Tema 5: Organización de la memoria: memoria principal.

Objetivos:

- Conocer las características generales de los diferentes tipos de memoria que aparecen en un computador digital y analizar la necesidad de su organización jerárquica.
- Estudiar las diferentes formas de configuración interna de la memoria principal de un computador.
- Conocer las técnicas de diseño de memorias a partir de módulos más elementales así como las funciones de selección que permiten ubicarlas en el espacio de direcciones.
- Diseñar sistemas de memoria con detección y corrección de fallos.
- Aumentar el ancho de banda de la memoria principal utilizando un diseño modular con acceso simultáneo a cada módulo y técnicas de entrelazado de direcciones.

Contenido

1. Características generales de las memorias

Las memorias se pueden clasificar atendiendo a diferentes criterios. Revisaremos en los apartados siguientes los más significativos:

1.1. Método de acceso

- Acceso aleatorio (RAM): acceso directo y tiempo de acceso constante e independiente de la posición de memoria.
- Acceso secuencial (SAM): tiempo de acceso dependiente de la posición de memoria.
- Acceso directo (DAM): acceso directo a un sector con tiempo de acceso dependiente de la posición, y acceso secuencial dentro del sector.
- Asociativas CAM): acceso por contenido

1.2. Soporte físico

- Semiconductor
- Magnéticas
- Ópticas
- Magneto-ópticas

1.3. Alterabilidad

- RAM: lectura y escritura
- ROM (*Read 0nly Memory*): Son memorias de sólo lectura. Existen diferentes variantes:
  - ROM programadas por máscara, cuya información se escribe en el proceso de fabricación y no se puede modificar.
  - PROM, o ROM programable una sola vez. Utilizan una matriz de diodos cuya unión se puede destruir aplicando sobre ella una sobretensión.
  - EPROM (*Erasable PROM)* o RPROM (Reprogramable *ROM),* cuyo contenido puede borrarse mediante rayos ultravioletas para volverlas a escribir.
  - EAROM (*Electrically Alterable ROM)* o EEROM (*Electrically Erasable ROM),* son memorias que están entre las RAM y las ROM ya que su contenido se puede volver a escribir por medios eléctricos. Se diferencian de las RAM en que no son volátiles.
  - Memoria FLASH. Utilizan tecnología de borrado eléctrico al igual que las EEPROM, pero pueden ser borradas y reprogramadas en bloques, y no palabra por palabra como ocurre con las tradicionales EEPROM. Ofrecen un bajo consumo y una alta velocidad de acceso, alcanzando un tiempo de vida de unos 100.000 ciclos de escritura.

1.4. Volatilidad con la fuente de energía

- Volátiles: necesitan la fuente de energía para mantener la información.
- No volátiles: mantienen la información sin aporte de energía.

1.5. Duración de la información

- Estáticas: el contenido permanece inalterable mientras están polarizadas.
- Dinámicas: el contenido sólo dura un corto período de tiempo, por lo que es necesario refrescarlo (reescribirlo) periódicamente.

1.6. Proceso de lectura

- Lectura destructiva: necesitan reescritura después de una lectura.
- Lectura no destructiva

1.7. Ubicación en el computador

- Interna (CPU): registros, cache(L1), cache(L2), cache(L3), memoria principal
- Externa (E/S): discos, cintas, etc.

1.8. Parámetros de velocidad

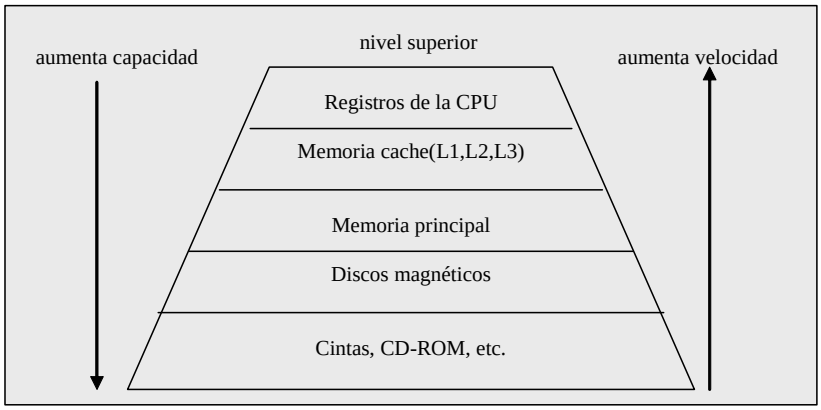
- Tiempo de acceso
- Tiempo de ciclo
- Ancho de banda(frecuencia de acceso)

1.9. Unidades de transferencia

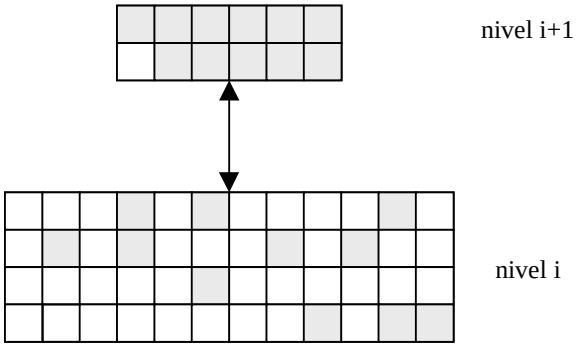
- Palabras
- Bloques

1.10. Jerarquía de las unidades de memoria de un computador

Las distintas memorias presentes en un computador se organizan de forma jerárquica:



En el nivel *i+1* se ubica una copia de aquellos bloques del nivel *i* que tienen mayor probabilidad de ser referenciados en el futuro inmediato



Este mecanismo de migración entre niveles es efectivo gracias al principio de localidad referencial que manifiestan los programas:

- espacial
- temporal

Se consigue que el mayor número de referencias generado por los programas correspondan a informaciones ubicadas en los niveles más altos de la jerarquía.

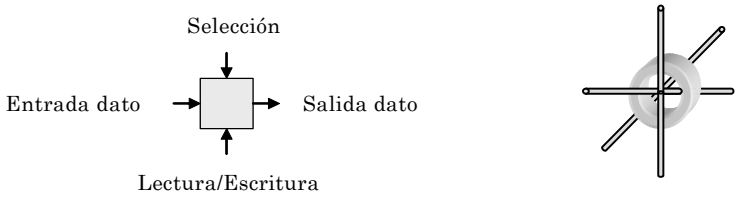
2. Organización interna de la memoria principal.

Una memoria principal se compone de un conjunto de celdas básicas dotadas de una determinada organización. Cada celda soporta un bit de información. Los bits se agrupan en unidades direccionables denominadas palabras. La longitud de palabra la determina el número de bits que la componen y constituye la resolución de la memoria (mínima cantidad de información direccionable). La longitud de palabra suele oscilar desde 8 bits (*byte*) hasta 64 bits.

Cada celda básica es un dispositivo físico con dos estados estables (o semi-estables) con capacidad para cambiar el estado (escritura) y determinar su valor (lectura). Aunque en los primeros computadores se utilizaron los materiales magnéticos como soporte de las celdas de memoria principal (memorias de ferritas, de película delgada, etc.) en la actualidad sólo se utilizan los materiales semiconductores.

Dentro de las memorias electrónicas de semiconductor podemos distinguir dos grandes grupos: las estáticas (SRAM: *Static Random Access Memory*) y las dinámicas (DRAM: *Dynamic Random Access Memory*). Las estáticas utilizan el principio de biestabilidad que se consigue con dos puertas inversoras (NAND ó NOR) realimentadas, mientras que las dinámicas aprovechan la carga o ausencia de carga de un pequeño condensador creado en un material semiconductor. Debido a la descarga natural que sufren las celdas cargadas, las memorias dinámicas necesitan un sistema de refresco que periódicamente - y antes que la carga eléctrica del condensador se haga indetectable - recargue las celdas que se encuentran en estado de carga.

Desde un punto de vista conceptual y con independencia de la tecnología, consideraremos la celda básica de memoria como un bloque con tres líneas de entrada (entrada dato, selección y lectura/escritura) y una de salida (salida dato). La celda sólo opera (lectura ó escritura) cuando la selección está activa.

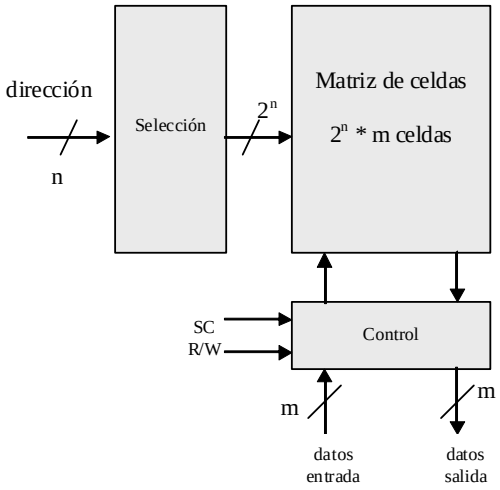


2.1. Organización interna de la memoria.

Las celdas de memoria se disponen en el interior de un chip atendiendo a dos organizaciones principales: la organización por palabras, también denominada 2D, y la organización por bits, también denominada 2 ½ D o 3D.

2.2. Organización 2D

Es la organización más sencilla que responde al esquema mostrado en la siguiente figura:

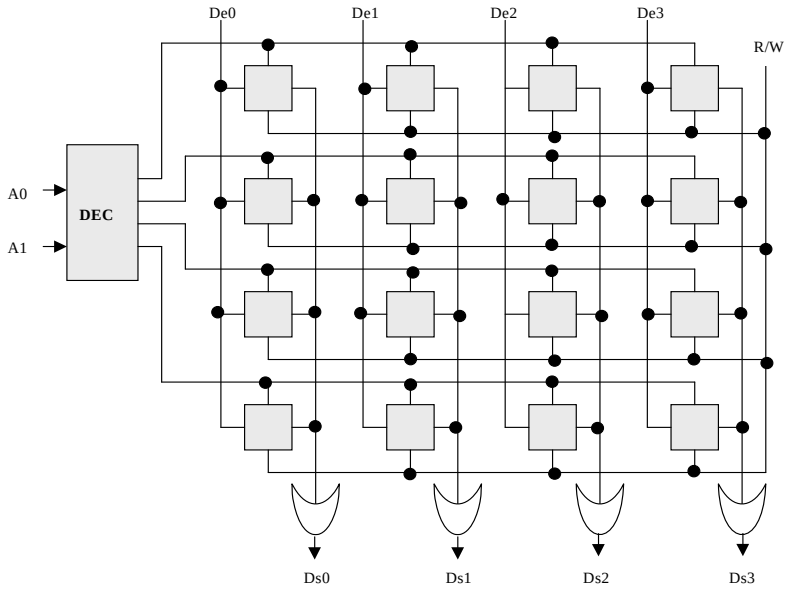


Las celdas forman una matriz de 2^n filas y m columnas, siendo 2^n el número de palabras del chip y m el número de bits de cada palabra. Cada fila es seleccionada por la decodificación de una configuración diferente de los n bits de dirección.

Esta organización tiene el inconveniente que el selector (decodificador) de palabras crece exponencialmente con el tamaño de la memoria. Igual le ocurre al número de entradas (*fan-in*) de las puertas OR que generan la salida de datos.

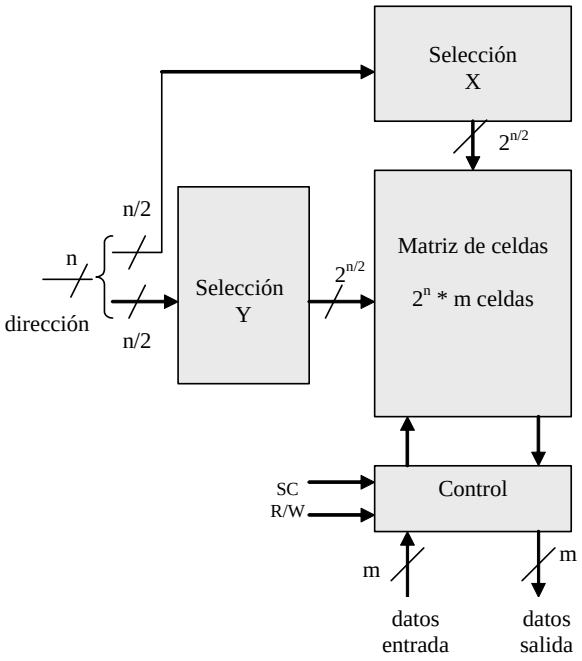
Ejemplo

En la siguiente figura se muestra la organización 2D de un chip de memoria con 4 palabras de 4 bits:

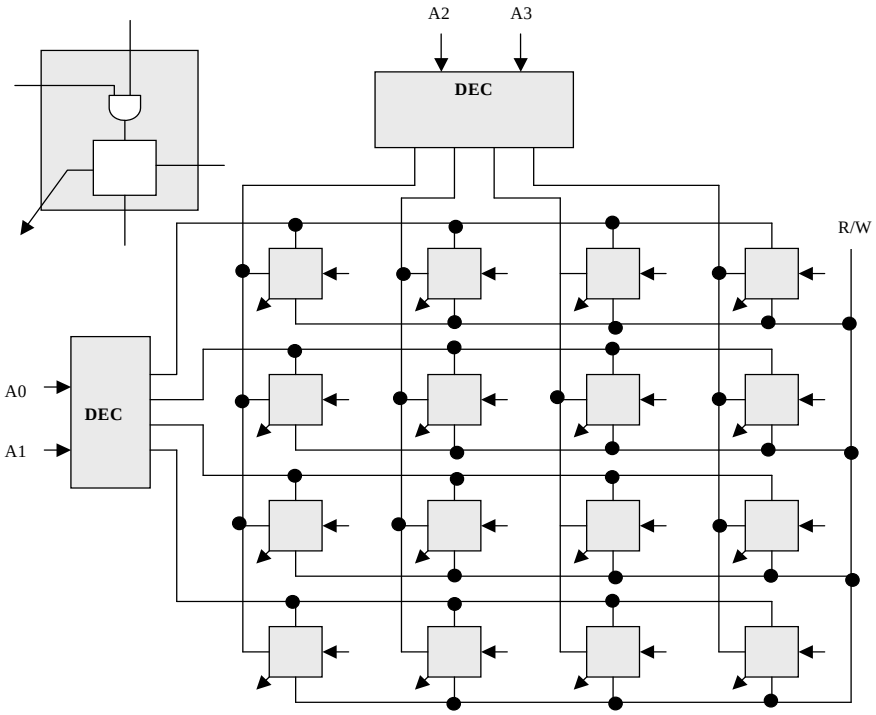


2.3. Organización 3D

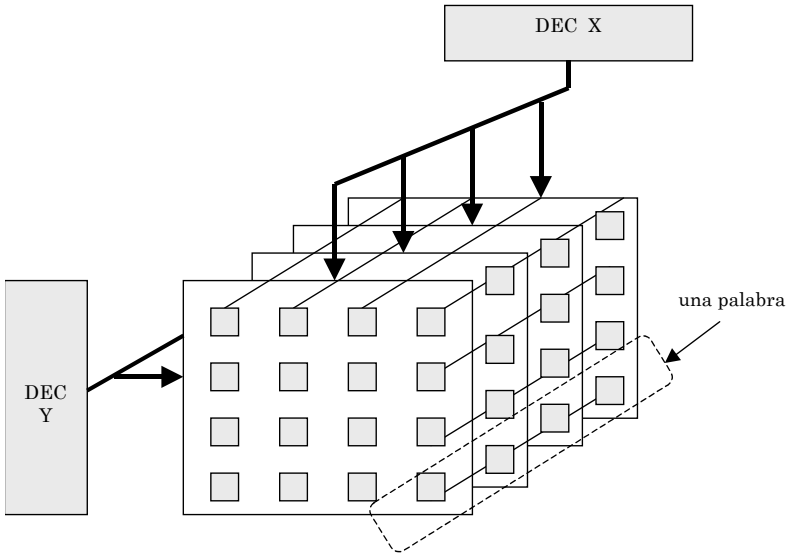
En lugar de una única selección (decodificador) de 2^n salidas en esta organización se utilizan dos decodificadores de $2^{n/2}$ operando en coincidencia. Las líneas de dirección se reparten entre los dos decodificadores. Para una configuración dada de las líneas de dirección se selecciona un único *bit* de la matriz. Por ello se la denomina también organización por bits.



Ejemplo:



En esta organización se necesitan varias matrices de celdas básicas, tantas como bits deba tener la palabra de memoria, actuando sobre ellas en paralelo los circuitos de decodificación:

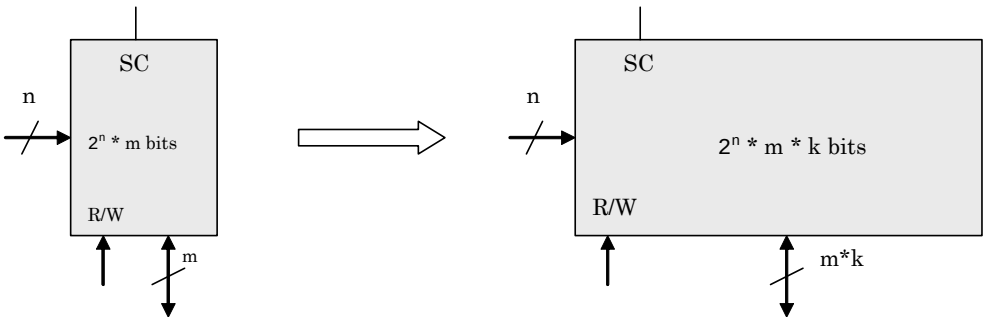


3. Diseño de memorias

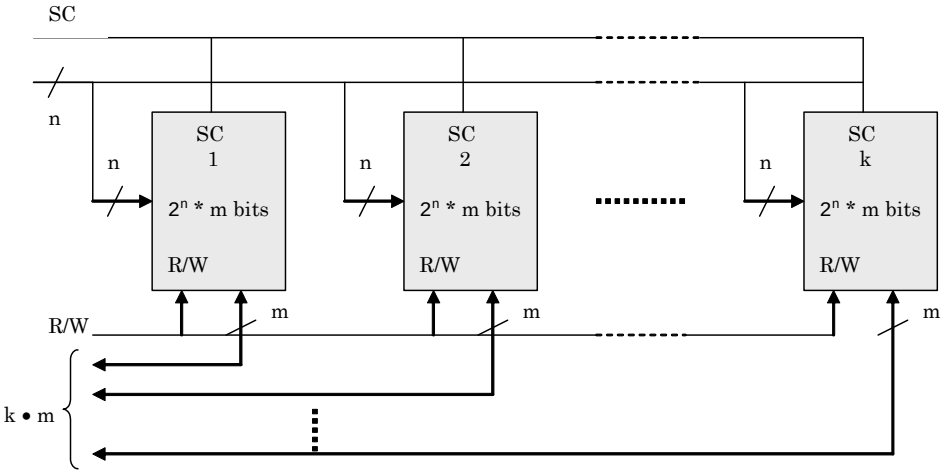
Cuando se ha de diseñar una memoria principal cuyas dimensiones (número de bits y número de palabras) exceden a las de un chip, se tienen que disponer varios chips en una placa de circuito impreso para alcanzar las dimensiones requeridas. Para mayor claridad trataremos independientemente cada una de las dimensiones.

3.1. Ampliación del número de bits de la palabra de memoria

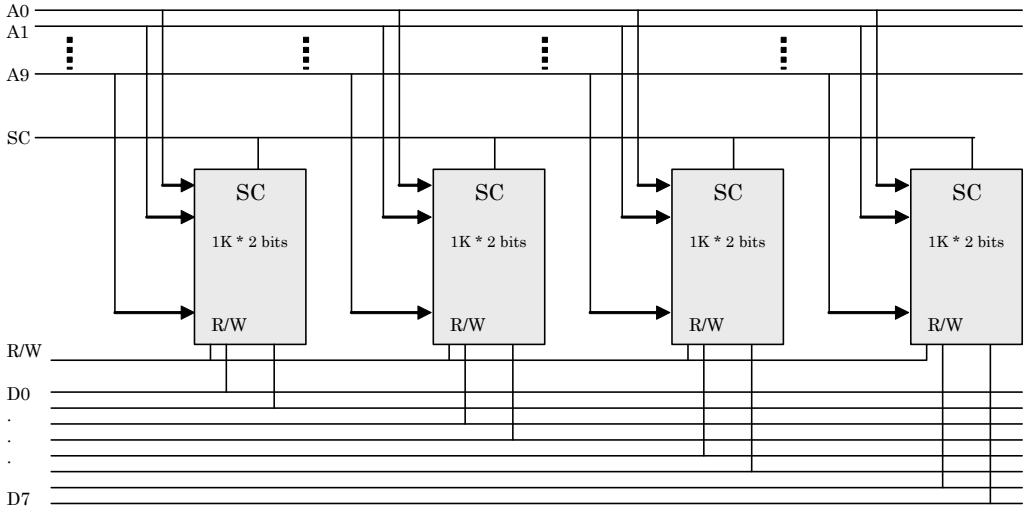
Se trata de formar una memoria de $2^n * (m * k)$ *bits* a partir de chips de $2^n * m$ *bits*



El esquema general se muestra en la siguiente figura. Simplemente se disponen en paralelo k chips de $2^n * m$ *bits* al que llegarían las mismas líneas de dirección y control. Cada chip aportaría *m* líneas de datos a la palabra de la memoria total.

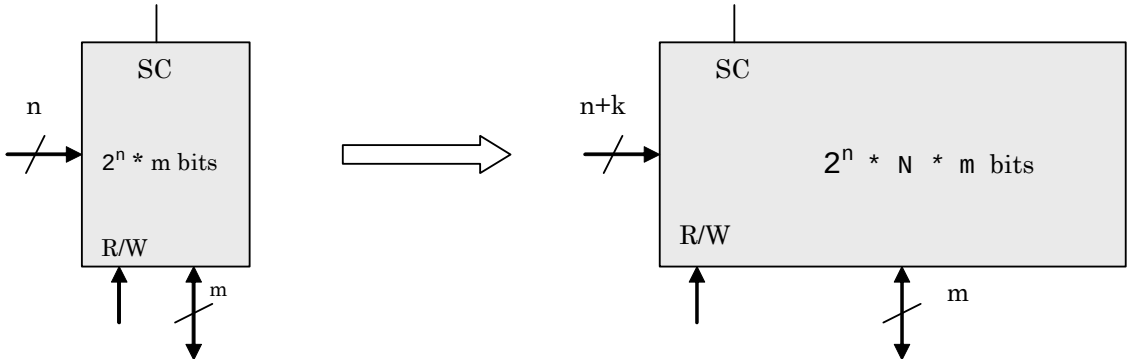


Ejemplo: Diseño de una memoria de 1K * 8 bits a partir de módulos (chips) de 1K * 2 bits;

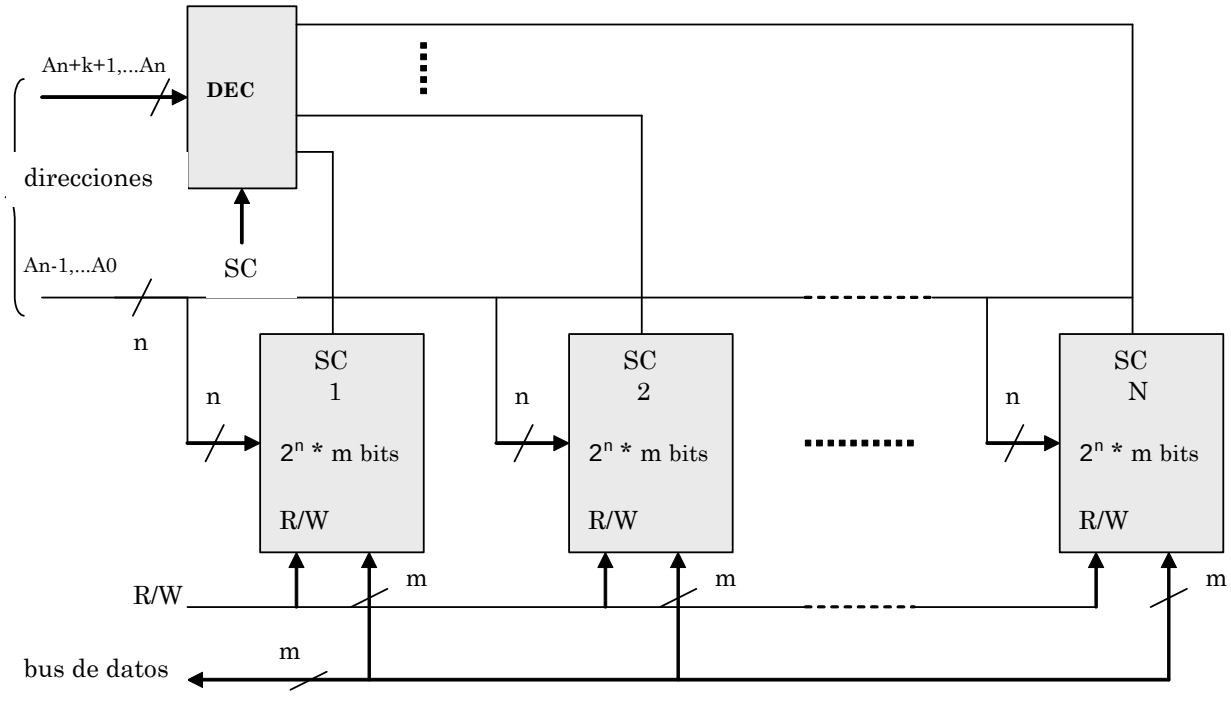


3.2. Ampliación del número de palabras de memoria

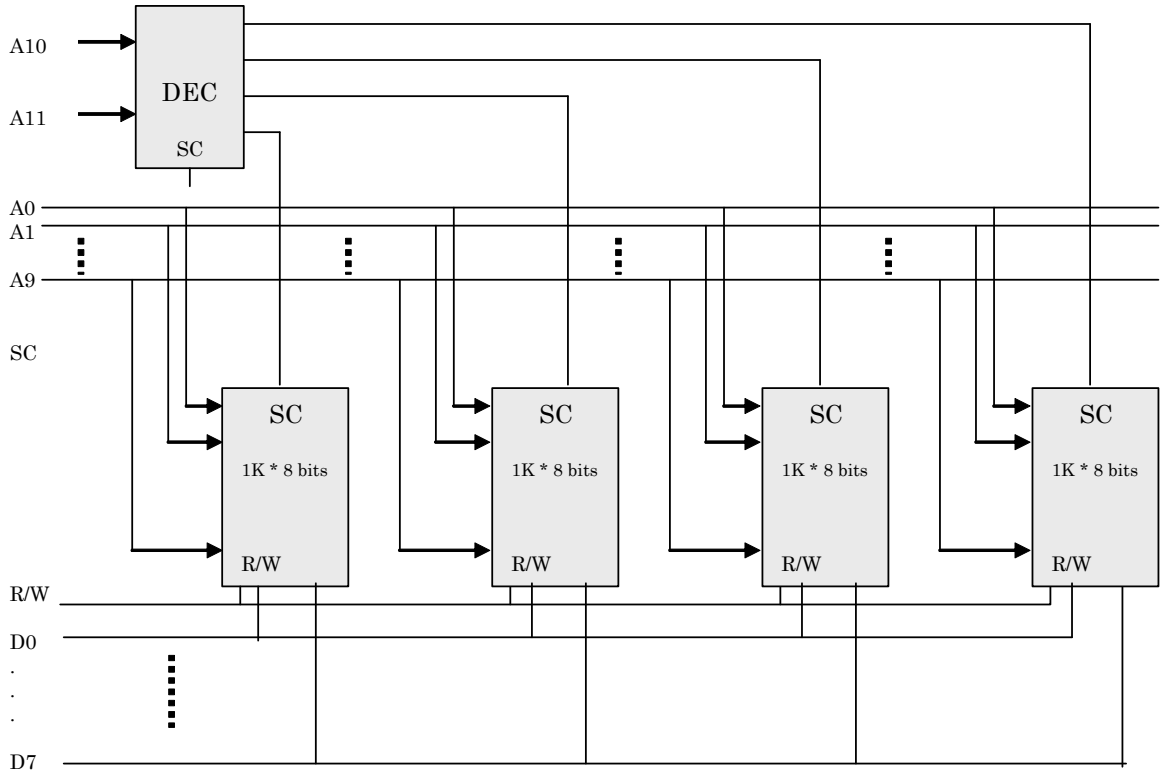
Se trata de formar una memoria de $2^n * m * N$ *bits* a partir de chips de $2^n * m$ *bit,* es decir, aumentar el número de palabras manteniendo la misma longitud de palabra.



El esquema general se muestra en la siguiente figura. La nueva memoria tendrá n + k líneas de dirección. Se disponen $N=2^k$ chips en paralelo a los que se llevan las mismas m líneas de datos, las mismas n líneas de dirección menos significativas y la misma línea de lectura/escritura (R/W). Las k líneas de dirección más significativas se decodifican para activar con cada salida del decodificador el selector de chip (SC) de cada uno de los $N=2^k$ chips.

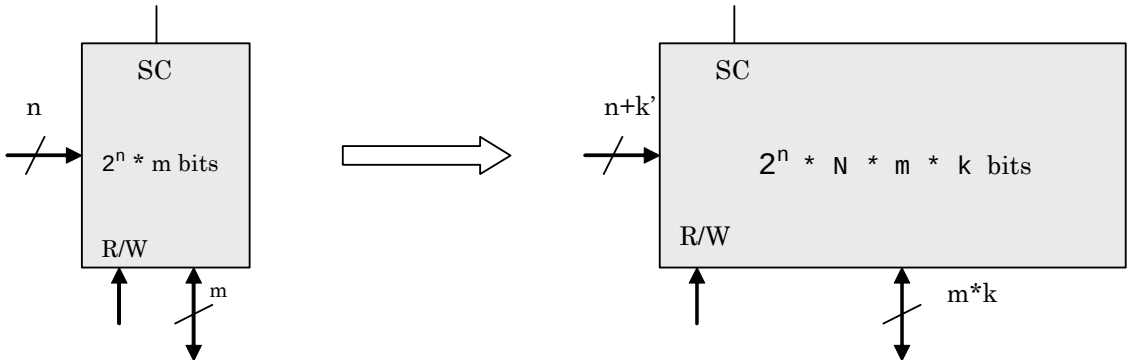


Ejemplo: Diseño de una memoria de 4K * 8 bits con módulos de 1K * 8 bits;



3.3. Ampliación de la longitud y el número de palabras de memoria

En este caso ampliaríamos en primer lugar el número de líneas de datos (longitud de palabra) y con los bloques resultantes diseñaríamos una memoria con mayor número de palabras.

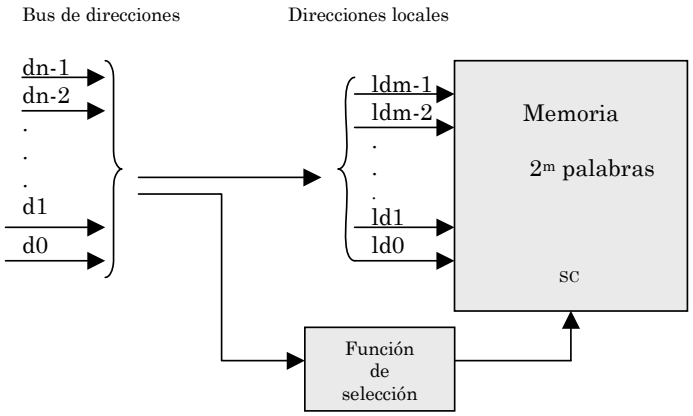


3.4. Ubicación en el espacio de direcciones.

Dado un espacio de direcciones *dn-1, dn-2,...,d1, d0* y una memoria de capacidad 2^m palabras($m \leq n$)se trata de hallar la función lógica de selección (SC) que ubique a la memoria en un segmento de direcciones consecutivas *Di,...Dj*, así como las señales locales de dirección de la memoria.

	dn-1 dn-2..... d1 d0
D0	0 0 ... 0 0
D1	0 0 ... 0 1
D2	0 0 ... 1 0
D3	0 0 ... 1 1
.	.
.	.
.	.
Di	----------
.	Memoria
.	2^m palabras
.	
Dj	----------
.	
.	
.	
$D2^n-1$	1 1 ... 1 1

Debe cumplirse que $Dj - Di + 1 = 2^m$



Si la memoria debe ocupar posiciones consecutivas, las líneas menos significativas del bus de direcciones se conectarán con las líneas de direcciones locales de la memoria:

$$ld0 = d0$$
$$ld1 = d1$$
$$\vdots$$
$$ldm\text{-}2 = dm\text{-}2$$
$$ldm\text{-}1 = dm\text{-}1$$

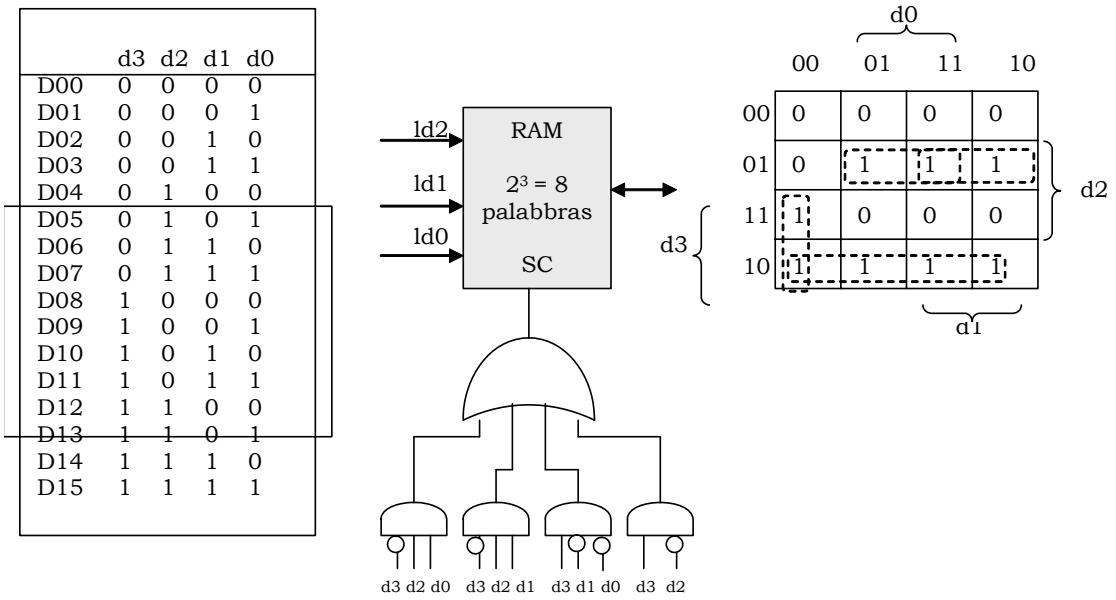
La función de selección será una función booleana de las señales del bus de direcciones:

$$SC = fb(dn\text{-}1, dn\text{-}2,...,d1, d0)$$

Ejemplo: Ubicar una memoria de 8 palabras en un espacio de 4 líneas de dirección (d3,d2,d1,d0).

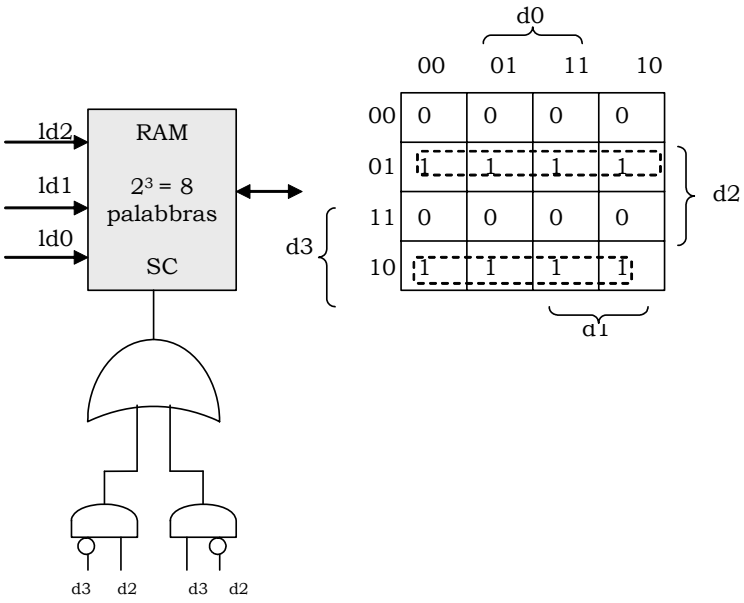
<u>Ubicación 1</u>: a partir de la dirección 5 (0101)

	d3	d2	d1	d0
D00	0	0	0	0
D01	0	0	0	1
D02	0	0	1	0
D03	0	0	1	1
D04	0	1	0	0
D05	0	1	0	1
D06	0	1	1	0
D07	0	1	1	1
D08	1	0	0	0
D09	1	0	0	1
D10	1	0	1	0
D11	1	0	1	1
D12	1	1	0	0
D13	1	1	0	1
D14	1	1	1	0
D15	1	1	1	1

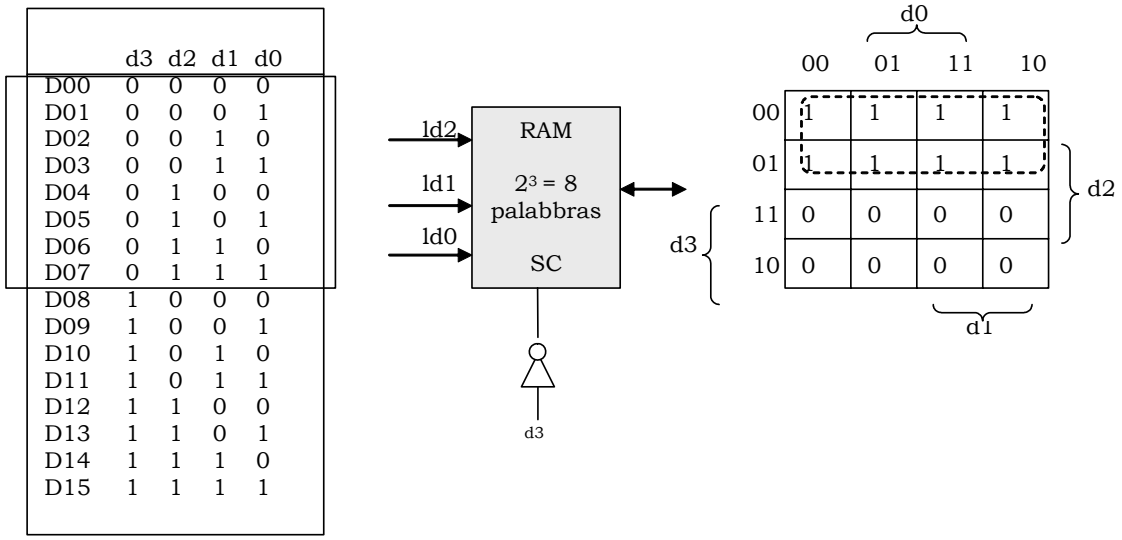


<u>Ubicación 2</u>: a partir de la dirección 4 (0100)

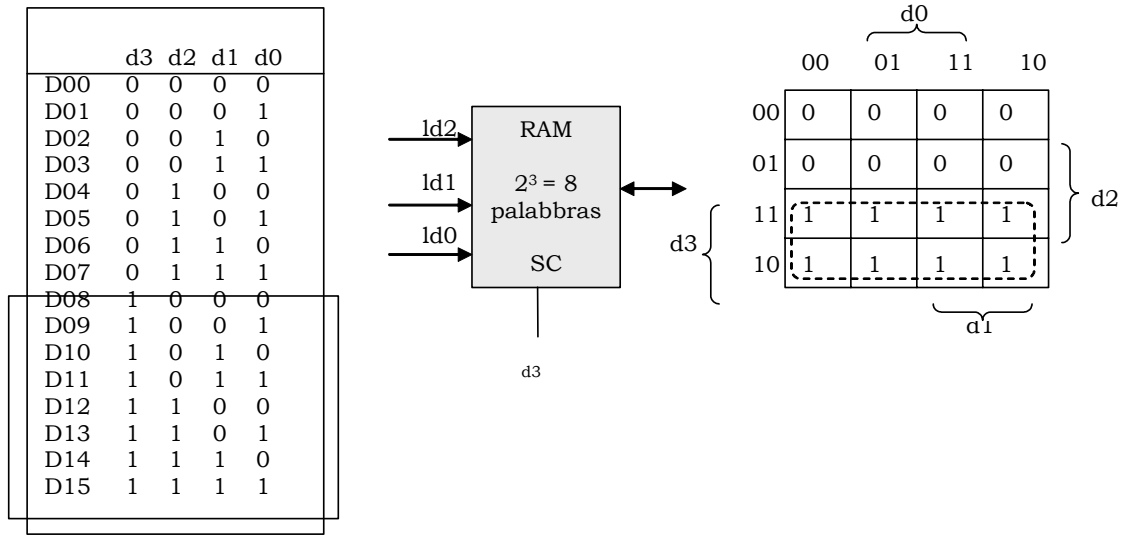
	d3	d2	d1	d0
D00	0	0	0	0
D01	0	0	0	1
D02	0	0	1	0
D03	0	0	1	1
D04	0	1	0	0
D05	0	1	0	1
D06	0	1	1	0
D07	0	1	1	1
D08	1	0	0	0
D09	1	0	0	1
D10	1	0	1	0
D11	1	0	1	1
D12	1	1	0	0
D13	1	1	0	1
D14	1	1	1	0
D15	1	1	1	1



<u>Ubicación 3</u>: a partir de la dirección 0 (0000)



<u>Ubicación 4</u>: a partir de la dirección 8 (1000)



<u>Conclusión</u>: la función de selección es más simple cuanto más alineado se encuentre el bloque de memoria en el espacio de direcciones. La máxima alineación la tendremos cuando se cumpla que:

$$dirección\ inicial\ mod\ 2^n = 0$$

Es decir, el resto de dividir la dirección inicial por el tamaño de bloque es igual a 0.

<u>Caso de módulos con más de un selector</u>

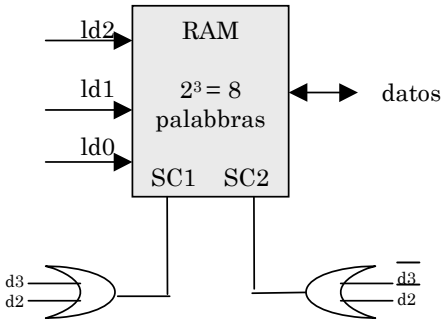
Se simplifica la lógica global de selección, pero hay que descomponer la función en un producto.

<u>Ejemplo</u>: ubicación 2 (anterior)

función de selección: $SC = d2 \bullet \overline{d3} + d2 \bullet d3$

que en forma de producto resulta:

$$SC = (d2 + d3) \bullet \overline{(d2 + d3)}$$

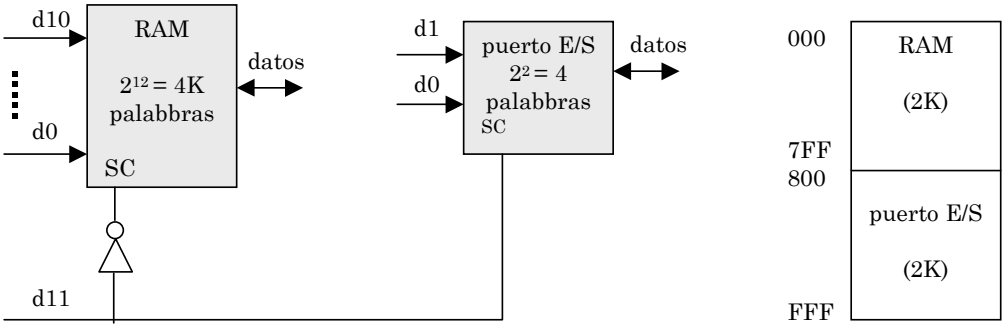


3.4.1. Decodificación parcial y total

Cuando se generan las funciones de selección de varios bloques de memoria o puertos de E/S que han de ubicarse en el espacio de direcciones tiene que evitarse siempre que una dirección sea asignada a más de una posición de memoria o registro de E/S. En caso contrario el resultado de las operaciones de lectura/escritura puede resultar incierto. Sin embargo, no es necesario que a una posición de memoria o registro le corresponda sólo una dirección. En algunos casos, para simplificar la decodificación, y siempre que no se vaya a utilizar totalmente el espacio disponible, se puede realizar una decodificación parcial.

Ejemplo:

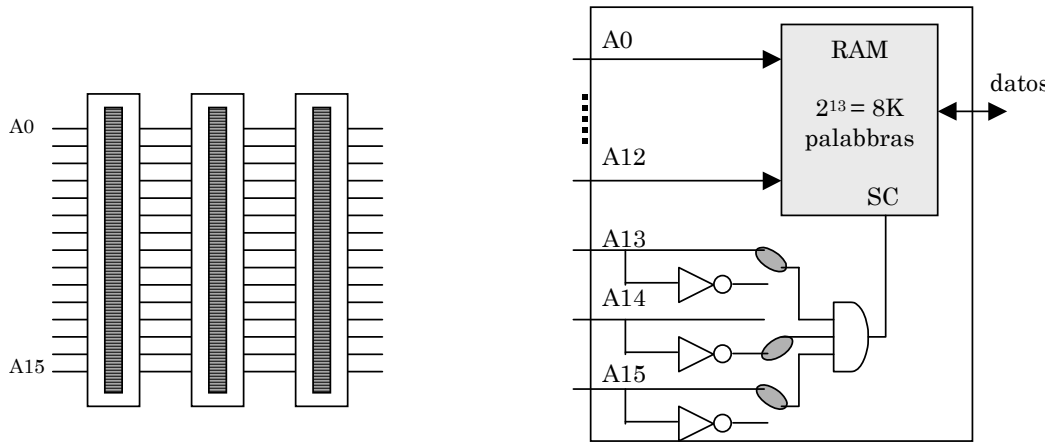
Ubicación en un espacio de direcciones de 12 líneas d11, d10,...,d1, d0 (2^{12} = 4.086 = 4K) de un bloque de RAM de 2K y un puerto de E/S de 4 registros.



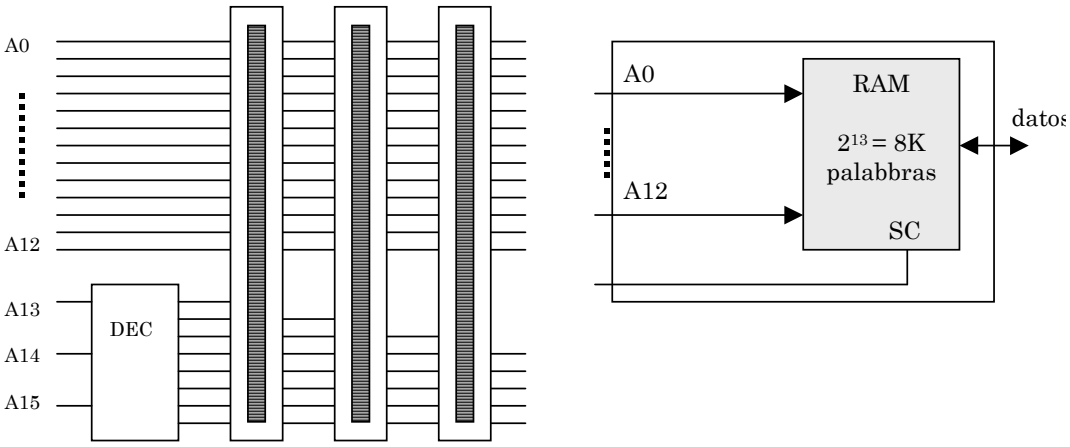
La RAM la hemos decodificado completamente, en cambio el puerto de E/S presenta una decodificación parcial, ya que hemos utilizado 2K del espacio disponible para ubicar tan sólo 4 palabras. Esto significa que los 4 registros presentan múltiples direcciones cada uno (exactamente 512):

R0	800	804	808	80C	...
R1	801	805	809	80D	...
R2	802	806	80A	80D	...
R3	803	807	80B	80F	...

Decodificación en placa



Decodificación en bus (*back plane*)



3.5. Disposición de los módulos de memoria

3.5.1. SIMM (Single In-line Memory Module).

Un SIMM típico consta de varios chips de DRAM instalados en una pequeña placa de circuito impreso (*PCB*) que se fija verticalmente a través de un conector a la placa del sistema. Los SIMMs disponen de varios formatos y número de contactos. Una de las ventajas de la memoria SIMM es la posibilidad de instalar gran cantidad de memoria en un área reducida. Algunos SIMMs de 72 contactos contienen 20 ó más chips de DRAM; 4 de estos SIMMs contienen, pues, 80 ó más chips de DRAM. Ocupan un área de 58 cm2 , mientras que si los chips se instalaran horizontalmente en la placa del sistema ocuparían 135 cm2.

3.5.2. DIMM (Dual In-line Memory Module).

Los módulos de memoria DIMM, al igual que los SIMMs, se instalan verticalmente en los conectores de expansión. La diferencia principal estriba en que en los SIMMs los contactos de cada fila se unen con los contactos correspondientes de la fila opuesta para formar un solo contacto eléctrico; mientras que en los DIMMs los contactos opuestos permanecen eléctricamente aislados para formar dos contactos independientes.

4. Detección y corrección de errores.

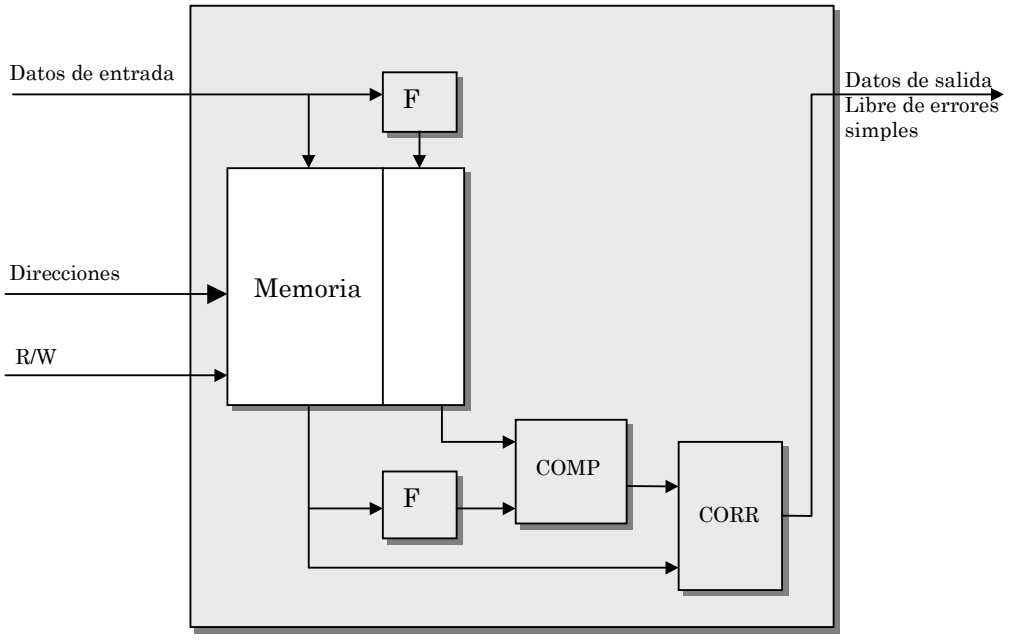
4.1. Códigos detectores de errores

Para la detección de errores simples en dispositivo de memoria se suelen utilizar los bits de paridad. Es decir, se amplía en 1 la longitud de palabra para almacenar en las operaciones de escritura la paridad total (par o impar) de la palabra. En las operaciones de lectura se comprueba si se mantiene la paridad convenida. Las detecciones de fallos de la memoria se suelen traducir en excepciones (interrupciones) del sistema.

4.2. Códigos correctores de errores (Hamming)

El código de Hamming permite la corrección de un error simple. El dato a transmitir de *m bits* de longitud se amplía con *k bits* de paridad, suficientes para codificar la posición de un posible bit erróneo en el mensaje completo de *m* + *k bits*. Es decir, deberá cumplirse que $2^k - 1 \geq m + k$.

Nº de bits del dato	Nº de bits de paridad	Nº total de bits	Incremento porcentual
8	4	12	50 %
16	5	21	31,25 %
32	6	38	18,75 %
64	7	71	10,94 %
128	8	135	6,25 %
256	9	265	3,52 %



Ejemplo

Supongamos que la longitud de palabra de la memoria es m = 8.

Los datos de m = 8 bits

(*d8, d7, d6, d5, d4, d3, d2, d*1)

los ampliamos con k = 4 bits de paridad

(*C8, C4, C2, C1*)

para formar un mensaje de m + k = 12 bits

(*m12, m11, m10, m9, m8, m7, m6, m5, m4, m3, m2,* m1).

Los bits de paridad *Ci* ocupan las posiciones en el mensaje correspondientes a las potencias de 2, es decir la 1, 2, 4 y 8 (de aquí el subíndice i).

Con los bits (*C8, C4, C2, C1*) se codifican todas las posiciones del mensaje (12), es decir:

(*C8, C4, C2, C1*) = 0 0 0 1 = 1 en decimal para *m1*, ...

(*C8, C4, C2, C1*) = 1 1 0 0 = 12 en decimal para *m12,*

de manera que cuando se produzca un error simple (complementación de un bit) en el bit *mj* del mensaje, (*C8, C4, C2, C1*) sea igual a *j* en decimal.

Es decir, *Ci* deberá recoger la paridad de todos los *mj* que tengan *Ci = 1* en su codificación, incluido el propio *Ci*:

d8	d7	d6	d5		d4	d3	d2		d1			datos
				C8				C4		C2	C1	paridades
m12	m11	m10	m9	m8	m7	m6	m5	m4	m3	m2	m1	mensaje
1	1	1	1	1	0	0	0	0	0	0	0	C8
1	0	0	0	0	1	1	1	1	0	0	0	C4
0	1	1	0	0	1	1	0	0	1	1	0	C2
0	1	0	1	0	1	0	1	0	1	0	1	C1

d8	⊕d7	⊕d6	⊕d5									= C8
d8					⊕d4	⊕d3	⊕d2					= C4
	d7	⊕d6			⊕d4	⊕d3			⊕d1			= C2
	d7		⊕d5		⊕d4		⊕d2		⊕d1			= C1

El sistema de memoria deberá generar en la escritura los bits de paridad Ci para almacenarlos junto a los bits de datos

Supongamos que el dato **escrito es**: d8 d7 d6 d5 d4 d3 d2 d1 = 100**1** 1101

Los bits de paridad para el código de Hamming valdrán:

Función F:

C8 = d8⊕d7⊕d6⊕d5 = 1 ⊕ 0 ⊕ 0 ⊕ 1 = 0

C4 = d8⊕d4⊕d3⊕d2 = 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1

C2 = d7⊕d6⊕d4⊕d3⊕d1 = 0 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 1 = 1

C1 = d7⊕d5⊕d4⊕d2⊕d1 = 0 ⊕ 1 ⊕ 1 ⊕ 0 ⊕ 1 = 1

En la lectura se generará de nuevo los bits de paridad con el bloque F' a partir de los bits de datos leídos de memoria. El bloque F' responde a la misma función lógica que F. Supongamos que en la lectura se produce un error simple en d5, es decir, escribimos un **1** y hemos leído un **0**, el valor de F' será:

Función F':

C'8 = d'8⊕d'7⊕d'6⊕d'5 = 1 ⊕ 0 ⊕ 0 ⊕ **0** = **1**

C'4 = d'8⊕d'4⊕d'3⊕d'2 = 1 ⊕ 1 ⊕ 1 ⊕ 0 = 1

C'2 = d'7⊕d'6⊕d'4⊕d'3⊕d'1 = 0 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 1 = 1

C'1 = d'7⊕d'5⊕d'4⊕d'2⊕d'1 = 0 ⊕ **0** ⊕ 1 ⊕ 0 ⊕ 1 = **0**

Para detectar la existencia de un error simple y su posición (para poderlo corregir) comparamos (Comparador COMP) los bits de paridad generados en la lectura, C'i con los almacenados en la escritura, Ci:

Comporador COMP:

C"8 = C8 ⊕ C'8 = 0 ⊕ 1 = 1

C"4 = C4 ⊕ C'4 = 1 ⊕ 1 = 0

C"2 = C2 ⊕ C'2 = 1 ⊕ 1 = 0

C"1 = C1 ⊕ C'1 = 1 ⊕ 0 = 1

Si el resultado de la comparación es C"8 C"4 C"2 C"1 = 0 0 0 0 significa que no se ha producido error. Si se ha producido un error, los Ci" codificarán su posición.

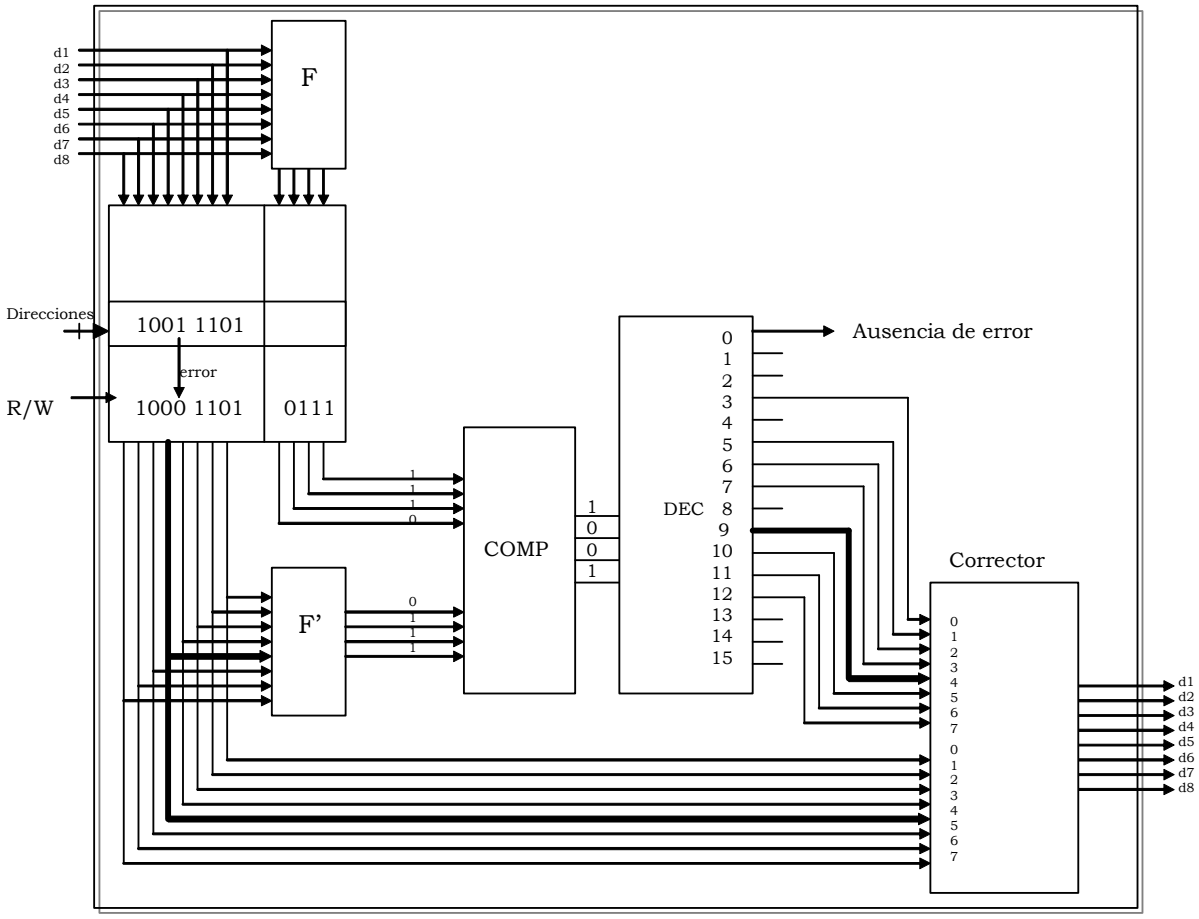
En nuestro ejemplo el resultado de la comparación vale

C"8 C"4 C"2 C"1 = 1 0 0 1 = 9 (decimal) ➔ el bit *m9* del mensaje ha fallado, es decir el bit de datos *d5*

Para corregir el error decodificamos su posición, es decir, los Ci" (decodificador DEC), con lo que disponemos de una línea individual para cada posible posición de error.

Con cada una de estas líneas podemos controlar la inversión o no de la correspondiente línea de datos *di'* leída de memoria, según que se haya producido o no error en dicha línea. Para la inversión controlada de estas líneas utilizamos una etapa de puertas XOR.

En la siguiente figura aparece el esquema completo de la memoria tolerante a un fallo simple y el proceso de corrección del fallo que hemos supuesto en el ejemplo



F' = F

F

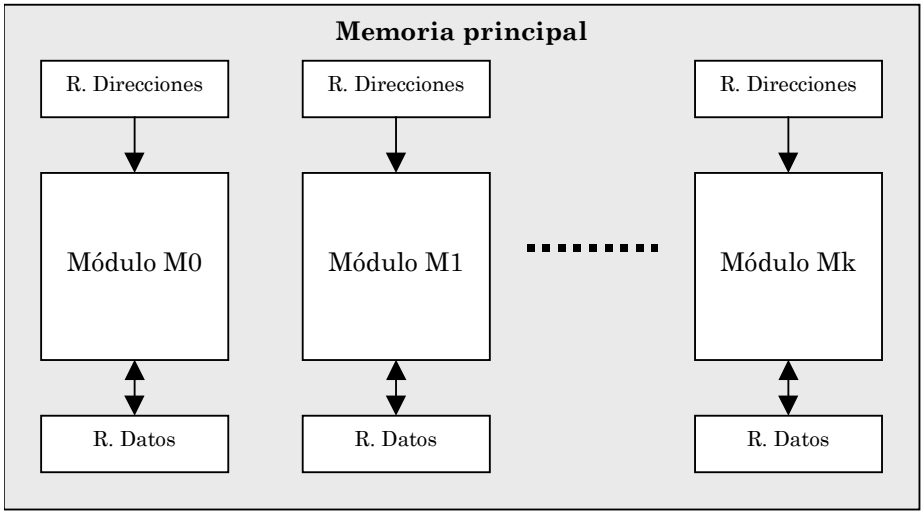
C8 = d8⊕d7⊕d6⊕d5
C4 = d8⊕d4⊕d3⊕d2
C2 = d7⊕d6⊕d4⊕d3⊕d1
C1 = d7⊕d5⊕d4⊕d2⊕d1

Corrector

D8 = dec12 ⊕ d'8
D7 = dec11 ⊕ d'7
D6 = dec10 ⊕ d'6
D5 = dec9 ⊕ d'5
D4 = dec7 ⊕ d'4
D3 = dec6 ⊕ d'3
D2 = dec5 ⊕ d'2
D1 = dec3 ⊕ d'1

5. Memoria entrelazada.

Para aumentar el ancho de banda de una memoria principal se puede descomponer en módulos con accesos independientes, de manera que se pueda acceder simultáneamente a una palabra de cada uno de los módulos.

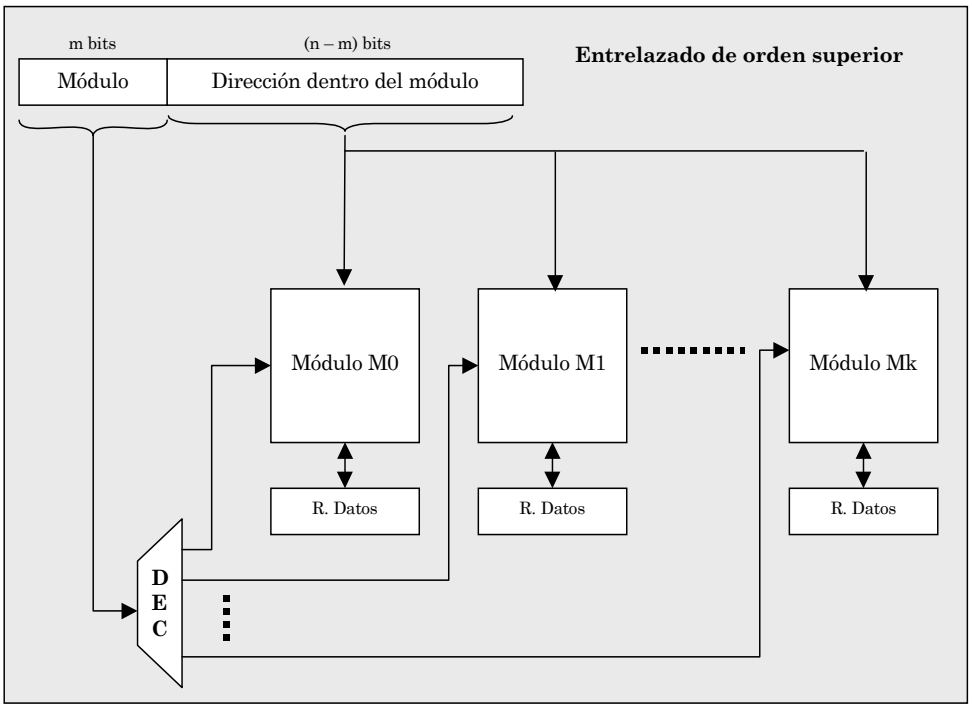


Existen diferentes elementos de diseño en una memoria modular, siendo el orden del entrelazado de las direcciones uno de los principales. Básicamente existen dos tipos de entrelazado para el espacio de direcciones de una memoria: entrelazado de orden superior y entrelazado de orden inferior.

5.1. Entrelazado de orden superior

Mp = 2^n palabras

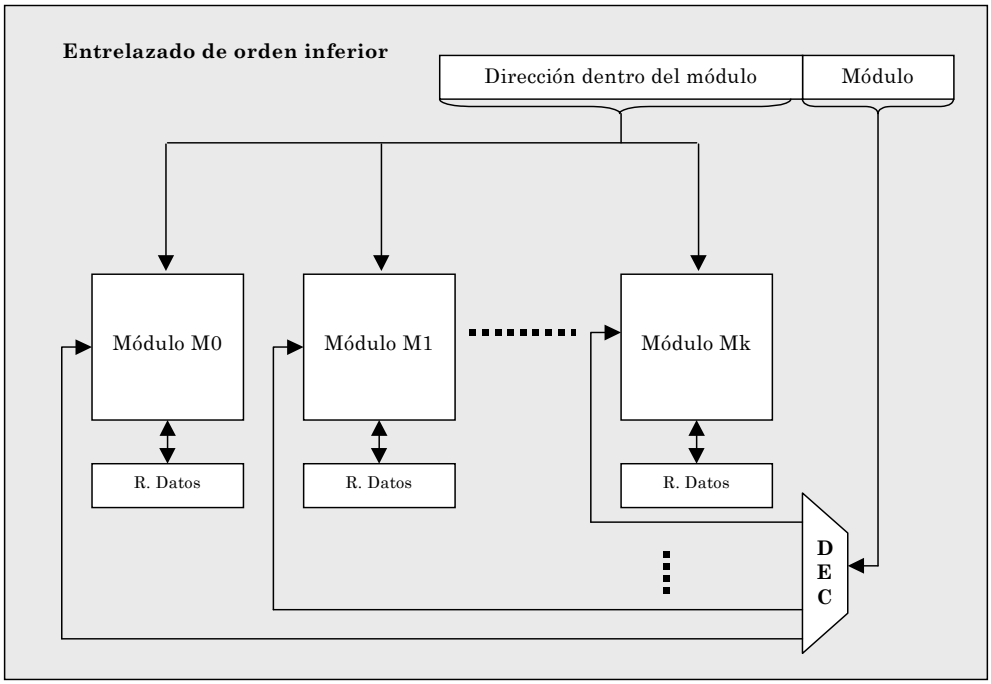
Se divide Mp en 2^m módulos de 2^{n-m} palabras



Ejemplo: Para n = 5 y m = 2 tendríamos una distribución de direcciones de manera que las consecutivas irán en el mismo módulo:

módulo	Dir.módulo	M0	módulo	Dir.módulo	M1	módulo	Dir.módulo	M2	módulo	Dir.módulo	M3
00	000	0	01	000	8	10	000	16	11	000	24
00	001	1	01	001	9	10	001	17	11	001	25
00	010	2	01	010	10	10	010	18	11	010	26
00	011	3	01	011	11	10	011	19	11	011	27
00	100	4	01	100	12	10	100	20	11	100	28
00	101	5	01	101	13	10	101	21	11	101	29
00	110	6	01	110	14	10	110	22	11	110	30
00	111	7	01	111	15	10	111	23	11	111	31

5.2. Entrelazado de orden inferior

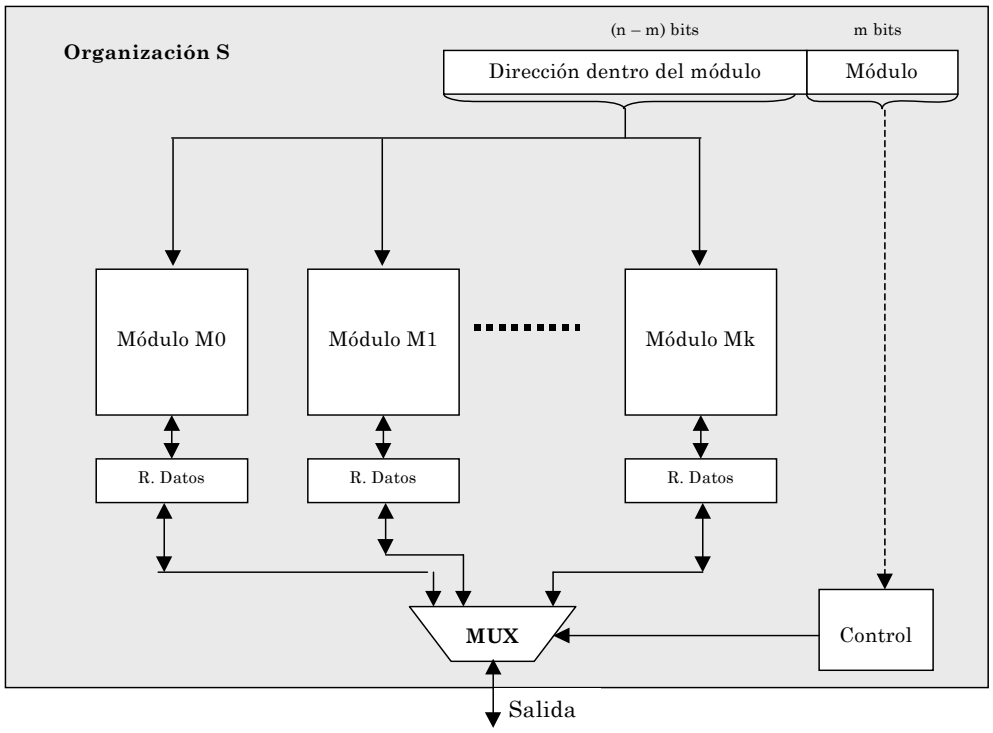


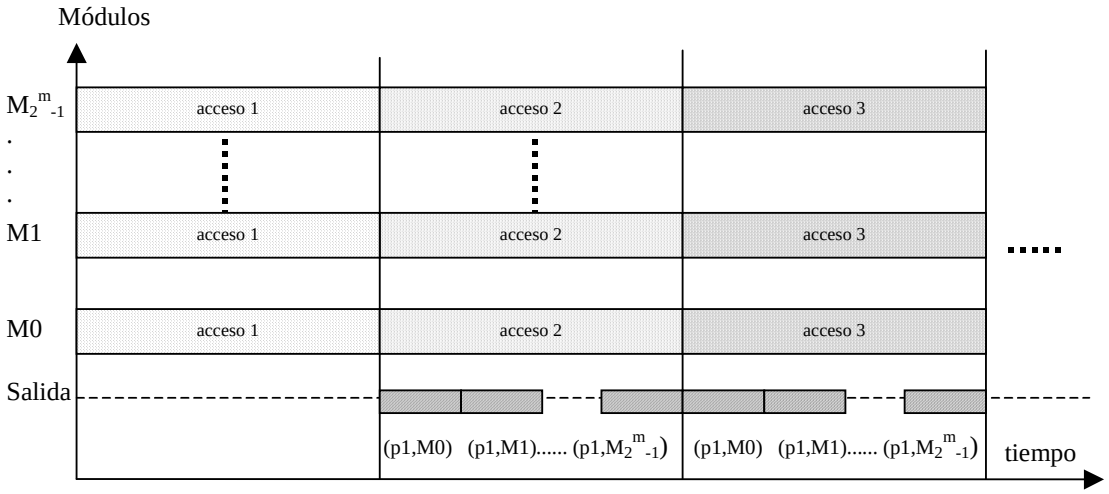
Ejemplo: Para n = 5 y m = 2 el entrelazado de orden inferior produciría una distribución de direcciones de manera que las consecutivas irán en módulos diferentes:

Dir.módulo	módulo	M0	Dir.módulo	módulo	M1	Dir.módulo	módulo	M2	Dir.módulo	módulo	M3
000	00	0	000	01	1	000	10	2	000	11	3
001	00	4	001	01	5	001	10	6	001	11	7
010	00	8	010	01	9	010	10	10	010	11	11
011	00	12	011	01	13	011	10	14	011	11	15
100	00	16	100	01	17	100	10	18	100	11	19
101	00	20	101	01	21	101	10	22	101	11	23
110	00	24	110	01	25	110	10	26	110	11	27
111	00	28	111	01	29	111	10	30	111	11	31

5.3. Memoria entrelazada con organización S

La organización S (Sencilla) utiliza entrelazado de orden inferior, accediendo simultáneamente, con la misma dirección local, a la misma palabra de cada uno de los módulos





5.4. Memoria entrelazada con organización C