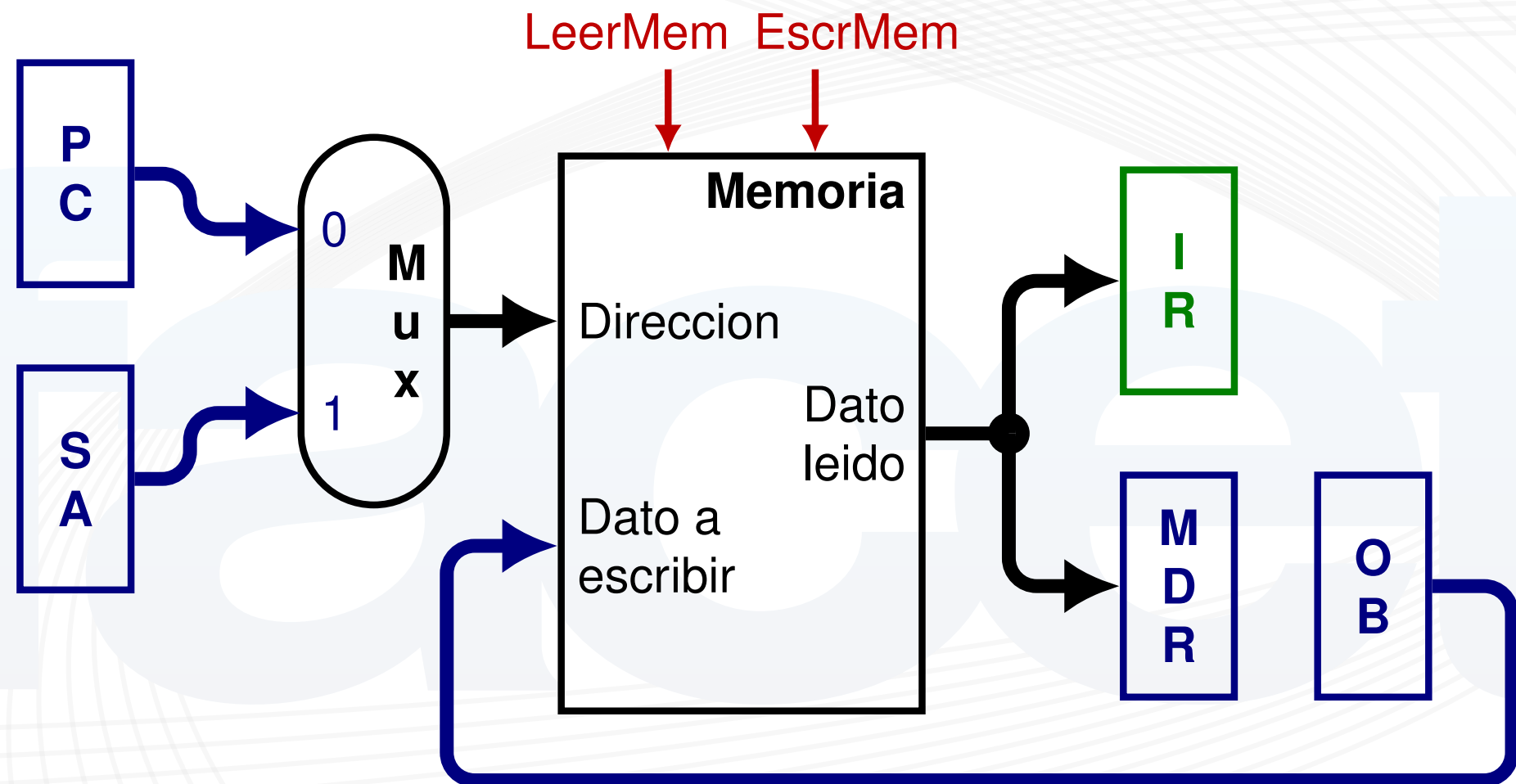
Componentes secuenciales (7/7)

► Funcionamiento de la Memoria “Ideal”

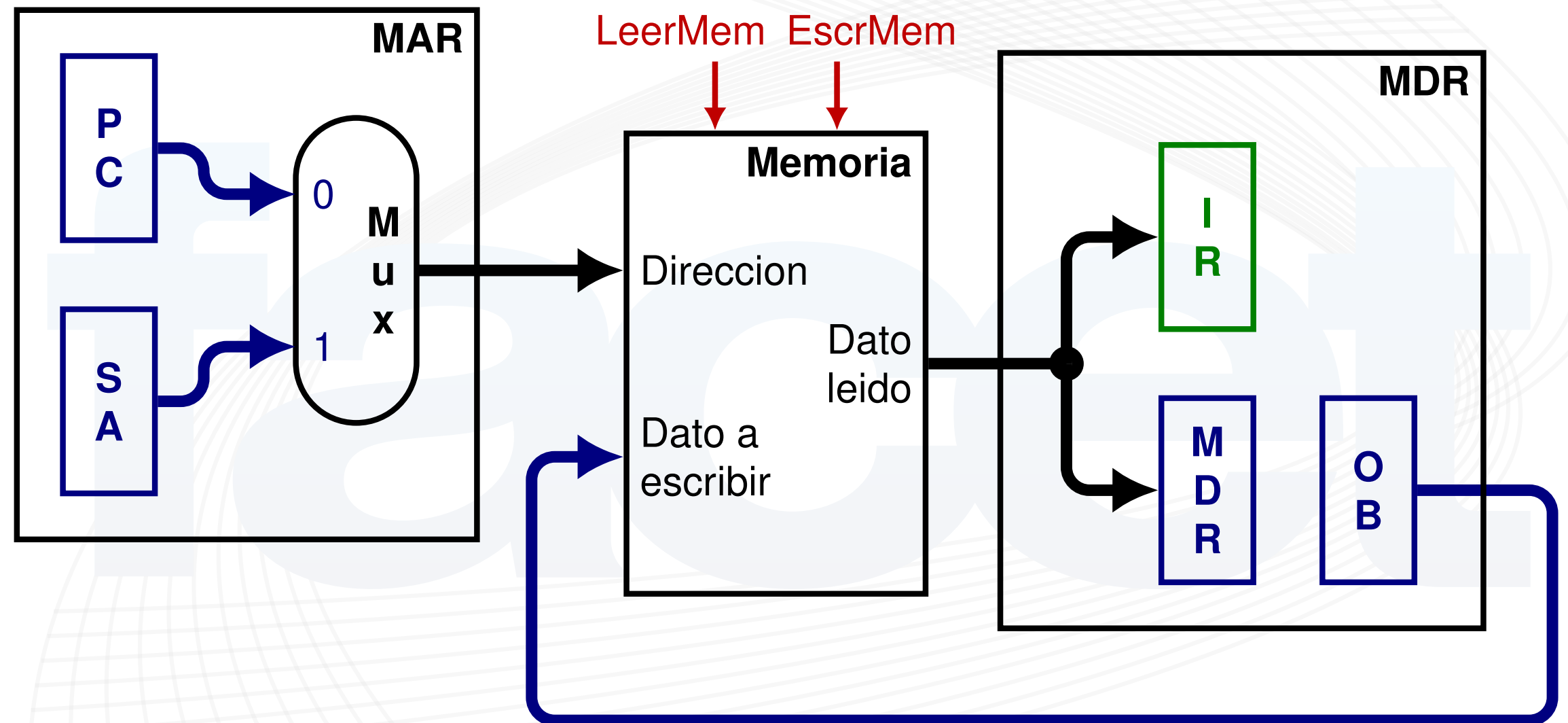
- ▶ Cuando la señal de control de lectura está activa, se muestra en la salida el contenido apuntado por la dirección de entrada, después de un cierto tiempo de acceso
- ▶ Cuando la señal de control de escritura está activa, se guarda en el contenido apuntado por la dirección el valor en la entrada de datos, después de un cierto tiempo de acceso
- ▶ Se podría considerar como un banco de registros externo más grande y más lento



Conexión de la memoria



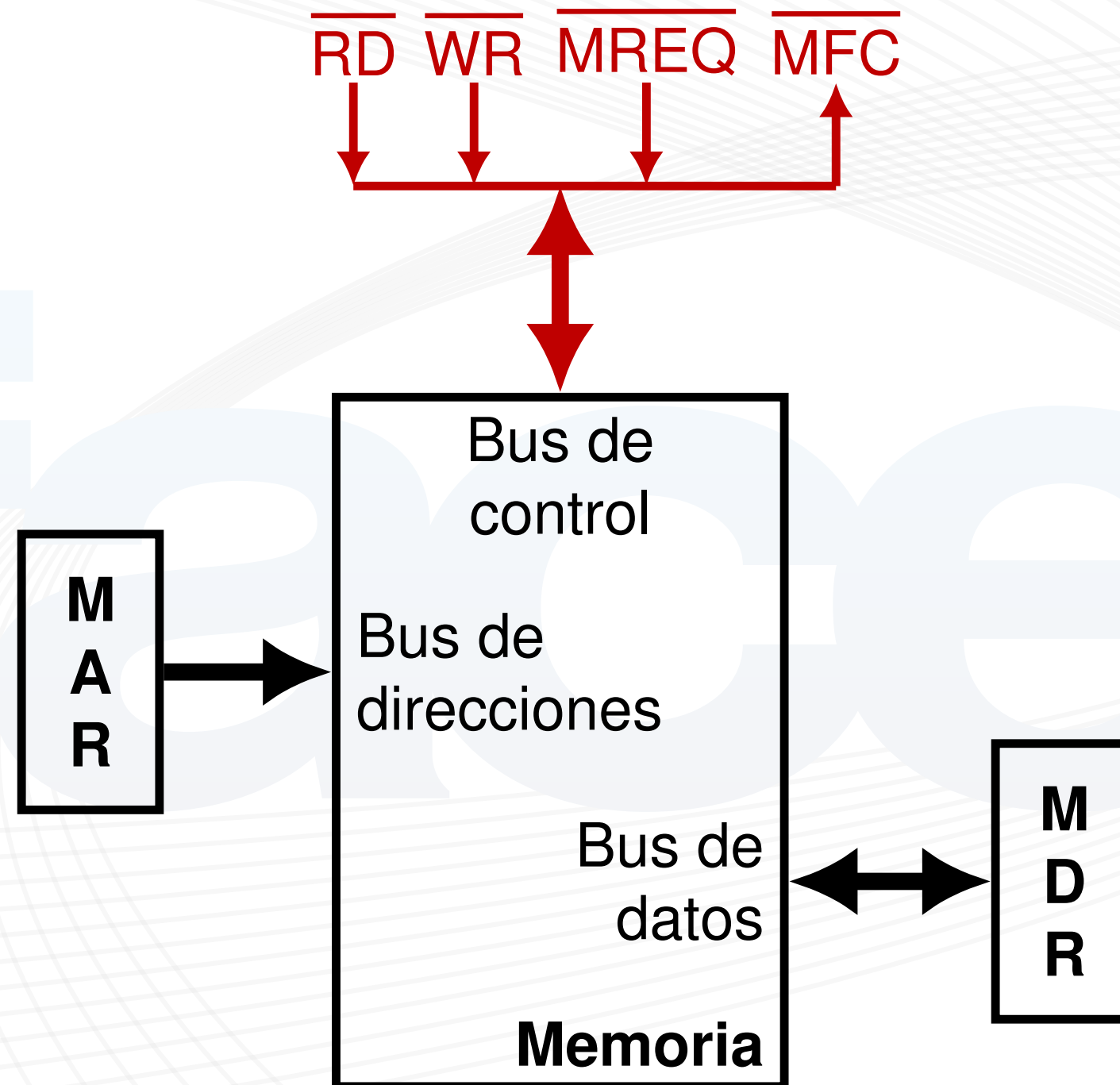
Conexión de la memoria



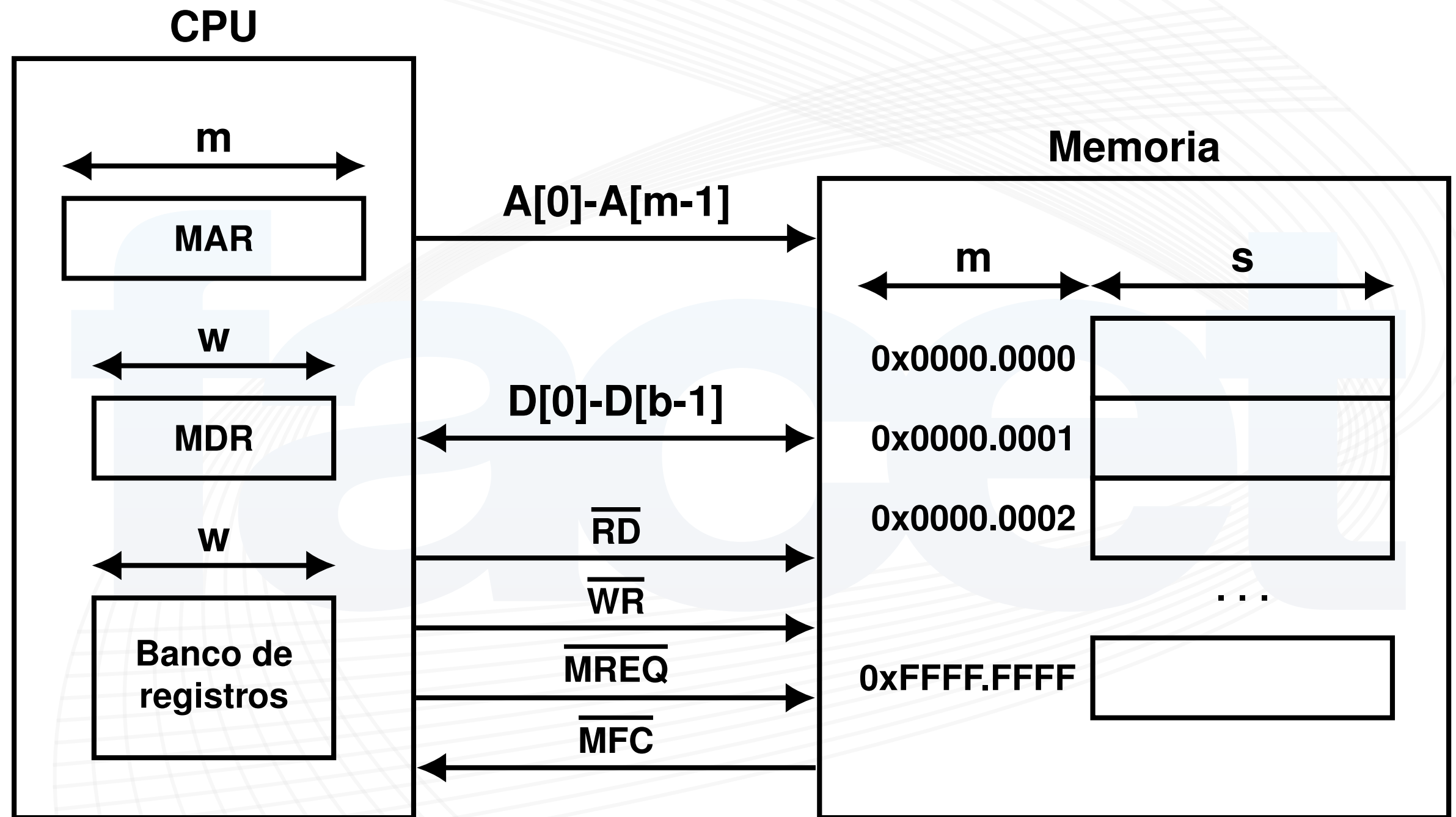
Se podrían hacer algunas mejoras

- ▶ En procesador moderno cada conjunto de señales son, en general, de 32 bits
 - ▶ Las señales DatoLeido y DatoAEscribir no se utilizan al mismo tiempo
 - ▶ Utilizando salidas en tercer estado se pueden combinar estas señales y disminuir la cantidad de conexiones
- ▶ En este caso no se deben activar LeerMem y EscrRead al mismo tiempo
 - ▶ No es una combinación válida
 - ▶ Es preferible utilizar RD/WR y MREQ
 - ▶ En estas señales no hay una combinación inválida

Conexión de la memoria



Conexión de la memoria



Conexión de la memoria

- ▶ Bus de direcciones (Address Bus, $A_0 - A_{m-1}$)
 - ▶ Contiene la dirección de memoria para leer o escribir los datos
 - ▶ 2^m es la dimensión del “espacio de direcciones”
- ▶ Bus de datos (Data bus, $D_0 - B_{b-1}$)
 - ▶ Contiene los datos a escribir o que se leen de memoria
 - ▶ w es el “ancho de palabra” del procesador
 - ▶ s es el “ancho” de una dirección de memoria
 - ▶ b es el “ancho” del bus de datos
- ▶ Bus de control (RD/WR, /MREQ, /MFC)
 - ▶ Contiene las señales de control necesarias para la operación
 - ▶ La señal /MFC puede no existir, en ese caso la operación se completa en base a tiempos prefijados

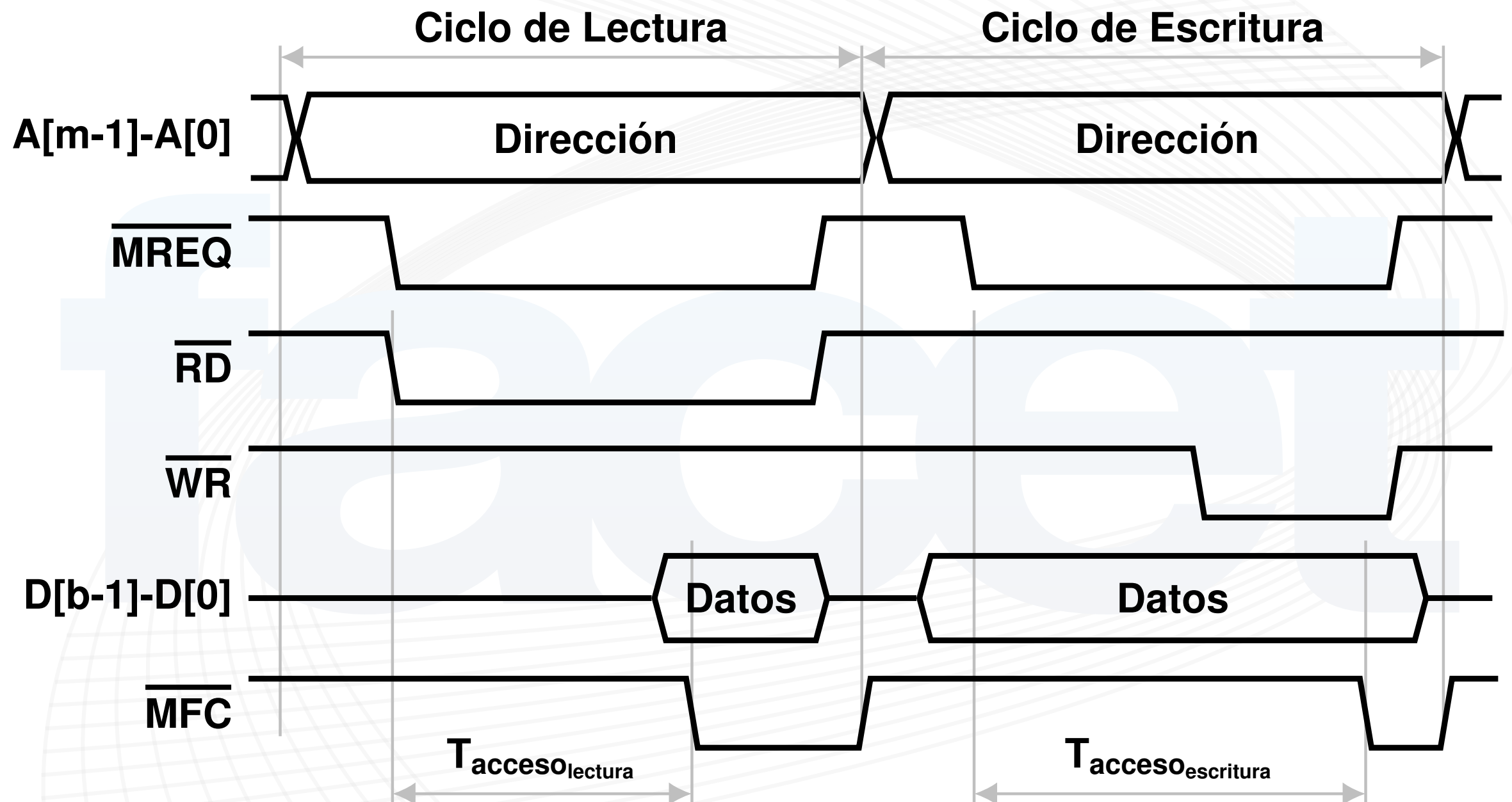
Lectura de memoria

- ▶ El procesador envía por las líneas $A[0]$ a $A[m-1]$ la dirección de memoria almacenada en el registro MAR
- ▶ También fija el valor adecuado a la línea $/RD$
- ▶ Finalmente, activa la línea $/MREQ$ para iniciar la operación
- ▶ La memoria recupera el dato almacenado en la dirección recibida por las líneas $A[x]$ y lo envía por las líneas $D[x]$
- ▶ A continuación activa la línea $/MFC$ (si existe)
- ▶ El procesador desactiva la línea $/MREQ$ para terminar la operación
- ▶ La memoria desactiva la línea $/MFC$ (si existe)

Escritura en memoria

- ▶ El procesador envía por las líneas $A[0]$ a $A[m-1]$ la dirección de memoria almacenada en el registro MAR
- ▶ Al mismo tiempo envía por las líneas $A[0]$ a $A[b-1]$ el dato almacenado en el registro MDR
- ▶ También fija el valor adecuado a la línea $/WR$
- ▶ Finalmente, activa la línea $/MREQ$ para iniciar la operación
- ▶ La memoria almacena el dato recibido por las líneas $D[x]$ en la dirección recibida por las líneas $A[x]$
- ▶ Una vez completada la escritura activa la línea $/MFC$ (si existe)
- ▶ El procesador desactiva la línea $/MREQ$ para terminar la operación
- ▶ La memoria desactiva la línea $/MFC$ (si existe)

Diagramas de tiempos



Espacio de memoria

- ▶ Es la máxima cantidad de memoria que se puede conectar a un procesador
- ▶ Esta dado por su capacidad de direccionamiento
- ▶ En general no todo se usa
- ▶ La ubicación de las partes usadas forman el Mapa de Memoria

Direccionamiento a palabra

- ▶ En general el ancho de la memoria debería ser igual que el ancho de la palabra del procesador
- ▶ Pero no se pueden leer o escribir datos más chicos que una fila de memoria
- ▶ Sin embargo en los procesadores mas modernos los programas todavía utilizan muchos datos pequeños (de 8 bits o incluso de menos)
- ▶ Se desperdicia mucha memoria al almacenar datos pequeños en filas de memoria con muchos bits

Direccionamiento a bytes

- ▶ En consecuencia lo mas común es mantener la memoria organizada en bytes
- ▶ De esta forma se pueden operar con datos de un byte como mínimo
- ▶ Para datos de mayor tamaño se deberían realizar varias lecturas sucesivas
- ▶ Esta opción tiene problemas de velocidad

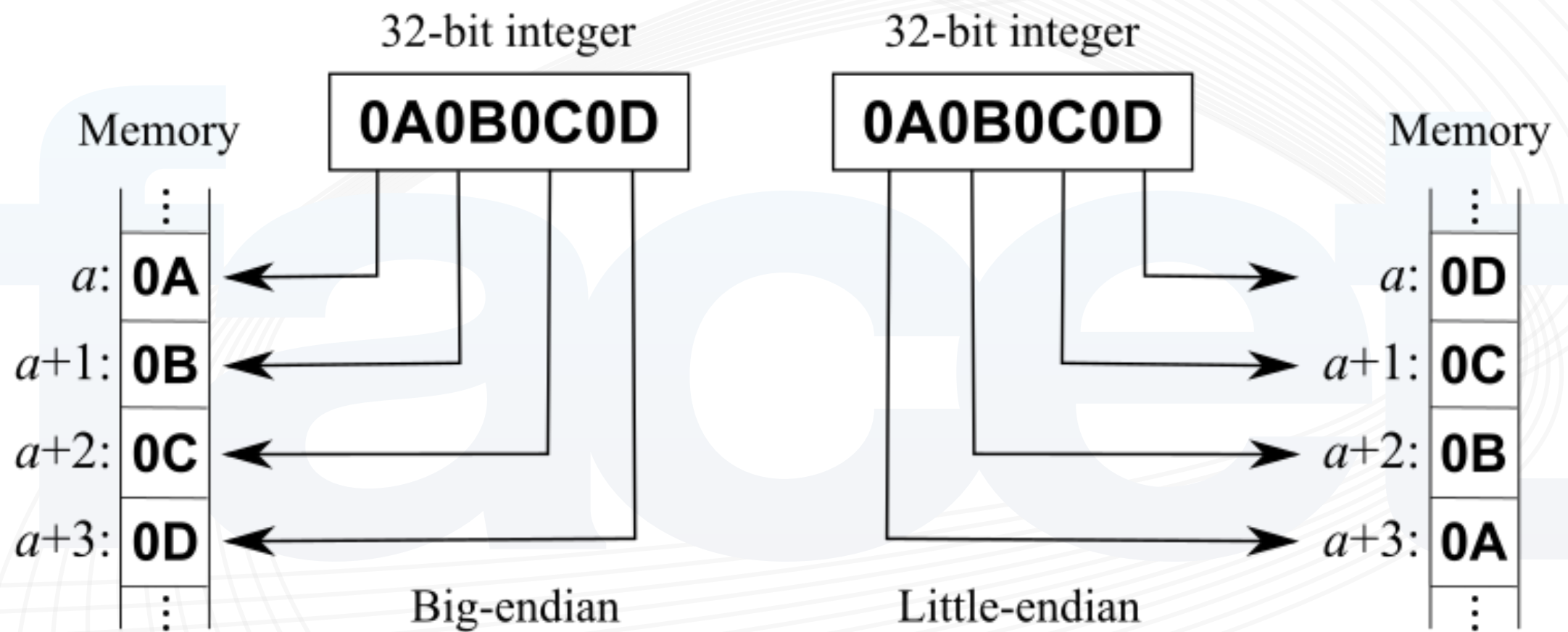
Direccionamiento a bytes

- ▶ Para mejorar la performance se lee una palabra completa en cada acceso
- ▶ Cada byte de la palabra tiene una dirección y se puede individualizar
- ▶ Cada palabra se representa por la dirección del primer byte que la compone
- ▶ Cada palabra ocupa más de una dirección de memoria

Big-endian y Little-endian

- ▶ En una memoria organizada por bytes los números que ocupan más de un byte se pueden almacenar de dos formas diferentes:
- ▶ Big-endian
 - ▶ El Byte más significativo en la dirección baja
 - ▶ Es más fácil para leer (humanos)
- ▶ Little-endian
 - ▶ El Byte menos significativo en la dirección baja
 - ▶ Es más fácil para operar (maquinas)

Big-endian y Little-endian



Presentación de un CPU de 32 bits

- ▶ Consideraremos un procesador de 32 bits con un bus de conexiones simplificado
 - ▶ Este tipo de buses se utilizó en procesadores de hasta 16 bits y ya no se utilizan en la actualidad
 - ▶ Sin embargo es más fácil de entender y no hay diferencias tan significativas en la operación con los buses actuales
- ▶ Líneas de Datos: Dependen de cuántos bits se almacena en cada dirección de memoria
 - ▶ Por razones de velocidad la organización física de la memoria será en palabras de 32 bits
 - ▶ Por lo tanto se utilizarán 32 líneas de datos, de D0 a D32

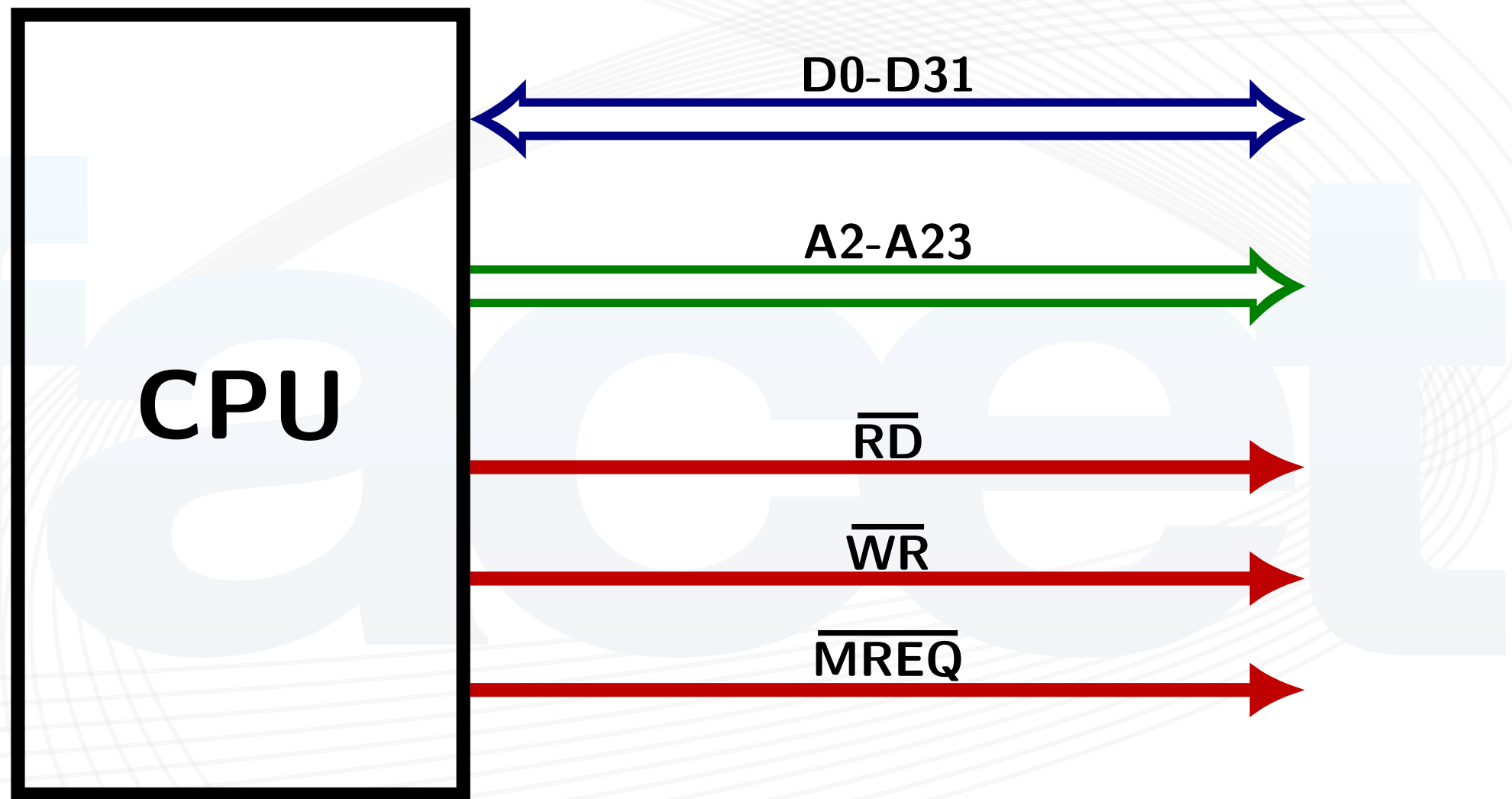
Presentación de un CPU de 32 bits

- ▶ Líneas de Direcciones: Dependen de la capacidad de direccionamiento del procesador, en este caso deberían ser 32 bits
- ▶ Sin embargo dado que se utilizará direccionamiento a bytes en una memoria organizada físicamente por palabras los bits A0 y A1 no existen en el bus
- ▶ Muchas veces se utiliza en el bus de direcciones una cantidad menor de líneas que las que corresponden según la capacidad teórica de direccionamiento
- ▶ Finalmente se utilizarán 22 líneas de direcciones, de A2 a A23 con un espacio de direcciones real de 16 MiB

Presentación de un CPU de 32 bits

- ▶ Líneas de control: dependen de las operaciones que se pueden realizar en el bus, en este caso se utilizarán tres líneas de control:
 - ▶ Una línea /RD para indicar que se realiza una lectura de datos
 - ▶ Una línea /WR para indicar que se realiza una escritura de datos
 - ▶ Una línea /MREQ para validar la dirección de memoria presente en el bus de direcciones
- ▶ No se utilizará la línea /MFC para que la memoria confirme que se completó la operación
 - ▶ La operación dura un tiempo fijo determinado por el procesador y la memoria debe adecuarse a esta especificación

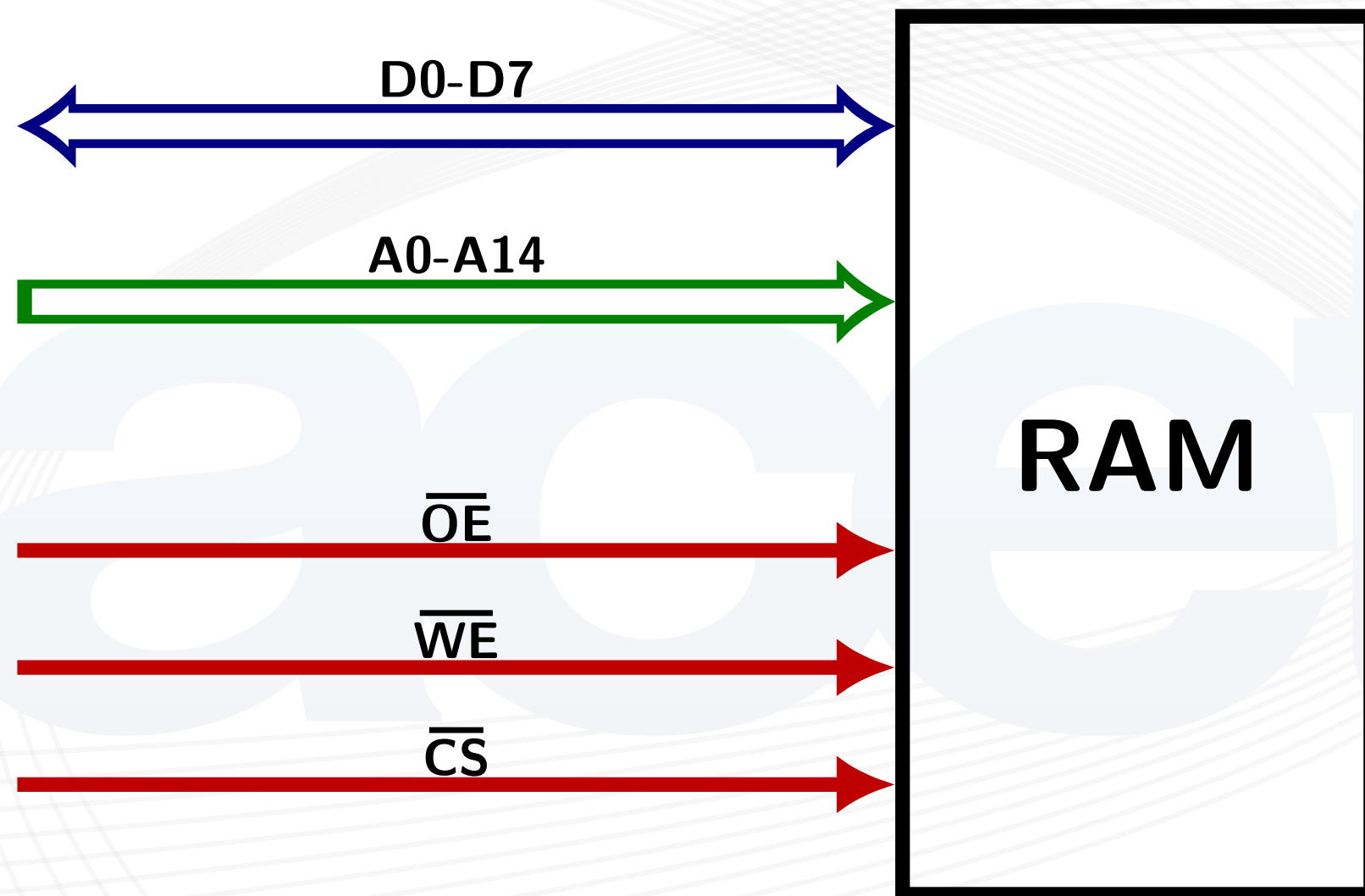
Presentación de un CPU de 32 bits



Presentación de la RAM 62C256

- ▶ Memoria de 256 Kibits organizada como 32 Ki x 8 bits
- ▶ Líneas de Direcciones: Depende de cuántas direcciones tenga la memoria, en este caso son 15 líneas, de A0 a A14
- ▶ Líneas de Datos: Depende de cuántos bits se almacena en cada dirección, en este caso son 8 líneas, de D0 a D7
- ▶ Líneas de control: dependen de las operaciones que se pueden realizar en la memoria, en este caso se utilizan tres líneas:
 - ▶ Una línea /OE para que la memoria escriba en el bus de datos el contenido de la dirección de memoria presente en el bus de direcciones
 - ▶ Una línea /WE para indicar a la memoria que almacene el contenido del bus de datos en la dirección de memoria presente en el bus de direcciones
 - ▶ Una línea /CS para validar la dirección de memoria presente en el bus de direcciones y que la memoria comience la decodificación de la misma

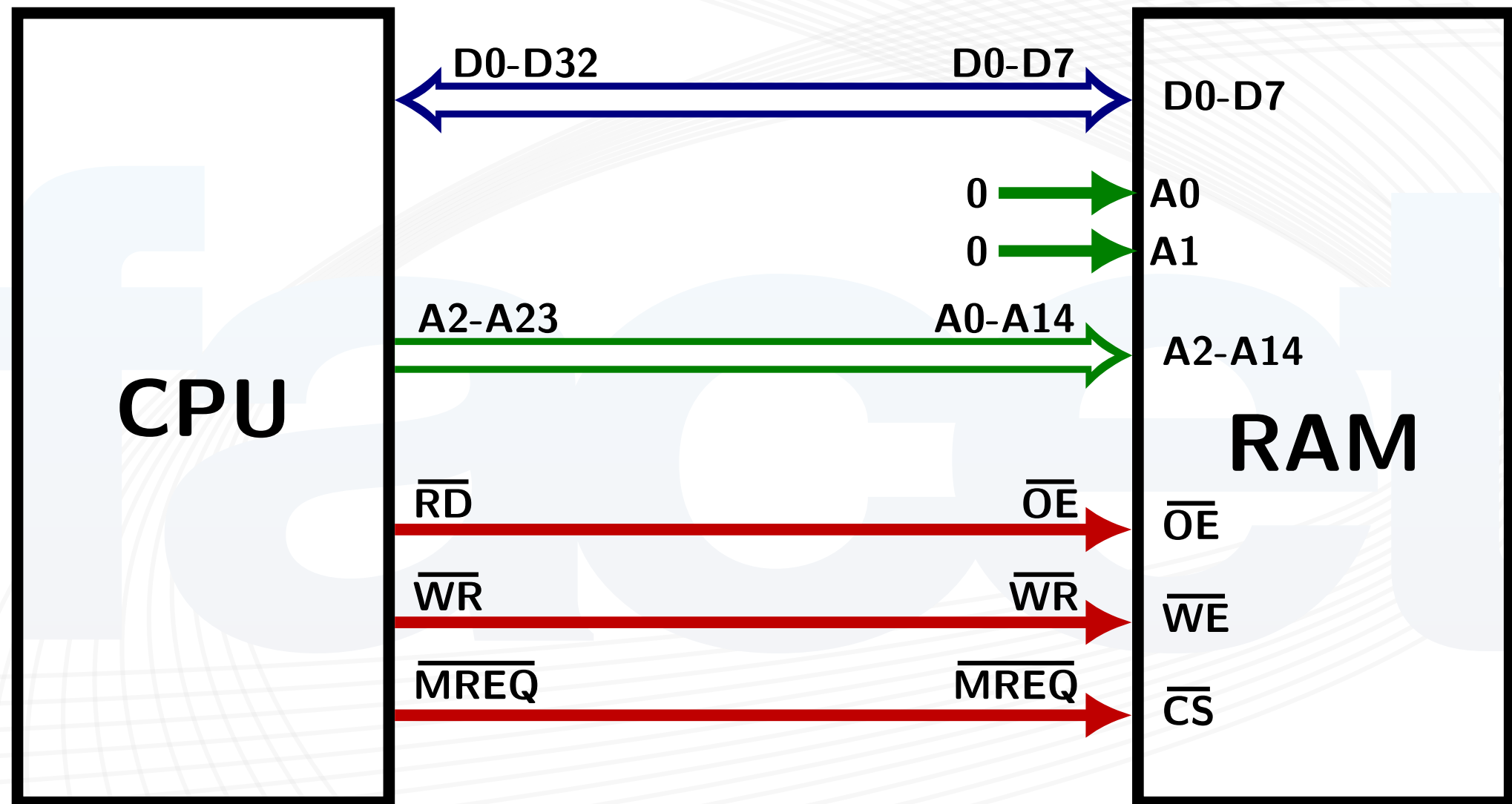
Presentación de la RAM 62C256



¿Y como se conectan?

- ▶ Se podría conectar las líneas con funciones compatibles
- ▶ El procesador tiene 32 líneas de datos y la memoria solo 8
- ▶ El procesador tiene 22 líneas de direcciones y la memoria solo 14
- ▶ Pero de estas solo 12 líneas tienen el mismo nombre:
 - ▶ En el procesador no existen A0 y A1
 - ▶ En la memoria no existen de A15 a A23
- ▶ Una posible solución sería
 - ▶ Se podría utilizar solo de D0 a D7 y de A2 a A14
 - ▶ Dejar sin usar de D8 a D31 y de A15 a A23
 - ▶ Conectar A0 y A1 a masa

Ejemplo de aplicación

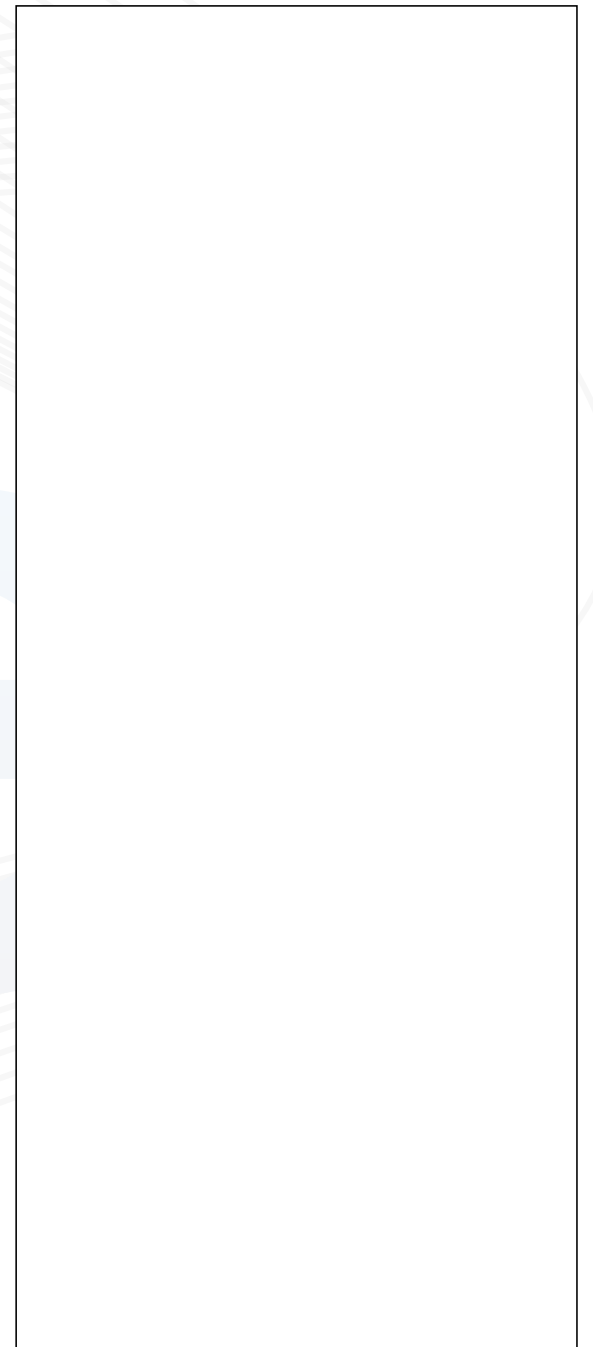


Mapas de memoria

- ▶ Se llama espacio de direcciones (o de memoria) al conjunto de todas las direcciones posibles a las que podría acceder un procesador
- ▶ En el ejemplo es 2^{22} , es decir 4 MiB
- ▶ No siempre se usa la totalidad del espacio disponible, puede no hacer falta
- ▶ Los mapas de memoria muestran las partes del espacio de direcciones a las que se conecta Memoria Física
- ▶ En nuestro ejemplo podemos representar en un mapa de memoria en qué direcciones está conectada la RAM

0x000000

0xFFFFF



Ejemplo de aplicación

- ▶ Al acceder a la palabra 0x000000 la memoria RAM entrega un byte en las líneas D0 a D7
- ▶ Este byte es el que corresponde a la dirección 0x000000

0x000000
0x000001

RAM 0 (0x00000)

0xFFFFF

Ejemplo de aplicación

- ▶ Al acceder a la palabra 0x000000 la memoria RAM entrega un byte en las líneas D0 a D7
- ▶ Este byte es el que corresponde a la dirección 0x000000
- ▶ Pero las líneas D8 a D31 no tienen un valor definido ya que no hay memoria conectada en esas líneas
- ▶ Es decir que los bytes 0x000002 al 0x000003 no existen

0x000000

0x000001

0x000002

0x000003

0x000004

RAM 0 (0x00000)

Sin Memoria

Sin Memoria

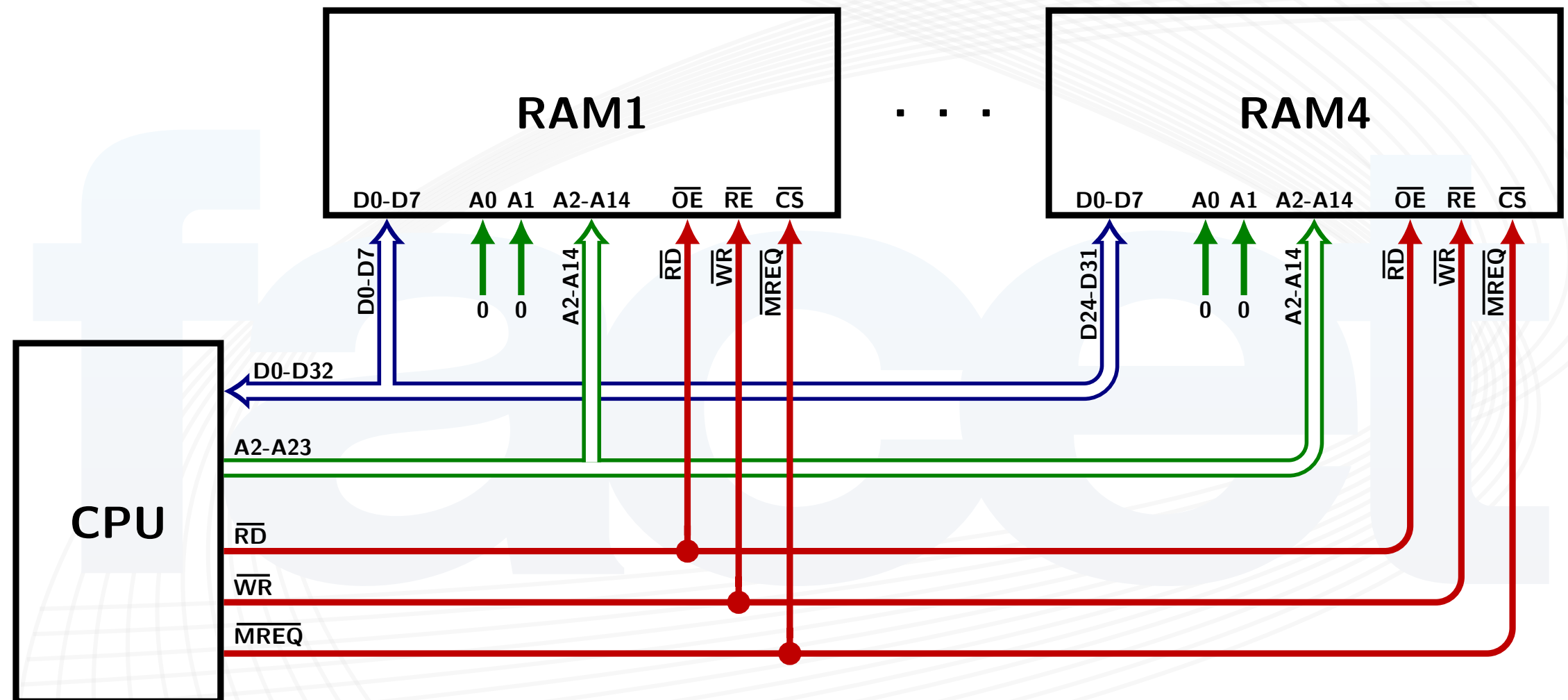
Sin Memoria

0xFFFFF

Algo no funciona bien

- ▶ Esto no funciona correctamente: tres bytes de cada palabra no existen
- ▶ Esto es la consecuencia de no utilizar D8 a D31
- ▶ La única solución es utilizar cuatro memorias, una para cada byte de la palabra
- ▶ Todas las lineas de dirección y de control se conectan en paralelo
- ▶ Esto es equivalente a utilizar construir una memoria de 32 bits con memorias de 8 bits

Ejemplo de aplicación



Mejor, pero sigue sin estar bien

- Ahora la palabra está completa, sin embargo todavía hay un problema

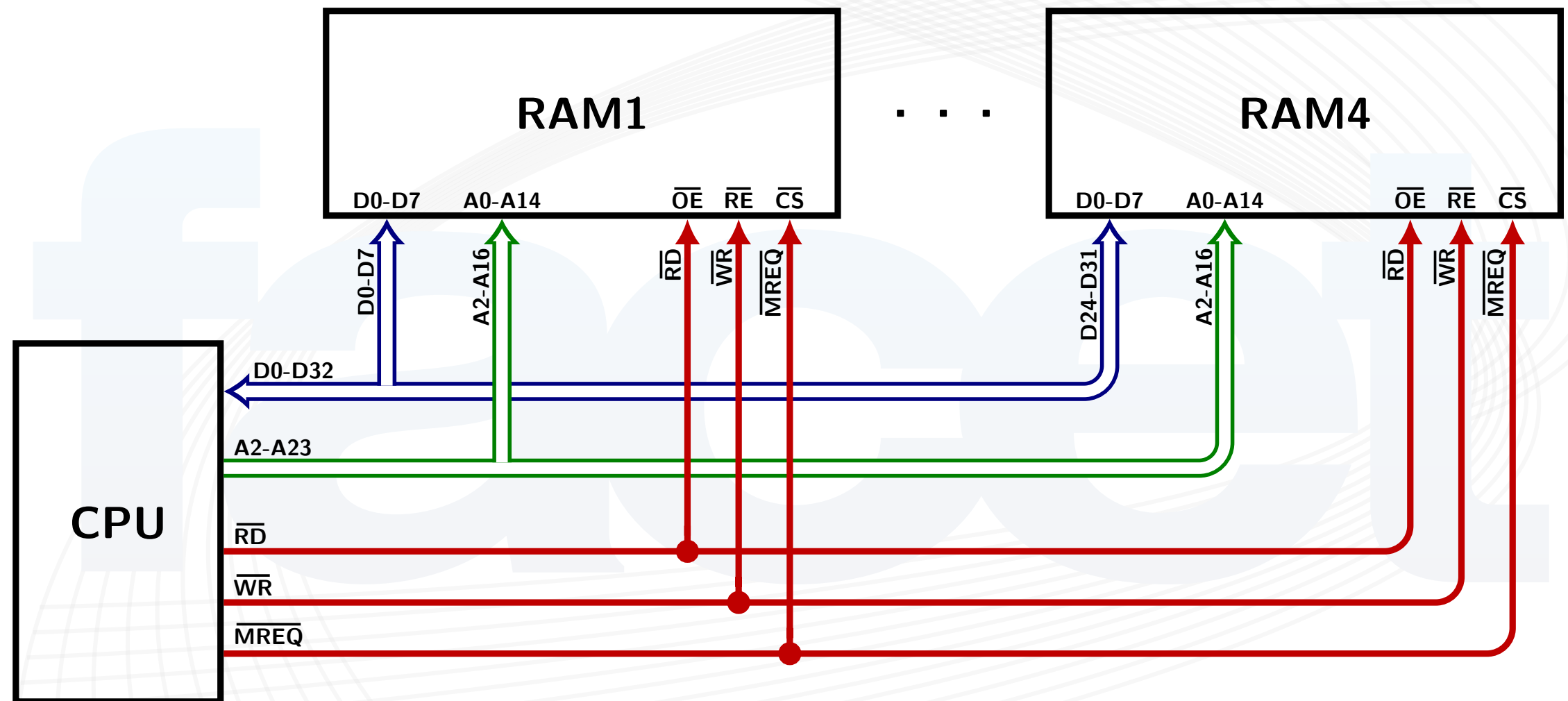
[illegible]

Mejor, pero sigue sin estar bien

- ▶ Ahora la palabra está completa, sin embargo todavía hay un problema
- ▶ En la memoria RAM se puede acceder a solo una dirección de cada cuatro
 - ▶ Esto es la consecuencia de conectar a masa las líneas A0 y A1
- ▶ El resultado que se desperdician tres cuartas partes del almacenamiento
 - ▶ La solución es utilizar todas las líneas de direcciones de la memoria para poder acceder a todas las celdas
 - ▶ Se deben conectar A2 del CPU con A0 de la memoria y así sucesivamente

0x000000	RAM 0 (0x00000)
0x000001	RAM 1 (0x00000)
0x000002	RAM 2 (0x00000)
0x000003	RAM 3 (0x00000)
0x000004	RAM 0 (0x00004)
0x000005	RAM 1 (0x00004)
0x000006	RAM 2 (0x00004)
0x000007	RAM 3 (0x00004)
0x000008	
0xFFFFF	

Ejemplo de aplicación



Mejor, pero sigue sin estar bien

- ▶ Ahora se puede acceder a todas las celdas de las memorias RAM, sin embargo

0x000000
0x000003
0x000004
0x000007
0x000008

RAM (0x00000)

RAM (0x00001)

0xFFFFF

Mejor, pero sigue sin estar bien

- ▶ Ahora se puede acceder a todas las celdas de las memorias RAM, sin embargo

0x000000	RAM (0x00000)
0x000003	
0x000004	
0x000007	RAM (0x00001)
0x000008	
0x01FFFB	.
0x01FFFC	
0x01FFFF	
0x020000	RAM (0x1FFFF)
0xFFFFF	

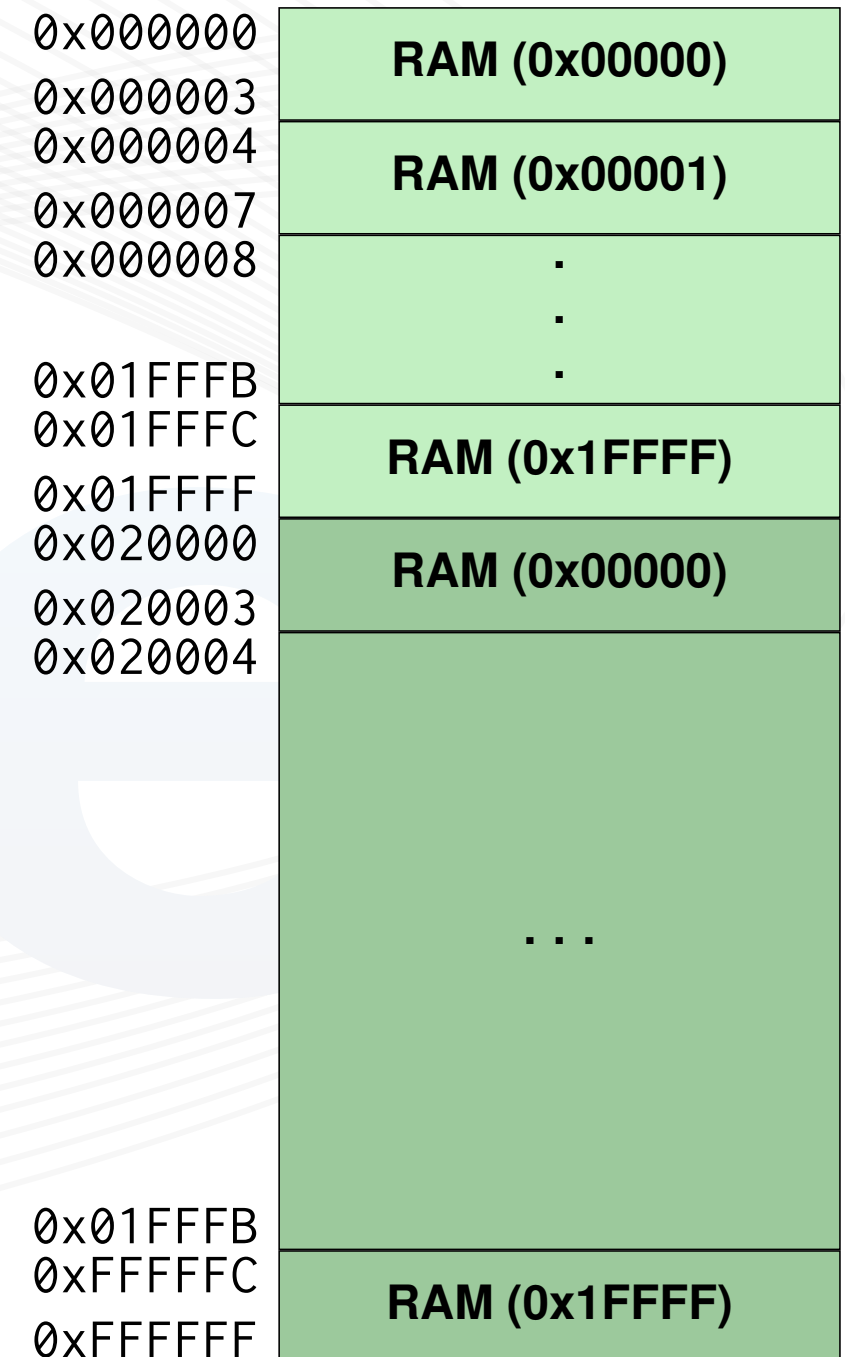
Mejor, pero sigue sin estar bien

- ▶ Ahora se puede acceder a todas las celdas de las memorias RAM, sin embargo
- ▶ Cuando la dirección accedida por el procesador es mayor que la dirección más alta de la memoria RAM esta todavía responde con el contenido de la primera dirección

[illegible]

Mejor, pero sigue sin estar bien

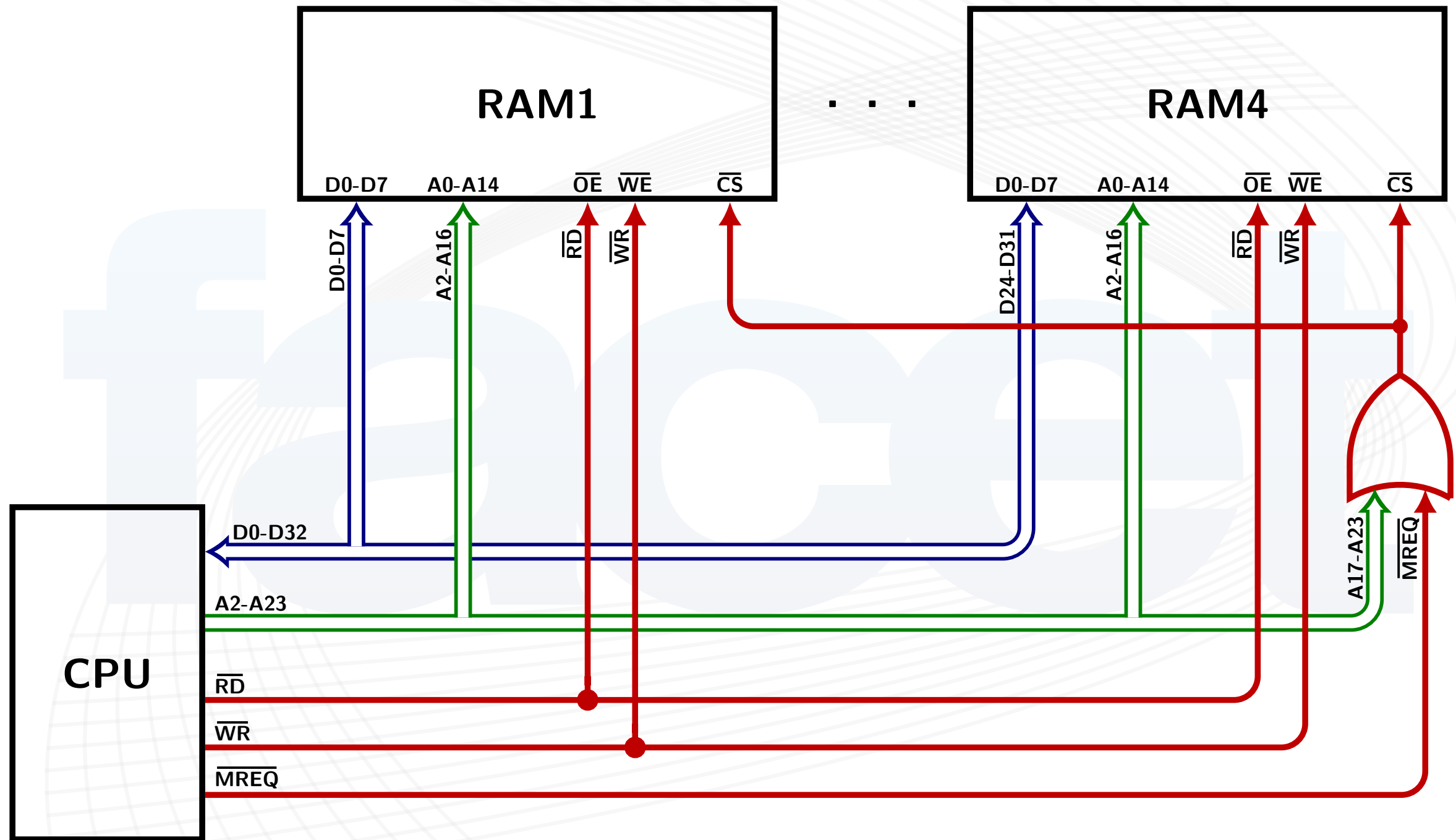
- ▶ Ahora se puede acceder a todas las celdas de las memorias RAM, sin embargo
 - ▶ Cuando la dirección accedida por el procesador es mayor que la dirección más alta de la memoria RAM esta todavía responde con el contenido de la primera dirección
- ▶ Esto es consecuencia de no utilizar algunas de las líneas de direcciones del procesador
 - ▶ Por cada línea que no se utiliza aparecen dos copias de la memoria en el mapa



Conexión con redundancia (100%)

- ▶ Este tipo de conexión se llama redundante
- ▶ La principal ventaja es la simplicidad
- ▶ Sin embargo ocupa todo el mapa e impide conectar más memoria en el futuro
- ▶ ¿Como sería la conexión para que RAM ocupe la primera parte del mapa de memoria sin redundancia?

Ejemplo de aplicación



Conexión sin redundancia

- ▶ En este tipo de conexión asignamos un mínimo posible del mapa de memoria
 - ▶ La principal ventaja es el aprovechamiento óptimo del espacio en el mapa de memoria
 - ▶ Sin embargo este esquema requiere mayor cantidad de componentes
- ▶ Además el tiempo de acceso total aumenta debido a los tiempos de propagación a través de las compuertas

Presentación de la EEPROM 27C4096

- ▶ Memoria de 4 Mibits organizada como 256 Ki x 16 bits
- ▶ Líneas de Direcciones: Depende de cuántas direcciones tenga la memoria
 - ▶ En este caso son 18 líneas de dirección, de A0 a A17
- ▶ Líneas de Datos: Depende de cuántos bits se almacena en cada dirección
 - ▶ En este caso son 16 líneas de datos, de D0 a D15
- ▶ Lectura: una línea que habilita la salida de datos
- ▶ Selección: Una línea que habilita la operación del chip

El sistema completo

- ▶ Se desea implementar un sistema con 1 MiB de memoria FLASH y 128 KiB de memoria RAM sin redundancia
- ▶ La FLASH deberá operar en las direcciones 0x000000 a 0x0FFFFFFF
- ▶ La RAM deberá operar en las direcciones 0xC00000 a 0xC1FFFFF
- ▶ ¿Como fijamos las direcciones para cada bloque de memoria?

Ocupación en mapa de memoria

- ▶ La memoria EPROM es de 256 Ki palabras por 16 bits en cada palabra
 - ▶ Pero como el procesador direcciona bytes esto equivale a 512 KiBytes
- ▶ Para proveer los 32 bits que requiere el procesador se necesitan 2 memorias EPROM
 - ▶ Lo que implica que las dos memorias EPROM ocupan, como mínimo, 1MiBytes en el mapa
- ▶ Las memorias RAM son las mismas del ejemplo anterior
 - ▶ Se requieren 4 memorias y ocupan 128 KiBytes en el mapa

División del Mapa de Memoria

- ▶ La memoria EPROM debe ocupar 1MiB,
- ▶ Se divide el mapa de memoria en fracciones de este tamaño



División del Mapa de Memoria

- ▶ La memoria EPROM debe ocupar 1MiB,
- ▶ Se divide el mapa de memoria en fracciones de este tamaño
- ▶ Cada fracción se caracteriza por mantener constante una combinación de valores de las 4 líneas superiores de direcciones

0x000000	$\overline{A23} \overline{A22} \overline{A21} \overline{A20}$
0x0FFFFFF 0x100000	$\overline{A23} \overline{A22} \overline{A21} A20$
0x1FFFFFF 0x200000	$\overline{A23} \overline{A22} A21 \overline{A20}$
0x2FFFFFF 0x300000	$\overline{A23} \overline{A22} A21 A20$
0x3FFFFFF 0x400000	.
0xDFFFFFF 0xE00000	$A23 A22 A21 \overline{A20}$
0xEFFFFFF 0xF00000	$A23 A22 A21 A20$
0xFFFFFFF	

División del Mapa de Memoria

- ▶ De acuerdo a la zona en la que debe operar el chip, se elige la combinación de líneas de direcciones que los seleccionará
- ▶ Una opción es utilizar compuertas y negadores para sintetizar la función de selección
- ▶ Otra opción es utilizar un decodificador de 4 a 16 para generar una línea de selección independiente para cada fragmento del mapa
- ▶ Estos fragmentos se suelen denominar bancos

El sistema completo

