MEMORIA

Arquitectura de Computadores I

3er tema

Introducción

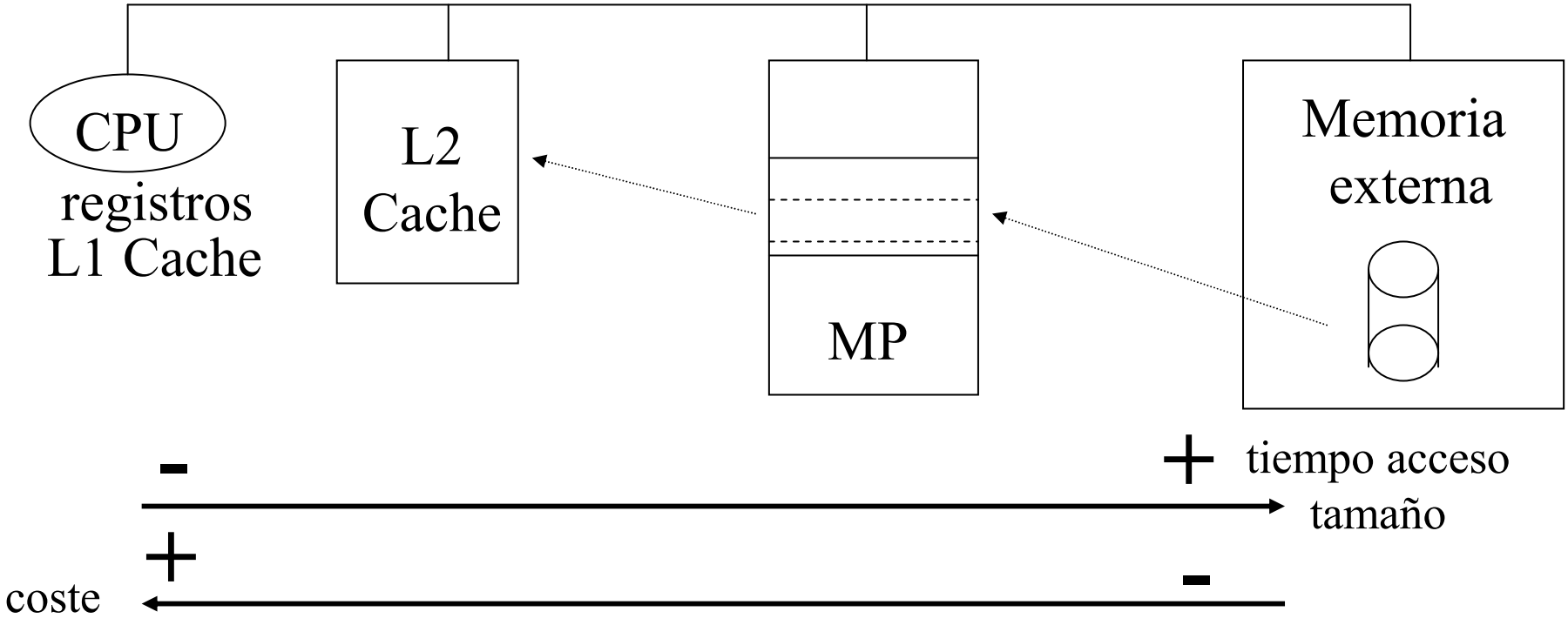
- **Almacén: programa+datos**
- **Problema**:
  - Velocidad entre CPU y el sistema de memoria
  - Necesidad de alta capacidad de almacenamiento con bajo tiempo de acceso
- **Objetivos del tema**:
  - Acelerar el acceso al sistema de memoria
    - estructura de la memoria: interconexiones entre módulos
    - jerarquía de memoria
  - Proceso de traducción: @lógica / @física

Tiempos de una memoria

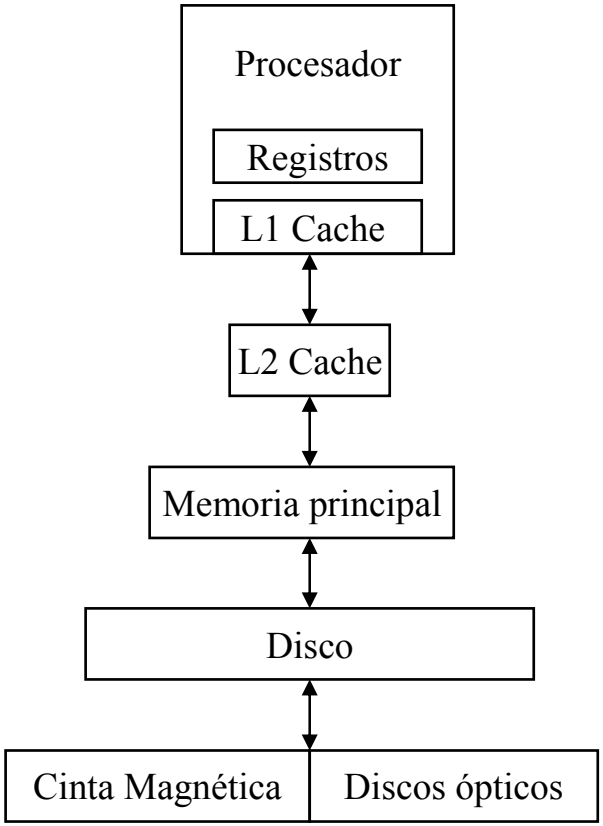
- **Tiempo de acceso o tiempo de respuesta:**
  - Tiempo que transcurre desde el envío a la memoria de la dirección de una celda hasta la obtención del dato
- **Tiempo de ciclo:**
  - Tiempo que transcurre desde que la memoria acepta una petición hasta que queda lista para aceptar la siguiente
- Tiempos condicionados a la tecnología de la memoria y a la mejora del protocolo de diálogo memoria/CPU

Jerarquía de memoria

- **Memoria ideal:**
  - capacidad de almacenamiento “ilimitada”
  - tiempo de acceso “nulo”
- **En la realidad →jerarquía de memoria:**



Jerarquía de memoria



- **Principio de localidad:**
  - espacial y temporal

Tipo de memoria	Tecnología	Tamaño	Tiempo acceso
Registros	SRAM	512 bytes	0.25/0.5 ns
L1 Cache	SRAM	32/64 KB	1 ns
L2 Cache	SRAM	512KB / 2MB	< 25 ns
Memoria principal	DRAM	1 GB	50 /200 ns
Disco magnético	Disco duro	1 TB	10 ms
Disco óptico	CD-ROM	GB	300 ms
Cinta magnética	Cinta	GB/TB	seg-min

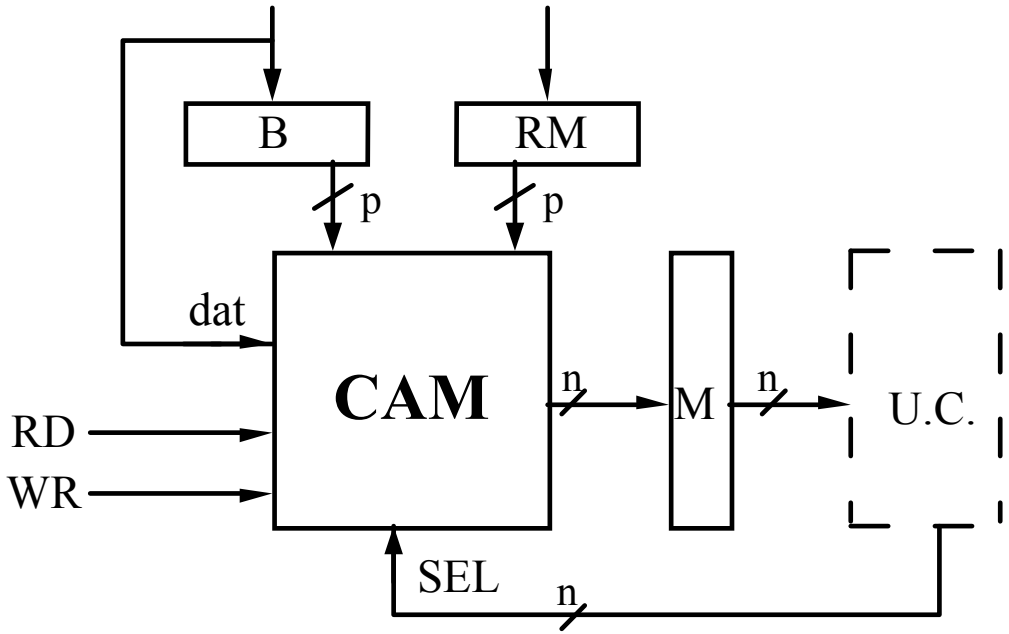
Tipos de Memoria

- **RAM Dinámica (DRAM):** Dynamic Random Access Memory
  - Núcleo básico: condensador
  - Alta densidad, bajo coste → Memoria principal
  - Necesita refrescar la información, elevado Tacceso

- **RAM Estática (SRAM)**: Static Random Access Memory
  - Núcleo básico: flip-flop
  - Baja densidad, alto coste → Memoria cache
  - No necesita refresco, menor Tacceso (6/7 veces menor)

Tipos de Memoria

- **Memorias asociativas (CAM, Content-addresable memory):**
  - SRAM con acceso por contenido
  - Búsqueda en paralelo → hardware complejo y caro
  - Utilización: TLB/estructura interna cache

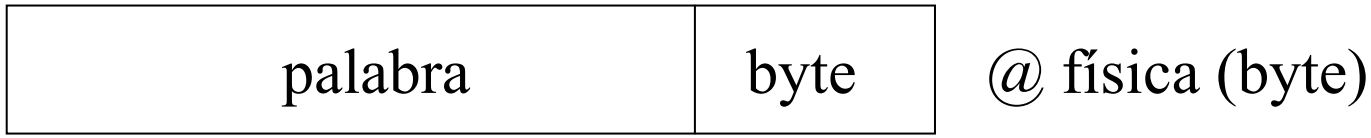


Tipos de Memoria

- **ROM** (Read Only Memory): **[120/200ns]**
  - Memoria no volátil → BIOS y software de arranque
  - Distintos tipos:
    - +PROM (Programmable Read Only Memory)
      → ROM no borrable
    - +EPROM (Erasable Programmable Read Only Memory)
      → ROM borrable: luz ultravioleta, chip completo
    - +EEPROM (Electrically Erasable Programmable ROM)
      → ROM borrable: eléctricamente, actualizable a nivel de byte
    - +FLASH
      → Densidad de EPROM, borrable eléctrico de EEPROM
      → Borrable por bloques → proceso de borrado rápido

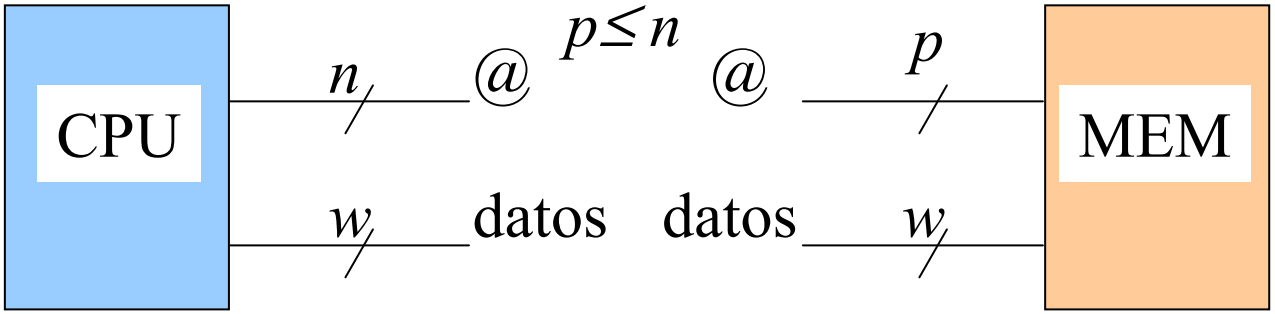
Sistema de memoria

- **Estructura de la memoria**:
  - Dirección, posición, contenido, bus @, bus datos, bus control
  - Palabra del procesador (32, 64 bits)
- **Direccionamiento**:
  - Byte: bus @ direcciona todos los bytes de memoria
  - Palabra: bus @ sólo direcciona palabras
  - Normalmente: byte
    - palabra = @ física *div* tamaño de palabra (byte)
    - byte = @ física *mod* tamaño de palabra (byte)

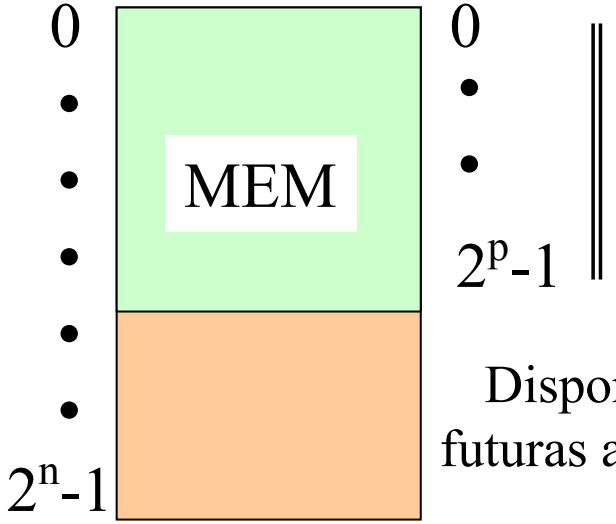


Sistema de memoria

- **Conexión memoria-procesador**:



- **Mapa de memoria del procesador**:
  - espacio direccionable
  - bus direcciones *n* bits → 2^n posiciones máximo



Mapa de memoria:

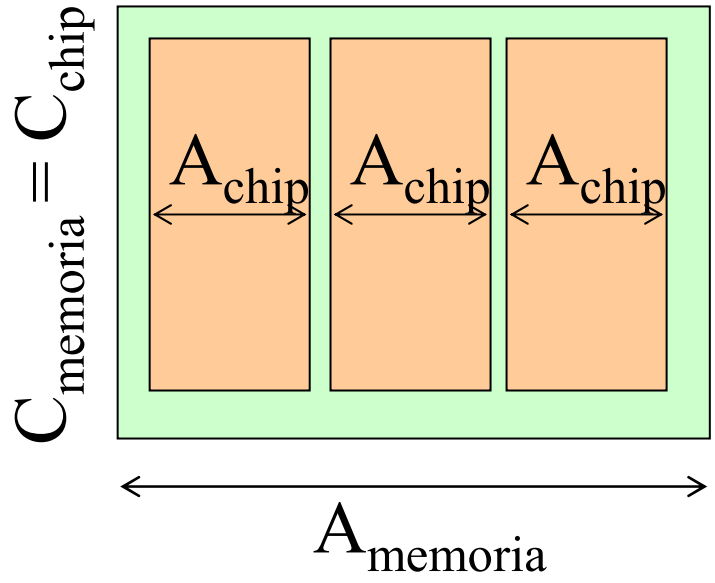
DIRECCIONES		MAPA DE MEMORIA	
DECIMAL	HEXADECIMAL	FUNCIONAL	FÍSICO
8.191 7.168	1FFFH 1C00H	ROM con aplicaciones	Pastilla 5 ROM 1Kx8
7.167 6.144	1BFFH 1800H	RAM del monitor	Pastilla 4 RAM 1Kx8
6.143 5.120	17FFH 1400H	LIBRE	
5.119 4.096	13FFH 1000H	ROM periféricos	Periféricos
4.095 3.072	0FFFH 0C00H	LIBRE	
3.071 2.048	0BFFH 0800H	RAM para programas	Pastilla 3 RAM 1Kx8
2.047 1.024	07FFH 0400H	RAM para la pila	Pastilla 2 RAM 1Kx8
1.023 256	03FFH 0100H	LIBRE	
255 0	00FFH 0000H	ROM de arranque	Pastilla 1 ROM 256x8

- **Funcional**: indica el uso que el sistema hace de cada una de las posiciones de memoria
- **Físico (hardware)**: indica qué chip de memoria o dispositivo físico contiene cada una de las direcciones

Sistema de memoria

Memoria principal

- **Construcción física de la memoria**:
  - Tamaño de mem. principal ≠ tamaño chips comerciales
  - **Expansión** de los chips de memoria
- **Expansión en anchura** (anchura de palabra):

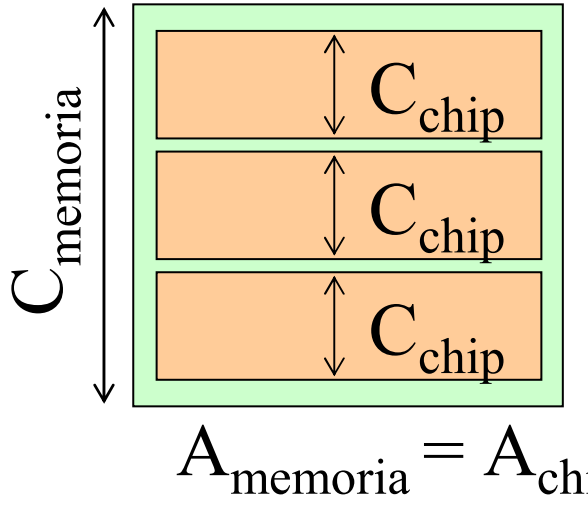


$$\#\,chips\,necesarios = \frac{A_{memoria}}{A_{chip}}$$

C: capacidad (número de palabras)
A: anchura de la palabra

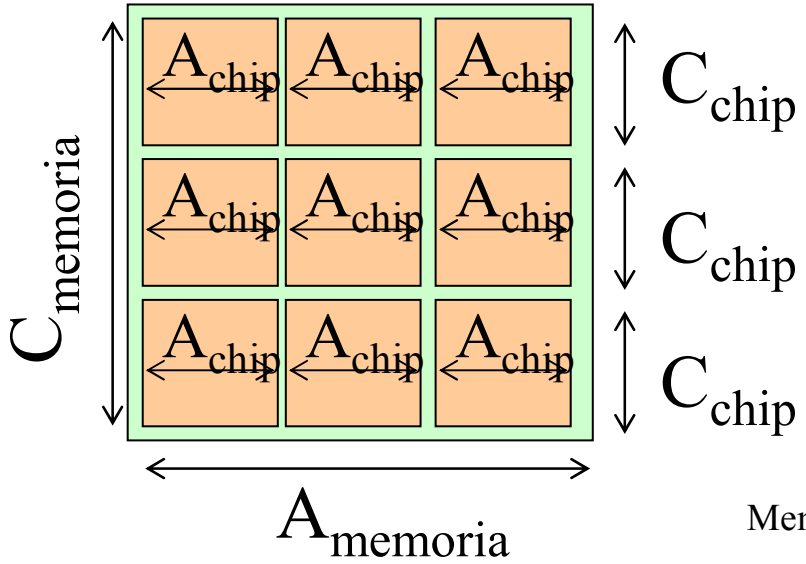
Memoria principal

- **Expansión en longitud** (número de palabras, capacidad):



$$\#chips\ necesarios = \frac{C_{memoria}}{C_{chip}}$$

- **Expansión en anchura y longitud:**



$$\#chips = \frac{A_{memoria}}{A_{chip}} \times \frac{C_{memoria}}{C_{chip}}$$

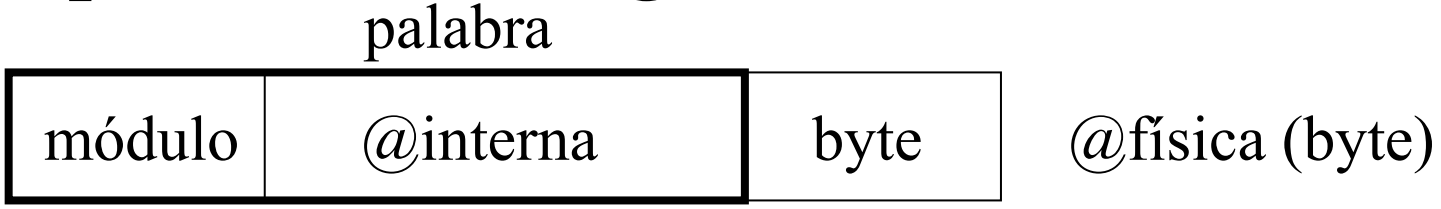
Conexiones entre módulos

- **Objetivo**: expansión en longitud
  - Módulos de *N* bytes $\Rightarrow$ memoria de *M* bytes
  - Minimizar el tiempo de acceso a memoria

- **Opciones**:
  - Módulos consecutivos
  - Módulos entrelazados
  - Bancos consecutivos de módulos entrelazados

Módulos consecutivos

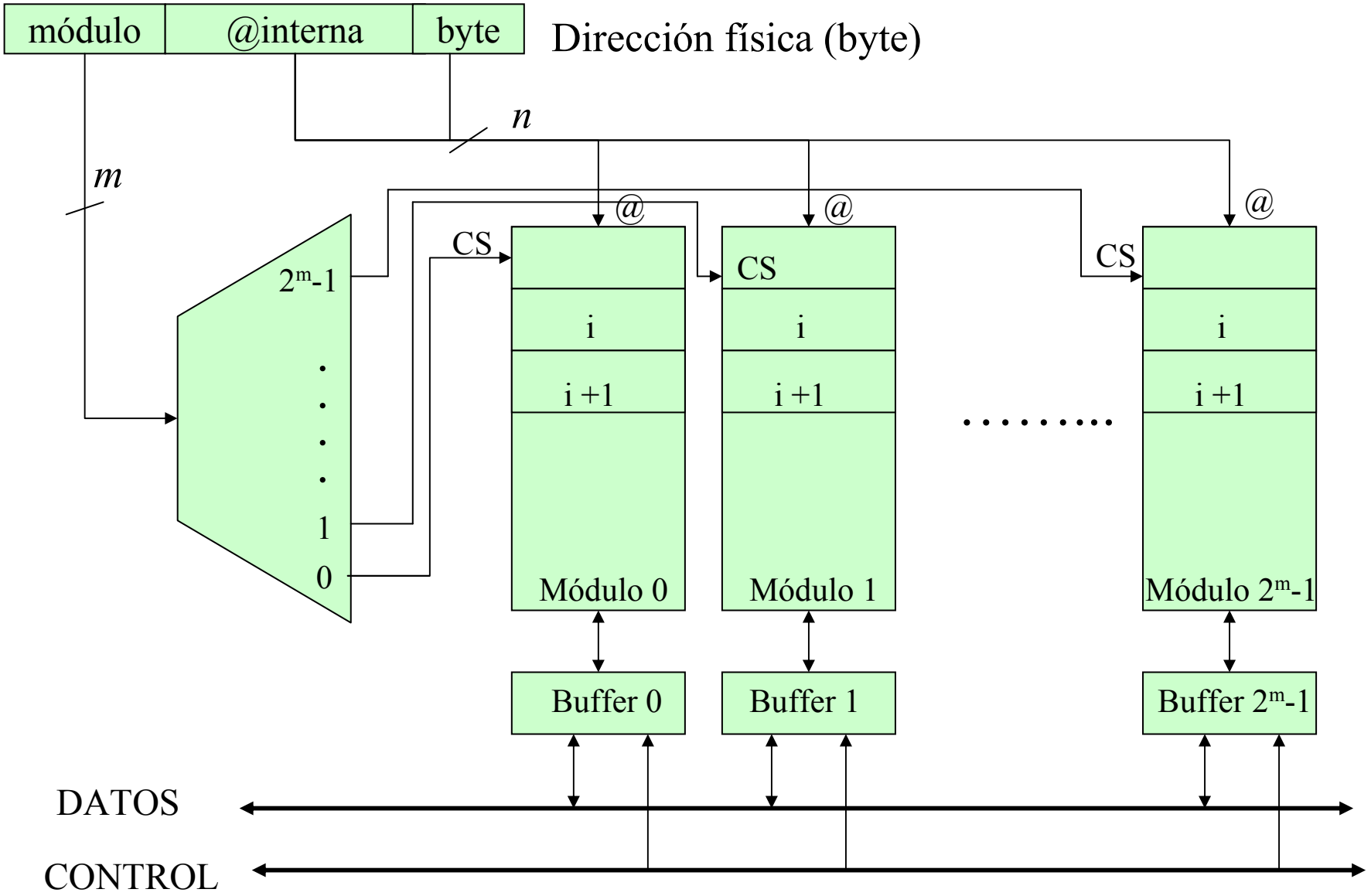
- **Estructura**: palabras consecutivas en el mismo módulo
- **Componentes de la @física**:



  - palabra = @física *div* tamaño de palabra (byte)
  - módulo = palabra *div* tamaño de módulo (palabras)
  - @ interna= palabra *mod* tamaño de módulo (palabras)

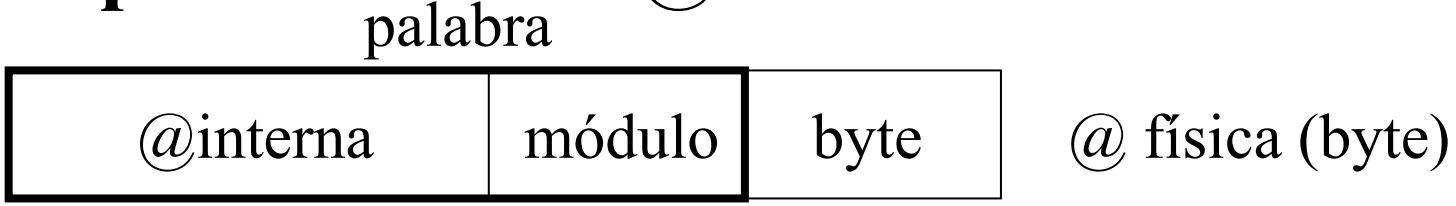
- **Problema:** accesos secuenciales
  - no sigue el comportamiento de los programas

Módulos Consecutivos



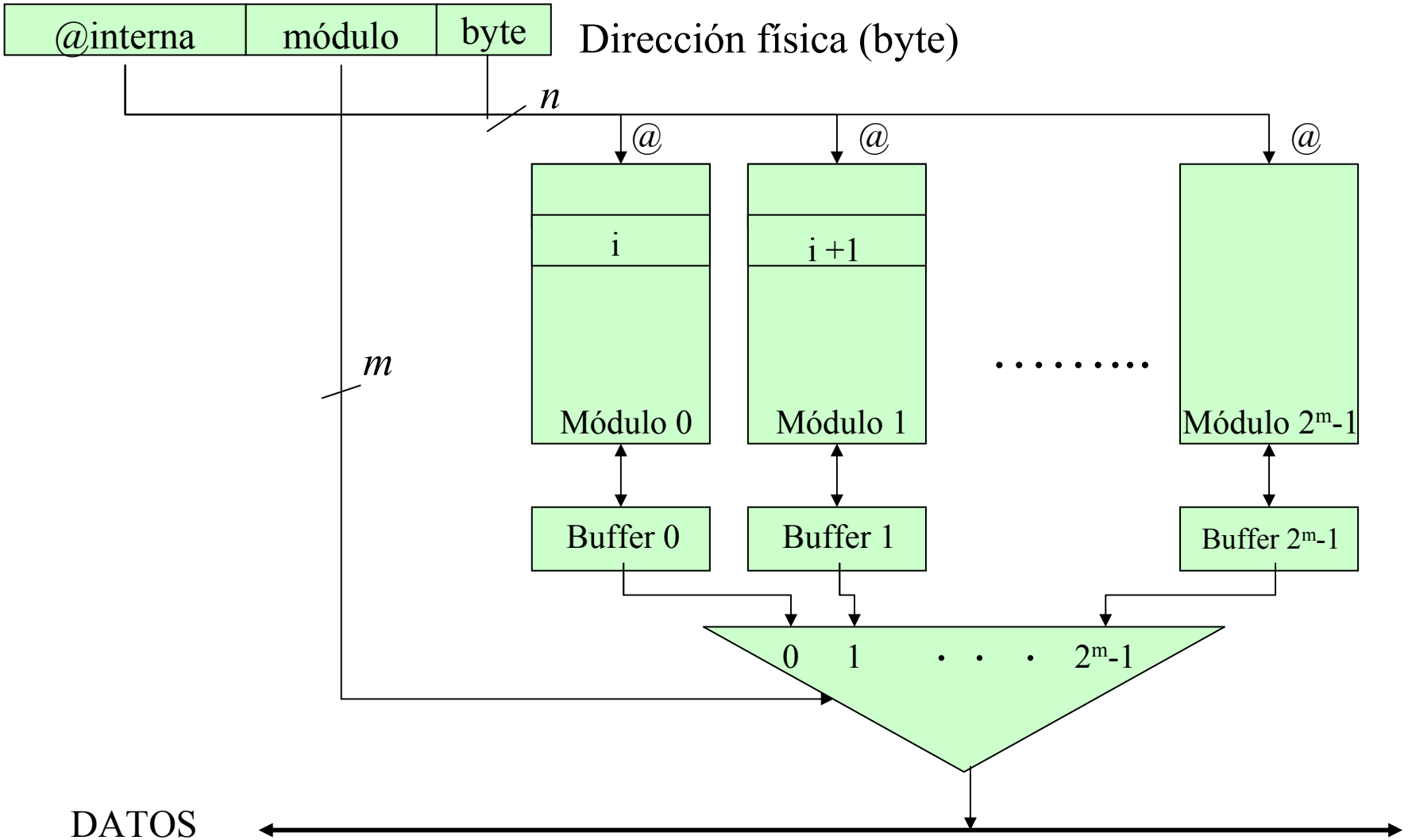
Módulos entrelazados

- **Estructura**:
  - palabras consecutivas en módulos consecutivos
- **Componentes de la @física:**



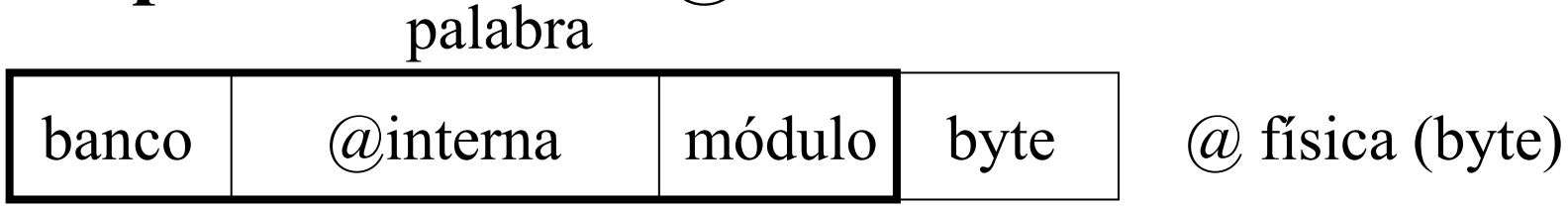
  - palabra = @física *div* tamaño de palabra (byte)
  - módulo = palabra *mod* número de módulos (n)
  - @interna = palabra *div* número de módulos
- **Ventaja:** 1 acceso $\Rightarrow$ ***n*** palabras
  - se considera el comportamiento de los programas

Módulos Entrelazados



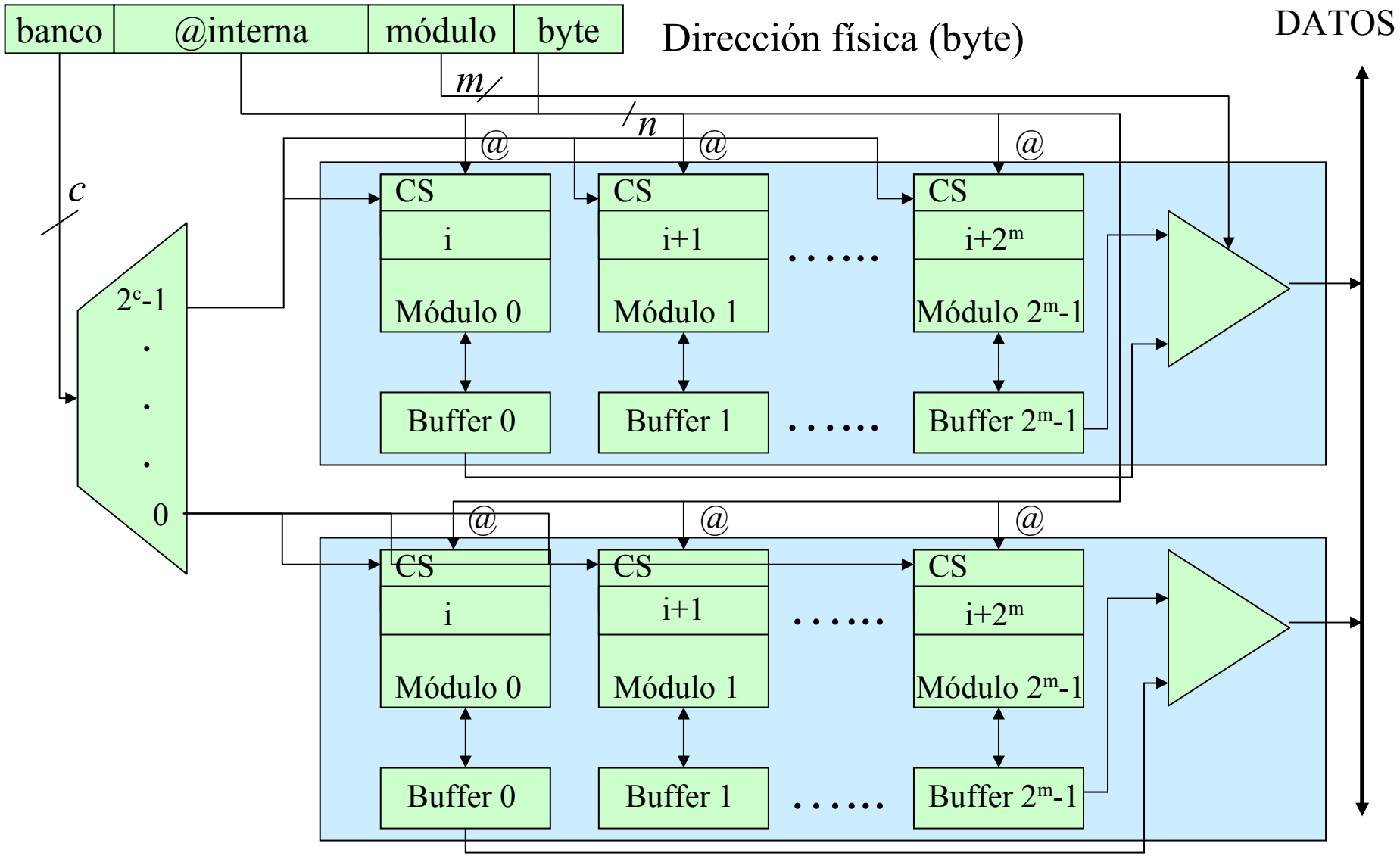
Bancos consecutivos de módulos entrelazados

- **Estructura**:
  - los módulos se reparten en bancos consecutivos
  - dentro de un banco se entrelazan los módulos
- **Componentes de la @física**:



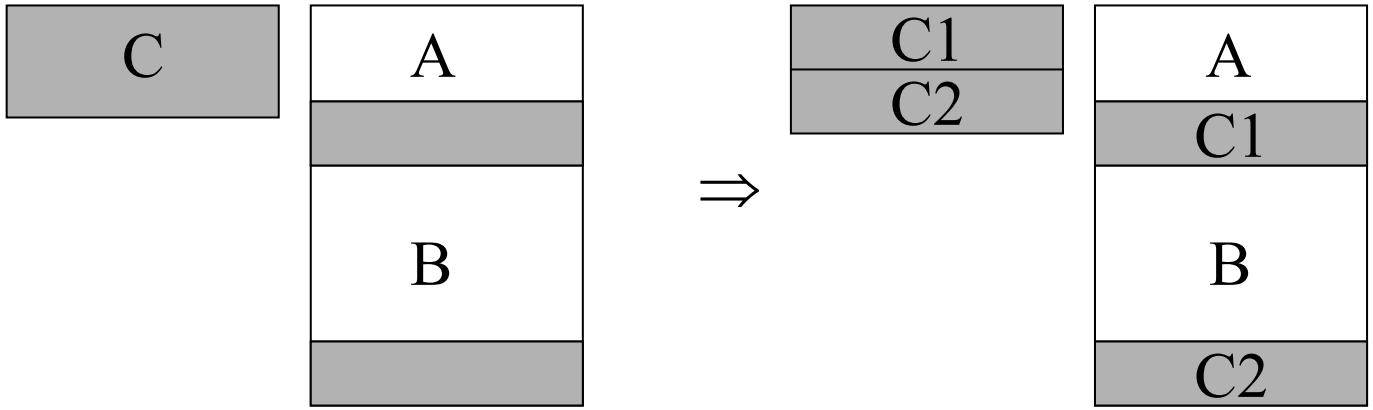
  - palabra = @física *div* tamaño de palabra (byte)
  - banco = (palabra *div* número de módulos) *div* tamaño del módulo (palabras)
  - módulo = palabra *mod* número de módulos
  - @interna = (palabra *div* número de módulos) *mod* tamaño del módulo (palabras)

Bancos consecutivos de módulos entrelazados



Segmentación/Paginación

- **Objetivo**:
  - Cargar un programa en cualquier @física
    - CPU ⇒ direcciones lógicas
  - Aprovechar mejor la memoria

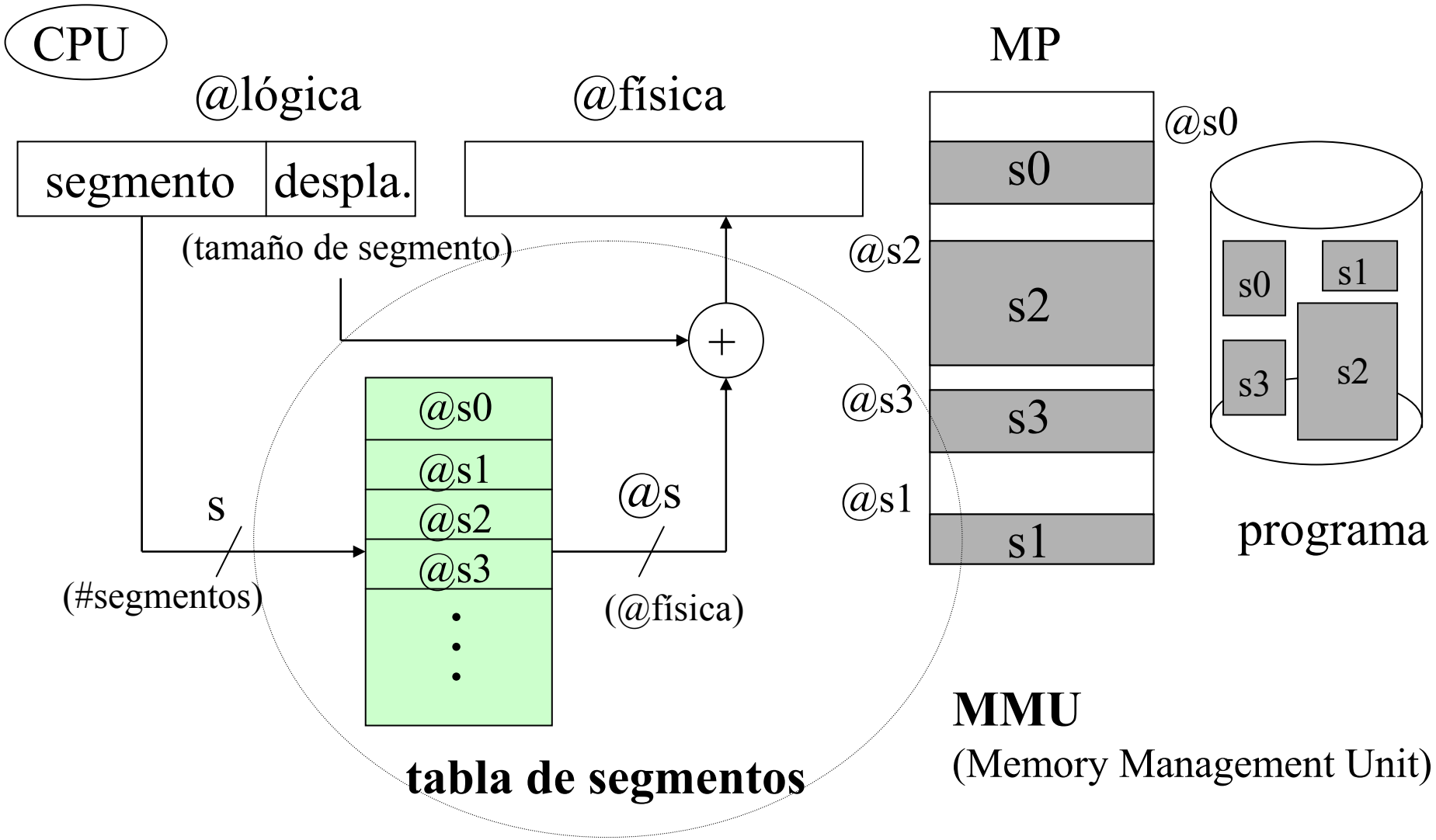


- **Proceso de traducción:** @lógicas ⇒ @físicas

Segmentación

- El programa se divide en entidades lógicas: pila, datos, código, etc. $\Rightarrow$ **segmentos**
- Segmentos de distinto tamaño
- El cargador los carga en memoria, cada uno a partir de una dirección de inicio (@base)
- Dirección lógica:
  - segmento
  - desplazamiento

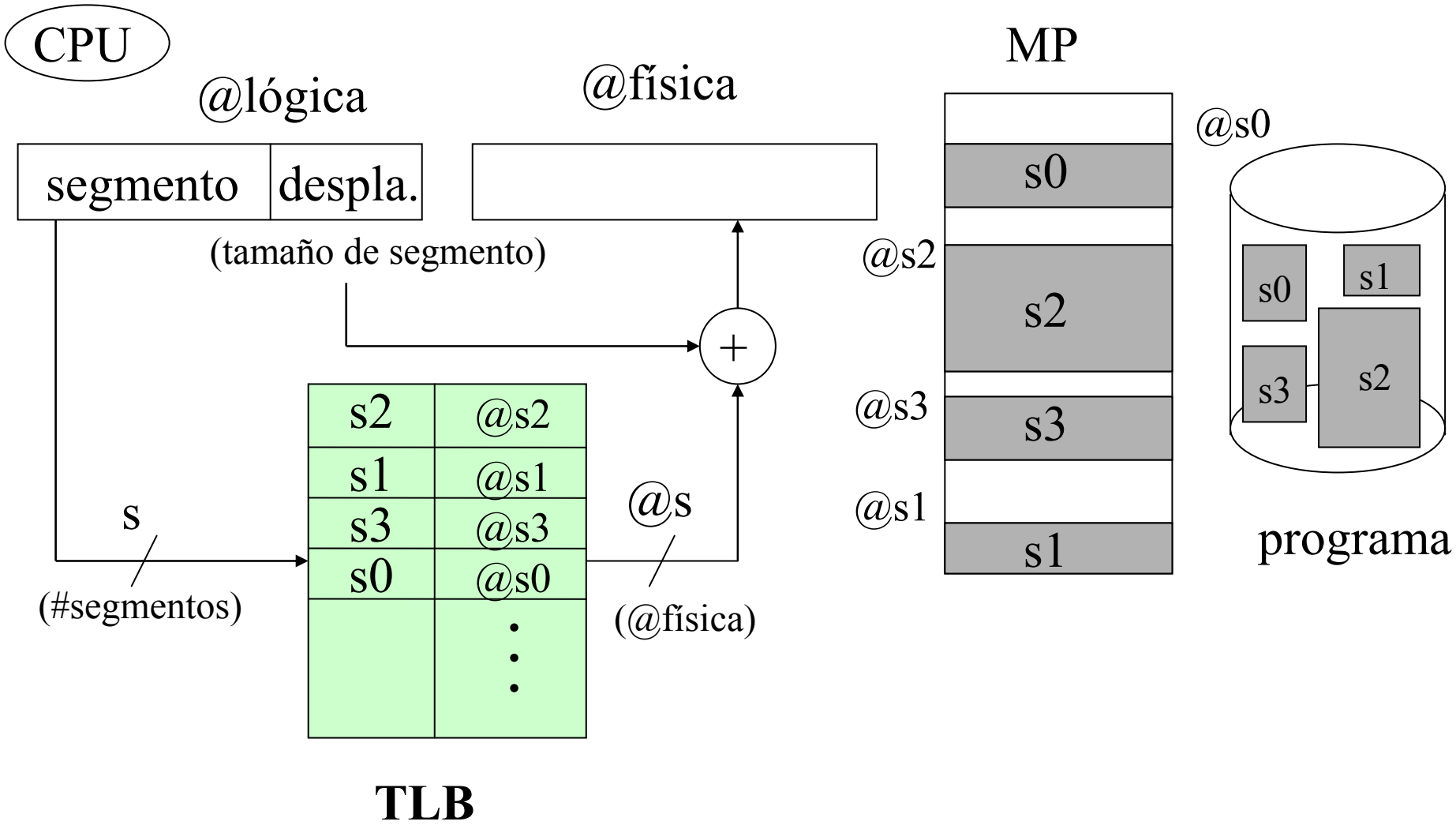
Segmentación



Segmentación

- Traducción: **tabla de segmentos**
- ¿Dónde se guarda?
  - Registros
  - Memoria principal
    - 2 accesos para cualquier referencia
- Hardware especial: **TLB** (Translation Lookaside Buffer)
  - Memoria asociativa
  - Fallo / Acierto (acceso rápido)
  - Caro, tamaño limitado (32/1024 entradas)

Segmentación

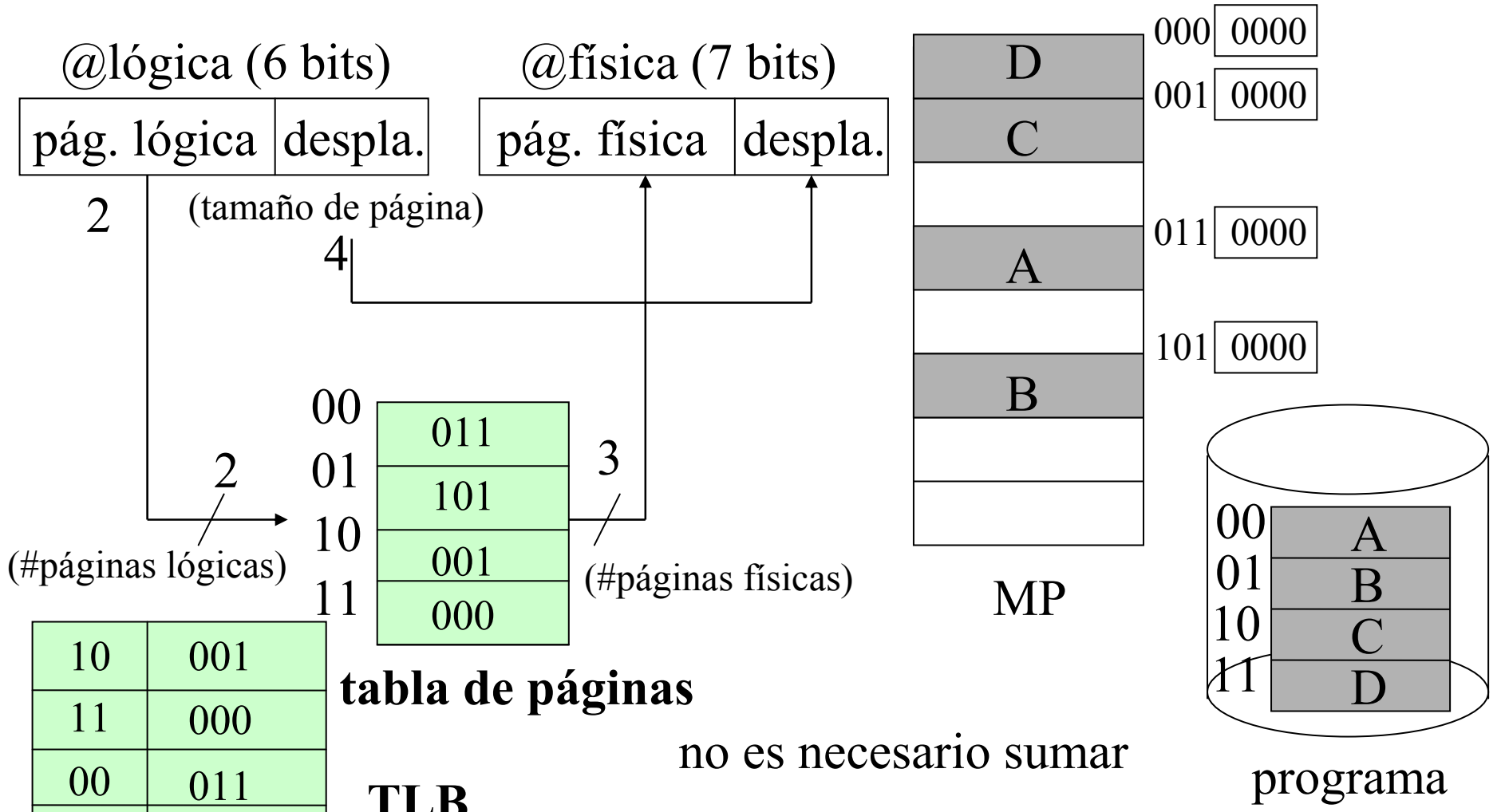


Paginación

- El programa y la memoria se dividen en trozos del mismo tamaño $\Rightarrow$ **páginas**
  - potencia de dos (4K / 16 Mbytes)
  - sin sentido lógico
- El cargador las carga en memoria, cada una en una página física libre
- @lógica: página lógica / desplazamiento
  @física: página física / desplazamiento

Paginación

Ejemplo: 4 páginas lógicas / 8 páginas físicas / páginas de 16 bytes



no es necesario sumar

Segmentación/Paginación

- Segmentación:
  - entidades lógicas (definidas por el programador)
  - tabla de segmentos con @físicas
  - fragmentación externa
- Paginación:
  - entidades sin sentido lógico
  - traducción sencilla (sin necesidad de sumar)
  - la tabla de páginas indica páginas físicas
  - fragmentación interna

Segmentación/Paginación

- Normalmente:
  - los programas se segmentan
  - los segmentos se paginan

  $\Rightarrow$**segmentación paginada**
- @lógica: 40/80 bits @física: 32/44 bits
- **Memoria virtual**
  - no es necesario cargar todo el programa
  - aprovechamiento de memoria
  - posibilidad de ejecutar un programa de mayor tamaño que la memoria principal

Detección y correción de errores

- Memoria:
  - →circuito de alta complejidad
  - →considerar la posibilidad de la existencia de fallos en su funcionamiento
  - →fallos hardware: no recuperables
  - →fallos software: transitorios, pueden recuperarse
  - →añadir bits de control al código original
- Dos posibidades:
  - Bits de paridad
  - Códigos basados en los códigos de Hamming

Bits de paridad

- Añadir un bit de paridad a cada palabra de memoria:
  →se determina con la XOR de todos los bits
  →**paridad par**: el número total de unos sea par
  →**paridad impar**: el número total de unos sea impar
- Ejemplo:
  →paridad par: 10010010 → 1 // 00001100 →0
  →paridad impar: 10010010 → 0 // 00001100 →1
- Permite detectar errores de un bit, pero no corregirlos
  →reintentar la lectura de memoria
  →si persiste, error del sistema

Códigos correctores de errores

- Basados en los códigos de Hamming:
  - →código **SEC**: *código corrector de errores simples*
  - →código **SEC-DED**: *código corrector de errores simples y detector de errores dobles*

- Ejemplo de código SEC con palabras de 8 bits:
  - →añadir 4 bits de control: c0, c1, c2 y c3 intercalados en las posiciones potencias de 2

posición	12	11	10	9	8	7	6	5	4	3	2	1
bit		d7	d6	d5	d4	**c3**	d3	d2	d1	**c2**	d0	**c1**	**c0**

  - →bits de control: bits de paridad de un subconjunto de los bits de la palabra original, de acuerdo a la descomposición binaria de la posición que ocupan en la nueva palabra

Códigos correctores de errores

- Ejemplo de código SEC con palabras de 8 bits (cont.):

  →bits de control:

  $c0 = d6 \oplus d4 \oplus d3 \oplus d1 \oplus d0$ $\quad$ $c2 = d7 \oplus d3 \oplus d2 \oplus d1$

  $c1 = d6 \oplus d5 \oplus d3 \oplus d2 \oplus d0$ $\quad$ $c3 = d7 \oplus d6 \oplus d5 \oplus d4$

  bit d6 → posición 11 = 8 + 2 + 1 → cálculo de c3, c1 y c0

  →se guardan junto con la palabra en memoria

  →para detectar un error

  – calcular los bits de control de la palabra leída
  – efectuar la XOR entre éstos y los originales → síndrome
  – si el resultado es 0000, el dato es correcto. De lo contrario, el valor obtenido indica la posición del bit erróneo

Códigos correctores de errores

- Ejemplo de código SEC con palabras de 8 bits (cont.):
  →ejemplo:

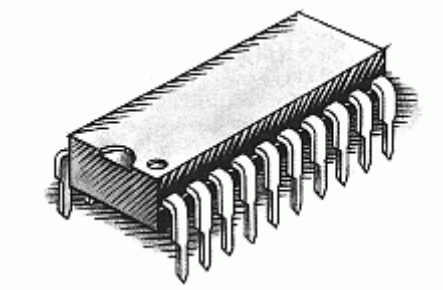
dato	01110011 → c3=1, c2=1, c1=1, c0=0
nueva palabra	0111**1**001**1**1**10** (se escribe en mem.)
error en un bit	01***0***110011110 (se lee de mem.)
cálculo de los bits de control	c3'=0, c2'=1, c1'=0, c0'=0
XOR → síndrome	1010
	→ bit de la posición 10 (d5) erróneo

Códigos correctores de errores

- Los códigos SEC y SEC-DED suponen un incremento en el tamaño de la memoria.
- El número de bits a añadir en un código SEC viene dado por la expresión: $(d+c+1) \leq 2^c$. En un código SEC-DED se añade un bit más a la expresión anterior.
- Algunos ejemplos:
  + IBM 30xx → SEC-DED de 8 bits para 64 bits →12%
  + VAX → SEC-DED de 7 bits para 32 bits → 22%
  + DRAM → ECC (Error Correction Code) de 8 bits para 64 bits → 12%

Formatos de memoria RAM (PC)

- **DIP (*Dual In-line Package*):**
  - Formato más antiguo: formato chip
  - 64 Kbytes → 8 chips de 64 Kbits



- **SIP (*Single In-line Package*):**
  - 8 chips soldados en una placa de circuito impreso
  - Módulos de *N* Kbytes
  - Problema: patillas frágiles

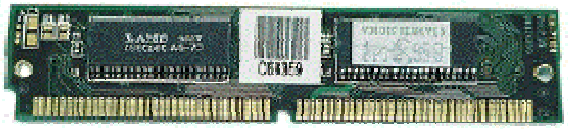
Formatos de memoria RAM (PC)

Tipo Zócalo	Número contactos	Tipo de memoria	Anchura Bus datos	Tamaño módulos
SIMM	30	FPM	8 bits	256 kB, 1/4/16 MB
SIMM	72	FPM, EDO, BEDO	32 bits	1/2/4/8/16/32/64/128 MB
DIMM	168	EDO, BEDO, SDRAM	64 bits	8/16/32/64/128/256 MB 1 GB
DIMM	184	DDR-SDRAM	64 bits	128/256/512 MB, 1GB
DIMM	240	DDR2/3-SDRAM	64 bits	256/512 MB, 1/2/4 GB
RIMM	184	RDRAM	16 bits	128/256/512 MB

SIMM (*Single In-line Memory Module*)
DIMM (*Dual In-line Memory Module*)
RIMM (*Rambus In-line Memory Module*)

Formatos de memoria RAM (PC)

SIMM (30/72 contactos)



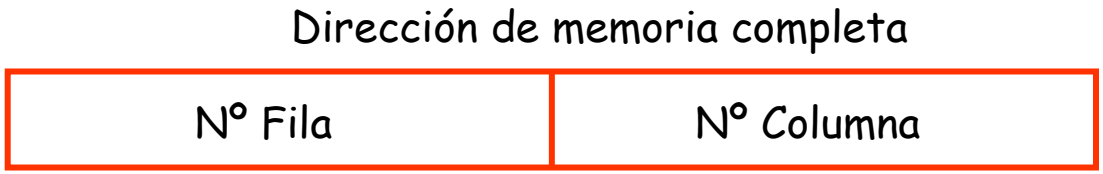
DIMM 168 contactos

DIMM DDR 184 contactos

RIMM 184 contactos

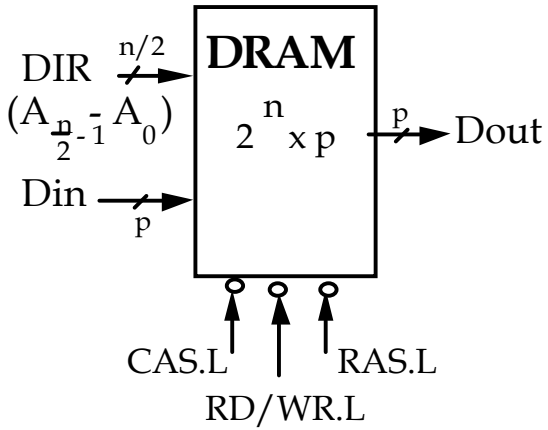
Memoria principal PC (DRAM)

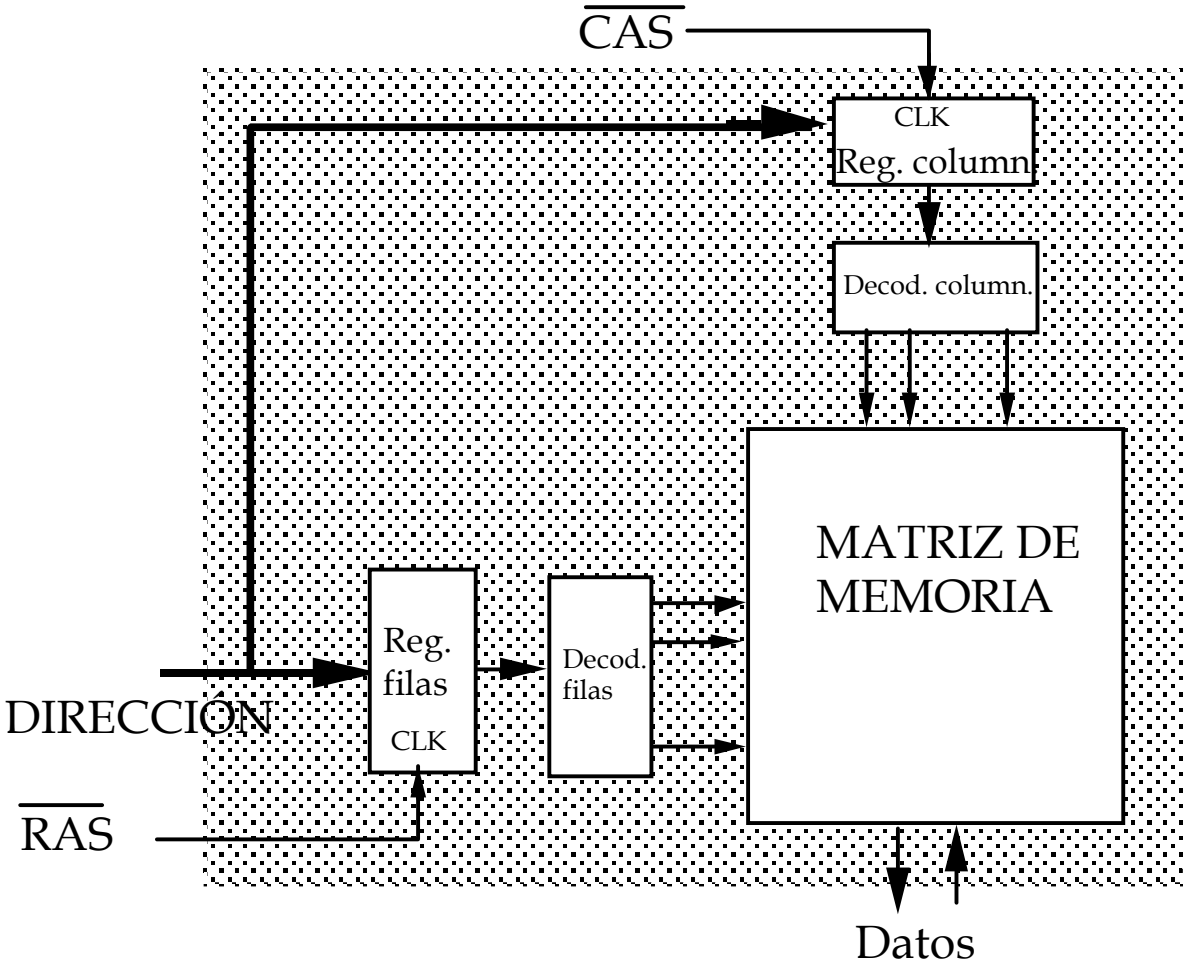
- **Funcionamiento:**
  - Matriz de celdas de memoria
  - DRAM estándar:
    - acceso por filas (RAS) y luego por columnas (CAS)
    - desactivar RAS y CAS
    - memorias asíncronas



Memoria DRAM

- **Esquema:**





Memoria DRAM

- ***Fast Page Mode* DRAM:**
  - 1 activación de RAS y varias activaciones de CAS
  - acceso a distintas direcciones en la misma fila
  - tiempo de acceso:
    - → $T_{fila}+T_{columna}$ para el primer dato de una fila
    - → $T_{columna}$ para el resto de la misma fila
      - →entrelazado en longitud
  - tiempo de acceso: 60/80 ns [DRAM convencional, 80/150ns]
  - temporización: 5-3-3-3 (66 MHz)

Memoria DRAM

- **EDO RAM (*Extended Data Out*):**
  - minimiza el tiempo de acceso a cada uno de los *n-1* datos de la misma fila
  - almacena el dato en un buffer de salida, solapando la salida de un dato con el acceso al siguiente y evita el reseteo del bus de salida de datos

    →reduce $T_{columna}$
  - tiempo de acceso: 50/60 ns
  - temporización: 5-2-2-2 (66 MHz)
  - módulos SIMM y DIMM

Memoria DRAM

- **BEDO RAM (*Burst Extended Data Out*):**
  - acceso en modo ráfaga a datos consecutivos de la misma fila
  - genera de forma automática las direcciones de los datos en la misma fila, eliminando el tiempo de introducción de las señales de cada columna

    → reduce $T_{columna}$
  - mejora la segmentación entre el acceso a un nuevo dato y la salida del anterior
  - tiempo de acceso: 40/50 ns
  - temporización: 5-1-1-1 (66 MHz)
- **Fiabilidad FPM/EDO/BEDO: buses de 66 MHz**

Memoria DRAM

- **SDRAM (*Synchronous DRAM*):**
  - sincronización con el reloj de la placa base mediante latches o registros temporales:
    - simplifica las señales de control
    - elimina problemas de temporización
    - elimina tiempos de espera del procesador
  - acceso en modo ráfaga
  - tiempo de acceso: 1er dato→20/50ns // resto →7/15ns
  - temporización: (2+)3-1-1-1 (133 MHz → 37.5 ns // 7.5 ns)
  - sólo módulos DIMM
  - buses de 166 MHz

Memoria DRAM

- **DDR SDRAM (*Double Data Rate SDRAM*):**
  - SDRAM sincronizada en flanco de subida y de bajada
  - buses con frecuencias superiores a 100 MHz (K7 AMD)
  - temporización: (2+)2.5-0.5-0.5-0.5
  - 184 conectores, 2,5V, hasta 500 MHz (4GB/s)
  - DDR2: 240 conectores, 1,8V, hasta 800 MHZ (6,4 GB/s)
  - DDR3: 1,5V, hasta 1.600 MHz (12,8 GB/s)
  - *Dual channel*: 2 canales de acceso independientes
    - DDR400: 3,2 GB/s → 6,4 GB/s
    - DDR2-800: 6,4 GB/s → 12,8 GB/s
    - DDR3-1600: 12,8 GB/s → 25,6 GB/s

Memoria DRAM

- **RDRAM (*Rambus DRAM*):**
  - diseñada para el Pentium IV (bus de la placa a 400 MHz)
  - organización interna completamente diferente:
    - buses internos de 16 bits
    - mayores frecuencias de funcionamiento: 400, 500, 600 MHz
  - memoria síncrona: 2 datos por ciclo de reloj
  - temporización: (9+)9-0.5-0.5-0.5
    memoria de 400 MHz → 45 ns // 1.25 ns
  - mayor coste: tecnología propietaria
  - *dual channel, quad channel*

Memoria DRAM

Memoria	Descripción	Anchura bus	Frecuencia reloj	Frecuencia efectiva	Ancho de banda
SDRAM	PC133	8 bytes	133 MHz	133 MHz	1,06 GB/s
DDR	PC3200*	8 bytes	200 MHz	400 MHz	3,2 GB/s
DDR	PC4000*	8 bytes	250 MHz	500 MHz	4 GB/s
DDR2	PC2-5300**	8 bytes	333 MHz	667 MHz	5,3 GB/s
DDR2	PC2-6400**	8 bytes	400 MHz	800 MHz	6,4 GB/s
DDR3	PC3-8500***	8 bytes	533 MHz	1066 MHz	8,5 GB/s
DDR3	PC3-10660***	8 bytes	667 MHz	1333 MHz	10,6 GB/s
DDR3	PC3-12800***	8 bytes	800 MHZ	1600 MHz	12,8 GB/s
DRDRAM	PC1600	2 bytes	400 MHz	800 MHz	1,6 GB/s
DRDRAM	PC3200	2 bytes	800 MHz	1600 MHz	3,2 GB/s

* También se nombran como DDR400, DDR500, etc.

** También se nombran como DDR2-667, DDR2-800, etc.

*** También se nombran como DDR3-1066, DDR3-1333, DDR3-1600