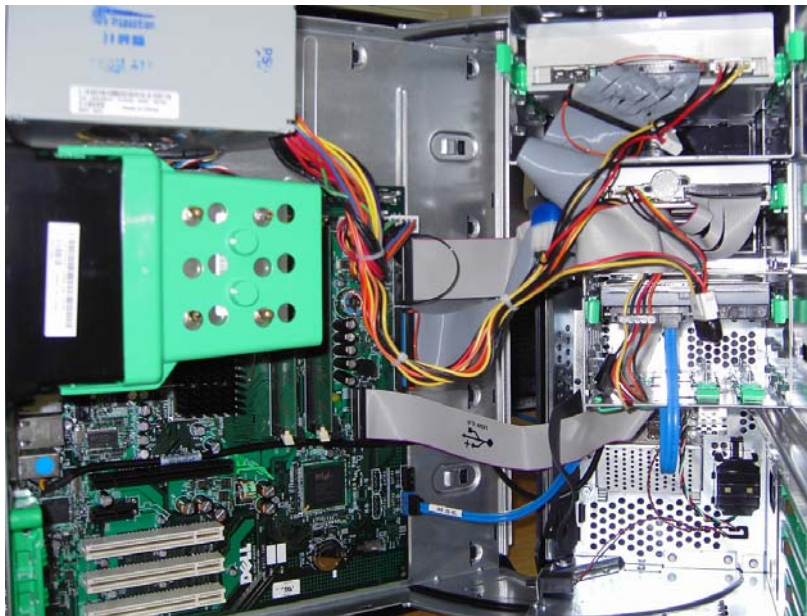


Guía Práctica

de la asignatura

Estructura de Computadores II

Versión 1.3



SEGUNDO CURSO DE INGENIERIA TÉCNICA INDUSTRIAL
ESPECIALIDAD EN
INFORMÁTICA DE GESTIÓN



E.U.I.T.I. de Bilbao
Universidad del País Vasco / Euskal Herriko Unibertsitatea



I. OBJETIVOS DE LA ASIGNATURA

La asignatura *Estructura de Computadores II* se imparte en el primer cuatrimestre del segundo curso de la especialidad Informática Técnica de Gestión. El carácter de esta asignatura es troncal por lo que deberán cursarla todos los alumnos matriculados en Ingeniería Técnica especialidad Informática Técnica de Gestión. El programa se ha previsto para un curso académico de 15 semanas de duración, siendo su carga docente semanal de tres horas teóricas (4.5 créditos) y una hora de prácticas (1.5 créditos), aunque este último aspecto se resolverá agrupando alumnos para disponer de mayor tiempo de uso de ordenador en el Laboratorio. Las horas teóricas se repartirán entre las clases magistrales y las dedicadas a discutir y resolver ejercicios relacionados con las mismas.

Los objetivos a alcanzar por esta asignatura, teniendo en cuenta el perfil que se desea que los alumnos obtengan una vez finalizada la carrera, se pueden resumir en los siguientes puntos:

1. Objetivo General: será dotar al alumno de los conocimientos amplios y generales del funcionamiento de un computador.
2. Objetivos específicos:
 - a. Conocer la estructura de un ordenador y su funcionamiento.
 - b. Conocer los diversos periféricos que conforman un ordenador y su programación.
 - c. Conocer los diversos componentes que conforman un ordenador.
 - d. Familiarizarse con los lenguajes de bajo nivel de programación.
 - e. Familiarizarse con los conceptos de Ensamblado y Enlazado de programas.
 - f. Conocer las arquitecturas de Procesamiento Paralelo.

El programa propuesto y su carga aproximada, para la parte teórica, es el siguiente:

Presentación. Presentación de la Asignatura (2 horas)

Tema 1. Memorias (15 horas)

Tema 2. Dispositivos de Almacenamiento Secundario (4 horas)

Tema 3. Buses y Dispositivos de Entrada/Salida (9 horas)

Tema 4. Lenguajes Máquina y Ensamblador (5 horas)

Tema 5. Arquitectura de Computación en Paralelo (5 horas)

El programa propuesto para la parte práctica (Laboratorio) se explicita en el cuaderno dedicado al Laboratorio: *Ejercicios de Laboratorio de la Asignatura Estructura de Computadores II*.

NOTA: De los autores que a continuación se reseñan, han sido extraídos los gráficos que se detallan
- A.S. Tanenbaum (Página Web del autor): 45, 46..56, 62, 71, 77,79, 80, 85..88, 108.
- T.L. Floyd. (*Fundamentos de Sistemas Digitales*): 43, 44, 57 .. 61.
- J.L. Hennessy, D.A. Patterson (*Organización de Computadoras: Un Enfoque Estructurado*): 99

II. BIBLIOGRAFÍA

La bibliografía disponible para asignaturas que traten temas como *Arquitectura de Computadores*, *Estructura de Computadores*, *Diseño de Computadores*, etc., es muy extensa. Debido a ello, el objeto de este apartado no es ofrecer una recopilación enciclopédica de títulos, sino más bien al contrario, nombrar y comentar exclusivamente los textos que han sido seleccionados para la confección del programa en sus diversos contenidos. Como ya se comentará, unos títulos han sido mas determinantes que otros en la elaboración de esta extensa guía, pero no por ello deben dejar de consultarse los restantes porque siempre nos aportarán una visión o complemento a lo expuesto y enriquecerán nuestro bagaje técnico.

La selección se ha realizado atendiendo a diversos criterios, a saber, la adecuación de los contenidos del texto al programa propuesto, la adecuación del nivel de complejidad presentado por los contenidos, la disponibilidad de los volúmenes (algunos solo en biblioteca), la calidad pedagógica de la exposición de la materia y la posibilidad que ofrecen algunos textos para “profundizar y ampliar conocimientos”.

En varias de estas referencias se ha valorado también la manera de tratar los temas por parte de los autores, el enfoque y lenguaje utilizado motivarán que el alumnado se acerque con menos temor que el que inspiran normalmente los libros especializados.

- A. S. Tanenbaum. “*Organización de Computadoras: Un Enfoque Estructurado*”. Ed. Prentice-Hall.
Libro básico para esta asignatura, trata todos los temas con rigor, profusión y amenidad.
- D. A. Paterson, J. L. Hennessy. “*Estructura y Diseño de Computadores*”. Vol. 2 y 3. Ed. Reverté, S.A. 2004.
Estos dos volúmenes tratan los temas de la asignatura con rigor, son el complemento del anterior. El volumen 3 resulta especialmente relevante en el tema “Lenguaje Máquina y Ensamblador”. Tratamiento de las memorias Caché muy didáctico. Los tres volúmenes no deberían faltar en nuestra biblioteca personal/profesional.
- J.L. Hennessy, D.A. Patterson. “*Arquitectura de Computadores: Un Enfoque Cuantitativo*”. Ed. McGraw-Hill. 2002.
Libro de los mismos autores que en un solo volumen tratan los tres con menor alarde gráfico pero con gran rigor. Muy buen tratamiento de aspectos relacionados con la memoria Caché. Libro también básico.
- M. Morris Mano, C. R. Kime. “*Fundamentos de Diseño Lógico y Computadoras*”. Ed. Prentice-Hall. 3ª Edición. 2005.
Énfasis en el diseño Digital de un Computador y el tratamiento de las instrucciones a nivel máquina. Excelente consulta para la memoria Virtual y Caché.
- I. Englander. “*Arquitectura Computacional*”. Ed. Cecs. 2ª Ed. 2002.
Complementa aspectos y enfoque. Lectura amena. Consulta general
- Brey. “*Los Microprocesadores Intel*”. Ed. Prentice-Hall.
Visión del mundo “Intel”. Imprescindible para el diseñador de computadores en base a CPU’s de Intel
- M. A. de Miguel, T. Higuera. “*Arquitectura de Ordenadores*”. Ed. Ra-Ma.
Arquitectura basada en el entorno “Motorola”. Tratamiento de la memoria Virtual muy didáctica.

INDICE

1. MEMORIAS	1
1.1 Introducción	1
1.2 Características	1
1.2.1 Localización	1
1.2.2 Capacidad	2
1.2.3 Método de Acceso	2
1.2.4 Velocidad	2
1.2.5 Dispositivo Físico	3
1.2.6 Aspectos Físicos	3
1.2.7 Organización	5
1.3 Jerarquía de Memoria	7
1.4 Memoria Principal Semiconductora	7
1.4.1 Tipos de memorias	7
□ Memorias de solo Lectura	7
▪ Escritura Destructiva (ROM-PROM) (solo Lectura)	7
▪ Escritura no Destructiva (EPROM) (Lectura y Escritura)	7
□ Memorias de Lectura/Escritura	8
▪ Lectura principalmente (EEPROM-FLASH)	8
▪ Lectura/Escritura (RAM)	8
1.4.2 Diseño	9
1.4.2.1 Dispositivo (Chip) de memoria	9
1.4.2.2 Mapa de Memoria	10
1.4.2.3 Ejemplos	12
1.5 Memoria Caché	12
1.5.1 Principio de Localidad	12
1.5.2 Concepto de Memoria Caché	13
1.5.2.1 Organigrama de Funcionamiento	14
1.5.3 Parámetros de Diseño	15
1.5.3.1. Tamaño de la memoria Caché	15
1.5.3.2. Tamaño del Bloque	15
1.5.3.3. Número de Cachés	15
1.5.3.4. Contenido de la Caché	16
1.5.3.5. Estrategia de Escritura	16
1.5.3.6. Función de Correspondencia	17
□ Correspondencia Directa	18
□ Correspondencia Asociativa	20
□ Correspondencia Asociativa por Conjuntos	21
1.5.3.7. Algoritmos de Substitución	23
1.5.3.8. Ejemplos de Memoria Caché	23
1.6 Memoria Asociativa	23
1.6.1 Concepto	23
1.6.2 Estructura de una CAM	23
1.7 Memoria Compartida	24
1.7.1 Concepto	24
1.7.2 Memorias de Doble Puerta	25
1.8 Memoria Virtual	26
1.8.1 Concepto	26
1.8.2 Diseño	26
1.8.2.1 Paginación	26
1.8.2.2 Implementación	27

1.8.2.3	Gestión	28
1.9	Otras Memorias	29
1.9.1	Introducción	29
1.9.2	Memorias entrelazadas	29
□	Organización S	29
□	Organización C	30
□	Organización C/S	31
1.9.3	Memorias Dinámicas	31
1.9.3.1	Fast Page Mode RAM (FPM RAM) (EDO RAM)	31
1.9.3.2	Extended Data Output RAM (EDO RAM)	32
1.9.3.3	Burst Extended Data Output RAM (BEDO RAM)	32
1.9.3.4	Synchronous DRAM (SDRAM)	32
1.9.3.5	Double Data Rate SDRAM (DDR RAM)	33
1.9.3.6	Rambus DRAM (RDRAM)	33
1.9.3.7	Synchronous Link DRAM (SLDRAM)	33
1.9.3.8	Video DRAM (VRAM)	33
1.9.3.9	Synchronus Graphics RAM (SGRAM)	33
1.9.3.10	Pipeline Burst Static RAM (PBSRAM)	34
1.9.3.11	Tabla Resumen	34
2.	DISPOSITIVOS DE ALMACENAMIENTO SECUNDARIO	35
2.1.	Introducción	35
2.2.	Discos Magnéticos	35
2.2.1	Discos Duros	35
2.2.2	Discos Flexibles	37
2.2.3	Discos IDE	37
2.2.4	Discos SCSI	38
2.2.5	Discos RAID	38
2.3.	CD-ROM	41
2.3.1.	Gravables	42
2.3.2.	Regravables	43
2.4.	DVD	43
2.5.	Discos Magneto-Ópticos	44
2.6.	Cintas magnéticas	45
3.	BUSES Y DISPOSITIVOS DE E/S	46
3.1.	Introducción	46
3.2.	Buses	47
3.2.1.	Ancho de Bus	47
3.2.2.	Temporización de Bus	47
3.2.2.1	Buses Síncronos	47
3.2.2.2	Buses Asíncronos	48
3.2.3.	Arbitraje de Bus	49
□	Centralizado	49
□	Descentralizado	50
3.2.4.	Operaciones de Bus	50
3.2.5.	Ejemplos de Buses: ISA, PCI, VME, USB	52
3.2.5.1	ISA	52
3.2.5.2	PCI	53
3.2.5.3	VME	55
3.2.5.4	USB	56
3.2.5.5	FIREWIRE	59

3.3. Terminales	60
3.3.1. Monitores de Tubo de Rayos Catódicos (CRT)	60
3.3.2. Pantallas planas LCD (TFT)	60
3.3.3. Terminal de Mapa de Caracteres	62
3.4. Ratones	64
3.5. Impresoras	65
3.5.1 Impresoras Monocromáticas	65
3.5.2 Impresoras de Color	67
3.6. Modems	69
3.7. Código de Caracteres	70
3.7.1. ASCII	70
3.7.2. UNICODE	71
4. LENGUAJE MÁQUINA Y ENSAMBLADOR	73
4.1 Introducción	73
4.2 Lenguaje Ensamblador	73
4.2.1 Uso del lenguaje	74
4.2.2 Formato de las Instrucciones	75
4.2.3 Pseudoinstrucciones	76
4.3 Macroinstrucciones	77
4.4 Ensamblador	79
4.4.1 Primera Pasada	79
4.4.2 Segunda Pasada	80
4.4.3 Estructura de un Módulo Objeto	81
4.5 Enlazado y Carga	81
4.5.1 Enlazado Estático	81
4.5.2 Enlazado Dinámico	85
5. ARQUITECTURA DE COMPUTACIÓN EN PARALELO	87
5.1 Introducción	87
5.2 Modos de Interconexión	88
5.2.1 Multiprocesadores Conectados por un solo Bus	88
❑ Protocolo Snooping	89
5.2.1.1 Sincronización	91
5.2.2 Multiprocesadores Conectados por una Red	91
❑ Protocolo de Directorios	92
5.2.3 Comparativa de Arquitecturas	93
5.3 Clusters	93
5.4 Topologías de Red	94
5.4.1 Introducción	94
5.4.2 Topologías	94
5.4.3 Otras Topologías – Redes Multietapa	96
❑ Crossbar	96
❑ Omega	96
5.4.4 Conmutadores	97
5.5 Ley de Amdahl	97
5.6 Perspectiva Histórica	98

TEMA 1: MEMORIAS

1.1 Introducción

Lo primero que debemos definir es el concepto de “memoria”, “*Dispositivo capaz de almacenar información*”. No se precisa el tipo de dispositivo (soporte), si es mecánico, de estado sólido, magnético u óptico, etc. Tampoco importa el tipo de información, que es, como está codificada, etc...

Desde el punto de vista de un Computador, vamos a distinguir tres apartados

- La memoria va a contener el Programa (conjunto de instrucciones) y los Datos de trabajo
- Se trata de un elemento sencillo pero con una gran diversidad de tipos, tecnologías, prestaciones, etc.
- Se da una jerarquía entre los “diversos” dispositivos que componen la memoria, localizándose unos en el interior del computador y otros en su exterior.

Por otro lado, un dispositivo de memoria, debe presentar una determinada estructura para poder reconocerlo como tal; deberá disponer de los siguientes elementos:

- **Medio o Soporte:** donde se almacenarán los distintos estados que codifiquen la información (óptico, magnético, eléctrico, etc.).
- **Transductor:** convierte una magnitud física en otra, por ejemplo, una presión a corriente (sensor) o un voltaje a movimiento (actuador). (Mecanismos de lectura óptica y su paso a eléctrico, etc...).
- **Mecanismo de Direccionamiento:** que permita procedimientos de Lectura/Escritura de información en el lugar e instante deseado.

1.2 Características

1.2.1 Localización

Dependiendo de donde se ubiquen físicamente las memorias, distinguiremos tres tipos

- **Memoria Interna del Procesador:** Memoria de alta velocidad, presenta un tamaño pequeño en comparación con las siguientes. Memoria de trabajo para almacenamiento temporal de Instrucciones y Datos. Podrán ser Registros o la Caché
- **Memoria Interna del Computador:** Se trata de la Memoria Principal o Primaria, en ella residen los Programas que van a ser ejecutados o bien los Datos obtenidos. Es relativamente grande pero mas pequeña y mas rápida que la siguiente
- **Memoria Externa al Computador:** Memoria Secundaria o Auxiliar, es la mas lenta pero es la que mas información puede almacenar (programas y grandes ficheros); está limitada a la velocidad del Bus y a aspectos electro-mecánicos. Podemos citar a unidades Zip, CD's, etc..

Esta característica de localización la podemos ver en la Figura 1:

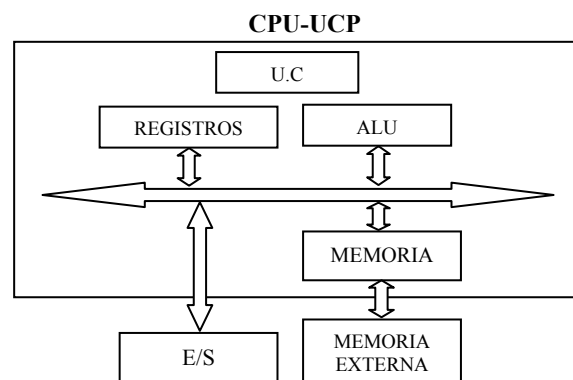


Figura 1

1.2.2 Capacidad

Es la cantidad de información que puede almacenar el dispositivo, se cuantifica en múltiplos de “bit” (0 o 1), la práctica habitual es medirla en múltiplos de “byte” (octetos).

La nomenclatura habitual es la siguiente:

- 1B significa “un byte”.
- 1Kb significa 1K bits (1024 bits, potencia más cercana de 2 a 1000).
- 1KB es una memoria de extensión 1K (posiciones) de 8bits de tamaño cada posición.

Resumiendo, en la Tabla 1 vamos a ver las unidades y múltiplos mas habituales.

1 bit			1 Mb	1024 Kb	2^{20} bits
1 nibble	4 bits		1 Gb	1024 Mb	2^{30} bits
1 byte - octeto	8 bits		1 Tb	1024 Gb	2^{40} bits
1 Kb	1024 bits	2^{10} bits			
1 word	16 bits	32 bits			
1 long word	32 bits	64 bits			

Tabla 1

1.2.3 Método de Acceso

Es el modo de acceder a las celdas de memoria

- **Acceso Secuencial** (SAM: *Sequential Acces Memory*): Para acceder a un Dato, hay que recorrer todos los datos anteriores. Es un método lento (memoria lenta) pero útil en grandes sistemas de almacenamiento, como p.ej. cintas.
- **Acceso Aleatorio** (RAM: *Random Acces Memory*): Se accede a un Dato de manera directa a través de su dirección. El tiempo de acceso es siempre el mismo.
- **Acceso Asociativo** (CAM: *Content Addressable Memory*): Se accede a un Dato por su contenido, se busca en toda la memoria simultáneamente. Una vez encontrado el Dato, se da la dirección donde se ubica.

1.2.4 Velocidad

Para medir el rendimiento de una memoria, se recurre a tres parámetros

- **Tiempo de Acceso** (T_A): Según el tipo de memoria, se darán dos tipos de T_A :
 - Si es un acceso aleatorio RAM: Tiempo que transcurre desde el instante en el que se presenta una dirección a la memoria hasta que el dato es memorizado o disponible para su lectura.
 - Si es un acceso CAM o SAM: Tiempo empleado en actuar el mecanismo de L/E en la posición deseada, es decir, tiempo empleado en localizar la dirección.
- **Tiempo de Ciclo de Memoria** (T_C): Tiempo transcurrido desde la orden de operación de L/E hasta que se puede dar la siguiente orden de L/E. Por debajo de este tiempo la memoria no responde. Interesa que este tiempo sea lo menor posible.

Gráficamente el T_A y el T_C se pueden representar, en el tiempo, según el siguiente gráfico. T_A debe ser menor que T_C .

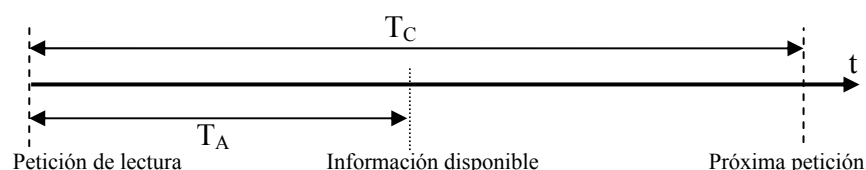


Figura 2 Tiempos de Acceso y Ciclo de Memoria

- **Velocidad de Transferencia (V_T):** Es la velocidad a la que se pueden transferir datos a, o desde, una unidad de memoria. Según el tipo de memoria, existen dos casos:
 - **Caso aleatorio (RAM)** : $V_T = 1/T_C$
 - **Caso no aleatorio (CAM-SAM)** : $T_N = T_A + (N/V_T)$

T_N : Tiempo medio de L/E de N bits. Tiempo de disponibilidad de datos
 T_A : Tiempo de acceso al dato, es variable en acceso secuencial. Fijo si CAM
 N : Número de bits a transferir
 V_T : Velocidad de transferencia (bits/s) una vez encontrado el dato

Los dos primeros parámetros definen la *Latencia* de una memoria.

1.2.5 Dispositivo Físico

Los sistemas de memoria empleados en los computadores utilizan diferentes dispositivos físicos. En un principio se empleaba la memoria de ferrita que era de “lectura destructiva”, ya que tenía que volver a escribir lo que se leía. Esto las hacía lentas; si no hubiera tenido que realizarse la sobre escritura, las habría convertido en memorias muy rápidas.

Actualmente los tipos mas usados son:

- **Memoria principal:** dispositivos semiconductores
- **Memoria secundaria:** debido a la cantidad de información se recurre a:
 - Memorias magnéticas: Cintas, discos, etc.
 - Memorias ópticas: CDROM
 - Memorias magneto-ópticas

1.2.6 Aspectos Físicos

Las principales características físicas a tener en cuenta para trabajar con determinados tipos de memoria son:

- **Alterabilidad:** Esta propiedad hace referencia a la posibilidad de alterar el contenido de una memoria, las hay de solo lectura o de lectura/escritura. Memorias ROM (*Read Only Memory*) y RWM (*Read Writable Memory*).

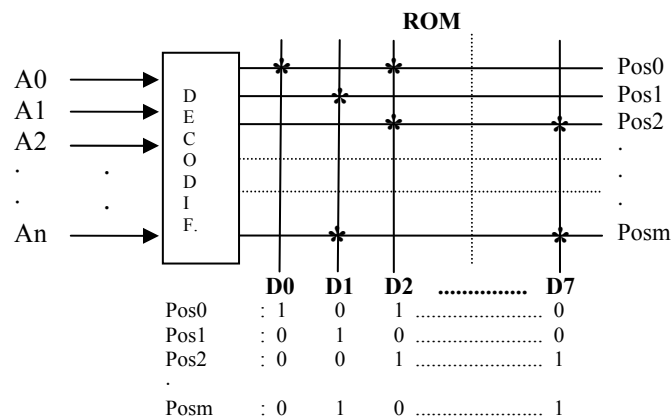


Figura 3 Esquema de una ROM

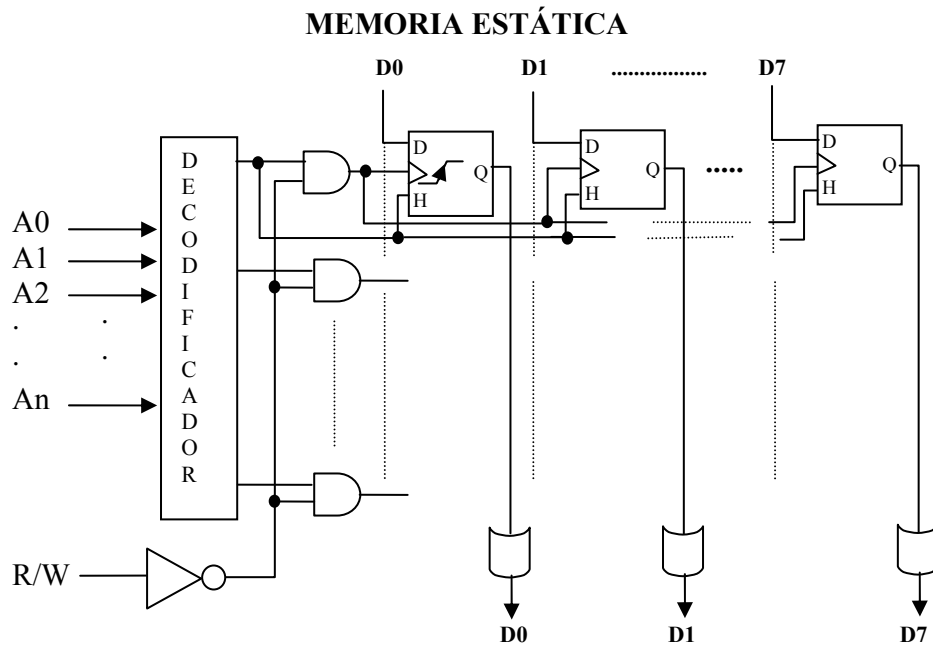


Figura 4 Esquema de una RAM

- **Permanencia de la Información:** Relacionado con la duración de la información almacenada en memoria
 - **Lectura destructiva:** Memorias de lectura destructiva DRO (*Destructive Read Out*) y memorias de lectura no destructiva NDRO (*Non Destructive Read Out*)
 - **Volatilidad:** Esta característica hace referencia a la posible destrucción de la información almacenada en un cierto dispositivo de memoria cuando se produce un corte en el suministro eléctrico. Memorias Volátiles y no Volátiles.
 - **Almacenamiento Estático/Dinámico:** Una memoria es estática SRAM (*Static Random Access Memory*) si la información que contiene no varía con el tiempo. Una memoria es dinámica DRAM (*Dinamic Random Access Memory*) si la información almacenada se va perdiendo conforme transcurre el tiempo; para que no se pierda el contenido se debe “refrescar” la información.

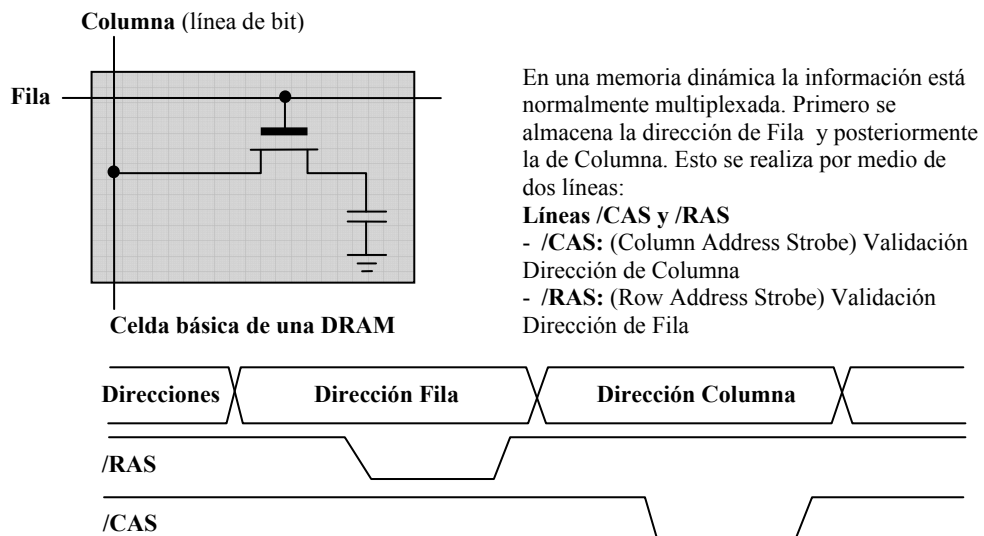
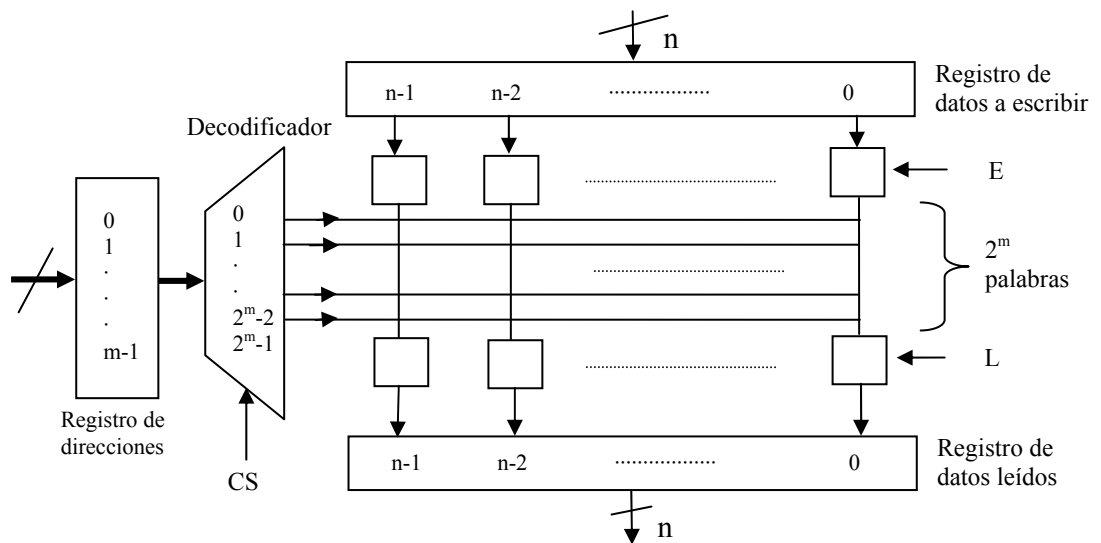


Figura 5 Célula y Direccionamiento en una DRAM

1.2.7 Organización

Hace referencia a la disposición física de los bits para formar palabras. La organización depende del tipo de memoria que se trate. Para una memoria semiconductora distinguimos tres tipos de organización:

- Organización 2D.
 - Organización $2^{1/2}D$.
 - Organización 3D.
- **Organización 2D:** RAM de 2^m palabras de ancho “n” bits cada una, la matriz de celdas está formada por 2^m filas (nº de posiciones de la memoria interna que van de la 0 a la 2^m-1 en el bus de direcciones) y n columnas (nº de bits en el registro de la memoria o ancho de palabra). Se trata de una organización lineal que se emplea en memorias de poca capacidad y gran rapidez de acceso.



Si tuviéramos, p.ej. 1K x 8 de memoria tendríamos 1024 posiciones que son 2^{10} , por lo tanto las direcciones van de la 0 .. 1023, o bien, de la 0000H .. 03FFH. Si se aumenta el ancho de palabra a mas de 8 bits entonces, la superficie de silicio de la memoria, se vería ampliada

Figura 6 Organización 2D

- **Organización $2^{1/2}D$:** Utiliza dos decodificadores con $m/2$ entradas y $2^{m/2}$ salidas. El bus de direcciones de la organización 2D se divide en dos buses encaminados a dos decodificadores que van de 0 .. $[(m/2)-1]$ y de $[m/2]$.. m-1 y donde coincidan las líneas de salidas de los decodificadores estará el bit que busco, p.ej. bus de direcciones de 16 pasa a dos de 8 bits que implican 2^8 (256) posiciones. Esta organización se emplea en memorias de un solo bit como unidad de transferencia pues solo coincide un bit entre los dos decodificadores y precisa menor tamaño para la memoria que la de la organización 2D por lo que requiere menor nº de puertas lógicas.

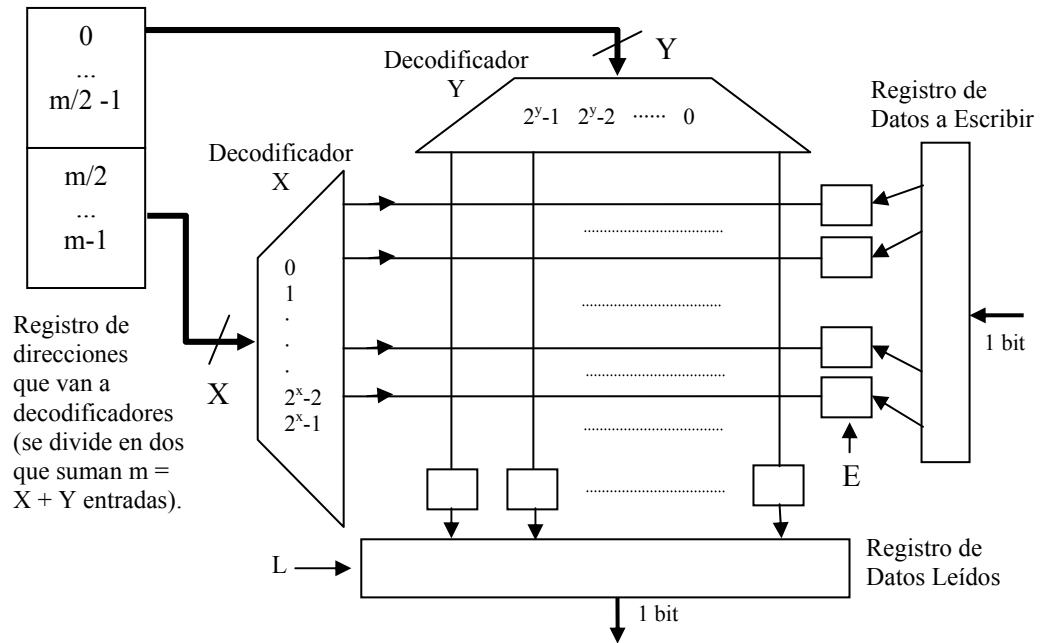


Figura 7 Organización 2 1/2 D

- Organización 3D:** Es similar a la organización $2^{1/2}D$ pero la palabra de n bits se almacena en n planos y dentro de cada plano se selecciona la posición “ x ” y la posición “ y ”, por lo que solo se habilitarán los bits que estén en el mismo plano; p.ej., si tenemos 16 bits tendremos 16 planos. Tiene un diseño mas complicado y es una organización para memorias de acceso lento. Son chips de mayor densidad de silicio y, por lo tanto, mas anchos para así poder ampliar el ancho de palabra mas fácilmente.

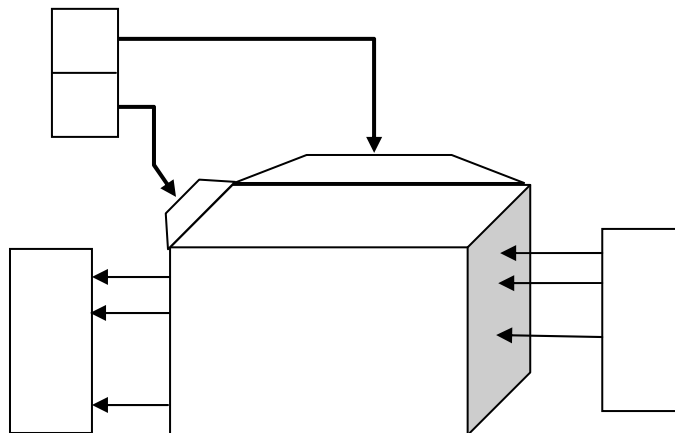


Figura 8 Organización 3D

1.3 Jerarquías de Memoria

La manera usual de almacenamiento de información, comenzando por poca información y llegando a una cantidad masiva de información, es una jerarquía planteada en la siguiente imagen:

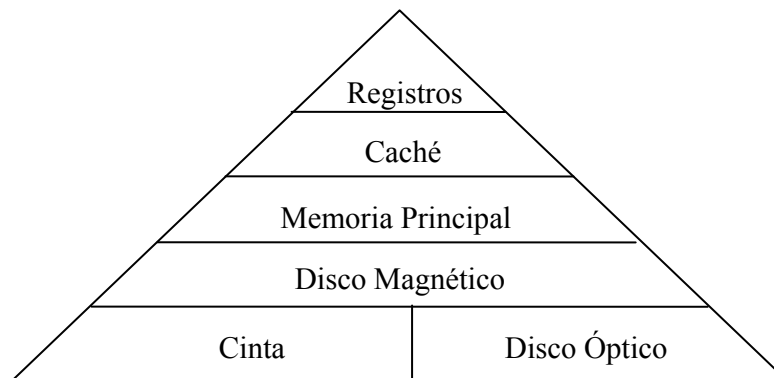


Figura 9 Jerarquía de Memoria de cinco Niveles

En esta jerarquía se establecen tres parámetros fundamentales:

- **Velocidad:** Tiempo de acceso a la memoria. Si bajamos en la jerarquía, este tiempo aumenta. Los registros de la CPU se acceden en unidades de ns, la caché en decenas de ns., etc..
- **Capacidad:** Almacenamiento de información. Si bajamos en la jerarquía, este parámetro aumenta; pasamos de p.ej 128 bytes en registros de CPU a Megabytes en caches, a decenas o miles de Megabytes en la principal, hasta Gigabytes y Terabytes en el resto de memorias. Los dispositivos que forman el 5º nivel son externos al computador
- **Coste:** El coste por bit desciende al bajar en la jerarquía.

1.4 Memoria Principal Semiconductora

Veamos a continuación los diferentes tipos de memorias y sus correspondientes diseños.

1.4.1 Tipos de memorias

Descripción según sus características de Lectura y/o Escritura

□ Memorias de solo Lectura

▪ Escritura Destructiva (solo Lectura)

- **ROM (Read Only Memory):** Memorias grabadas durante el proceso de fabricación del circuito (silicio). Esta memoria puede formar parte de microcontrolador o ser independiente. En ella residen programas de uso muy frecuente (partes de sistema) o bien programas muy contrastados que no van a sufrir modificaciones en el tiempo. Barata en grandes series.
- **PROM (Programmable Read Only memory):** Memoria grabada normalmente por el cliente (desarrollador), puede ser grabada por el fabricante. Presenta las mismas consideraciones que la anterior. Una única grabación. Mas cara que la ROM.

▪ Escritura no Destructiva (Lectura y Escritura)

- **EPROM (Erasable Programmable Read Only Memory):** Memoria grabada eléctricamente y borrable por luz ultravioleta. Presenta un nº no muy elevado de borrados. Tiempo de borrado en torno a los 15'-20'. En ella residen los programas. Muy extendido su uso.

□ Memorias de Lectura/Escritura

▪ Lectura principalmente

- **EEPROM (Electrically Erasable Programmable Read Only Memory):** Se trata de una Memoria grabable eléctricamente y borrable eléctricamente. Presenta una gran ventaja y es que puede ser grabada por el propio procesador de placa, así como almacenar información del tipo consignas variables en el proceso. Es mas lenta en Escritura que en Lectura. La grabación puede ser a nivel de byte o bloques de bytes. Mas cara que la EPROM.

- **FLASH** (Nombre debido a su velocidad de grabación eléctrica): Características similares a la EEPROM, presentan mayor capacidad de almacenamiento que las EEPROM; debido a esta característica substituyen, en ocasiones, a los discos duros. La grabación es a nivel de bloques.
- **Lectura/Escritura (RAM)**
 - **DRAM** (*Dinamic Random Access Memory*): Memorias que necesitan “refrescar” su información debido a que esta se almacena en condensadores (de estado sólido). (*Row Address Strobe-Column A S*). Es un proceso repetitivo que solamente se altera con un ciclo de Lectura/Escritura. Pierden la información al desaparecer la energía. De entre los diversos tipos destacamos dos:
 - **SDRAM**: *Synchronus DRAM*.
 - Alta velocidad.
 - “Pipeline” interno completo.
 - Interfase síncrono (sincronizado con el *clock* del μ P).
 - **DDR**: *Double Data Rate DRAM*
 - Memoria estándar en comunicaciones.
 - Salida de datos con flanco de subida y de bajada del reloj. Transfiere dos datos por pulso.
 - Dirección y señal de control con flanco de subida.
 - **SRAM** (*Static Random Access Memory*): Memorias que almacenan la información en biestables. No necesitan refresco. Pierden la información al desaparecer la energía. Son mas rápidas que las DRAM. Son mas caras que las DRAM. Destacamos dos tipos:
 - **QDR**: *Quad Data Rate SRAM*.
 - Read y Write separados trabajando como DDR.
 - “4” datos de salida por pulso.
 - Aplicaciones de alto ancho de banda (memoria principal para Look Up Tables, Listas ligadas, controlador Buffer de Memoria).
 - **ZBT**: *Zero Bus Turnaround SRAM*
 - Elimina tiempos muertos de Bus durante los ciclos de Read/Write y Write/Read (100% de utilización).
 - Otros nombres: *No Bus Latenci* (NBL), *No Turnarround RAM* (NTRAM).

En la Tabla 2 se refleja un resumen de los diferentes tipos de memorias.

Tipo	Clase	Borrado	Escritura	Volatilidad
RAM	Lectura/Escritura	Eléctrico por Bytes	Eléctricamente	Volátil
ROM	Solo Lectura	No	Máscara	No Volátil
PROM			Eléctricamente	
EPROM	Luz Ultravioleta			
FLASH	Lectura/Escritura Fundamentalmente Lectura	Eléctricamente por Bloques		
EEPROM		Eléctricamente por Bytes y Bloques		

Tabla 2 Tipos de memoria

1.4.2 Diseño

En el diseño de la memoria, hay que tener en cuenta dos aspectos muy importantes:

- Dispositivo de memoria
- Mapa de memoria

1.4.2.1 Dispositivo (Chip) de memoria

Está organizado internamente como una matriz de celdas de memoria de $n \times m$, donde n es el nº de palabras que puede almacenar el dispositivo y m es el nº de bits por palabra.

Ejemplo: Memoria de $4K \times 8 \rightarrow 4K$ palabras con ancho de 8 bits por palabra. La memoria tiene un total de:

- $4K = 4 \times 1024 = 4096$ palabras. $n = 4096$ direcciones o posiciones de memoria.
- 8 bits = m bits de ancho de palabra.

Gráficamente el dispositivo de memoria podría representarse de la siguiente manera.

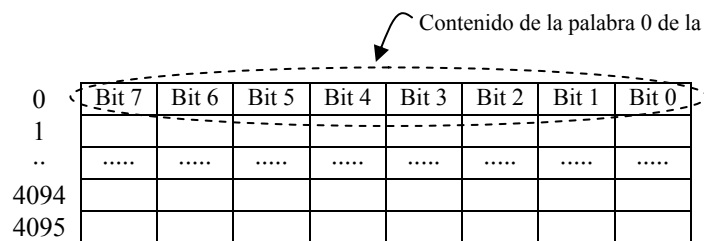


Figura 10 Contenido y Dirección en una Memoria

Un dispositivo físico de memoria presenta las siguientes señales en sus patillas:

- n patillas para el Bus de Direcciones, direccionándose 2^n palabras.
- m patillas para el Bus de datos, indicando esto el ancho de palabra de m bits.
- R/\overline{W} (Read/Write) indica si la operación es de Lectura o Escritura. Existen otras patillas de escritura WE (*Write Enable*) y lectura OE (*Output Enable*).
- $/CS$ (*Chip Select*) o $/CE$ (*Chip Enable*). Selecciona el chip de memoria al que se accede.
- V_{cc} . Positivo de la alimentación del chip
- V_{ss} . Negativo (0V. Lógico) de la alimentación.

La pastilla presentará una configuración como la siguiente:

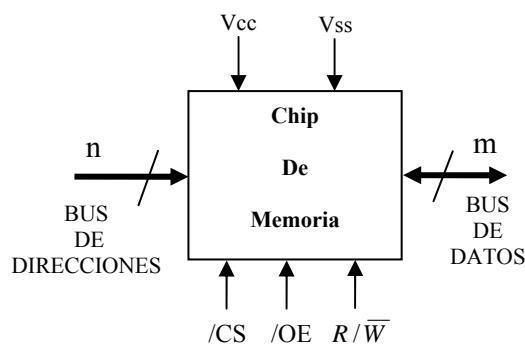


Figura 11 Esquema de señales de una memoria

Para el correcto funcionamiento de la memoria, que se rige por una tabla de verdad, es necesario incorporar una circuitería auxiliar que realice la decodificación, multiplexación, bufferización, etc. En la Tabla 3 se muestra el comportamiento de la memoria y en la Figura 12:

Entradas			Operación (Salida del Dispositivo)
/CE	(R/W) /WE	/OE	
H	X	X	Z
L	H	L	Lectura
L	L	X	Escritura
L	H	H	Power Down / Z

Tabla 3 Señales de Control de una memoria

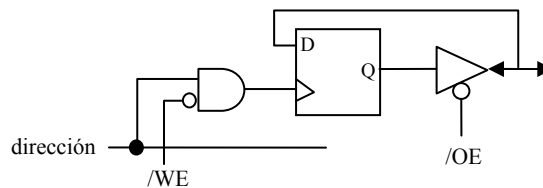


Figura 12 Célula básica de Memoria

Veamos un ejemplo de estructuración de una memoria:

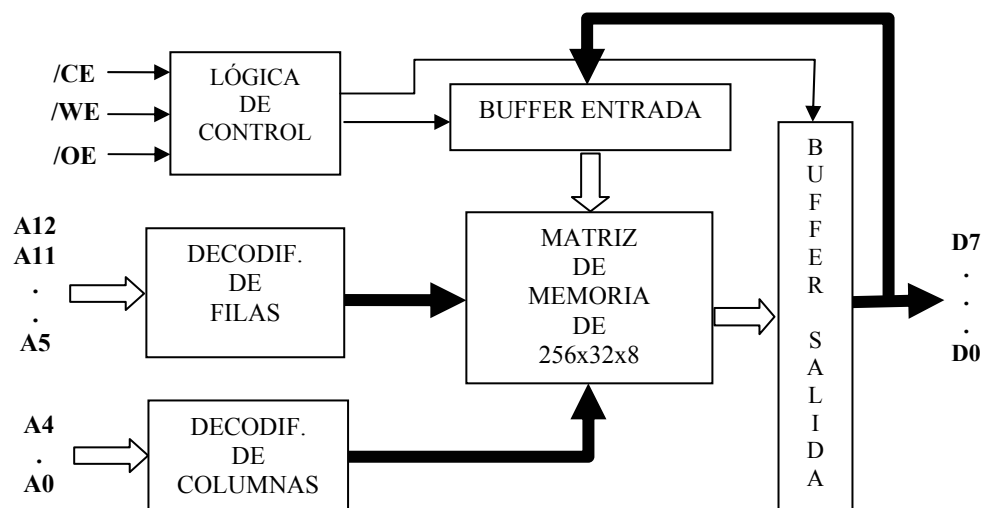


Figura 13 Estructuración en Bloques de memoria

1.4.2.2 Mapa de Memoria

Es el espacio que puede direccionar un procesador según la memoria física instalada. Este mapa indica al procesador donde comienza y donde acaba la memoria, así como el tipo de memoria empleado. Normalmente no se ocupa toda la extensión de memoria, sino que se ubican los dispositivos en fragmentos de la extensión total de memoria.

Para realizar el mapa de memoria, necesito conocer aspectos como los siguientes:

- Conocer el n° de líneas de Direcciones del Procesador, si son m, tendré un espacio total de 2^m localidades $[0 .. 2^m-1]$.
- Conocer el n° de líneas del Bus de Datos, si son n podré escribir hasta 2^n datos diferentes de n bits cada dato.
- Conocer el n° de líneas necesario para realizar las funciones de L/E. Normalmente solo es necesaria una línea, pues la señal de control es R/\overline{W} , por lo que con “1” realizará la Lectura y con “0” la Escritura.
Esto dependerá de cada Procesador, deberá estudiarse cada caso.

Veamos unos casos concretos para clarificar estos conceptos:

- Supongamos un computador que presenta una memoria de 64K x 8, esto nos dice que tiene 64K palabras de un ancho de 8 bits. El nº de líneas de dirección se calcula de la siguiente forma.
 - $64K = 64 \times 2^{10} = 2^6 \times 2^{10} = 2^{16} = 65.536$ posiciones de memoria accedidas por “16” líneas (bits) de dirección.
 - Estas $2^n = 2^{16}$ posiciones de memoria, codificadas en hexadecimal, se expresarán de la siguiente forma: 0000H .. FFFFH.
 - Los Datos son $2^m = 2^8 = 256$ datos diferentes accedidos por 8 bits. Expresado en hexadecimal, será 00H .. FFH.

La implementación física del mapa de memoria, se realiza con uno o varios chips de memoria. Los fabricantes de dispositivos de memoria disponen de una gran variedad de formatos. Se nos presentan dispositivos como los siguientes: nK x 1, nK x 4, nK x 8, nK x 16, nK x 32, nM x 1, nM x 4, nM x 8, nM x 16, nM x 32, etc.; Donde “n” es un múltiplo de 2.

P.ej. si disponemos de una memoria de 4K x 8, accederemos a 4096 posiciones de memoria de 8 bits de ancho de información.

- Si quisiéramos diseñar una memoria de n bits y dispusiéramos de dispositivos de 1 bit de tamaño 64K, dispondremos en paralelo n dispositivos para su realización.

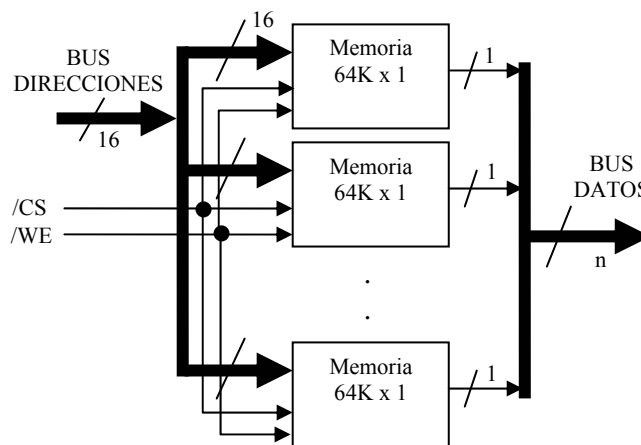


Figura 14 Ejemplo de Memoria de 64Kxn bits

- Si quisiéramos diseñar una memoria de 8 bits y dispusiéramos de dispositivos de 4 bit de tamaño 64K, dispondremos en paralelo 2 dispositivos para su realización.

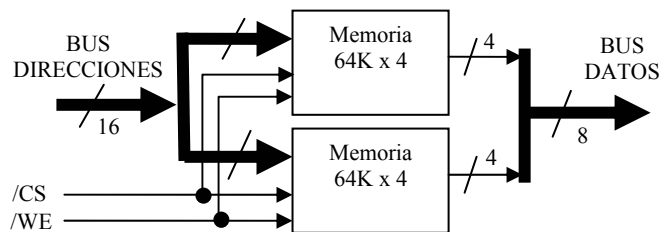


Figura 15 Ejemplo de Memoria de 64Kx8 bits

- Si quisiéramos diseñar una memoria de 64Kx8 bits y dispusiéramos de dispositivos de 8 bits de tamaño 32K, dispondremos 2 dispositivos para su realización. En este caso podremos utilizar A15 como señal de /CS.

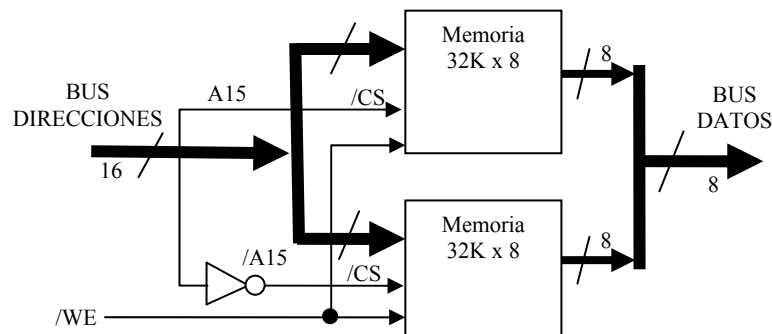


Figura 16 Ejemplo de Memoria de 64K(32K+32K)x8 bits

1.4.2.3 Ejemplos

- Queremos diseñar una memoria principal de 128 Kbytes
 - ¿Cuántos dispositivos de memoria de 32Kx8 necesitamos?
 - ¿Cuántos dispositivos de memoria de 64Kx4 necesitamos?
- Dado un computador con datos de 16 bits y que permite direccionar 1 Megapalabra, teniendo instaladas 128 Kpalabras con chips de 64Kx1.
 - Obtener el nº de bits del Bus de Direcciones
 - Averiguar el nº de bits necesarios para direccionar el chip de memoria utilizado
 - Calcular el nº de chips necesarios
 - Obtener el nº de bits del Bus de direcciones que permitan seleccionar los chips utilizados

1.5 Memoria Caché

1.5.1 Principio de Localidad

Antes de comenzar con la explicación de lo que es una memoria caché, es necesario introducir el concepto de “Localidad” para comprender porqué existe la memoria caché.

El principio de Localidad establece que *“las direcciones que genera un programa durante su ejecución, no son uniformes, sino que tienden a estar concentradas en pequeñas regiones del espacio de direcciones”*.

Los programas tienden a reutilizar los datos e instrucciones que han utilizado recientemente.

En la mayoría de los programas, un pequeño porcentaje del código, es el responsable de un gran tiempo de ejecución. Es corriente que el 10% del programa ocupe el 90% del tiempo de ejecución. Este comportamiento es debido al flujo secuencial y a las estructuras de control de flujo (bucles), así como a la agrupación en bloques, tanto de las instrucciones como de los datos dentro de los programas.

En la localidad de las referencias que genera un proceso, coexisten tres componentes, que pueden ser excluyentes:

- **Tiempo**
- **Espacio**
- **Orden**

- ❑ **Localidad Temporal:** Tendencia del proceso a hacer referencia a elementos a los que se ha accedido recientemente. Un elemento podrá ser referenciado pronto.
- ❑ **Localidad Espacial:** Tendencia del proceso a efectuar referencias en la vecindad de la última referencia que ha realizado. Referenciando a un elemento es probable que se referencie a elementos próximos a él.
- ❑ **Localidad Secuencial:** Tendencia del proceso a hacer referencia al siguiente elemento en secuencia al último referenciado.

1.5.2 Concepto de Memoria Caché

Recurriendo al Principio de Localidad podemos pensar que si disponemos de una memoria muy rápida, donde podamos almacenar bloques secuenciales de código y datos, conseguiremos aumentar el rendimiento del computador.

Esta memoria va a ser la caché y va a estar situada entre el procesador y la memoria principal. Se dispone en el mismo dispositivo del procesador por lo que va a tener la misma tecnología o lo que es lo mismo “velocidad”.

Debido al alto coste que supone el disponer en el silicio este tipo de almacenamiento, su tamaño no es grande pero se ha llegado a compromisos satisfactorios.

El funcionamiento de la memoria caché se basa en la transferencia de bloques de la memoria principal y la memoria caché, así como la transferencia de palabras entre la memoria caché y la CPU.

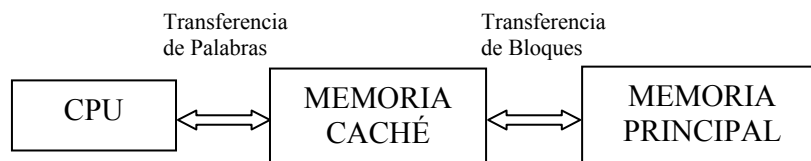


Figura 17 Esquema de transferencia de Datos entre CPU-MC-MP

La memoria Principal de 2^n palabras está organizada en M bloques de longitud fija (K palabras/bloque), capacidad de una línea de caché, donde $M = 2^n / K$ representa el nº de bloques totales. En la siguiente figura se muestra la configuración de la memoria:

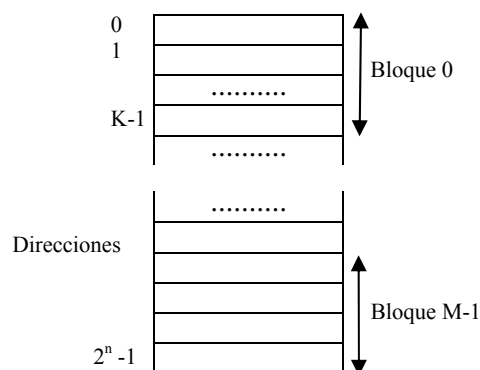


Figura 18 Estructuración en Bloques de la MP

Una Línea de caché contiene un Bloque de memoria, o lo que es lo mismo, K palabras consecutivas.

La memoria caché está dividida en C líneas o particiones de K palabras siendo $C \ll M$. En esta figura se muestra la configuración de la memoria:

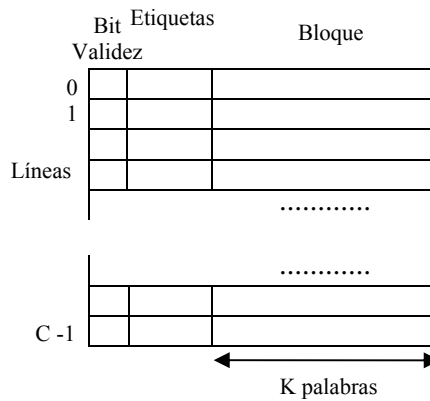


Figura 19 Líneas en la MC

En cada línea de caché cabe un bloque de la memoria principal. La “Etiqueta” o “Tag” representa la dirección del bloque en la MP.

Fue Wilkes en 1965 el que introdujo el concepto de memoria cache y poco tiempo después, en 1968 el modelo 360/85 la incluía en su arquitectura. El primer microprocesador con la memoria cache en el interior del chip fue el Motorola 68020. Posteriormente aparecieron computadores con dos niveles de cache uno interno y otro externo como por ejemplo el SUN 3/250 en 1986 y caches separadas para datos e instrucciones (4D/240 de Silicon en 1988).

- Tasa de acierto:** Cuando la CPU necesita una palabra de memoria y la encuentra en la memoria caché, se dice que se produce un acierto (“hit”); si la palabra no está en la memoria caché se contabiliza un fallo (“miss”).

La relación entre el nº de aciertos y el nº total de referencias a memoria (aciertos + fallos) es la tasa de acierto. En sistemas bien diseñados se suelen conseguir tasas de aciertos de 0,9. La tasa de acierto se calcula mediante la siguiente expresión:

$$tasa_de_aciertos = \frac{n^\circ \text{ aciertos}}{n^\circ \text{ aciertos} + n^\circ \text{ defallos}}$$

1.5.2.1 Organigrama de Funcionamiento

Veamos a continuación como se expresa, mediante un organigrama, el mecanismo de búsqueda y substitución en una memoria caché.

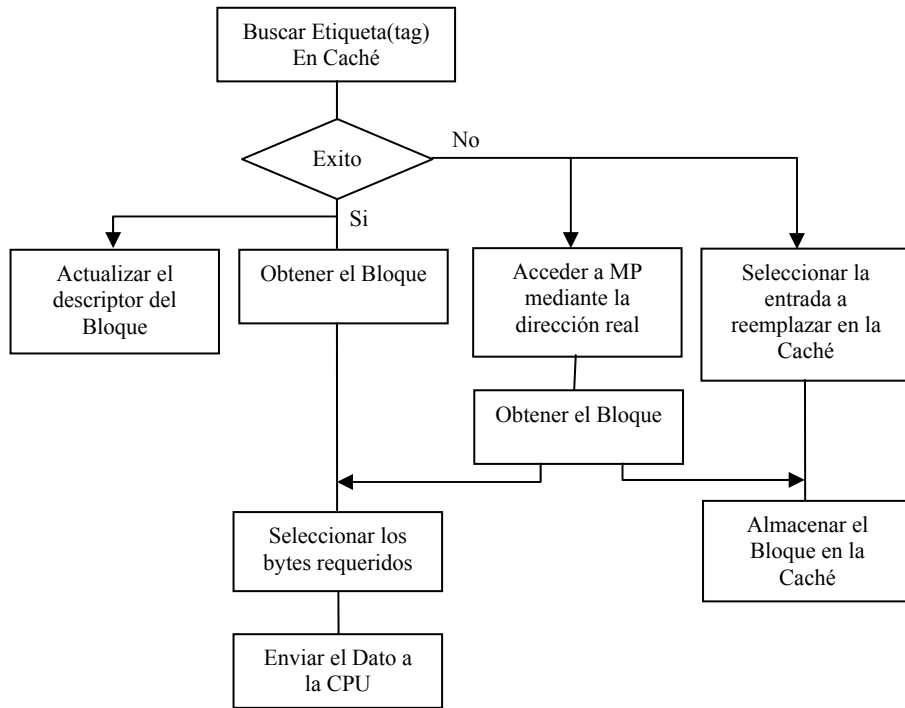


Figura 20 Organigrama de acceso a caché

1.5.3 Parámetros de Diseño

La memoria caché surge como consecuencia del aumento de la velocidad del sistema. A la hora de diseñar la memoria caché debemos de tener en cuenta los siguientes parámetros:

1.5.3.1 Tamaño de la Memoria Caché

Este parámetro plantea un cierto compromiso:

- Debería ser lo suficientemente pequeña como para que el coste medio por bit de información almacenada en la memoria interna del computador estuviese próximo al de la memoria principal.
- Tendría que ser lo suficientemente grande como para que el tiempo de acceso medio total (MP y MC) fuese lo mas próximo posible al de la memoria caché.

De acuerdo con estos estudios empíricos, se sugiere que el tamaño de una caché esté situado entre 1KB y 1MB.

1.5.3.2 Tamaño del Bloque

Partimos de un tamaño de caché fijo. Cuando se va aumentando el tamaño del bloque, a partir de valores muy pequeños, la tasa de aciertos inicialmente aumentará; pero a partir de un cierto tamaño del bloque la tasa de acierto comenzará a disminuir.

- Cuanto mayor sea el tamaño de los bloques, menos bloques se instalarán en la memoria y mas veces se ejecutará el algoritmo de substitución de bloques (mas fallos).
Se favorece la localidad “espacial” pero va en detrimento de la “temporal” al disminuir el nº de bloques.
- Cuando crece el tamaño de un bloque, cada nueva palabra añadida a ese bloque estará a mayor distancia de la palabra requerida por la CPU, y por tanto es menos probable que se necesite a corto plazo.

La línea de caché que es la que contiene el bloque, consta generalmente de entre 4 y 64 bytes (excepcionalmente 128) consecutivos (tamaño del bloque).

1.5.3.3 Número de Cachés

Este parámetro plantea dos tipos de caché distintos .

- Caché interna. Nivel 1 (primario): Físicamente está ubicada en el mismo chip que el procesador (CPU) y aumenta la velocidad de procesamiento. Los accesos a esta caché se efectúan mas rápido. La capacidad de esta caché es bastante pequeña. Oscila entre 16KB y 64KB.
- Caché externa. Nivel 2 (secundario): Físicamente está ubicada fuera del chip del procesador (CPU) por lo que será mas lenta que la caché de Nivel 1 pero seguirá siendo mas rápida que la memoria principal. Al estar fuera (situada entre la memoria principal y el procesador) la capacidad podrá ser mayor que la caché Nivel 1. Su uso se está extendiendo de forma que cada dispositivo se fabrica con su propia caché pero son dos dispositivos diferenciados. Oscila entre 512KB y 1MB.

Generalmente las cachés son inclusivas, es decir:

$$CN1 \subset CN2 \subset CN3 \subset MP$$

1.5.3.4 Contenido de la Caché

Lo ideal es que la memoria caché contenga instrucciones y datos, empleándose actualmente en diseños segmentados.

- Caché unificada: Datos e Instrucciones
- Caché dividida: Caché de Datos y caché de Instrucciones.

Una caché que contiene tanto datos como Instrucciones presenta las siguientes ventajas:

- Presenta una tasa de aciertos mayor ya que automáticamente equilibra la carga (instrucciones y datos), es decir, si un patrón de ejecución implica muchas más captaciones de Instrucciones que de Datos, la caché tenderá a llenarse con instrucciones y viceversa.
- Solo se necesita implementar y diseñar una caché, por lo que el coste será más reducido.

Hoy en día, la tendencia es hacia dos cachés. Se trabaja en ejecución paralela de instrucciones y los diseños “pipelining” en los que el uso de dos cachés da mejores prestaciones. (Ref. Harvard). Duplica el ancho de banda del sistema de memoria.

Ventaja: elimina la competición entre el procesador de instrucciones y la unidad de ejecución.

1.5.3.5 Estrategia de Escritura de Datos – Política de Actualización

Antes de que pueda ser reemplazado un bloque que está en la caché, es necesario saber si se ha modificado o no, es decir si el bloque es un bloque limpio o modificado (sucio). Los datos se modifican en la caché al ejecutar el programa. Si el bloque no fue modificado no hay problema pero si no, existe un problema en la escritura; esto nos lleva a dos tipos de escritura:

- **Escritura Inmediata o Directa** (*Write Through*): Todas las operaciones de escritura se realizan tanto en la caché como en la memoria principal, lo que asegura que los contenidos sean siempre válidos. El principal inconveniente es que genera mucho tráfico con la memoria principal, pudiendo causar un colapso (cuello de botella).

Este tipo de escritura también se conoce como “escritura a través”.

- **Post-Escritura** (*Write Back*): Las escrituras se realizan solo en la memoria caché. Asociada a cada línea de la caché existe un bit de “modificación”. Cuando se escribe en la caché, el bit de modificación, se pone a 1. En el caso de reemplazar una línea, se mira el bit de modificación y si está a 1, se escribirá la línea en la memoria principal, mientras que si está a 0 no. Se escribe el bloque en la MP cuando este es expulsado de la MC.

Este tipo de escritura se conoce también como “escritura diferida o reescritura”.

Inconveniente: se obliga a que los módulos de E/S accedan a la memoria principal a través de la memoria caché. Esto complica la circuitería y genera un cuello de botella. Incrementa la tasa de fallos de caché.

Ventaja: se utiliza menos ancho de banda de memoria principal haciendo idóneo su uso en multiprocesadores. Consistencia de la información.

Por último, este parámetro conlleva un problema de coherencia de datos si no se realiza la reescritura a la memoria principal.

NOTA: La idea genérica es mantener el máximo tiempo posible las líneas más utilizadas.

Las arquitecturas mas comunes de conexión entre la MP, la MC y las E/S son las siguientes:

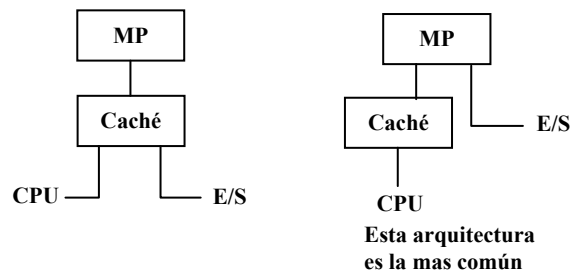


Figura 21 Arquitecturas de interconexión entre MP-MC-E/S

1.5.3.6 Función de Correspondencia – Organización de la Caché

Debido a que existen menos líneas que bloques, se necesita un algoritmo que haga corresponder bloques de memoria principal a líneas de memoria caché. Existen tres tipos de correspondencia por lo que existirán tres tipos de función de correspondencia distinta:

- **Correspondencia Directa:** Ejemplo para 8 líneas. El bloque 12 de memoria principal solo podrá almacenarse en la línea 4 (= 12 módulo 8). Es la Dirección del Bloque MODULO N° de bloques de la Caché.
- **Correspondencia Totalmente Asociativa:** Se puede almacenar en cualquier línea.
- **Correspondencia Asociativa por Conjuntos:** Si se escoge asociatividad 2 (2 bloques por conjunto). Se puede almacenar en cualquier línea del conjunto 0 (= 12 módulo (8/2)).

Este ejemplo de cada tipo de correspondencia se ve mejor en la Tabla 4:

Número de línea	Directa	Asociativa	Asociativa por conjuntos
0			Conjunto 0
1			
2			Conjunto 1
3			
4			Conjunto 2
5			
6			Conjunto 3
7			

Tabla 4 Correspondencias en memoria caché

A continuación vamos a describir las tres técnicas enumeradas. Veremos la estructura general en cada caso y un caso concreto.

En los tres casos para los ejemplos, trabajaremos con un sistema con las siguientes características:

- El tamaño de la memoria caché es de 4 KBy = 4096 Bytes..
- Los datos se transfieren entre memoria principal y la memoria caché en bloques de 16Bytes (las “K” palabras del gráfico). Esto nos indica que la caché está organizada en 256 líneas (4096/16) (la “C” del gráfico).
- La memoria principal consta de 64 KB, por lo que el bus de direcciones es de 16 bits (64 K = 2¹⁶). Esto nos indica que la memoria principal está constituida por 4096 bloques (la “M” del gráfico).

- ❑ **Correspondencia Directa:** Consiste en hacer corresponder cada bloque de memoria principal a solo una línea de memoria caché. La función de correspondencia se expresa mediante la siguiente expresión:

$$i = j \text{ módulo } m$$

donde:

- $i =$ n° de línea de caché.
- $j =$ n° de bloque de la memoria principal (dirección del bloque).
- $m =$ n° de líneas de la memoria caché (n° de bloques de la caché).

Cada dirección de la memoria principal puede verse dividida en tres campos:

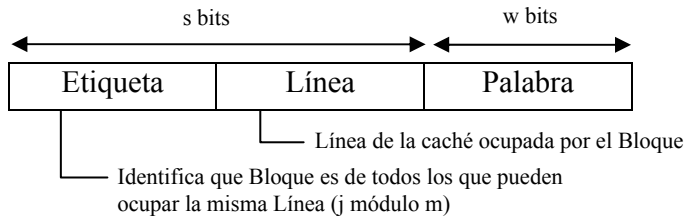


Figura 22 Partición de la Dirección en campos en Correspondencia Directa

Donde:

- w bits = identifica cada palabra dentro de un bloque.
- s bits = identifica el n° de bloque.

El uso de una parte de la dirección como número de línea proporciona una asignación única de cada bloque de memoria principal en la caché tal y como se muestra en la siguiente figura:

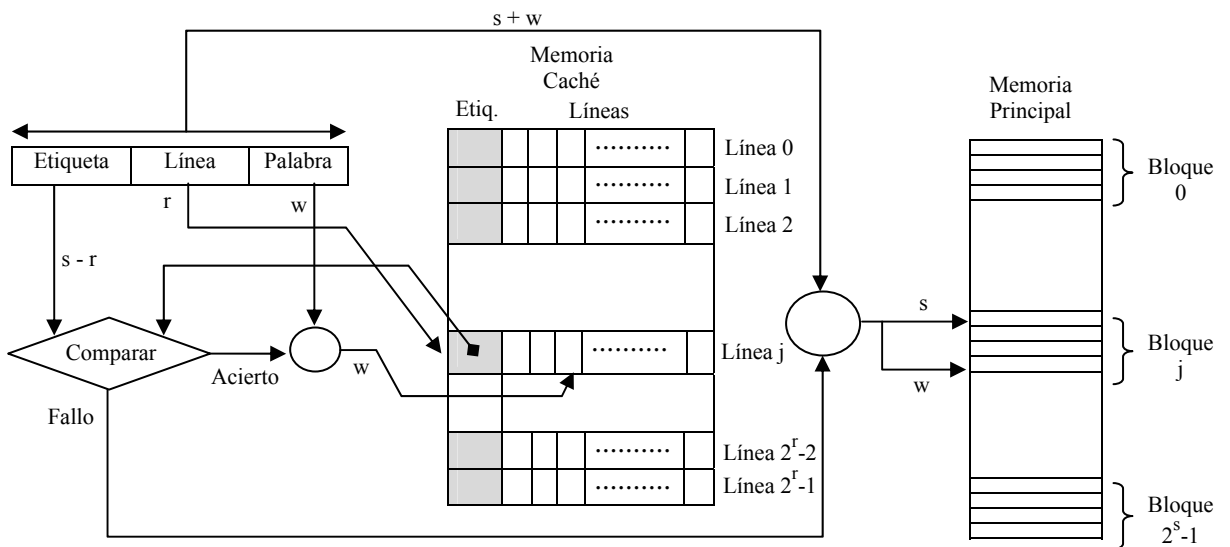


Figura 23 Caché de Correspondencia Directa

En el gráfico anterior vemos que “r” selecciona la “Línea” de la Caché y de ella extrae la “Etiqueta” (Bloque que ocupa la línea de entre todos los posibles) y la compara con el campo “Etiqueta” de la dirección de memoria.

Ejemplo:

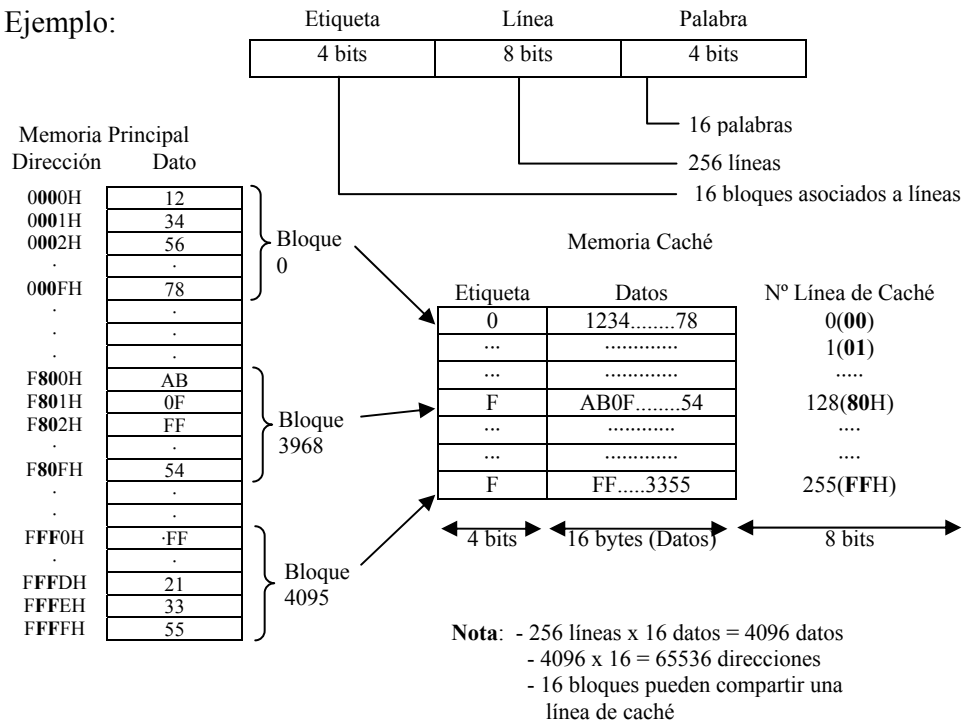


Figura 24 Ejemplo Caché de Correspondencia Directa

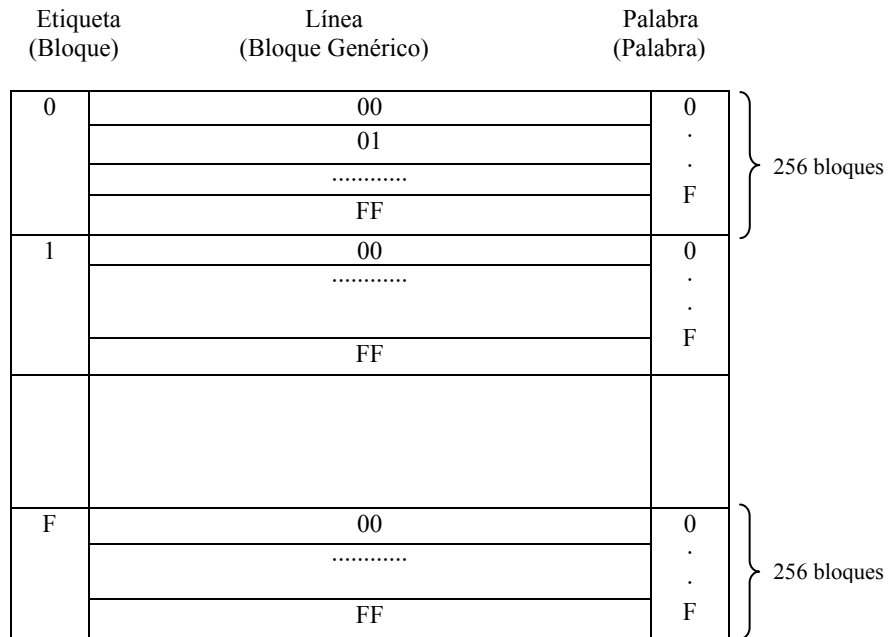


Figura 25 Ejemplo de estructuración de Bloques de Memoria

Ventaja: La técnica de correspondencia directa es simple y poco costosa de implementar (requiere poco hardware).

Inconveniente: Hay una posición concreta de caché para cada bloque dado. Puede existir un trasiego alternativo de bloques que ocupan el mismo lugar.

❑ **Correspondencia Asociativa**

Permite que se cargue un bloque de memoria principal en cualquier línea de la memoria caché. La lógica de control de la memoria caché interpreta una dirección de memoria como una etiqueta y un campo de palabra.

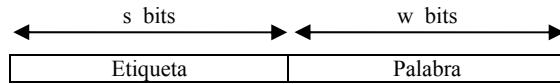


Figura 26 Partición de la Dirección en campos en Correspondencia Asociativa

Donde:

- Palabra: identifica cada palabra dentro de un bloque de MP.
- Etiqueta: identifica unívocamente un bloque de MP.

Para determinar si un bloque está en la memoria caché, se debe examinar simultáneamente todas las etiquetas de las líneas de memoria caché, si la etiqueta está, entonces es un “acierto” y se indica que la palabra escogida es la que corresponde en la caché, sino es un fallo y la dirección es la que nos dice donde está en la memoria principal.

Veamos como ocurre esto gráficamente:

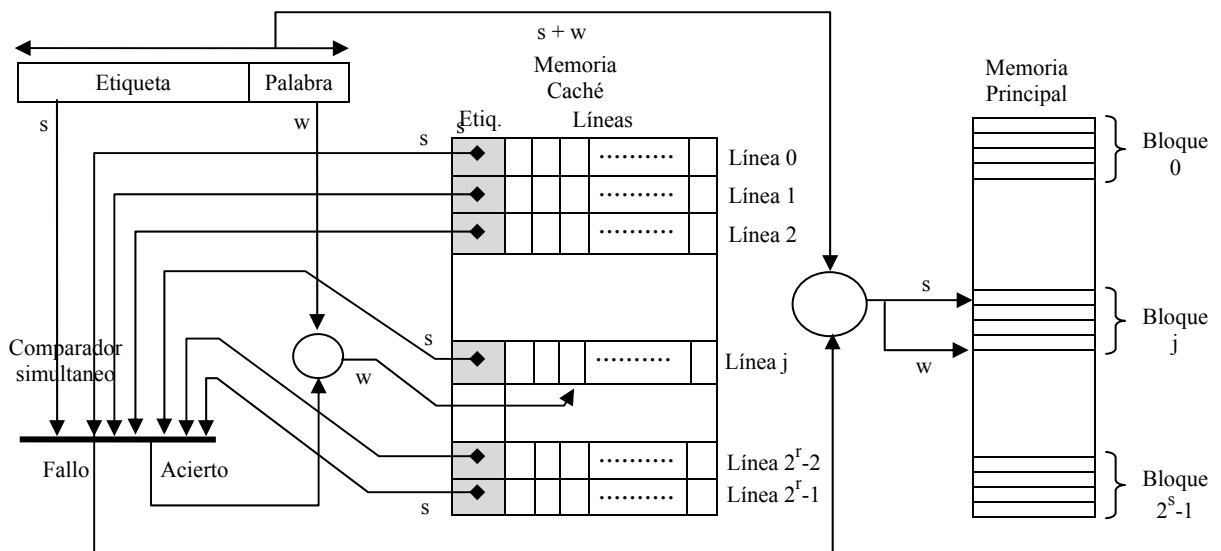


Figura 27 Caché de Correspondencia Asociativa

Ejemplo:

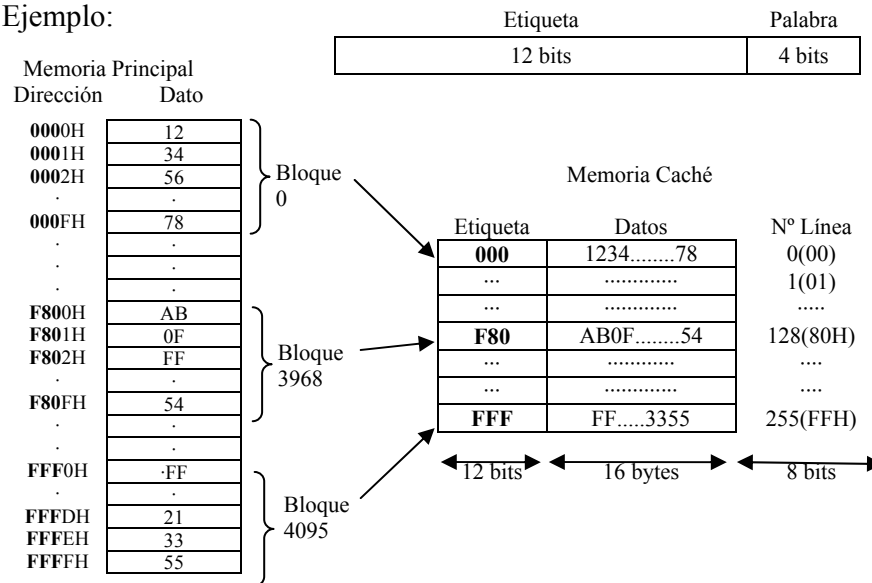


Figura 28 Ejemplo Caché de Correspondencia Asociativa

Ventaja: Mayor tasa de aciertos.

Inconveniente: Necesidad de una circuitería bastante compleja. No es tan rápida como la de acceso Directo.

❑ **Correspondencia Asociativa por Conjuntos**

Esta técnica es un compromiso que trata de aunar las ventajas de las dos técnicas vistas anteriormente. Memoria caché dividida en “T” conjuntos de “L” líneas. Las relaciones que se tienen son:

$$C = T \times L \text{ (nº de líneas de la MC) y } i = j \text{ módulo } T$$

Donde:

- i = número de conjunto de memoria caché (MC)
- j = número de bloque de memoria principal (MP)

El bloque Bj puede asociarse a cualquiera de las líneas del conjunto i. La lógica de control de la memoria caché interpreta una dirección de MP con tres campos.

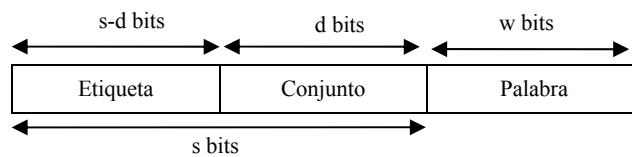


Figura 29 Partición de la Dirección en campos en Correspondencia Asociativa por Conjuntos

Donde:

- w bits de menor peso: palabra dentro de un bloque
- s bits: identifica un bloque de MP
- d bits: especifica uno de los conjuntos de la MC. Lleva directamente a la zona de MC donde está el conjunto.
- s-d bits: etiqueta asociada a las líneas del conjunto “d” bits. Lleva al bloque del conjunto.

Nota: Experimentalmente se ha trabajado con asociatividad entre 2 y 16. En la práctica entre 2 y 4 vías. No mas de 4.

Para saber si una dirección está o no en la memoria caché, lo primero se aplica correspondencia directa y luego correspondencia asociativa.

Veamos como ocurre esto gráficamente:

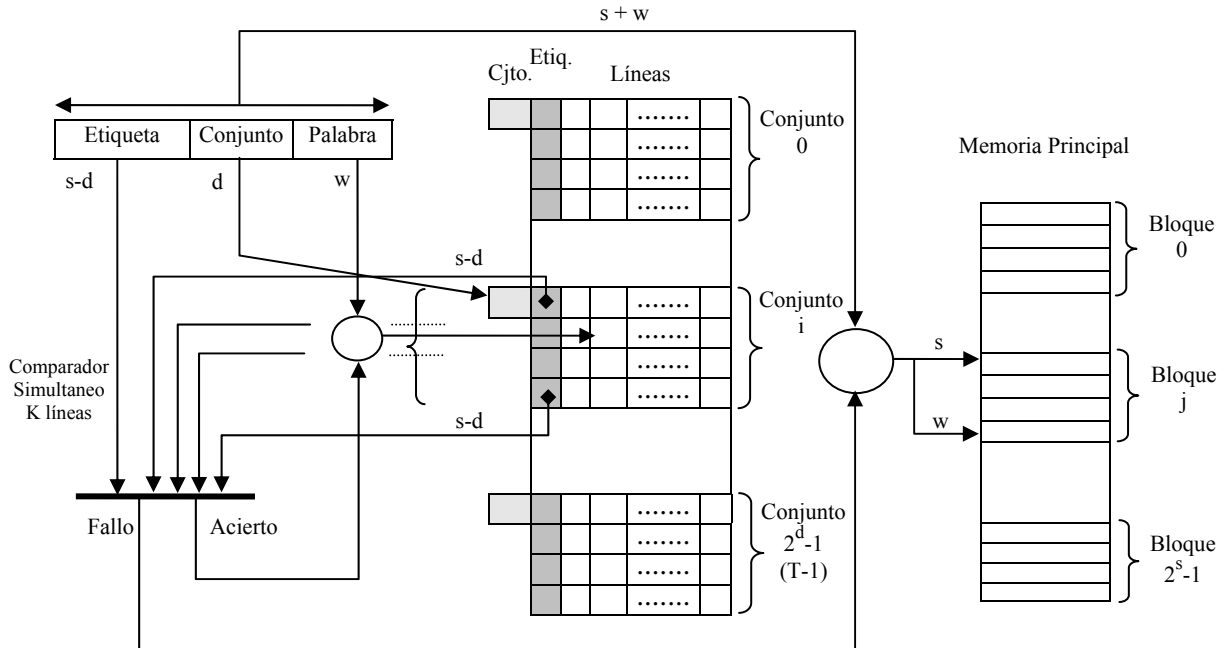


Figura 30 Caché de Correspondencia Asociativa por Conjuntos

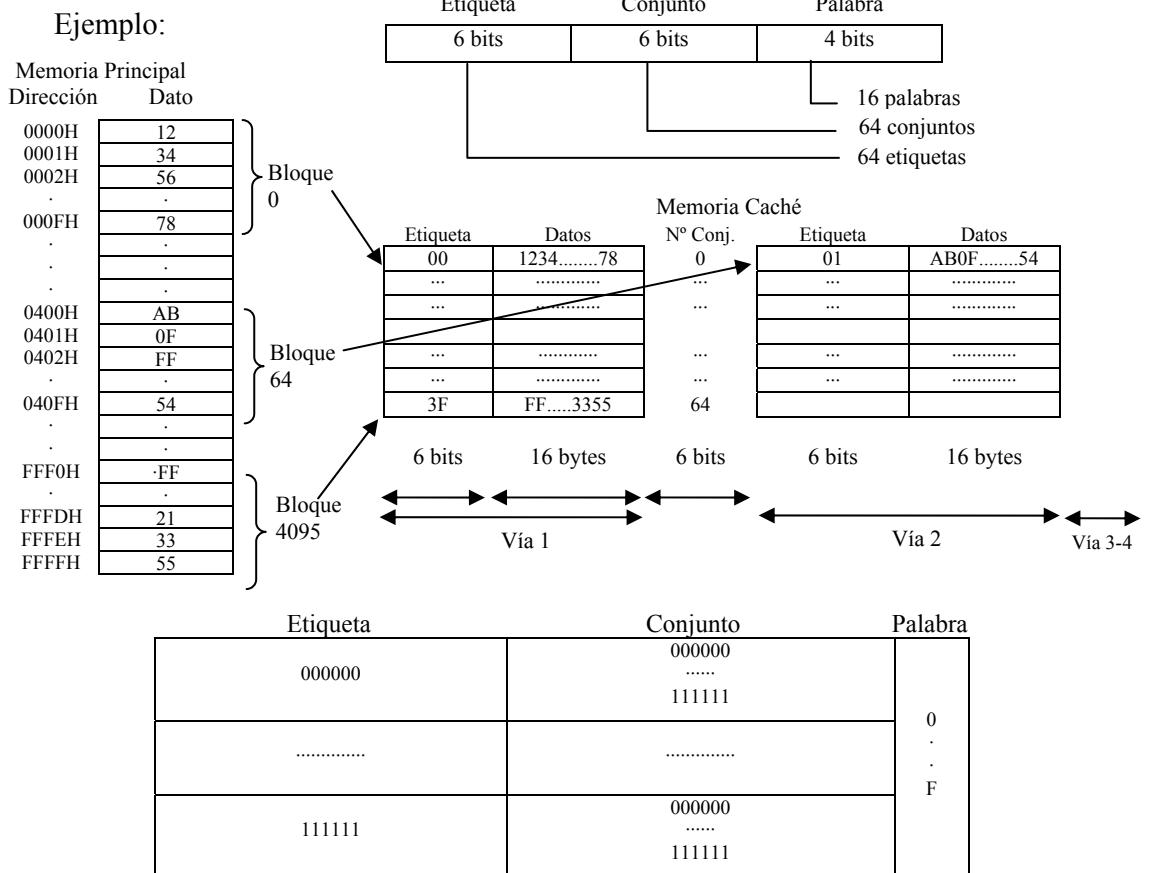


Figura 31 Ejemplo Caché de Correspondencia Asociativa por Conjuntos

Ventaja: Mas rápida y menos costosa que la Asociativa. No presenta el problema de la correspondencia Directa.

Inconveniente: Necesidad de una circuitería bastante compleja.

Nota: Si $T = 1$ es Asociativa, un único conjunto.

Si $L = 1$ es Directa, un único bloque.

1.5.3.7 Algoritmos de Substitución

Cuando un nuevo bloque se transfiere a la memoria caché, debe substituir a uno de los existentes si la línea estuviera ocupada. En el caso de correspondencia directa la línea no tiene sentido en estos algoritmos. Entre los diferentes algoritmos que se han propuesto destacan los siguientes:

- LRU: *Least Recently Used*
- FIFO: *First In First Out*
- LFU: *Least Frequently Used*

1.5.3.8 Ejemplos de Memoria Caché

Fijándonos en los procesadores de Intel, estos han ido incorporando la memoria caché para aumentar su rendimiento. En la Tabla 5 vemos una serie de procesadores de Intel con las características de sus memorias caché:

Procesador	Nivel 1	Nivel 2
Inferior al 80386	NO	
80386	NO	16K, 32K, 64K
80486	8K Datos/Instrucciones	64K, 128K, 256K
Pentium	8K Datos y 8K Instrucciones	Pentium_Pro: 256K, 512K, 1M
Procesador	Tipo Caché (Nivel 1 y Nivel 2)	Descripción
80486	Caché Interna	Tamaño de línea de 16 bytes Organización Asociación por Conjuntos de 4 vías
	Caché Externa	Tamaño de línea de 32, 64 o 128 bytes Org. Asoc. por Conjuntos de 2 vías
Pentium	Caché Interna	Tamaño de línea de 32 bytes Org. Asoc. por Conjuntos de 2 vías
	Caché Externa	Tamaño de línea de 32, 64 o 128 bytes Org. Asoc. por Conjuntos de 2 vías

Tabla 5 Tabla comparativa de cachés

1.6 Memoria Asociativa

1.6.1 Concepto

Una memoria asociativa se caracteriza por el hecho de que la posición de memoria a la que se desea acceder, se realiza especificando su contenido o parte de el y no por su dirección. A las memorias asociativas también se les denomina direccionables por contenido: CAM (*Content Addressable Memory*).

1.6.2 Estructura de una CAM

Una memoria asociativa consiste en un conjunto de registros y una matriz de celdas de memoria, con su lógica asociada, organizada en “n” palabras con “m” bits/palabra. El conjunto de registros está formado por un registro argumento (A) de “m” bits, un registro máscara (K) de “m” bits y un registro marca (M) de “n” bits.

Cada palabra de memoria se compara simultáneamente con el contenido del registro argumento, y se pone a “1” el bit de registro de marca asociado a aquellas palabras cuyo contenido coincide con el del registro argumentado. Al final del proceso, aquellos bits del registro de marca que están a “1” indican la coincidencia de las correspondientes palabras de la memoria asociativa y del registro argumento.

La comparación simultanea se realiza bit a bit. El bit A_j ($j=1,2, \dots, m$) del registro argumento se compara con todos los bits de la columna j si $K_j = 1$. Si existe coincidencia entre todos los bits $M_j = 1$. En caso contrario $M_j = 0$.

Resumen:

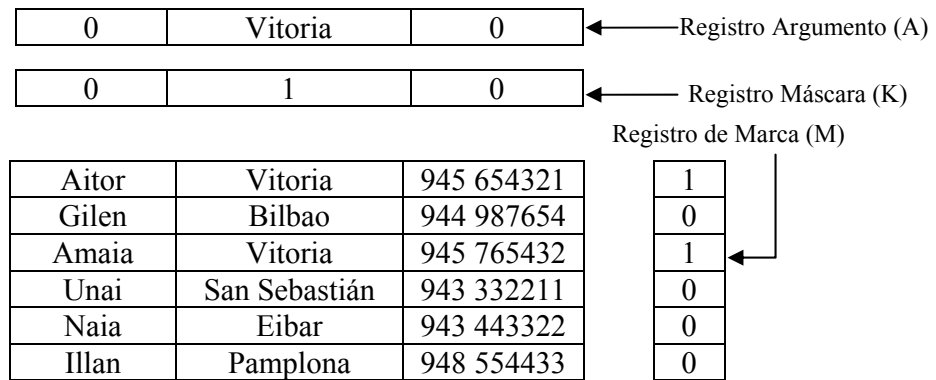


Figura 32 Ejemplo de memoria CAM

- **Registro Argumento:** Lo que quiero averiguar si está en el contenido de la memoria.
- **Registro de Máscara:** Podré quedarme con el que quiero averiguar que esté en la memoria. Determina filtrado del argumento a buscar.
- **Registro de Marca:** Tiene tantos bits como palabras tenga en memoria y pone un bit donde encuentre el argumento buscado a 1 y a 0 donde NO esté. Sirve para saber si en esa palabra (fila de la matriz) está el argumento que busco.

Por regla general, en la mayoría de las aplicaciones la memoria asociativa almacena una tabla que no tiene, para una mascara dada, dos filas iguales. Las memorias asociativas se utilizan sobre todo con memorias caché de tal forma que la identificación de la etiqueta de cada línea se realice de forma simultanea. La TAG RAM es un claro ejemplo de memoria asociativa utilizada como parte de memoria caché en los sistemas con Pentium de Intel.

Los tiempos de acceso a una CAM están entre 4ns y 20ns.

1.7 Memoria Compartida

1.7.1 Concepto

La memoria compartida surge por la necesidad de que dos o mas procesadores compartan información, para ello deberán acceder a la misma unidad de memoria, dejando o recogiendo la información adecuada.

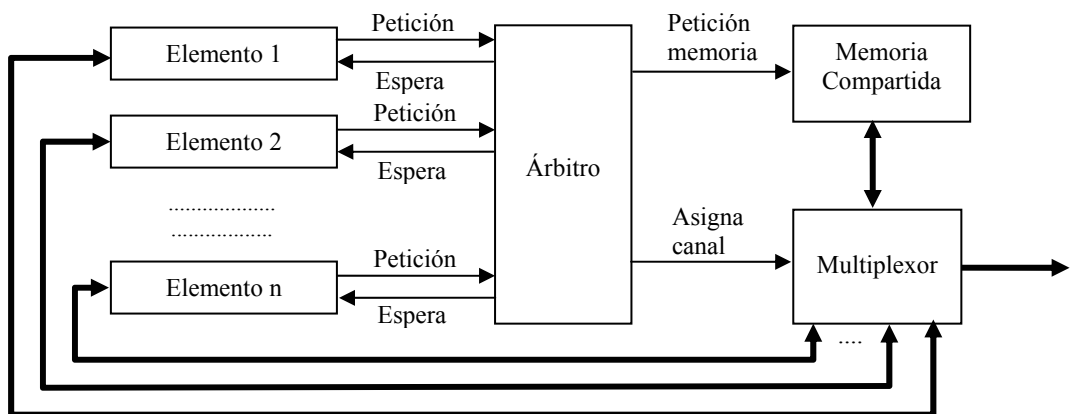


Figura 33 Esquema de Memoria Compartida

El árbitro es el elemento encargado de permitir el acceso a la unidad de memoria, en un instante dado, a cada uno de los elementos que solicitan ese recurso, es decir que arbitra que elemento en cada momento puede compartir el uso de la memoria. El árbitro se diseña de forma que asigne un tiempo de servicio, en promedio, análogo a todas las unidades que solicitan el recurso.

Existen diferentes estrategias:

- Asignación de la menor prioridad al elemento servido.
- Rotación de prioridades. En un estado cualquiera, el próximo estado se calcula rotando el orden de prioridades actual hasta que al elemento que se acaba de dar servicio tiene la menor prioridad.

1.7.2 Memorias de Doble Puerta

Surgen como consecuencia de la aparición de la memoria compartida. Las memorias de doble puerto son memorias compartidas que permiten trabajar con dos dispositivos a la vez (dos buses de datos y direcciones). Se basan en duplicar los buses de direcciones, los decodificadores, la selección de chip, la L/E y bits de semáforo para establecer prioridad de las CPUs que acceden a la misma dirección de memoria, p.ej., *con VRAM puede acceder la tarjeta gráfica y el monitor a la vez.*

La configuración de las memorias de doble puerto es la siguiente:

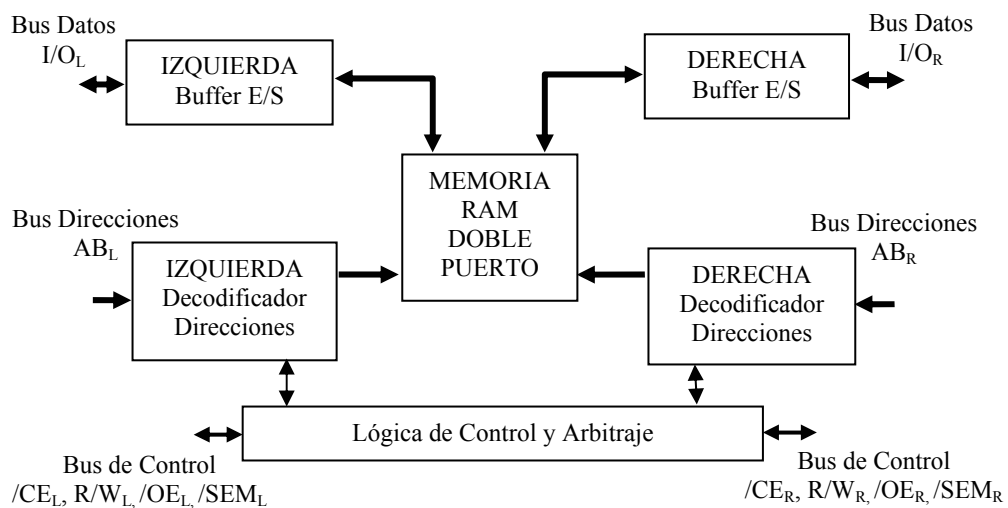


Figura 34 Esquema de Memoria de Doble Puerto

La memoria de doble puerto tiene prácticamente duplicados todos los componentes (puerto izquierdo L) y (puerto derecho R).

Puerto Izquierdo	Puerto Derecho	Descripción
I/O _L	I/O _R	Bus de Datos
AB _L	AB _R	Bus de direcciones
/CE _L	/CE _R	Selección Chip
R/W _L	R/W _R	Lectura/Escritura
/OE _L	/OE _R	Habilita Lectura
/SEM _L	/SEM _R	Habilita Semáforo

Figura 35 Relación de señales en una Memoria de Doble Puerto

1.8 Memoria Virtual

1.8.1 Concepto

El término “Virtual” hace referencia a sistemas en los que el usuario dispone de un espacio de direcciones virtual mas grande que el número de posiciones de memoria real con que cuenta.

En los primeros diseños de computadores, el bus de direcciones de la CPU se conectaba directamente a la memoria principal y no tenía mas que una única interpretación. Si un programa era muy grande y no cabía en la memoria principal, era tarea del programador ajustar los distintos módulos de programa.

El programador dividía los programas en módulos o bloques, llamados recubrimientos o superposiciones (*overlays*).

Los recubrimientos se cargaban o descargaban de memoria durante la ejecución del programa. El programa del usuario se encargaba de llevar a memoria los bloques que fueran necesarios, reemplazando los no necesarios. Esta técnica creaba una gran carga de trabajo tanto al programador como al sistema, que debía saber que partes de su programa necesitaban estar residentes en memoria en un momento dado.

En 1961 se propuso un método para efectuar de manera automática el proceso de los recubrimientos sin que el programador se diera cuenta de ello (grupo de Manchester). La idea consistía en separar los conceptos de espacio de direcciones y posiciones de memoria. Esto se conoce hoy en día como Memoria Virtual.

Supongamos un procesador (MC68000) en el que se ha implementado una memoria de 64 KB, si disponemos de un programa que exceda de la posición 65536, causará un fallo de ejecución. Sin embargo el procesador puede direccionar $2^{24} = 16$ MB de memoria. Es decir, podemos direccionar mas palabras que las que realmente se han implementado. Todo el conjunto de direcciones que pueden ser referenciadas por un programa se denomina espacio de direcciones virtual. Este espacio deberá estar soportado por una memoria secundaria.

Un sistema de memoria virtual gestiona automáticamente los dos niveles de la jerarquía de *memoria principal-memoria secundaria*.

Esta memoria la utilizó por primera vez el computador ATLAS diseñado en la Universidad de Manchester, aunque los primeros computadores comerciales que la utilizaron fueron los IBM/360. En 1974 la familia IBM/370 introdujo el mecanismo Translation Lookaside Buffer (TLB) para la traducción de direcciones.

1.8.2 Diseño

1.8.2.1 Paginación

La principal técnica utilizada en el uso de la memoria virtual consiste en dividir, tanto el espacio de direcciones virtual como el espacio de direcciones real en bloques de igual tamaño.

Un bloque en un sistema paginado se denomina página y un fallo de memoria virtual se denomina falta de página. El tamaño de página es potencia de 2 y oscila entre 512 y 4096 palabras.

Es necesario implementar una función para relacionar el espacio de direcciones de los programas y las posiciones de memoria real (memoria implementada en el diseño).

1.8.2.2 Implementación

La CPU produce una *dirección virtual* que debe ser traducida por una combinación del Hw y el Sw en una dirección real para acceder a memoria principal.

Debemos entender que la “página” tiene una ubicación fija en la memoria secundaria pero una ubicación variable en la memoria principal (física), esta ubicación se denomina “marco”, es decir, una página en la MP ocupará un determinado “marco” determinado por el sistema gestor de memoria.

Cada dirección virtual consta de dos partes:

- **Nº de página:** Parte superior de la dirección (bits de mayor peso) y determina el nº de páginas direccionables.
- **Desplazamiento:** Parte inferior de la dirección (bits de menor peso). Determina el tamaño de la página. Localización de la palabra en la página .

Y cada dirección real consta de dos partes:

- **Nº de Marco:** Parte superior de la dirección real y determina el nº de marcos direccionables de memoria real.
- **Desplazamiento:** Ídem anterior. Localización de la palabra en la página.

Dirección Virtual

PAGINA	DESPLAZAMIENTO
--------	----------------

Dirección Real

MARCO	DESPLAZAMIENTO
-------	----------------

Figura 36 Partición de la Dirección en Campos

En la traducción de una dirección virtual a una dirección real, se traduce el nº de página en un nº de marco y se concatena el desplazamiento.

Veamos un ejemplo:

MEMORIA PRINCIPAL		MEMORIA SECUNDARIA																																															
<table style="border-collapse: collapse;"> <tr><td style="padding-right: 10px;">0 00</td><td style="border: 1px solid black; padding: 2px;">A</td></tr> <tr><td style="padding-right: 10px;">0 01</td><td style="border: 1px solid black; padding: 2px;">B</td></tr> <tr><td style="padding-right: 10px;">0 10</td><td style="border: 1px solid black; padding: 2px;">C</td></tr> <tr><td style="padding-right: 10px;">0 11</td><td style="border: 1px solid black; padding: 2px;">D</td></tr> <tr><td style="padding-right: 10px;">1 00</td><td style="border: 1px solid black; padding: 2px;">E</td></tr> <tr><td style="padding-right: 10px;">1 01</td><td style="border: 1px solid black; padding: 2px;">F</td></tr> <tr><td style="padding-right: 10px;">1 10</td><td style="border: 1px solid black; padding: 2px;">G</td></tr> <tr><td style="padding-right: 10px;">1 11</td><td style="border: 1px solid black; padding: 2px;">H</td></tr> </table>	0 00	A	0 01	B	0 10	C	0 11	D	1 00	E	1 01	F	1 10	G	1 11	H	<table style="border-collapse: collapse;"> <tr><td style="padding-right: 10px;">00 00</td><td style="border: 1px solid black; width: 100px; height: 20px;"></td></tr> <tr><td style="padding-right: 10px;">00 01</td><td style="border: 1px solid black; width: 100px; height: 20px;"></td></tr> <tr><td style="padding-right: 10px;">00 10</td><td style="border: 1px solid black; width: 100px; height: 20px;"></td></tr> <tr><td style="padding-right: 10px;">00 11</td><td style="border: 1px solid black; width: 100px; height: 20px;"></td></tr> <tr><td style="padding-right: 10px;">01 00</td><td style="border: 1px solid black; padding: 2px;">A</td></tr> <tr><td style="padding-right: 10px;">01 01</td><td style="border: 1px solid black; padding: 2px;">B</td></tr> <tr><td style="padding-right: 10px;">01 10</td><td style="border: 1px solid black; padding: 2px;">C</td></tr> <tr><td style="padding-right: 10px;">01 11</td><td style="border: 1px solid black; padding: 2px;">D</td></tr> <tr><td style="padding-right: 10px;">10 00</td><td style="border: 1px solid black; width: 100px; height: 20px;"></td></tr> <tr><td style="padding-right: 10px;">10 01</td><td style="border: 1px solid black; width: 100px; height: 20px;"></td></tr> <tr><td style="padding-right: 10px;">10 10</td><td style="border: 1px solid black; width: 100px; height: 20px;"></td></tr> <tr><td style="padding-right: 10px;">10 11</td><td style="border: 1px solid black; width: 100px; height: 20px;"></td></tr> <tr><td style="padding-right: 10px;">11 00</td><td style="border: 1px solid black; padding: 2px;">E</td></tr> <tr><td style="padding-right: 10px;">11 01</td><td style="border: 1px solid black; padding: 2px;">F</td></tr> <tr><td style="padding-right: 10px;">11 10</td><td style="border: 1px solid black; padding: 2px;">G</td></tr> <tr><td style="padding-right: 10px;">11 11</td><td style="border: 1px solid black; padding: 2px;">H</td></tr> </table>	00 00		00 01		00 10		00 11		01 00	A	01 01	B	01 10	C	01 11	D	10 00		10 01		10 10		10 11		11 00	E	11 01	F	11 10	G	11 11	H
0 00	A																																																
0 01	B																																																
0 10	C																																																
0 11	D																																																
1 00	E																																																
1 01	F																																																
1 10	G																																																
1 11	H																																																
00 00																																																	
00 01																																																	
00 10																																																	
00 11																																																	
01 00	A																																																
01 01	B																																																
01 10	C																																																
01 11	D																																																
10 00																																																	
10 01																																																	
10 10																																																	
10 11																																																	
11 00	E																																																
11 01	F																																																
11 10	G																																																
11 11	H																																																

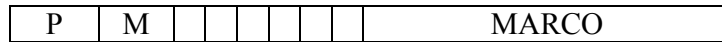
Correspondencia de Direcciones en un sistema Paginado

- Si dispongo de 64K de memoria y escojo 2 bits de página (4 páginas), tendré que 16bits-2bits = 14bits, por lo tanto 4 páginas de 16K
- Si dispongo de 64K de memoria y escojo 4 bits de página (16 páginas), tendré que 16bits-4bits = 12bits, por lo tanto 16 páginas de 4K

Figura 37 Relación entre Memoria Principal y Secundaria

1.8.2.3 Gestión

Si una página puede residir en cualquier marco, necesitamos un mecanismo para encontrarla. El mecanismo se denomina “Tabla de Páginas”, por cada página contiene un “descriptor” de página. Consta de los siguientes campos:



- Número de **Marco**
- Bit de **Presencia**
- Bit de **Modificado**

Figura 38 Descriptor de Página de Memoria Virtual

La tabla de páginas, que debe residir en memoria, está indexada por n° de página. La búsqueda de un descriptor en la tabla de páginas se realiza mediante el campo n° de página de la dirección virtual, que actúa como índice de la tabla.

Cuando el bit de presencia “P” está a “1”, la página está cargada en MP, y la tabla de páginas contiene la dirección del marco asignado a esta página. Si P = 0, no está la página en MP. Este n° de marco, concatenado con el desplazamiento de la dirección virtual, define la dirección real correspondiente.

Veamos esto en el esquema siguiente:

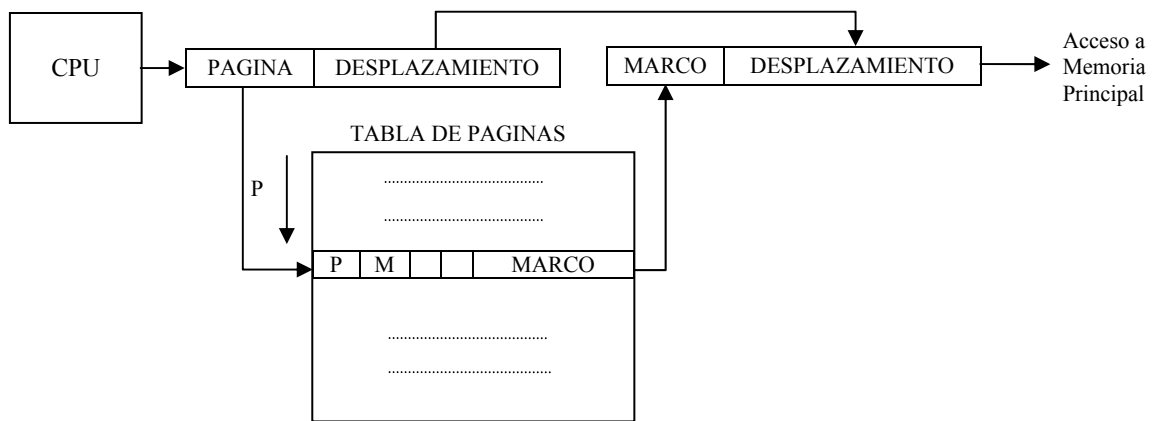


Figura 39 Tratamiento de Memoria Virtual

La no existencia de página en MP, provoca la “Excepción” de Falta de Página, transfiriéndose la página, que ha provocado la excepción, de la Memoria secundaria a la MP. Se expulsará una Página de la MP mediante la Política de Substitución.

Si la página que se decide expulsar tiene el bit M = 1, indica que esta página se ha modificado durante su estancia en la MP, por lo tanto habrá que transferirla a la MS copiando dicha página, antes de traer la nueva página.

Para el tratamiento de Páginas la CPU se complementa con un dispositivo denominado **MMU** (*Memory Management Unit*), este dispositivo puede estar dentro o fuera de la CPU. La incorporación de MMUs libera a la MP de la Tabla de Páginas, esta se ubica en los registros de la MMU.

Una solución intermedia consiste en casi toda la Tabla de Páginas. Esté en MP y un conjunto relativamente pequeño de Descriptores esté en la MMU. Estos serán los mas recientemente utilizados (referenciados). Estos registros se denominan Buffers de Traducciones Anticipadas (TLB: *Translation Lookahead Buffer*). “Principio de Localidad de las Referencias”.

1.9 Otras Memorias

1.9.1 Introducción

El rendimiento de un computador se ve afectado por la velocidad del bus, nos movemos entre 66 MHz, 100 MHz, 133 MHz, etc.. La velocidad final de un procesador, se obtiene a partir de un multiplicador interno.

Para no reducir el rendimiento de un computador, las memorias deberían ser igual de rápidas que el procesador, pero las tecnologías disponibles no lo permiten; por lo tanto, cada vez que el procesador accede a la memoria tiene que esperar (t_w : *time wait*). El uso de las memorias caché aumentan el rendimiento del sistema, pero debido a su alto coste y al espacio que ocupan físicamente no es viable añadir gran cantidad de memoria caché.

A medida que ha evolucionado la tecnología, las memorias cada vez se han fabricado mas rápidas para reducir los tiempos de espera del procesador. Así pues, otra alternativa para aumentar el rendimiento del sistema es utilizar memorias principales mas rápidas.

1.9.2 Memorias Entrelazadas

Se denomina “entrelazado” al diseño de memoria en módulos. Se dan dos tipos de entrelazado:

- **Orden Inferior:** Direcciones consecutivas en módulos consecutivos. Selección del módulo con los bits de menor peso.
- **Orden superior:** Cada módulo contiene direcciones consecutivas. Selección del módulo con los bits de mayor peso.

En este tipo de memorias, los módulos se organizan de tres formas que estudiaremos mas adelante:

- Organización con acceso S
- Organización con acceso C
- Organización con acceso C/S

□ Organización S

Es una de las configuraciones más sencillas:

- Permite acceder de manera simultánea a los módulos de memoria.
- Utiliza entrelazado de orden inferior y aplica los “n-m” bits superiores de la dirección (mayor peso) a todos los módulos $M = 2^m$ módulos de memoria simultáneamente en un acceso.
- En un único acceso se obtienen las M palabras consecutivas de información procedentes de los M módulos de memoria.
- Los m bits inferiores de la dirección (menor peso) se utilizan para seleccionar la información de un módulo particular.

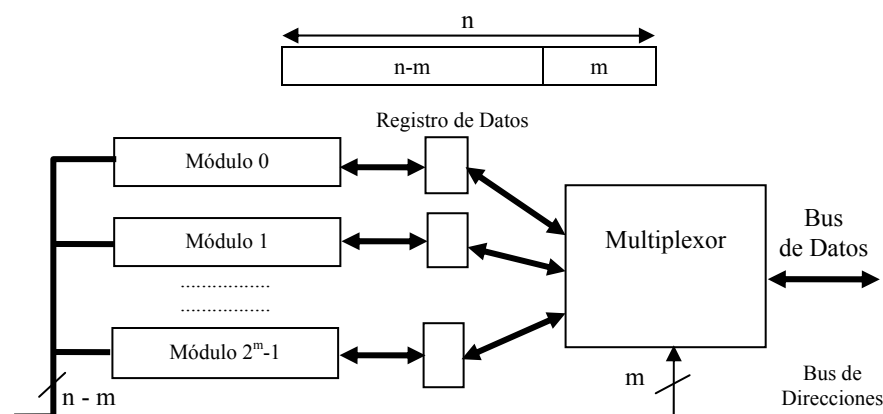


Figura 40 Organización S

Este tipo de configuración se denomina de acceso S porque a todos los módulos se accede simultáneamente tal y como se refleja en la Figura 40

El funcionamiento es sencillo:

- La dirección de memoria se divide en dos por tiempos de acceso y tiempo de recuperación; solo nos quedamos con una palabra del módulo de memoria, lo que implica que por cada módulo de memoria tendremos un registro de datos.
- Los módulos de memoria se organizan partiendo de las matrices de bits de las memorias; con los “n-m” bits mas significativos tengo la selección de cada uno de los módulos de memoria, de ellos leo a la vez así como de los registros, pero a la salida solo tengo una palabra por el multiplexor que se ha puesto antes del bus de datos.

Veamos un ejemplo para clarificar el funcionamiento:

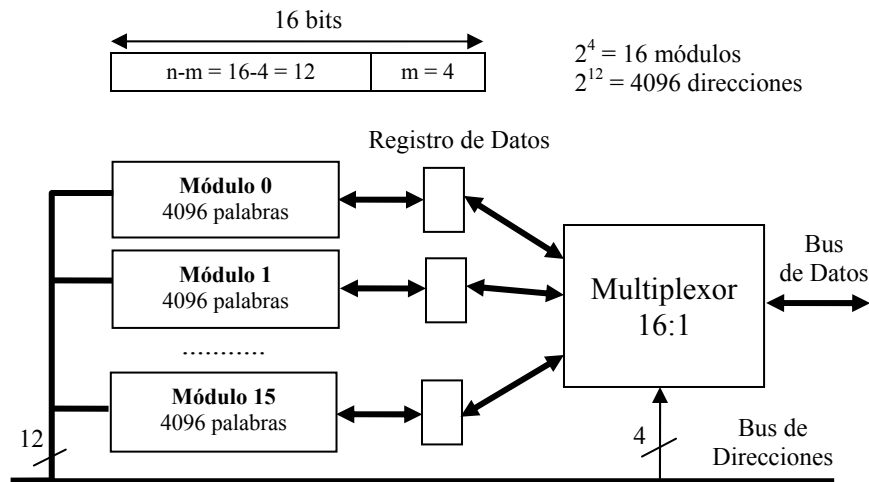


Figura 41 Ejemplo Organización S

Ventajas: Este tipo de organización es ideal para acceder a un vector de datos o para la búsqueda de instrucciones y operandos secuenciales. También es útil en la transferencia de bloques en sistemas con memoria caché.

Inconvenientes:

- Acceso a posiciones de memorias no consecutivas.
- Se utilizan todos los módulos para acceder a una posición.

□ **Organización C**

Presenta la siguiente organización:

- Permite acceder a posiciones distintas de los módulos concurrentemente
- Los “m” bits de orden inferior se utilizan para seleccionar el módulo y los “n-m” bits restantes direccional el elemento deseado dentro del módulo.
- El controlador de memoria se utiliza para mantener una petición que referencie un módulo ocupado. Existen configuraciones en las que el controlador dispone de una única cola FIFO o bien dispone de una cola FIFO por cada módulo de memoria.

Este tipo de configuración se denomina de acceso C porque a todos los módulos se accede de manera concurrente tal y como se refleja en la Figura 42

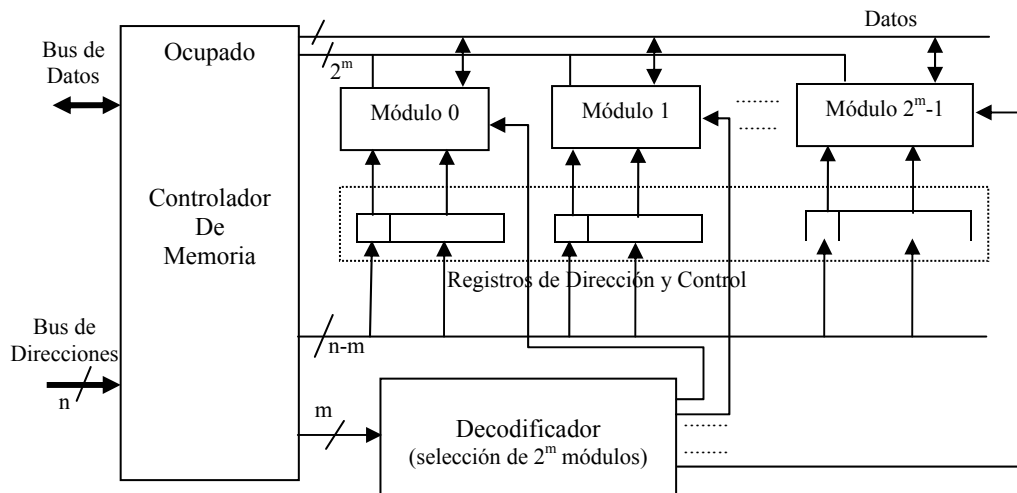


Figura 42 Organización C

Ventaja: para acceder a una posición de memoria ya no se ocupan todos los módulos sino que ocupa únicamente el módulo que contiene esa dirección y el resto están libres. Para seleccionar el módulo correspondiente se emplea el decodificador.

□ Organización C/S

Es una configuración mixta:

- Combina los esquemas con acceso S y acceso C.
- Los módulos se organizan en forma de matriz bidimensional. Si el acceso S es un entrelazado de M vías y el acceso C un entrelazado de L vías, hasta L accesos diferentes a bloques de M palabras consecutivas por estar en progreso simultáneamente.

1.9.3 Memorias Dinámicas

1.9.3.1 Fast Page Mode RAM (FPM RAM)

Es la primera memoria dinámica mas popular. Este tipo de memoria incorpora un sistema de paginado porque considera probable que el próximo dato referenciado esté en la misma columna, ganando tiempo en caso afirmativo. Incorpora un contador de columna automático.

Las transferencias de datos desde la memoria se realiza en paquetes de 4 datos denominados ráfagas (*burst*). El uso de ráfagas elimina los tiempos de establecimiento de dirección y de precarga de líneas de fila (también “página”) y columna posteriores al primer acceso.

Los tiempos de acceso están entre 120 ns. y 50 ns.

El rendimiento de un tipo de memoria se expresa con cuatro números separados por guiones que se corresponden con los ciclos de reloj que la memoria necesita para responder. La memoria ideal sería 1-1-1-1 que significa que en cada ciclo de reloj se transfiere un dato. Para una caché del tipo L2 (SRAM) es 2-1-1-2.

La memoria FPM RAM presenta un esquema, en el caso mas favorable, 5-3-3-3, es decir, cuatro estados de espera en el primer dato y dos en los sucesivos, haciendo un total de 10 por ráfaga.

Los anchos de banda están entre 16 MHz. Y 66 MHz.

1.9.3.2 Extended Data Output RAM (EDO RAM)

Es una modificación de las FPM RAM. Consigue una mejora entre el 10% y el 15% en velocidad con respecto a la FPM. Mientras se accede a los datos se prepara la siguiente dirección. La EDO RAM ha sido la memoria más popular debido a que su velocidad de acceso se incrementa y a que los fabricantes han tenido que hacer muy pocos cambios respecto de la FPM RAM.

La EDO RAM puede trabajar, en el caso más favorable, con un esquema 5-2-2-2, aumenta el ancho de banda de la memoria, lo que puede suponer una mejora de hasta un 40% en el rendimiento global. Aún así, el ancho de banda alcanzado con memorias de 60 ns, es de 40 MHz (este valor suele estar entre 33 MHz. Y 75 MHz.), todavía lejos de los 66 MHz del bus de un PC.

Algunos PCs permiten trabajar con memorias de 50 ns consiguiendo un ancho de banda de 50 MHz. Se consiguen tasas de transferencia de 264 MB/s.

La aparición, en los computadores, de un bus de 100 MHz deja a la memoria EDO sin sentido en su utilización ya que, como se ha visto, su ancho de banda llega a 50 MHz y esto reduce mucho el rendimiento del sistema.

1.9.3.3 Burst Extended Data Output RAM (BEDO RAM)

Esta memoria permite trabajar con un esquema 5-1-1-1 con 50 ns de tiempo de acceso consiguiendo alcanzar los 66 MHz. Estas memorias no han tenido mucho éxito debido a la aparición del bus de 100 MHz y las memorias SDRAM.

Las transferencias de datos son por ráfagas de 4 datos tanto en lectura como escritura. Mientras envía datos al procesador, está leyendo la siguiente dirección.

1.9.3.4 Synchronous DRAM (SDRAM)

Consigue una mejora del 25% en velocidad con respecto a la EDO. La memoria DRAM síncrona o SDRAM a diferencia de las DRAM típicas, que son asíncronas, intercambia datos con el procesador de forma sincronizada con una señal de reloj externa, que opera a la velocidad del bus sin imponer estados de espera. El reloj maneja una Máquina de Estados Finita (FSM).

El núcleo DRAM es mucho más rápido que el de la memoria convencional, llegando a alcanzar velocidades de hasta cuatro veces más. Una arquitectura de doble banco permite entrelazados, de forma que mientras un banco prepara los datos otro los proporciona. Así mismo, la longitud de la ráfaga es programable, permitiendo un diseño flexible en función del sistema que deba soportarlo.

El esquema de trabajo de la SDRAM es de 5-1-1-1. Con memorias de 15 ns se pueden alcanzar los 100 MHz. La firma IDT dispone de memorias con esquema 2-1-1-1 a 50 MHz. Los tiempos de acceso también han sufrido mejoras estando entre 12 ns. y 6 ns.

Aplicaciones de las SDRAM se dan en los Buses PC66 a 66 MHz., PC100 a 100 MHz. y PC133 a 133 MHz. La tasa de transferencia llega a 528 MB/s.

Un ejemplo de SDRAM es la PC-100 RAM que cumple determinadas restricciones establecidas por Intel para el correcto funcionamiento con el bus de 100 MHz implementado en su *chipset* 440BX.

En el año 2000 las SDRAM suplieron a las memorias de tecnologías anteriores. Suele ser utilizada en cachés grandes.

1.9.3.5 Double Data Rate SDRAM (DDR RAM)

Ya han sido enunciadas en el apartado de memorias RAM. Son un desarrollo posterior de las SDRAM. Presenta arquitecturas de 8 bytes y 16 bytes. Transfieren información en el flanco de subida y de bajada del reloj, duplicándose la cantidad de información transferida. De esta forma con 200 MHz. de ancho de banda, tendremos en el bus del PC una tasa de hasta 3.2 Gb/s. para arquitecturas de 8 bytes y de 6.4 Gb/s para arquitecturas de 16 bytes.

Las DDR han ido evolucionando hacia la DDR-2 y DDR-3. Siendo la DDR-2 estandarizada en el 2005 y la DDR-3 está en proceso de desarrollo y en espera de su estandarización bajo normas JEDEC. La diferencia entre ellas está en velocidades mayores, menores tensiones de alimentación y ciertas diferencias en sus interfaces.

Funciona a velocidades de 100 MHz, 133 MHz, 166 MHz y 266 MHz y frecuencias superiores. La nomenclatura del es PC1600 (se trata de PC100 para DDRAM), PC2100 (PC133 con DDRAM), siendo 1600, 2100 la tasa de transferencia en Mbps.

Como factor positivo presenta una arquitectura abierta, por lo que no paga derechos a fabricantes, pero como factor negativo es que Intel no apoya este tipo de arquitecturas, lo cual deja interrogantes en su evolución.

1.9.3.6 Rambus DRAM (RDRAM)

También se conoce como Direct Rambus. Son memorias construidas con bus de 16 bits y la frecuencia de reloj de 400 MHz. Trabaja, como las DDR, con flancos de subida y de bajada, alcanzando un ancho de banda de 1,6 GBs. Es el complemento de las tarjetas gráficas AGP, evitando cuellos de botella.

Los circuitos se empaquetan en tarjetas denominadas RIMMs (Rambus Inline Memory Module) no compatibles con los SIMMs (SDRAM).

Memoria desarrollada en cooperación con la empresa Rambus Company. Paga derechos a Intel.

1.9.3.7 Synchronous Link DRAM (SLDRAM)

Posee las ventajas de las SDRAM y la DDR RAM. Internamente es similar a una DDR. Incorpora un protocolo orientado a paquetes para el control/direccionamiento y un sistema de calibrado de tiempos para mantener la compatibilidad con las antiguas memorias.

Planteada para buses de 64 bits a velocidades de 200 MHz, 400 MHz efectivos, Alcanzando velocidades de 3.2 GB/s. Pensada para grandes servidores.

Esta memoria ha sido desarrollada por un consorcio de alrededor de 20 empresas, no debiéndose pagar licencias por su fabricación. Es la alternativa a las RDRAM y se plantea como estandar de los PCs de altas prestaciones.

1.9.3.8 Video DRAM (VRAM)

Se trata de la versión video de una FPM. Versión doble puerta de una DRAM. Uso para gráficos. Actualmente obsoleta, substituida por las SDRAM y por las SGRAM.

- Puerta 1: Como una DRAM
- Puerta 2: Solo salida Video

1.9.3.9 Synchronus Graphics RAM (SGRAM)

Versión video de una SDRAM para adaptar gráficos. Añade un bit de máscara y escritura de bloque.

1.9.3.10 Pipeline Burst Static RAM (PBSRAM)

Opera en modo ráfaga, típicamente 3-1-1-1. Requiere un ciclo extra de reloj. Típico 4 ns. de acceso. Amplio rango de relojes.

1.9.3.11 Tabla Resumen

A continuación veamos en la Tabla 6 donde se refleja la evolución histórica de las memorias dinámicas.

Año	Tipo de memoria	Tacceso o Velocidad
1987	FPM	50 ns.
1995	EDO	50 ns.
1997	PC66 SDRAM	66 MHz.
1998	PC100 SDRAM	100 MHz.
1999	RDRAM	800 MHz.
1999-2000	PC133 SDRAM	133 MHz.
2000	DDR SDRAM	266 MHz.
2001	DDR SDRAM	333 MHz.
2002	DDR SDRAM	434 MHz.
2003	DDR SDRAM	500 MHz.

Tabla 6 Evolución histórica de las DRAM

TEMA 2: DISPOSITIVOS DE ALMACENAMIENTO SECUNDARIO

2.1 Introducción

Surgen dos necesidades, una consiste en tener almacenados todos los archivos de programas generados por el equipo de desarrollo y otra en poder disponer del conjunto de particiones que forman el programa para ser cargado a Memoria Principal cuando así sea requerido por el sistema gestor.

Aun cuando los avances de la tecnología permitan disponer de dispositivos con mas capacidad de almacenamiento que sus antecesores, nunca podremos llegar (en determinadas aplicaciones) a poder almacenar la totalidad del programa en la memoria del computador. Es por ello que se hace necesario el disponer de dispositivos secundarios que permitan almacenar información de manera “masiva”.

2.2 Discos Magnéticos

2.2.1 Disco Duro (HD *Hard Disk*)

Consiste en uno o mas platos de aluminio con un recubrimiento de características magnéticas. Originalmente estos platos medían 50 cm., hoy en día estamos en 3 cm. Con el avance de la ciencia en materia de electromagnetismo, esta dimensión irá disminuyendo.

Una cabeza consistente en un entrehierro (u otro material magnético) bobinado, se sitúa sobre la superficie de la película magnética, no llega a tocarla por existir una fina lámina de aire que actúa como colchón (0.5 micras). La cabeza está situada en el extremo de un brazo mecánico que entra o sale de la superficie de manera radial.

La cabeza magnética polariza los dominios magnéticos de la superficie durante el proceso de escritura. Durante el proceso de lectura la cabeza presenta el proceso de inducción magnética a causa de los diferentes dominios magnéticos.

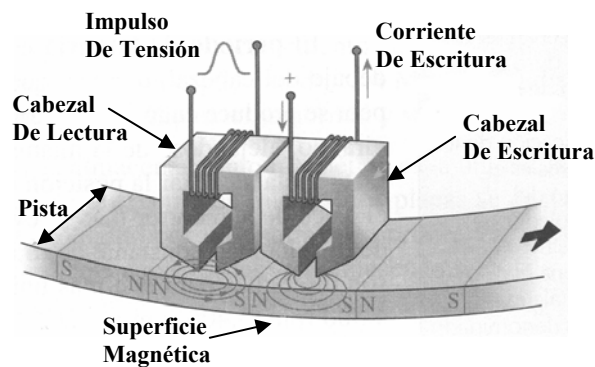


Figura 43 Cabezas de Lectura/Escritura

Como cada disco presenta dos caras magnéticas, se disponen dos brazos con sendas cabezas, aumentando la capacidad de almacenamiento el doble. La información está “serializada”.

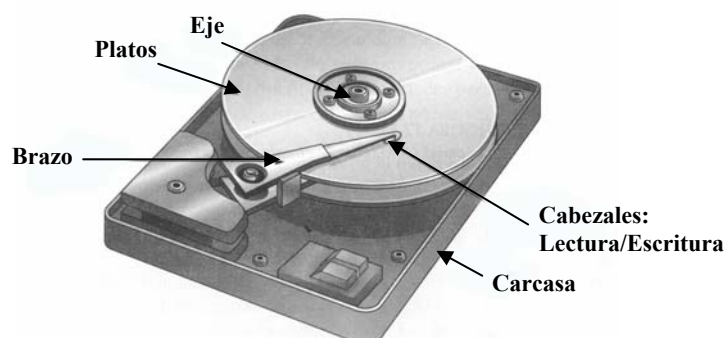


Figura 44 Aspecto físico de un Disco Duro

La estructuración de un disco es la siguiente:

- Una rotación completa del disco, crea una circunferencia imaginaria, se denomina “pista”. Se trata de una línea magnetizada.
- Una pista está dividida en “sectores” de longitud fija; normalmente son de un tamaño de 512 bytes (4096 bits).
- Cada sector está precedido de un “preámbulo” de bits que permiten sincronizar la cabeza antes de realizar las operaciones de lectura o escritura.
- Al final de cada sector hay un campo “corrección de errores” (ECC: *Error Correcting Code*). Estos códigos pueden ser *Haming* o *Reed-Solomon*.
- Entre sectores hay una pequeña separación. No contiene información.

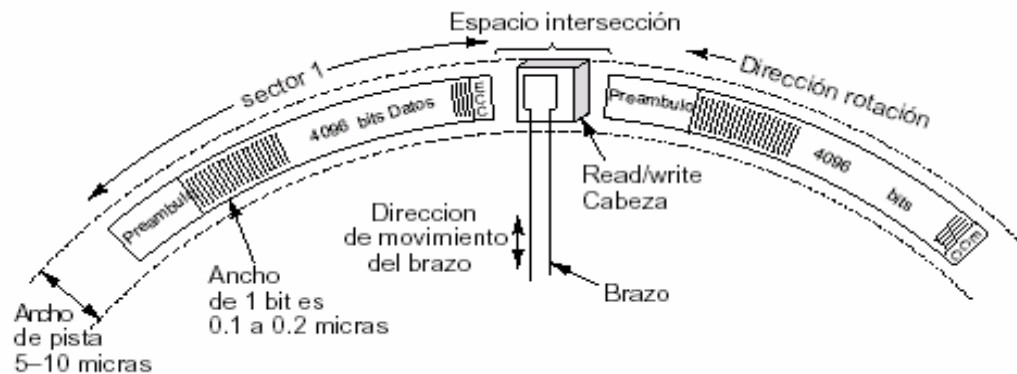


Figura 45 Información sobre el Disco

La práctica totalidad de discos están contruidos agrupando mas de un plato sobre el mismo eje, cada superficie tiene su propio brazo y cabeza magnética. Todos los brazos se mueven solidariamente.

El conjunto de todas las pistas, en una posición radial dada, se denomina “cilindro”.

La dirección de un sector es: nº de cabeza – nº de cilindro – nº de sector

El mecanismo de escritura/lectura de un sector, presenta dos aspectos:

- Movimiento del brazo a una posición radial determinada, esto se denomina “búsqueda” (*SEEK*). Entre 5 y 15 ms. entre pistas (medición al azar). Hay mediciones en el orden de 1 ms.
- Giro de la pista hasta que el sector deseado queda debajo de la cabeza, esto se denomina “latencia rotacional”. Entre 4 y 8 ms la rotación

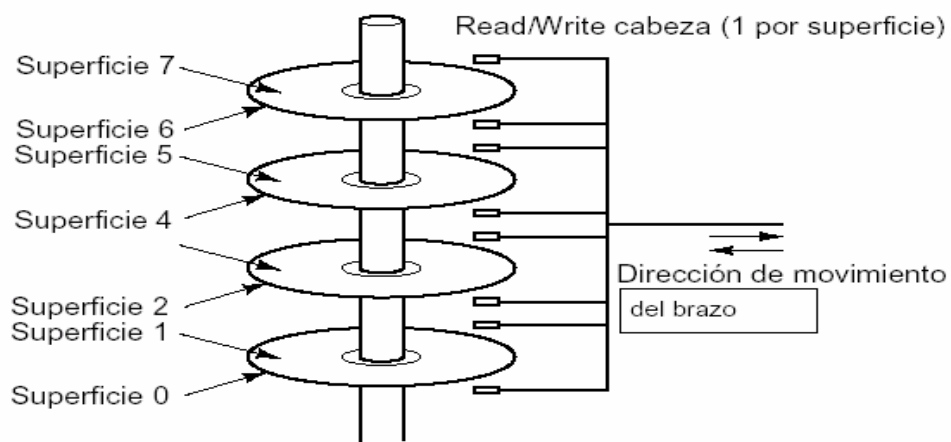


Figura 46 Esquema de Platos y Cabezas en un Disco Duro

Cuando vamos a almacenar un archivo, dependiendo de su tamaño, este podrá caber en unidades de almacenamiento consecutivo pero si hay unidades ocupadas anteriormente, tendrá que fragmentarse el archivo y ubicar los fragmentos en diferentes zonas o “clusters”.

Esta es una labor que la realiza el controlador. Un disco, con el tiempo, presenta problemas de fragmentación, lo que lleva a aumentar el tiempo de búsqueda de la información. Se hace necesario el uso de “desfragmentadores” que vuelvan a ubicar archivos en áreas contiguas.

Como un disco inicialmente es una superficie de material magnético, es preciso dar un formato a esta superficie, de esta manera se divide el disco en sectores y pistas y se descubren zonas con errores superficiales. Toda esta información se guarda en un archivo denominado FAT (*File Allocation Table*) que es la Tabla de Asignación de Archivos, ubicado en el sector cero. El Sistema Operativo almacena la estructura del directorio del disco, el nombre de los ficheros, su tamaño, la fecha y hora de su última modificación, los atributos de los ficheros y los “clusters” que se han utilizado para almacenar los archivos.

Cada unidad de disco contiene un dispositivo denominado “controlador de disco”. El procesador del computador se comunica con el controlador.

La velocidad de giro está entre los siguientes valores: 3600, 5400, 7200 y 10800 rpm.

La tasa de transferencia típica está entre 5 y 20 MB/s.

2.2.2 Discos flexibles (FD *Floppy Disk*)

Su aparición es determinante en cuanto la distribución del software. La necesidad de “portar” el software lleva a la industria a crear este tipo de soporte. IBM los introduce y en sus comienzos son “flexibles” puesto que el disco magnético está realizado sobre un soporte de poliéster, además la protección es una funda de cartón. Las primeras unidades tenían un tamaño de 8” y de simple densidad, podían almacenar 128 KB.

La evolución de las tecnologías nos lleva a tamaños de 3.5” (y menores) con capacidades de 1.44 MB. La fundas están realizadas en plástico rígido.

La velocidad de giro es de 360 r.p.m. Consta de 80 cilindros y 18 sectores. La Tasa de Transferencia es de 54 KB/s.

La diferencia fundamental con los discos duros, consiste en que las cabezas magnéticas están en contacto con la superficie, esto conlleva un desgaste de cabezas y de la superficie magnética. Para reducir este impacto, se retraen las cabezas cuando no hay lectura/escritura, deteniendo también la rotación del disco. Esto provoca retrasos considerables cuando se emite la orden de lectura/escritura ya que primero hay que poner en rotación al disco y posteriormente realizar la búsqueda.

$T_{\text{promedio búsqueda}} (T_{\text{seek}})$ es aproximadamente de 95 ms y la Latencia rotacional de 83 ms.

2.2.3 Discos IDE

Este tipo de disco duro surge a partir de los ordenadores personales (PC). En los PC XT, se introdujo un disco *Seagate* de 10MB controlado por un dispositivo *Xebec* dispuesto sobre una tarjeta insertable en una de las ranuras del Bus del PC, permitiéndose controlar 2 unidades de disco. La evolución consistió en disponer de un controlador ubicado en la electrónica del disco (solidario a el), denominándose a estas unidades IDE (*Integrated Device Electronics*).

La máxima capacidad estaba dada por:

- 16 cabezas
- 63 sectores
- 1024 cilindros

Esto permite 1.032.192 sectores lo que limita a 528 MB la capacidad máxima.

Debido a las anteriores limitaciones impuestas por el sistema BIOS (*Basic Input Output System*) de los PCs, pronto se pasó a realizar una extensión de este tipo de unidades

denominándose EIDE (*Extended IDE*), permitiéndose referenciar hasta $2^{24}-1$ sectores, ampliándose la capacidad hasta 8 GB y pudiéndose controlar hasta 4 unidades de disco.

2.2.4 Discos SCSI

Este tipo de discos presentan la misma estructura que los anteriores en cuanto a n° de platos, pistas, cilindros, etc., la diferencia consiste en el interfaz de transferencia de datos. Presentan una tasa de transferencia mucho mas alta que los IDE.

SCSI significa *Small Computer System Interface*, el nombre y estándar definitivo se establece en 1986 por el comité ANSI, efectuándose una revisión en 1994 denominándose SCSI-2, actualmente estamos en el SCSI-3.

La Tabla 7 nos muestra las potencialidades de las diferentes actualizaciones:

Nombre	N° Bits Datos	Transf. MB/s
SCSI-1 (conector de 25 pines)	8	5
SCSI-2 (conector de 50 pines)	8	5
SCSI-2 Rápido	8	10
SCSI-2 Rápido y Ancho	16	20
SCSI-3	16	40
SCSI-2 Ultra Rápido y Ancho	16	80

Tabla 7 Evolución del Bus SCSI

El estándar SCSI ha extendido su uso a otro tipo de periféricos, estos pueden ser, p.ej., CD-ROM, Impresoras, Scaners, etc..

El n° de dispositivos que pueden conectarse en un Bus SCSI es de 8, se referencian de 0 a 7 en la cadena. En el SCSI ancho se llega a 15 periféricos. Los dispositivos se conectan siguiendo una cadena, el 2° al 1°, el 3° al 2°, etc..., los identificadores no tienen porqué ser consecutivos. Disponen, por lo tanto, de un conector de entrada y uno de salida (siguiente dispositivo). El último dispositivo debe llevar un elemento terminador, se trata de una red de resistencias de 50 Ω , que evita las reflexiones eléctricas en los cables.

Este tipo de conexión en discos ha sido normalizado por Macintosh, HP y Sun, actualmente hay mas compañías que lo han ido incorporando.

El interfaz SCSI de 8 bits de Datos (mas común), requiere un cable de 50 hilos, de estos, 8 son bits de Datos, 1 bit de Paridad, 9 bits de Control el resto para alimentación y expansiones futuras.

2.2.5 Discos RAID

Más que un tipo de disco consiste en una organización distinta de los discos. Este tipo de organización surge por la confluencia de dos necesidades:

- a. Aumentar la tasa de transferencia de datos entre el procesador y el disco. Los procesadores aumentaban su rendimiento pero los discos sufrían un cierto estancamiento.
- b. Disminuir la tasa de errores y mejorar su recuperación. El fallo de un disco podía provocar la pérdida de datos definitiva.

Es por esto por lo que se idea una nueva estructura de organización de los discos, denominándose RAID (*Redundant Array Inexpensive Disks*); si la idea inicial también era el proveer discos económicos, pronto la industria cambió la "I" por Independientes, de tal manera que en una caja se incluyen una serie de discos independientes físicamente pero que van a presentar una organización entre ellos, de tal manera que el usuario o el computador lo ve como un solo disco. Su aplicación se da en grandes servidores o sistemas de control críticos.

Esta estructuración puede presentar las siguientes configuraciones:

- Se divide la información en bloques de tal manera que se graban en discos independientes y después son leídos en paralelo.
- Se duplica la información, grabándose en paralelo.
- Se generan bits de control (paridad) para recuperación de errores, según diferentes organizaciones.

Combinando estos modos de operación se establecen seis niveles de discos RAID, que se denominan RAID nivel 0, ..., RAID nivel 5. No establece una jerarquía, simplemente es una organización.

- **RAID nivel 0:** Disco dividido en tiras (*strips*) de K sectores cada una. De 0 a K-1 en la primera tira (1er. disco), de K a 2K-1 en la segunda (2º disco), etc.. La distribución de datos entre varias unidades se denomina *striping* (multiplexaje de datos). Si hay un comando de lectura de un bloque que ocupa varias tiras consecutivas (varios discos), el controlador del RAID genera procesos de lectura simultánea a los discos, entregando la información en bloque sin retrasos. Muy eficiente cuando se le piden grandes bloques de información.

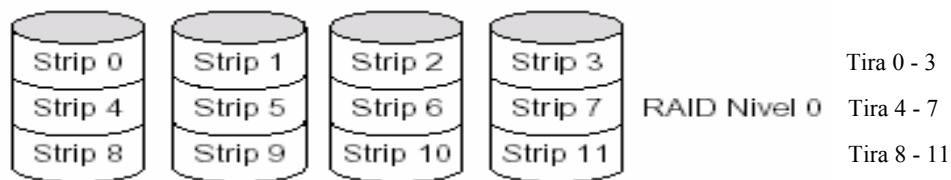


Figura 47 Raid nivel 0

- **RAID nivel 1:** Trabaja con tiras. RAID puro. Duplica la información. Muy alta tolerancia a fallos. El rendimiento en lectura es el doble, puesto que puede acceder a los dos conjuntos de discos. Ante fallo de disco, solamente hay que substituir el disco dañado.

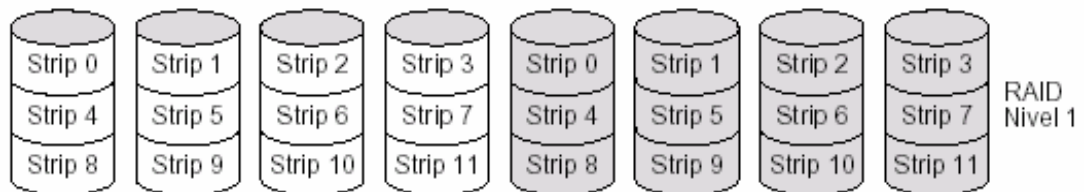


Figura 48 Raid nivel 1

- **RAID nivel 2:** Trabaja con palabras y con bytes. Se divide la información en nibbles y se introducen 3 bits de paridad código *Hamming* (bits 1, 2 y 4). Las unidades de disco están sincronizadas.

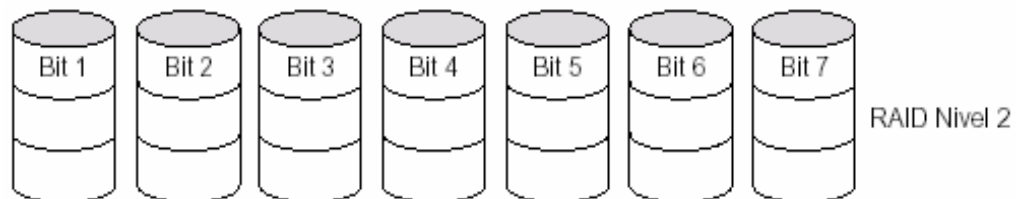


Figura 49 Raid nivel 2

- RAID nivel 3:** Trabaja con palabras. Simplifica el nivel 2, introduciendo un solo bit de paridad. Las unidades de disco están sincronizadas.

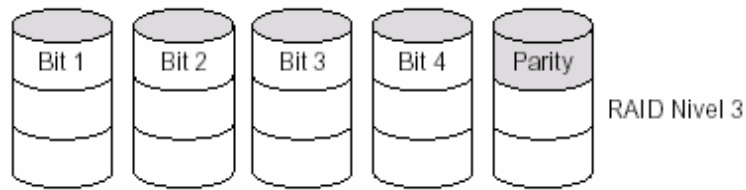


Figura 50 Raid nivel 3

- RAID nivel 4:** Trabaja con tiras. Parecido a nivel 0 pero introduciendo paridad por tira. Se calcula el OR Exclusivo de los bytes que componen todas las tiras. Protegido contra pérdidas pero de rendimiento bajo ante actualizaciones. Gran carga sobre el disco de paridades. Las unidades de disco no están sincronizadas.

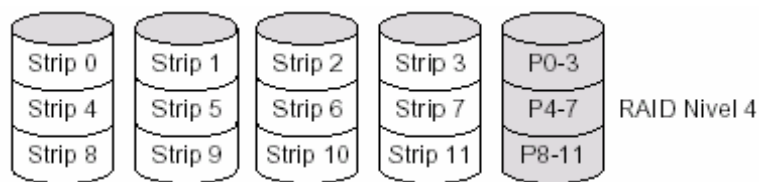


Figura 51 Raid nivel 4

- RAID nivel 5:** Trabaja con tiras. Distribuye los bits de paridad entre todas las unidades. Resuelve la carga de los bits de paridad. Si una unidad falla, su reconstrucción es un proceso complejo. Las unidades de disco no están sincronizadas.

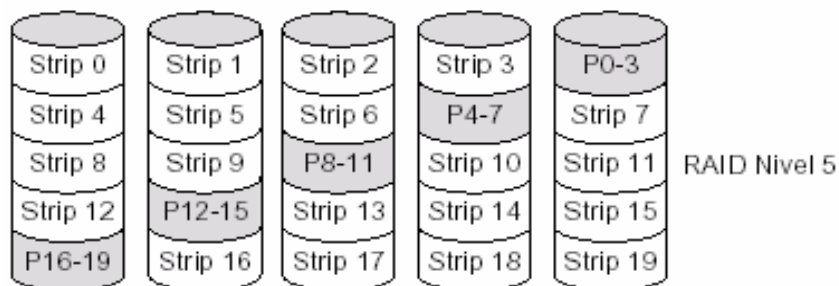


Figura 52 Raid nivel 5

2.3 CD-ROM

Compact Disk Read Only Memory. Es el primer medio de almacenamiento masivo no magnético, está basado en tecnología óptica láser. Su densidad de grabación es mucho mas alta que la magnética. Irrumpen en el mercado en 1980. Los inventores son Philips y Sony. Su diámetro es de 120 mm. Y su espesor de 1.2 mm.

La escritura se realiza de la siguiente forma, un láser infrarrojo de alta potencia quema orificios de 0.8 micras de diámetro en un disco maestro recubierto de vidrio. Con este disco maestro se genera un molde, presentando salientes donde han estado los orificios creados por el láser. Posteriormente se inyecta resina de policarbonato fundida para formar el CD, creándose el mismo patrón de orificios que el master de vidrio, finalmente se deposita una fina capa de aluminio reflectante sobre el policarbonato, seguido de una capa de resina protectora.

Las depresiones en el substrato se denominan “**fosos**” (*pits*) y el área entre fosos “**planicie**” (*lands*). La escritura se realiza en espiral desde el agujero central hacia el exterior hasta 3.2 mm del borde. La espiral describe 22,188 revoluciones alrededor del disco (aprox. 600 por mm), su longitud total llegaría a 5.6 Km.

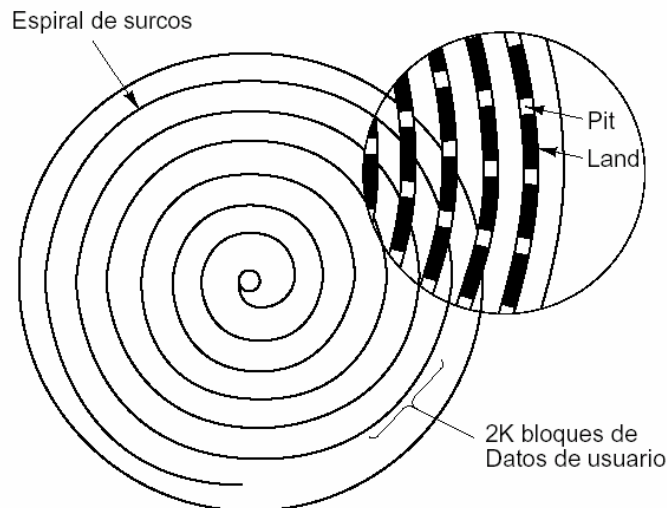


Figura 53 Pista de un CDROM. “Pits” y “Lands”

La reproducción se realiza moviendo la cabeza láser del interior hacia el exterior, en el interior la velocidad de rotación es de 530 rpm y en el exterior de 200 rpm, esto hace que la velocidad lineal sea constante y de valor 120 cm/s.

Un láser infrarrojo de baja potencia hace incidir su luz sobre los fosos y planicies, los fosos tienen una profundidad de $\frac{1}{4}$ de la longitud de onda de la luz láser, por lo que al ser reflejados, esta luz estará desfasada en $\frac{1}{2}$ de longitud de onda respecto a la luz reflejada circundante por lo que se produce una interferencia destructiva y llega menos luz al receptor, es así como se diferencia un foso de un planicies. Transición es “1”, no transición “0”.

La codificación es como sigue:

- Cada byte se codifica sobre 14 bits (código de corrección de error) denominándose “símbolo”.
- La agrupación de 42 símbolos forma un “cuadro” (588 bits).
- La agrupación de 98 cuadros forma un “sector” de cd-rom.
- Cada sector comienza con un “preámbulo” de 16 bytes. 12 bytes indican comienzo de sector, los 3 bytes siguientes el nº de sector, el último byte se denomina “modo” (se definen dos modos en el Libro Amarillo). El modo 1 es el que se ha comentado.

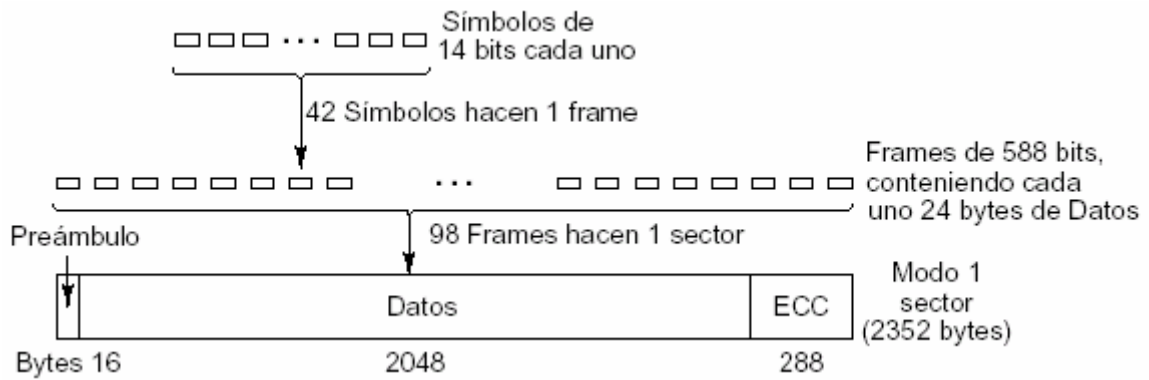


Figura 54 Empaquetamiento de Datos en un CDROM

Esto nos lleva a 7203 bytes para una información útil de 2048 bytes. Eficiencia del 28%. La corrección de errores se realiza siguiendo el siguiente esquema:

- Los errores de un solo bit se corrigen en el nivel inferior.
- Los errores de ráfaga corta se corrigen a nivel de cuadro.
- Resto de errores residuales se corrigen a nivel de sector.

2.3.1 Gravables

Los CD-R son parecidos a los CD-ROM, son discos de policarbonato blanco de 120 mm de diámetro. Se utiliza oro en lugar del aluminio para crear la capa reflectante.

Este tipo de CD no tiene fosos por lo que hay que simularlos, para que exista una diferencia de reflectividad entre foso y planicie. Esto se realiza añadiendo una capa de colorante (cianina o ftalocianina) entre el policarbonato y el oro.

En el estado inicial la capa colorante es transparente y permite el paso del haz de luz y su reflexión en la capa de oro.

Para realizar la escritura, se aumenta la potencia del láser entre 8 y 16 mW. Cuando el haz incide sobre un punto con colorante, este se calienta y rompe uno de sus enlaces químicos. Este cambio en la estructura molecular crea un punto oscuro.

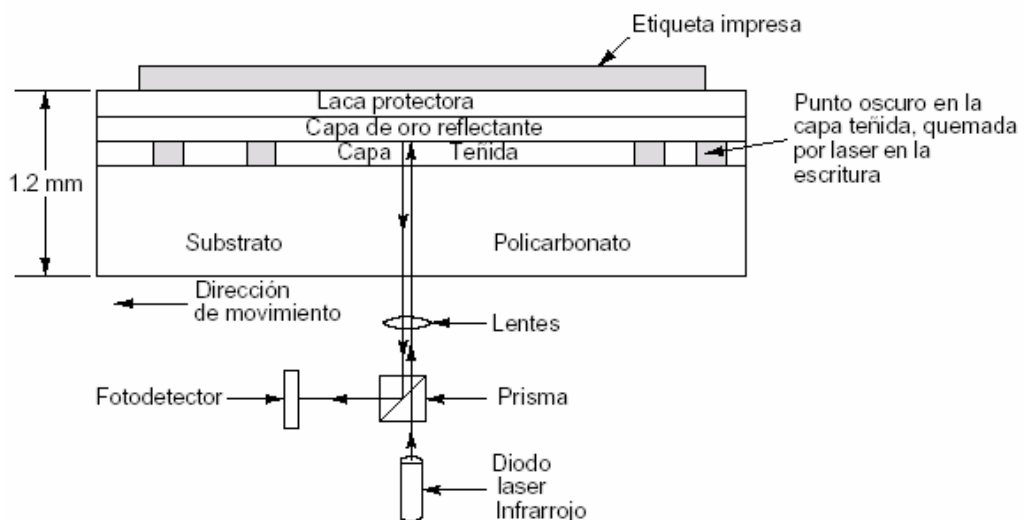


Figura 55 Estructura de un CDROM gravable

Durante la reproducción (potencia de 0.5 mW), el fotodetector percibe una diferencia entre los puntos oscuros (quemados) y las áreas transparentes (no quemadas), Interpretando esto como una transición entre foso y planicie.

Una evolución de este sistema es el permitir grabar por sesiones, no teniendo que grabar el CD completamente la primera vez que queremos realizar una grabación.

2.3.2 Regravables

Los CD-RW cambian el colorante de los CD-R por una aleación de plata-indio-antimonio-telurio. Esta aleación presenta dos estados estables: cristalina y amorfa con diferente reflectividad.

Se emplean tres potencias distintas en el haz del láser .

- Potencia alta: Funde la aleación, pasando de estado cristalino (alta reflectividad) a estado amorfo (baja reflectividad). Representa “foso”
- Potencia media: Funde la aleación recristalizando a su estado natural. Representa “planicie”.
- Potencia baja: Estado de lectura, no altera la estructura cristalina del material.

2.4 DVD

Se trata del Vídeo Disco Digital (Digital Video Disk). Actualmente se denomina Digital Versátil Disk. Existen dos tamaños reconocidos, uno de 8 cm y otro de 12 cm. La diferencia con los CD-ROM es la siguiente:

- Fosos de 0.4 micras.
- Espiral mas cerrada (0.74 micras entre pistas en vez de 1.6 micras).
- Láser rojo de 0.65 micras en vez de 0.78 (puede presentar problemas de compatibilidad con los lectores CD-ROM antiguos).

Esto multiplica la capacidad por siete, pasando a 4.7 GB.

Se han definido cuatro formatos:

- Un solo lado, una sola capa (4.7 GB).
- Un solo lado, capa dual (8.5 GB).
- Dos lados, una sola capa (9.4 GB).
- Dos lados, capa dual (17 GB) (8.5 horas de grabación como película).

La tecnología “dual” dispone de una capa semirreflectante y una capa reflectante, según la distancia focal del láser, trabaja con una o con otra.

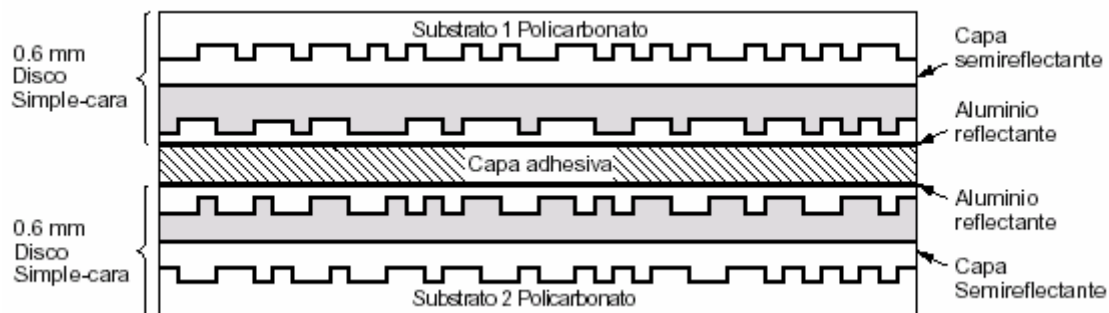


Figura 56 Estructura de un DVD

2.5 Discos Magneto-ópticos

Este tipo de discos presenta un método diferente en Lectura/Escritura con relación a los Discos Duros. En escritura, un haz de luz láser (potencia alta) calienta la superficie del disco (temperatura Curie aproximadamente 200°C) que está recubierta de un material cristalino, liberando cristales que pueden desplazarse por medio del campo magnético. Una cabeza de L/E parecida a la cabeza de los discos magnéticos, escribe los datos, cambia la dirección (polarización) de los cristales. La magnetización cambia el comportamiento óptico (efecto Kerr). Los puntos con una polaridad representan ceros y la polaridad opuesta unos.

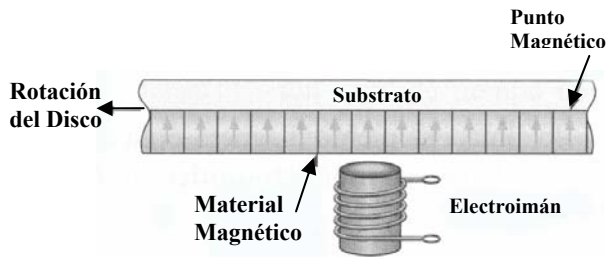


Figura 57 Disco no Grabado

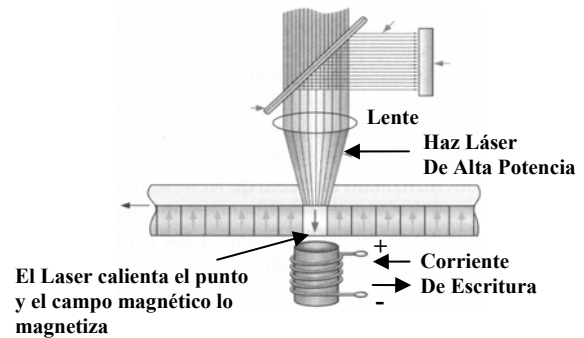


Figura 58 Operación de Escritura

La ventaja de este procedimiento radica en que la zona magnetizada se reduce únicamente a la parte calentada por el láser. De esta manera se pueden obtener discos de mas de 1 GB por cada cara (1.2 – 9.1 GB). Los lectores suelen incorporar una única cabeza y hay que dar la vuelta al disco.

El proceso de lectura requiere proyectar un haz de luz láser de baja potencia, donde la diferente orientación de los cristales creará dos diferentes intensidades de reflejo de luz. La operación de borrado requiere la proyección del haz laser de alta potencia y suministrar al electroimán una corriente que invierta el campo magnético.

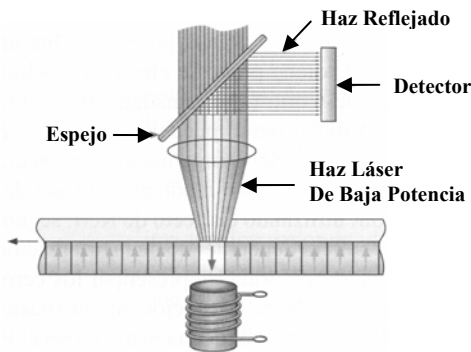


Figura 59 Operación de Lectura

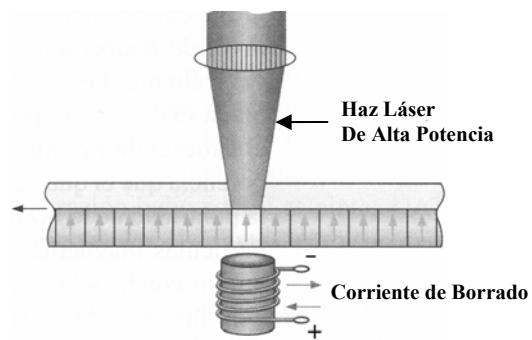


Figura 60 Operación de Borrado

Útiles tanto para copias de seguridad como para intercambio de datos.

2.6 Cintas Magnéticas

Son dispositivos de Lectura/Escritura de acceso secuencial. Especialmente útil cuando la información a almacenar es considerable. Especialmente recomendadas en salvaguarda de Sistemas completos. Dentro del abanico de cintas podemos destacar las tres siguientes: QIC, DAT y DLT.

La información está codificada digitalmente. La estructura del cabezal/cinta se refleja en el siguiente gráfico. La velocidad y anchura corresponden al tipo de cinta QIC.

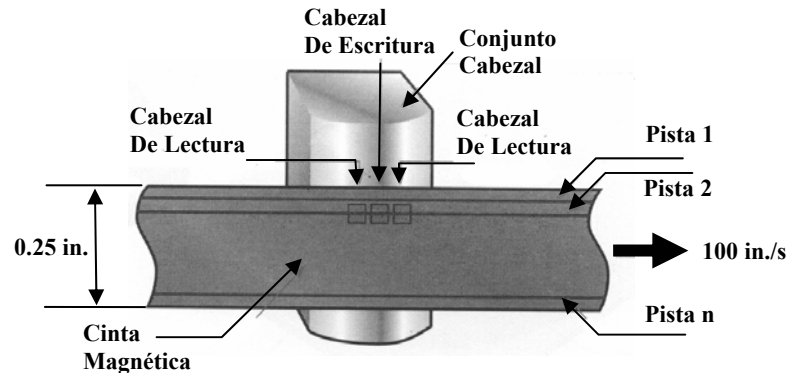


Figura 61 Cabezal y Pistas en una Cinta Magnética

Las tecnologías actuales permiten disponer de hasta 36 cabezas de L/E en un mismo cabezal. El nº de pistas puede llegar a 144 según tecnologías.

Las cintas **QIC** (*Quarter-inch Cartridge*) de 5,25" y 3,5", son parecidas a un casete de audio, variando el nº de pistas entre 36 y 72, pudiendo almacenarse desde 40 Mbytes, 1,2 Gbytes, 4 Gbytes hasta 25 Gbytes. En el gráfico superior se observa que el cabezal consta de una cabeza de escritura y dos cabezas de lectura a cada lado de la de escritura. Introducida por 3M en 1972.

Una de las cintas de menor tamaño es la denominada **DAT** (*Digital Audio Tape*), fue introducida para el almacenamiento de audio pero el comité ISO la adaptó al estándar **DDT** (*Digital Data Tape*) pero se sigue conociendo por DAT. Su capacidad de almacenamiento está entre los valores siguientes: 2,6 GB - 4 GB. - 8 GB. - 20 GB. y 36 GB (o 72 si comprimido). Se esperan evoluciones. Esta es una cinta muy extendida. Introducida por Sony en 1987.

Las cintas **DLT** (*Digital Linear Tape*) por su tecnología constructiva aporta la mayor capacidad de almacenamiento, según tecnologías estamos entre 35 Gbytes y 500 Gbytes (recientemente 1200 Gbytes). Fueron inventadas por Digital Equipment Corporation

TEMA 3: BUSES Y DISPOSITIVOS DE E/S

3.1 Introducción

En este capítulo se va a tratar de la interacción del usuario del computador con la información generada por el computador. El computador va a generar una información para el usuario que será inteligible solo si se dispone del periférico adecuado.

El usuario podrá introducir cierta información requerida por el computador a través de determinados dispositivos. Se trata de la relación Hombre-Máquina (*Man-Machine Interface*).

Esta información generada por la CPU del procesador o bien generada por el usuario para ser consumida por la CPU, necesita ser transportada por una serie de caminos denominados BUSES.

Los Buses son caminos eléctricos que unen dispositivos. Podemos distinguir tres tipos de Buses.

- Buses internos del Procesador: Unión de CPU con ALU y registros.
- Buses de comunicación CPU con Memoria.
- Buses destinados a E/S:

Los dos últimos Buses disponen de tres tipos de información:

- Bus de Datos
- Bus de Direcciones
- Bus de Control

Normalmente estos Buses discurren a través de lo que se denomina una “Placa Madre”, en ella se insertan las distintas tarjetas electrónicas controladoras de dispositivos y se comunican con el Procesador por medio del Bus de la Placa Madre. Podemos insertar tarjetas controladoras de Discos, tarjetas controladoras de Monitores, etc..

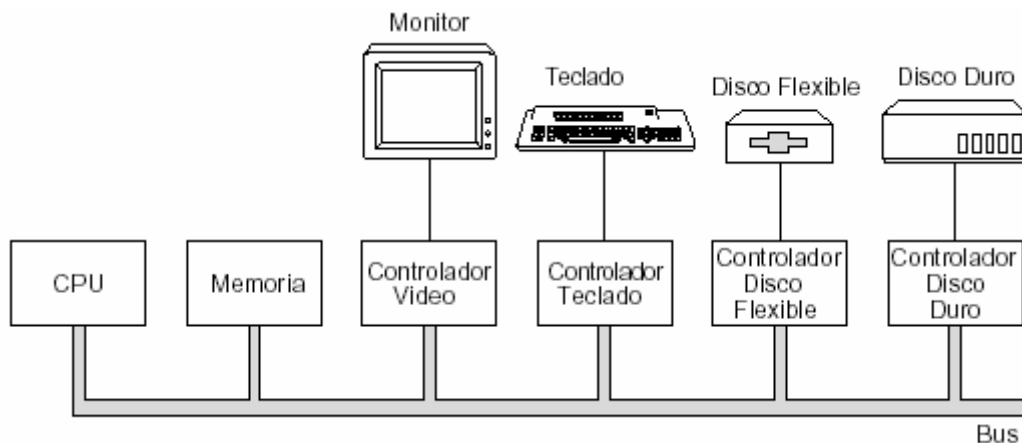


Figura 62 Conexión de Dispositivos en un Bus

Según criterios constructivos, la placa Procesadora junto con la memoria puede presentar los siguientes modos de montaje:

- Tarjeta insertada en la Placa Madre: Soluciones industriales (Incluso PCs industriales).
- Componentes montados sobre la Placa Madre: Mundo del PC.

Los aspectos mas importantes, en cuanto al diseño de un Bus, son los siguientes:

- **Ancho del Bus**
- **Temporización**
- **Arbitraje**
- **Operaciones**

3.2 Buses

3.2.1 Ancho de Bus

Por ancho de Bus entendemos el n° de señales eléctricas que transportan algún tipo de información (pistas de cobre). Ya se ha comentado que va a estar formado por Direcciones, Datos y Control.

Con respecto al Bus de Direcciones, según distintas soluciones, nos movemos entre 16 bits y 32 bits. En máquinas con necesidad de mayor potencia de direccionamiento se llega a 64; téngase en cuenta que el direccionamiento es 2^n .

En cuanto a los Buses de Datos nos movemos entre 8 y 32 bits. Los Buses de control no suelen presentar problemas en cuanto a su tamaño.

Cuanto mas denso sea un Bus mas cara será la placa. Mayor densidad de pistas y mas finas, además será necesario disponer de mas capas para poder realizar una conexión correcta entre placas.

En el mundo del PC, una modificación del Bus puede provocar la incompatibilidad de las tarjetas existentes en el mercado con la nueva solución. Por ello los crecimientos de Bus o incorporación de nuevos Buses, debe dejar inalterada la forma de conexión con lo ya existente.

Esta modificación puede ser debida a un aumento de la velocidad o bien a la incorporación de nuevas señales con nuevos cometidos.

3.2.2 Temporización del Bus

El intercambio de información entre diversos dispositivos, a través del Bus, puede estar regido por un Reloj (Bus Síncrono) o bien no depender de un reloj (Bus Asíncrono).

En el caso síncrono, un reloj maestro genera una onda cuadrada que marca las transiciones. La frecuencia está comprendida entre 5 y 100 MHz. Un Bus ISA de PC trabaja a 8.33 MHz y un PCI trabaja a 33 MHz o a 66 MHz.

En el caso asíncrono, los ciclos de bus pueden tener un ancho cualquiera.

La mayoría de los Buses implementados son síncronos.

3.2.2.1 Buses Síncronos

Un Bus síncrono puede estar movido por las siguientes señales:

- **Petición de Memoria:** /MREQ (indica acceso a memoria, no a E/S)
- **Señal de Lectura:** /RD (si "1" es una Escritura - WR)
- **Señal de Espera:** /WAIT (ciclo de espera generado por el dispositivo si este es lento)
- **Buses de Datos y Direcciones.**

La secuenciación de las señales se refleja en la Figura 63

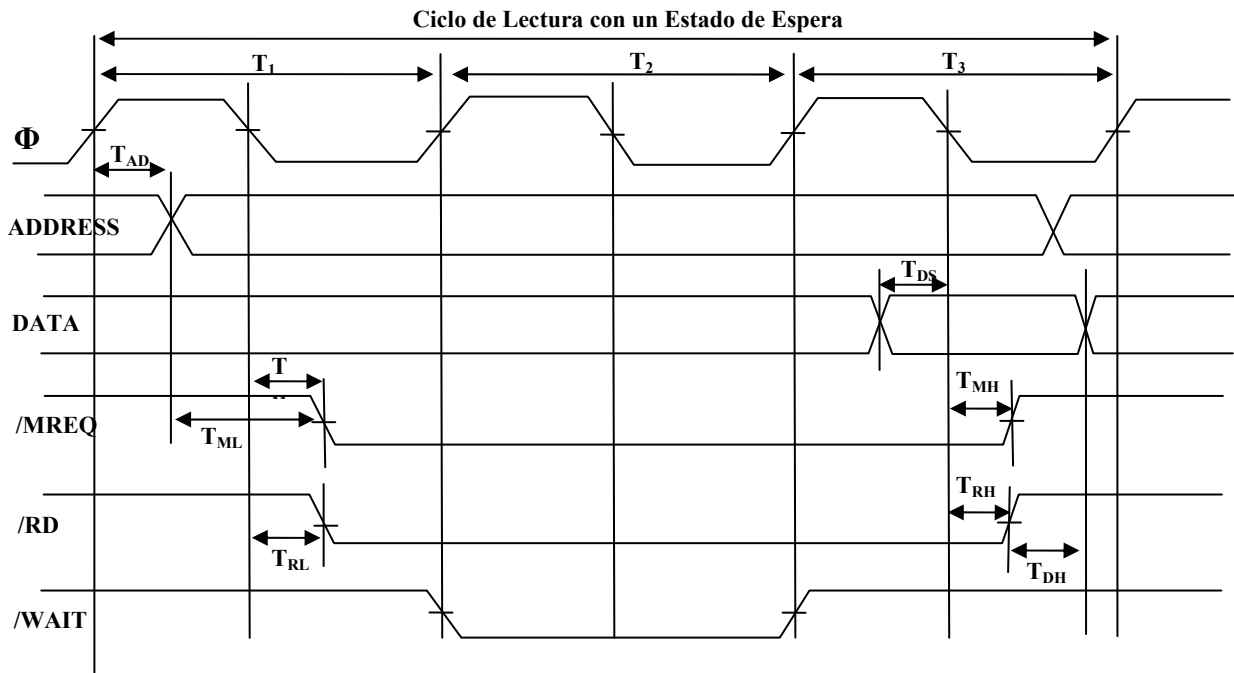


Figura 63 Bus Síncrono

3.2.2.2 Buses Asíncronos

La introducción de este tipo de Buses, libera a los dispositivos de la dependencia del reloj. Los dispositivos irán a la velocidad que puedan ir sin necesidad de supeditarse al ancho del periodo del reloj.

Las señales que se van a necesitar, además de las vistas anteriormente, son las siguientes:

- /MSYN : Master SYNchronitation
- /SSYN : Slave SYNchronitation

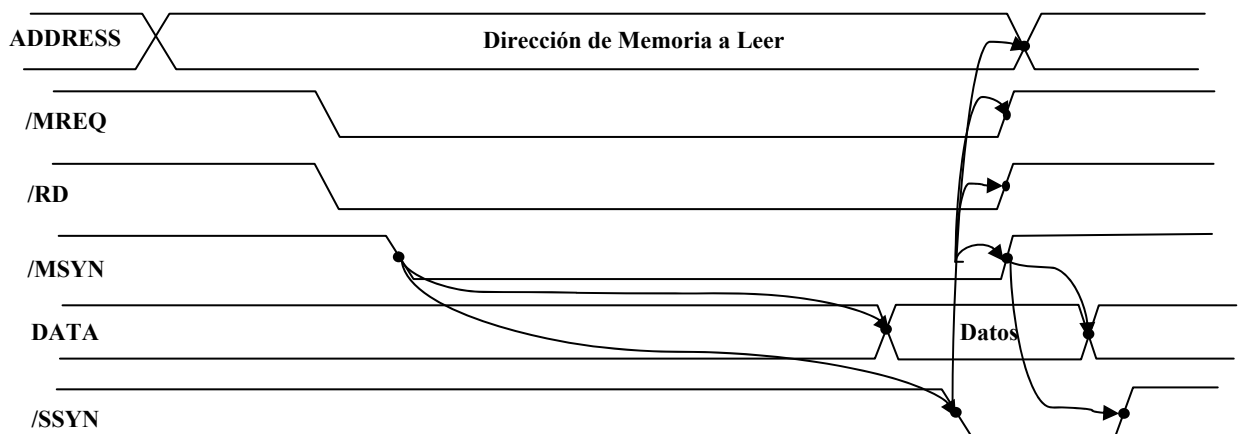


Figura 64 Bus Asíncrono

El dispositivo que controla el Bus (Master) habilita una señal de sincronización /MSYN, al ser entendida esta por el dispositivo Esclavo, este realiza su actividad y cuando termina, este genera la señal /SSYN, es en este momento cuando el Master reconoce que están disponibles los datos y los lee, deshabilitando posteriormente las señales de /MREQ, /RD y /MSYN. El Esclavo, una vez que detecta la desactivación de /MSYN desactiva /SSYN. Resumiendo:

- Activación de /MSYN
- Activación de /SSYN al reconocer /MSYN
- Desactivación de /MSYN al reconocer a /SSYN
- Desactivación de /SSYN al reconocer la desactivación de /MSYN

Este tipo de dialogo o protocolo hardware se denomina “Handshake” (apretón de manos). Algún autor lo denomina “saludo completo”.

3.2.3 Arbitraje de Bus

Cuando dos o mas dispositivos quieren acceder al Bus al mismo tiempo, se presenta un conflicto de colisión. No se permite que todos accedan en el mismo instante de tiempo. El conflicto se resuelve introduciendo un **Arbitro de Bus**, este mecanismo determinará quien de los dispositivos se lleva el control.

Los mecanismos de arbitraje pueden presentar dos estructuras:

- **Centralizado**
- **Descentralizado**

Centralizado: Este árbitro puede estar integrado en la CPU o bien ser un dispositivo diferenciado. Este árbitro determina quien es el siguiente en ser servido.

El Bus presenta una única línea de petición que consiste en un OR de todas las peticiones; no tiene manera de conocer quién es el peticionario. Cualquiera de los participantes puede pedir el Bus.

La línea de “Concesión de Bus” sale del árbitro y se encadena entre todos los participantes hasta llegar al último. La prioridad mas alta le corresponde al dispositivo mas próximo al árbitro y la mas baja al mas lejano (último). Este mecanismo se conoce como **encadenamiento circular**.

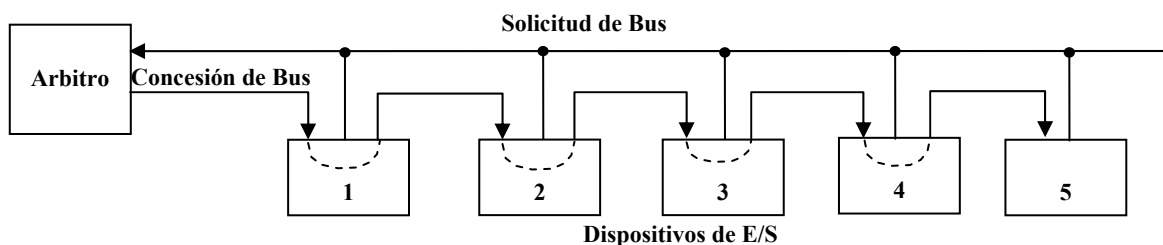


Figura 65 Bus Centralizado: Encadenamiento Circular de un Nivel

De lo anterior se desprende que si siempre pide el dispositivo mas próximo al árbitro, nunca los menos prioritarios podrán acceder al Bus. Esta situación se resuelve disponiendo de mas de una línea de petición, agrupando en ella (OR) los dispositivos de prioridad similar. De esta manera puede llegar al árbitro solicitudes diferenciadas, pudiendo dar servicio a estos dispositivos.

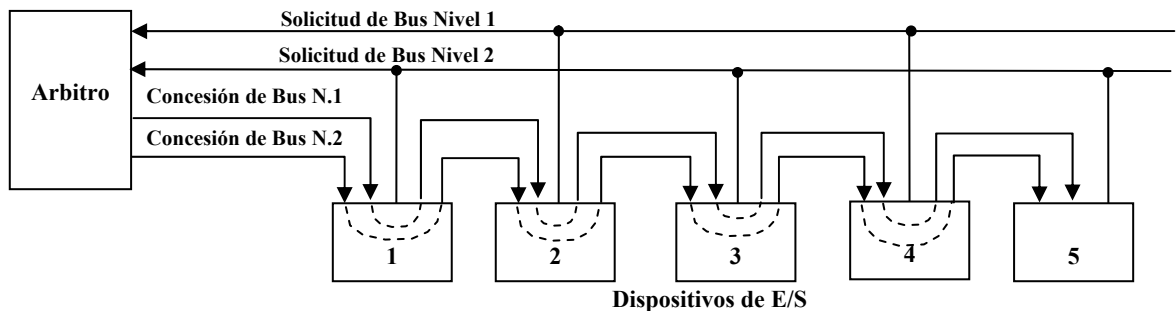


Figura 66 Bus Centralizado: Encadenamiento Circular de Dos Niveles

Mientras un dispositivo tiene la concesión del Bus (está realizando alguna transacción) no podrá darse servicio a otros dispositivos mas prioritarios. Estos detectarán que está activa la señal de “Concesión de Bus” y no activarán su señal de “Petición”. Cuando se desactive aquella, el árbitro detectará la nueva petición y volverá a activar la concesión.

Se presenta un problema si la CPU comparte el Bus de la memoria con el de E/S, si el dispositivo de E/S es de carácter prioritario, la CPU retrasará su actividad con la memoria. Para evitar este tipo de conflictos, se recurre a disponer la memoria en un Bus distinto al dedicado a E/S.

Descentralizado: Un tipo de arbitraje descentralizado puede ser el disponer de un nº elevado de líneas de solicitud en un orden de prioridad. A cada dispositivo le llegan todas las líneas de solicitud pero solo se le asigna una. Todos los dispositivos ven el estado del resto de las líneas. Esto les permite conocer si son ellos los servidos o si serán los próximos en ser servidos.

Esta estructuración requiere mas líneas de control pero evita el uso de árbitro, además limita el nº de dispositivos participantes.

Otro tipo de arbitraje descentralizado, es el que utiliza tres líneas. Una línea es un OR de las peticiones de los dispositivos, otra se corresponde con la función BUSY (ocupado) que es activada por el dispositivo que tiene concedido el Bus, la tercera y última es la señal de prioridad encadenada, de tal manera que si un dispositivo de mayor prioridad tiene concedido el Bus, esta señal pasará a los siguientes dispositivos desactivada, indicando que no pueden pedir Bus.

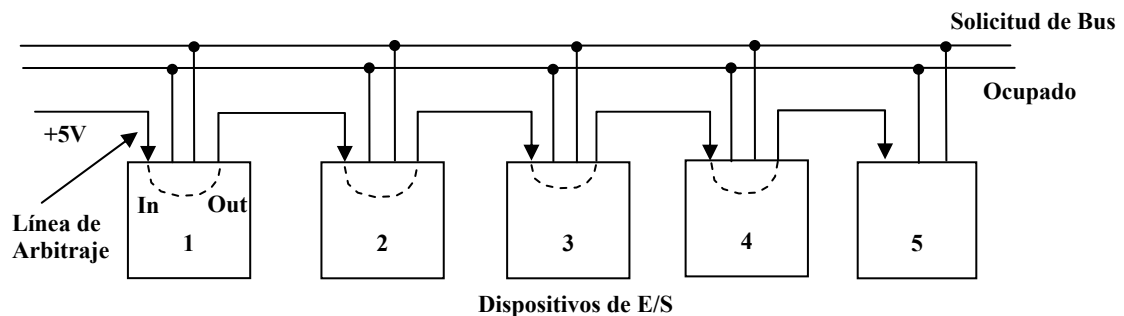


Figura 67 Bus Descentralizado: Arbitraje

En este tipo de arbitraje si un dispositivo de menos prioridad tiene concedido el Bus, habrá activado la señal de BUSY y si un dispositivo de mayor prioridad quiere activar su petición, no podrá realizar la petición porque verá esta señal activa, deberá esperar a su desactivación.

3.2.4 Operaciones de Bus

Lo que se ha visto hasta el momento, es el acceso al Bus para lectura o escritura de un Dato principalmente de Memoria. Se dan otro tipo de transacciones como las que vamos a ver a continuación:

- **Transferencia por Bloques:** Pensemos en el caso de una memoria caché, cuando hay que traer de la MP a la MC una palabra, es necesario el traer la Línea (bloque) que contiene la palabra requerida, esta línea estará formada p.ej. por 16 palabras consecutivas de n bits cada palabra. Si se organiza una transferencia por bloque, será mas eficiente que traer palabra a palabra accediendo a MP de manera consecutiva.

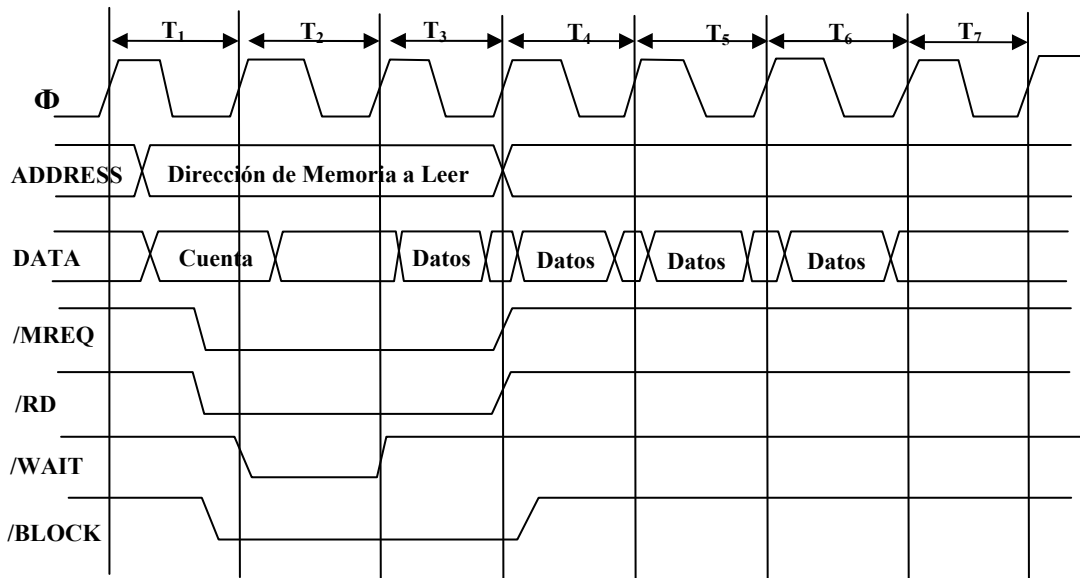


Figura 68 Transferencia de un Bloque

Mediante una señal de solicitud de transferencia de bloque, un primer dato indicando al dispositivo esclavo el nº de palabras a mover, podrá desencadenarse el acceso indicando solo la dirección de la primera palabra.

- **Acceso por Semaforización:** Se da el caso de disponer de una arquitectura multiprocesador, que comparten información de una misma área de memoria, entonces el acceso puede realizarse “escribiendo y leyendo” unos bits de control situados en la memoria RAM, de tal manera que cuando un procesador quiera acceder al área de memoria, lea primero el bit “semáforo” y si está “verde”, lo ponga “rojo” para que no sea utilizado por otro de los procesadores. Si está en “rojo”, deberá esperar a que cambie.
- **Servicio de Interrupciones:** El procesador central puede desencadenar una acción en un dispositivo periférico y en vez de escrutar constantemente a este dispositivo, puede continuar su proceso principal y entrar en un estado de “espera” que es resuelto por la llegada de una “interrupción” generada por el dispositivo periférico. La señal generada se denomina “Petición de Interrupción” (IRQx o IRx: Interrupt Request).

Cuando el procesador va a reconocer qué periférico ha provocado la interrupción (señal INTA: INTerrupt Acknowledge), se dirige a un dispositivo especial, denominado “Controlador de Interrupciones” que le devuelve en el Bus un código del dispositivo que ha provocado la interrupción. Este código es utilizado por el procesador como un “índice” que va a apuntar (Tabla de Apuntadores) a una dirección de la memoria donde se va a tratar la interrupción. Este índice recibe el nombre de “Vector de Interrupciones”.

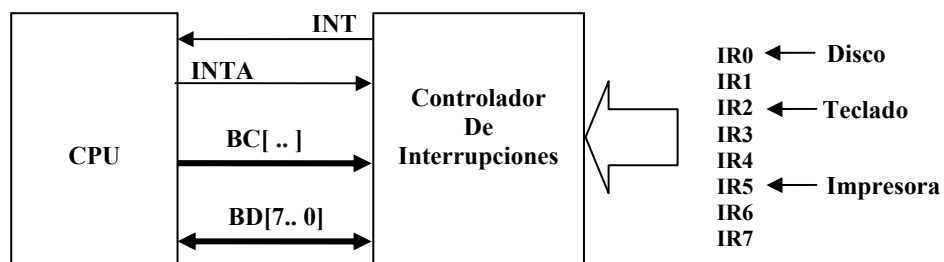


Figura 69 Controlador de Interrupciones

Para finalizar, comentar que los dispositivos controladores de interrupciones, pueden conectarse en cascada, de tal manera que pueden manejarse 8x8 interrupciones.

3.2.5 Ejemplos de Buses: ISA, PCI, VME, USB

3.2.5.1 ISA

El Bus inicial de los IBM PC está basado en el Procesador 8088 de Intel. Este Bus comprende 62 líneas de señales, con la siguiente distribución:

- 20 para la Dirección de memoria.
- 8 para los Datos.
- 4 para Lectura Memoria, Escritura Memoria, Lectura E/S y Escritura E/S.
- Señales de solicitud y Reconocimiento de Interrupciones.
- Señales para trabajo en DMA.

Estas señales discurren por la Placa Madre y van pasando por conectores insertados en ella, es en estos conectores, separados 2 cm., donde se van a insertar las tarjetas controladoras de diversos dispositivos.

Cuando se introdujo el Procesador 8086, este disponía de un Bus de Datos de 16 bits, necesitándose multiplexar el Bus de Datos al disponer solo de 8 patas para el Bus.

Al introducirse el Procesador 80286, se produjo el siguiente problema; este procesador presenta un ancho de Bus de datos de 16 bits, por lo tanto si se hubiese diseñado un nuevo Bus (nuevas tarjeta madre), las tarjetas diseñadas hasta el momento, habrían sido incompatibles con el nuevo Bus, esto hubiera provocado un cierto caos en la industria, tanto a nivel de usuario como de fabricante de tarjetas. Por ello, se implementó una extensión del Bus, creando una nueva fila de pistas y unos nuevos conectores, dejando los ya existentes sin tocar. Esto permitía utilizar las tarjetas desarrolladas hasta el momento y las de nuevo desarrollo.

Este segundo conector consta de 36 contactos, ocupados por mas líneas de Dirección (4 mas, pasando de 20 a 24), Datos, Control de Bus de Datos (8 o 16bits), Interrupciones, canales de DMA y líneas de alimentación.

En un momento determinado, (aparición del PS2) IBM decide dejar de utilizar este Bus y comienza a incorporar uno nuevo, el Microchannel. Es en este momento cuando el consorcio de fabricantes de PC compatibles y tarjetas para PC, decide normalizar el Bus existente, con ciertas adaptaciones, denominándolo ISA (Industry Standard Architecture). Básicamente es un Bus PC/AT a 8,33 MHz. Esta decisión mantuvo la compatibilidad con todo el mercado existente.

Posteriormente el Bus ISA se extendió a 32 bits con funciones adicionales, denominándose EISA (Extended ISA).

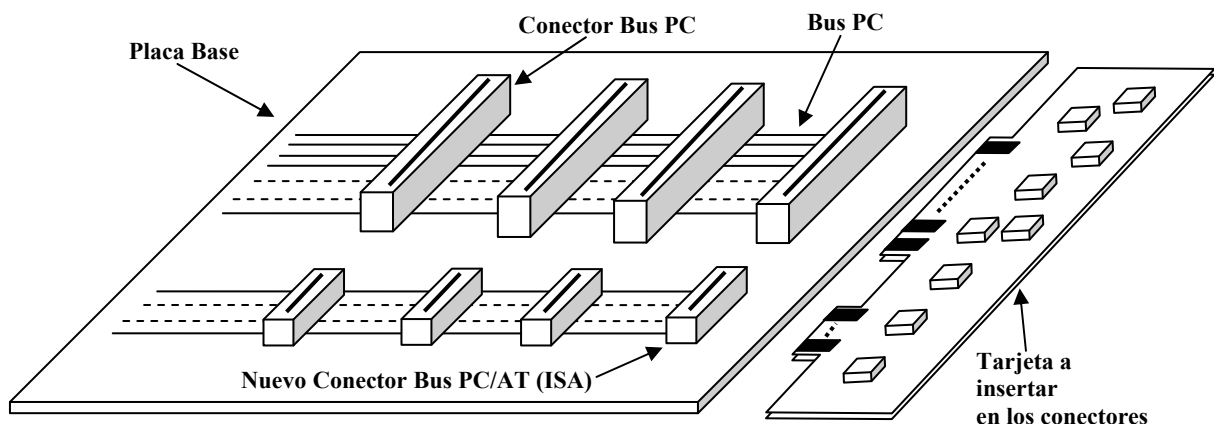


Figura 70 Placa Base con Bus PC/AT (ISA)

3.2.5.2 PCI

Debido al incremento de velocidad de los procesadores y a los mayores requerimientos de todos los equipos periféricos, había que aumentar el ancho de banda del Bus actual. Se podía detectar necesidades como las siguientes:

- 67.5 MB/s para pantallas en continuo movimiento.
- 135 MB/s para video.
- Etc..

El Bus ISA puede transferir datos a 16.7 MB/s y el EISA a 33.3 MB/s, de estos datos se concluye que era necesario el desarrollar otro Bus que pudiera cumplir con estos requerimientos. Es Intel quien en 1990, diseña el Bus PCI (Peripheral Component Interconnect). Es un Bus propietario pero de dominio público. Existe un consorcio para el control y desarrollo de equipos basados en el Bus PCI.

EL Bus PCI ha pasado por las siguientes versiones: PCI 1, PCI 2.0, PCI 2.1 y PCI 2.2. Actualmente opera a 66 MHz con un ancho de banda total de 528 MB/s.

Para continuar manteniendo la compatibilidad con “casi todo” lo existente en el mercado, Intel ideó una solución consistente en disponer de tres o mas Buses por computador. La CPU se comunica con un Bus especial para Memoria, crea un puente a PCI y este crea un puente a ISA y a controladores IDE.

Actualmente se da una circunstancia especial, debida a la evolución tecnológica de la industria del hardware, cuando se crea el Bus PCI, todos los circuitos operan a 5 Vcc. pero la evolución de la microelectrónica, incorpora tecnologías de 3,3 Vcc, por lo tanto tarjetas PCI que trabajan a esta tensión no deben conectarse a los 5 Vcc, y viceversa, por ello se introducen unas pestañas que impiden conectar tarjetas de 3,3 Vcc en 5 Vcc. También existen tarjetas universales que trabajan con ambas tensiones (conv. 5/3.3).

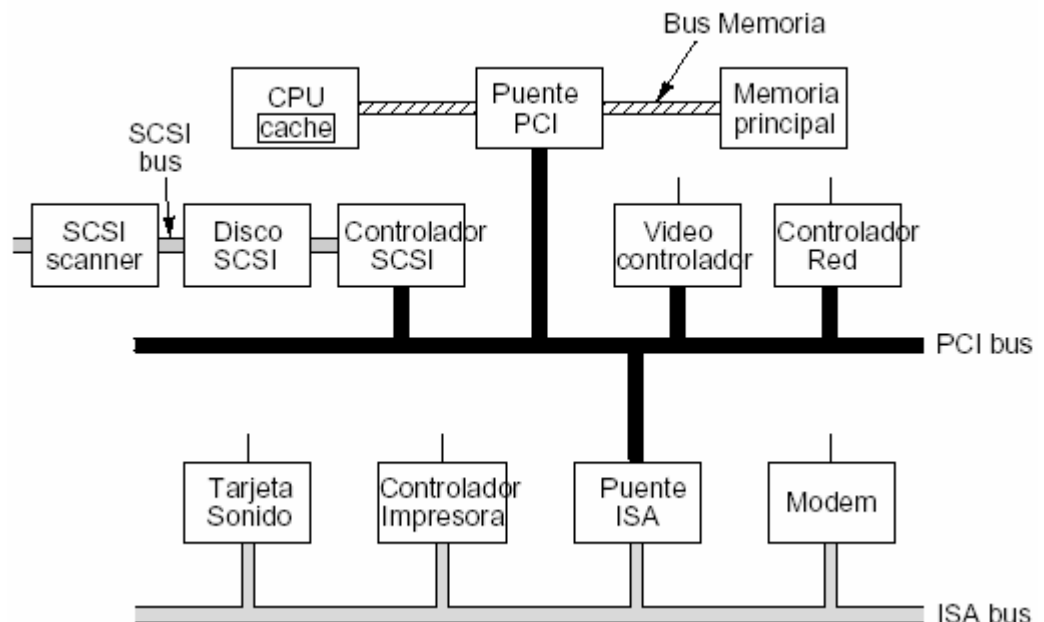


Figura 71 Interconexión de Bus PCI con ISA

Existen versiones de 32 bits (conector de 120 terminales) y de 64 bits (conector de 120 terminales y conector adicional de 64 terminales).

Las tarjetas PCI funcionan a 33 MHz, 66 MHz y frecuencias superiores. Es un Bus síncrono. Todas las transacciones se realizan entre un Maestro y un Esclavo. Las líneas de Direcciones (64) y de Datos (64), están multiplexadas.

Tratamiento de la Información: El tratamiento de la información es como sigue:

- Las transacciones se inician con el flanco de bajada del reloj, a diferencia del bus ISA que comienza con el flanco de subida.
- Durante una Lectura, en el ciclo n° 1 (T1), el Master coloca una Dirección en el Bus, en el flanco de bajada del reloj. Coloca la señal C/BE# para indicar Lectura de Palabra, Bloque, etc.; termina habilitando FRAME# para desencadenar la transacción.
- En el ciclo n° 2 (T2), se invierte el Bus para que pueda utilizarlo el Esclavo. El Master deja el Bus en Alta Impedancia. El Master modifica C/BE# para indicar que bytes de la palabra direccionada quiere leer.
- En el ciclo n° 3 (T3), el esclavo envía los datos solicitados. El Esclavo activa DEVSEL#, coloca los Datos y activa TRDY# para avisar al Master.
- Si el Esclavo es mas lento que el Master, activa DEVSEL# pero introduce ciclos de espera y activa TRDY# cuando ya está listo.
- El ciclo n° 4 (T4), no tiene efecto externo, se trata de un ciclo muerto.
- Si la transacción es de escritura, se requieren también tres ciclos de reloj, aunque no necesita inversión durante el ciclo n° 2.

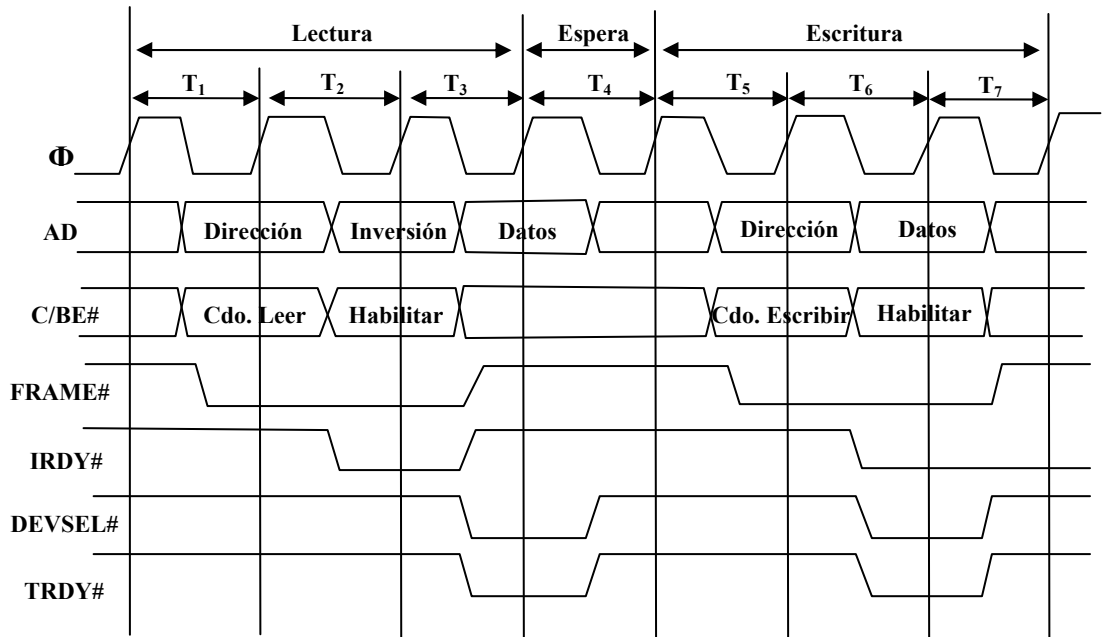


Figura 72 Ciclo de Bus PCI

Arbitraje: El problema del arbitraje se resuelve de la siguiente manera:

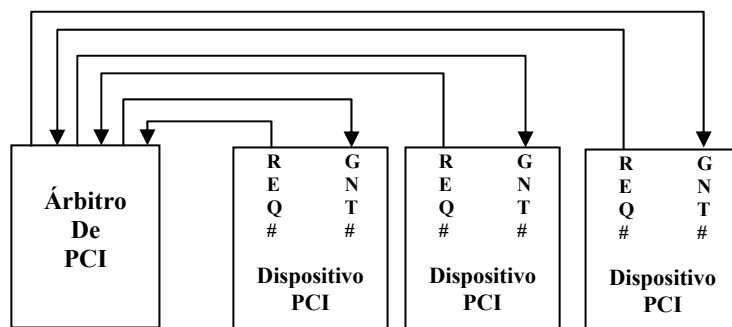


Figura 73 Arbitraje en un Bus PCI

Un dispositivo realiza una petición REQ# (REQUEST) y el Arbitro la concede GNT# (GRANT). En este momento el dispositivo puede utilizar el Bus en el siguiente ciclo. El arbitraje se resuelve de las formas ya comentadas, circular, por prioridad y otros sistemas.

El paso de un dispositivo a otro introduce siempre un “ciclo muerto”, este no se producirá solo si el que pide el Bus es el mismo dispositivo que lo tiene en su poder y quiere continuar transacciones, no existiendo ningún otro dispositivo que lo esté pidiendo en ese instante.

Si un dispositivo lleva utilizando mucho tiempo el Bus, el árbitro puede quitárselo, deshabilitando la señal GNT#, de esta forma el dispositivo se da cuenta y termina para el siguiente ciclo.

Conexión y Operación (Plug and Play): Los dispositivos PCI disponen de un área de configuración de 256 bytes. Esta área puede ser leída por otros dispositivos, activando la señal IDSEL. Este espacio de memoria contiene información sobre las propiedades del dispositivo. La capacidad de Plug and Play de algunos sistemas operativos, visitan esta zona de configuración para reconocer al dispositivo.

3.2.5.3 VME

Bus definido por varias compañías donde destaca Motorola. Se trata de una redefinición de un Bus existente, el Versabus, a las Normas Europeas, estas fundamentalmente diferían en los tamaños y en el formato de las conexiones. Su uso se da fundamentalmente en el entorno industrial.

El Bus nace por el impulso dado por la familia de microprocesadores 680xx de Motorola, esta familia pasa por los siguientes micros: 68000, 68008, 68010, 68012, 68020, 68030 y 68040.

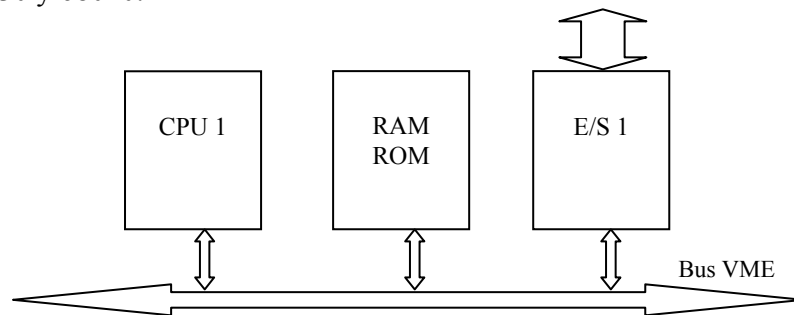


Figura 74 Bus VME

Las tarjetas normalizadas en Europa presentan dos tamaños:

- Simple Europa : 100mm x 160mm
- Doble Europa : 233,68mm x 160mm

Y su sistema de conexión se realiza por medio de conectores DIN.

La Norma define dos conectores denominados P1 y P2, ambos de 3x32 pines. La tarjeta Simple Europa lleva el P1 y la Doble Europa P1 y P2.

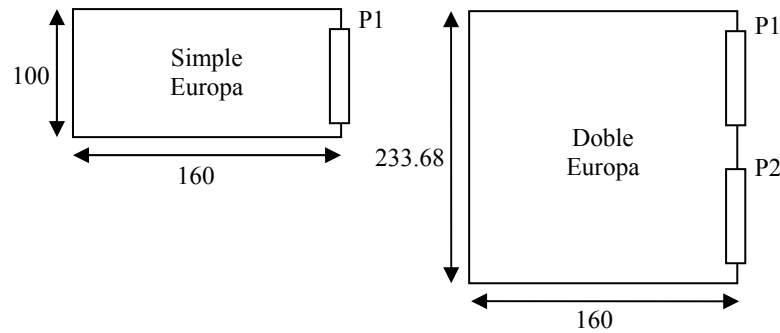


Figura 75 Tarjetas Simple y Doble Europa

El conector P1 permite trabajar con la siguiente arquitectura:

- Bus de Datos de 16 bits.
- Bus de Direcciones de 24 bits: 16 M.
- Arbitraje de Bus: 4 líneas (trabajo en modo DMA).
- Interrupciones: 8 líneas.
- Resto de señales de Control.
- Alimentaciones.

La extensión del conector permite la siguiente ampliación:

- Bus de Datos de 32 bits.
- Bus de Direcciones de 32 bits: 4 G.
- Resto de líneas de libre disposición.

3.2.5.4 USB

Con el fin de facilitar la integración de nuevos dispositivos al entorno del computador, nueve compañías, líderes mundiales del sector informático, crean un consorcio para diseñar un nuevo Bus para un nuevo modo de conexión al computador. Crean el Bus USB: Bus Serie Universal.

Las ideas iniciales del planteamiento, son las siguientes:

- a. El usuario no tendrá que actuar sobre microinterruptores en el dispositivo.
- b. El usuario no tendrá que abrir el computador para incorporar nuevos dispositivos.
- c. Solo existirá un único tipo de cable de interconexión.
- d. Los dispositivos de E/S serán alimentados desde el cable de Bus.
- e. Podrán conectarse hasta 127 dispositivos por computador.
- f. Podrán conectarse dispositivos que trabajen en tiempo real (teléfono, etc.).
- g. Los dispositivos podrán conectarse estando en funcionamiento el computador.
- h. No se necesitará reiniciar el computador cada vez que se instale un dispositivo.
- i. El Bus y sus dispositivos de E/S deberán ser de bajo costo.

La versión USB 1.0 (1.1) presenta una velocidad máxima es de 12 Mbits/s, denominada “*full-speed*” que permite un ancho de banda total del Bus USB de 1.5 MB/s (12/8), tasa suficiente para la mayoría de dispositivos periféricos. Los dispositivos mas comúnmente utilizados son: teclados, ratones, *scanners*, etc. .Dispone de otra velocidad de transmisión de 1.5 Mbits/s, denominada “*low-speed*”. El controlador detecta cual es la velocidad máxima de trabajo del periférico conectado y se adapta.

La versión USB 2.0 alcanza una velocidad máxima de 480 Mbits/s lo que permite un ancho de banda de 60 MB/s. Este modo se denomina “*high speed*”.

De un cable principal se van extrayendo las ramificaciones para los dispositivos.La conexión de un nuevo dispositivo es detectado por el controlador del Bus,

interrumpiendo al S.O y comenzando un ciclo de reconocimiento del tipo de dispositivo y su ancho de banda permitido. Si el ancho de banda es permitido por el S.O, teniendo en cuenta los dispositivos existentes en el Bus, le asigna una dirección única (1-127). La dirección “0” está asignada a los dispositivos que no han sido reconocidos todavía.

El Bus consta de cuatro hilos, dos de alimentación (5Vcc, 0V) y dos de Datos. El tráfico de información va de Controlador a Dispositivo o viceversa, nunca entre dispositivos. Cada milisegundo (1.00 ± 0.05 ms) se transmite una trama de sincronización de dispositivos.

- Las transacciones se denominan “tramas”.
- Las tramas están formadas por “paquetes”.
- Las tramas comienzan por SOF: Start Of Frame.

El bus USB reconoce cuatro clases de tramas:

- **Control:** Configuran dispositivos, emisión de órdenes y revisión del estado.
- **Isócronas:** Dispositivos de tiempo real, p.ej. altavoces. Se envían datos o reciben en tiempos precisos. No se retransmite en caso de error.
- **Volumen:** Transferencias de datos de gran volumen, como impresoras. No requiere tiempo real.
- **Interrupción:** USB no reconoce interrupciones, por lo tanto es necesario desencadenar escruciones a tiempos fijos para recoger informaciones pendientes.

Los paquetes se dividen en cuatro tipos:

- **Testigo:** Viajan del controlador al dispositivo. Son de control. Estos son SOF, IN (pide datos), OUT (indica que se envían datos) y SETUP (configuración).
- **Datos:** Paquetes DATA que transmiten hasta 64 bytes de información en ambos sentidos.
- **Saludo:** ACK (paquete anterior recibido correctamente), NAK (error en paquete – CRC), STALL (ocupado) y NYET.
- **Especiales:** PRE, ERR, SPLIT, PING, RESERVED.

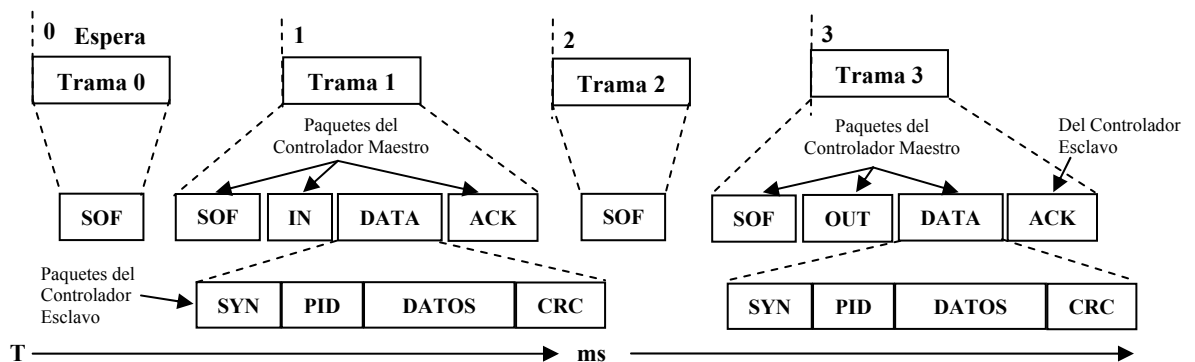


Figura 76 Secuencia de Datos en el Bus USB

SYN: Sincronización. 8 bits
 PID: *Packet Identifier Field*. 8 bits
 CRC: Código de Redundancia Cíclica

3.2.5.5 FIREWIRE

Bus serie creado por Apple Computer en 1986. Escogen el nombre debido a su velocidad. También se conoce como i.LINK en Japón. La primera especificación se completa en 1987.

Pensado para conexión multimedia de bajo coste, gran ancho de banda y tiempo real

Característica Plug and Play y conexión Peer-to-Peer (todos los participantes son Maestros)

Conexión entre ordenadores, periféricos y dispositivos de electrónica de consumo (cámaras de vídeo, cámaras digitales, impresoras, etc...). Una red puede soportar hasta 63 nodos. Conectando múltiples redes, en una súper red, puede llegarse a 1000 nodos (teóricamente).

En 1995 la organización IEEE adopta como norma las especificaciones del bus FireWire como IEEE 1394. Actualmente existe la organización 1394 TA compuesta por aproximadamente 170 compañías de todo el mundo (www.1394ta.org).

Evoluciona a 1394a (2000) y 1394b (2002).

- 1394 y 1394a: 400 Mbits/s sobre 4.5 m. Conector 4 pines
- 1394b: 1394b "High-Performance Serial Bus" 800Mb/s – 3.2 Gb/s sobre 100 m. Con CAT-5 (par trenzado) conector 6 pines y fibra óptica plástica.

El flujo de datos entre nodos es segmentado en canales síncronos y asíncronos.

- Canal asíncrono: Comandos, Funciones de Control y Ficheros.
- Canal síncrono: Audio y Video

Estos flujos de Datos son válidos para las tres versiones (1394-1394a-1394b). La versión 1394b incorpora el denominado Beta-Mode.

La incorporación de buses paralelo de 32 y 64 bits ha permitido el aumento de velocidad en los buses serie, por ello la versión “b” admite llegar a los Gigabits.

Para evitar las copias ilegales se ha desarrollado un protocolo denominado 5C (creado por las cinco industrias mas importantes). Presenta tres opciones:

- copiar una vez
- copiar libremente
- no copiar nunca

3.3 Terminales

Se denomina terminal al equipo que permite la relación de un operador con el Computador Central. Se trata del interfase entre el Hombre y la Máquina. Un terminal de ordenador consta de dos partes, teclado y monitor. En muchas ocasiones son dos elementos solidarios. La conexión se realiza vía serie bien directamente bien línea telefónica.

En el mundo del PC son dos elementos separados e independientes.

3.3.1 Monitores de Tubo de Rayos Catódicos (CRT)

Se basan en el principio de la emisión de electrones por calentamiento de un filamento, estos electrones forman un haz que atraviesa el espacio existente entre cuatro placas paralelas, dispuestas dos en orientación vertical y dos en orientación horizontal.

Si a estas placas se aplica un potencial eléctrico, el haz de electrones verá modificada su trayectoria según la polaridad de las placas. Tengamos en cuenta que la carga del electrón es negativa, por lo tanto será repelida por la placa de polaridad negativa y atraída por la positiva.

El haz de electrones incide sobre una pantalla de fósforo que presenta la propiedad de emitir luz ante la colisión de los electrones; a esta propiedad se denomina fosforescencia. En la pantalla hay una rejilla con potencial $+o-$, si es $+$ atrae al ekectrón y si $-$ lo repele.

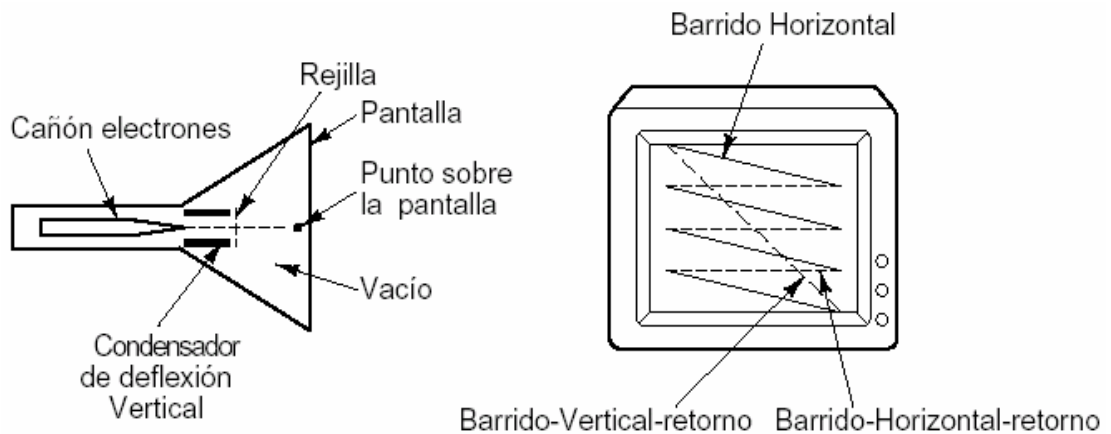


Figura 77 Tubo de Rayos Catódicos

Si se trata de un monitor a color, existen tres haces uno para el rojo, otro para el verde y otro para el azul, de esta manera se forma el color por adición de los tres colores complementarios que impactan sobre fosforo rojo-verde-azul, se denominan RGB (Red – Green – Blue).

El haz realiza un barrido horizontal y un barrido vertical.

3.3.2 Pantallas Planas LCD (TFT)

Este tipo de pantallas están basadas en la tecnología LCD (Liquid Crystal Display) Visualizador de Cristal Líquido. El origen de esta tecnología hay que situarlo en la necesidad planteada por la industria de disponer de visualizadores de bajo consumo en equipos de muy bajo consumo (tecnologías CMOS) y/o alimentados por baterías.

En la década de los sesenta se implanta esta tecnología en calculadoras y relojes digitales de pulsera, aunque tardará casi dos décadas en ser una solución industrial universal.

Los visualizadores mas comúnmente empleados estaban basados en tecnología LED (*Light Emission Diode*) Diodo Emisor de Luz, Tubos de Vacío., presentando ambos un consumo alto para lo requerido.

De la evolución de estos pequeños visualizadores, Una fila de 16 caracteres o dos filas de 16 o 20 caracteres, etc., surgen las pantallas que trabajan con puntos “píxel” al estilo de los monitores CRT.

El cristal líquido consiste en moléculas orgánicas viscosas que fluyen como un líquido pero que presentan una estructura espacial como un cristal. Esto es descubierto por el botánico Rheinitzer en 1888.

Cuando todas las moléculas están alineadas en la misma dirección, las propiedades ópticas del cristal dependen de la dirección y de la polarización de la luz incidente. Mediante un campo eléctrico, es posible modificar la orientación de las moléculas y por tanto las propiedades ópticas del cristal. Colocando una luz en la parte posterior, se puede controlar la intensidad de luz transmitida a través del cristal. Para completar el efecto, es necesario disponer en ambas caras de filtros polarizadores; uno de los filtros polariza horizontalmente y el otro verticalmente.

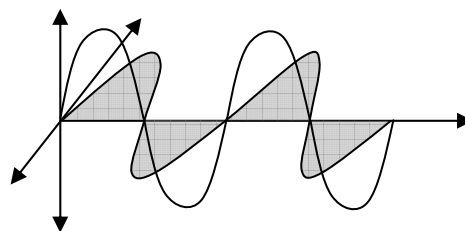


Figura 78 Vibración de la Luz en dos Planos

El cristal líquido se confina entre dos placas de vidrio transparente, en estas placas se han insertado matrices de hilos conductores transparentes, aplicando diferentes voltajes a estos hilos, se crearán campos eléctricos que crearán diferentes zonas de alineación. Colocando una fuente de luz en la parte posterior (o luz natural) veremos diferentes imágenes sobre la parte anterior.

Una tecnología muy aplicada es la TN (*Twisted Nematic*) Nemática Torcida. Consiste en la Luz, El Polarizador Horizontal, la Placa de cristal trasera con surcos Horizontales, el Cristal Líquido, la Placa de cristal delantera con surcos verticales y el Polarizador Vertical delantero.

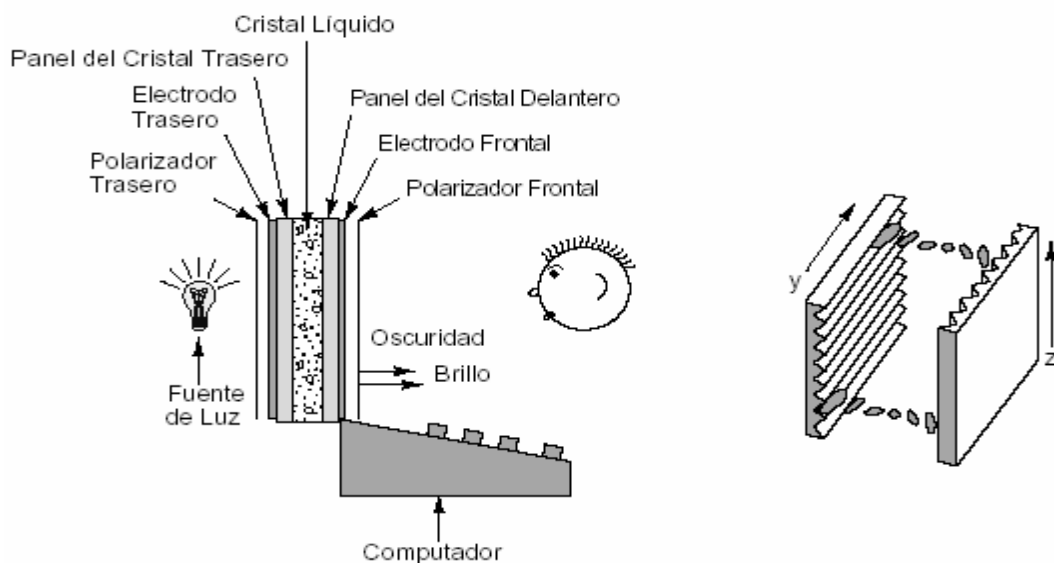


Figura 79 Pantallas Planas

Si incidiera luz en la parte posterior, debido a la posición de los polarizadores, la pantalla permanecería oscura; pero como el cristal líquido se va a alinear de surco horizontal a surco

vertical, sufre una rotación de 90°, formando una guía de luz, permitiendo el paso de la luz y mostrando la pantalla uniformemente iluminada. Es decir, en ausencia de campos eléctricos, la pantalla se verá iluminada. Si se aplican campos eléctricos a diferentes zonas de la pantalla, se rotará el camino de la onda de luz, apareciendo oscuridad en ese área.

La matriz de hilos conductores mencionada anteriormente, es la que va a dar la resolución de la pantalla en “pixels”. Si la matriz es de 640x480, tendremos el equivalente a un CRT de la misma resolución.

Aplicando un voltaje a uno de los hilos verticales y a otro de los horizontales, se habrá actuado sobre un píxel, oscureciéndolo. Si se realiza el barrido por los hilos horizontales y verticales, con una cadencia de 60 veces por segundo, el ojo tendrá la sensación de imagen estática del mismo modo a como se realiza en un CRT.

Las pantallas de color, utilizan el mismo principio pero disponen de tres filtros, rojo, verde y azul, para separar la luz blanca en cada píxel. Estos tres colores se denominan primarios por poder generar cualquier color con ellos por superposición lineal.

3.3.3 Terminal de Mapa de Caracteres

Hay tres tipos de terminales mas comúnmente utilizados:

- Mapa de Caracteres
- Mapa de Bits
- Terminales RS232C

Un computador genera una información para ser consumida en una pantalla de dos maneras diferentes: Mapa de Caracteres y Mapa de Bits.

Terminal de Mapa de Caracteres: La CPU envía caracteres (bytes) a la memoria de video. A cada carácter se asocia un byte de atributo, este atributo va a indicar al controlador de pantalla el color, intensidad, parpadeo, etc..

Como ejemplo, una pantalla de 25x80 caracteres, requiere 4000 bytes de memoria de video, 2000 para los caracteres y 2000 para los atributos. Es normal que las tarjetas controladoras de video disponga de memoria suficiente como para almacenar varias pantallas.

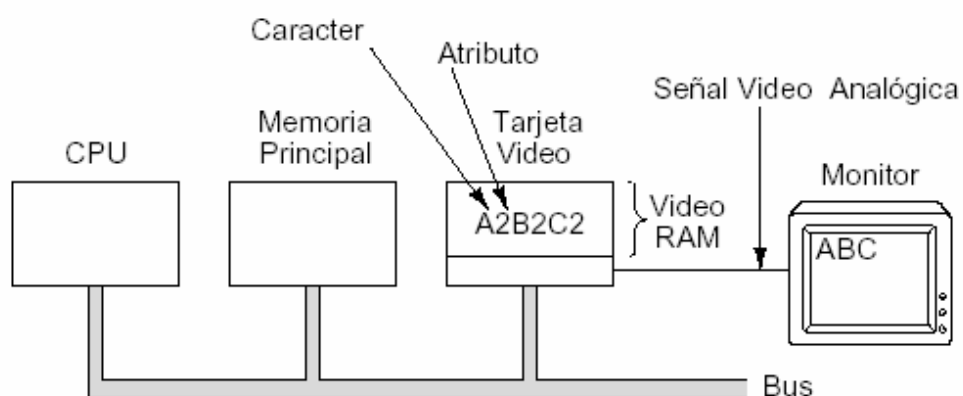


Figura 80 Terminal de Mapa de caracteres

Terminal de Mapa de Bits: La CPU ve a la pantalla como un mapa de puntos “pixels”, dividiendo la pantalla en 640x480 puntos u 800x600 o 1024x768, etc.. A cada bit se le asocia un atributo formado por 3 bytes (R-V-A) para poder definir bien el color. Por lo tanto, una pantalla de 1024x768 requiere una memoria de video de 2.3 MB por pantalla a visualizar. Como se puede realizar un Mapa de la memoria, si trabajamos en áreas bien especificadas

(ventanas), no será necesario cargar en la memoria de video toda la pantalla sino solamente el área necesaria.

Si se utiliza la pantalla para mostrar video, teniendo en cuenta que el ojo requiere 25 cuadros/s., pensando en la pantalla anterior, se requerirá una tasa de transferencia de 57.6 MB/s. Esto con un Bus EISA, no se consigue, hay que utilizar Bus PCI.

Terminal RS232C: Este tipo de conexión se creó con el fin de permitir una estandarización en las conexiones de los terminales con cualquier computador. La asociación Americana EIA (Electronics Industries Association) Asociación de Industrias Electrónicas creó este tipo de interfase.

La norma definía el tipo de conector que es un Sub D de 25 puntos. También define los niveles eléctricos de las señales transmitidas, el “1” lógico está entre -3 .. -24 Vcc, y el “0” lógico entre +3 .. +24 Vcc. Las tensiones mas normalizadas son $\pm 12Vcc$, aunque dispositivos actuales trabajan con $\pm 5Vcc$ o $\pm 9Vcc$.

La norma define dos tipos de equipos conectados, son los siguientes:

- **DTE** (Data Terminal Equipment): Equipo Terminal de Datos, va a ser el Terminal con el que estamos trabajando, intercambiando datos con el Computador o también es considerado el Computador. Las conexiones del conector presentan una determinada significación.
- **DCE** (Data Communication Equipment): Equipo de Comunicación de Datos, este equipo es el que va a unir el Terminal y el Computador a través de una línea telefónica de funcionamiento analógico.

Las conexiones del conector presentan el significado opuesto al del DTE, y está previsto de esta manera para poder conectar un cable “paralelo” sin necesidad de realizar cruzamientos de cables, p.ej.: si en DTE 2 es TxD, en DCE 2 es RxD (TxD del DCE), si en DTE 3 es RxD, en DCE 3 es TxD (RxD del DCE).

Otras señales son: RTS(4), CTS(5), DCD(8), DTR(20), DSR(6), 0V(7), GND(1), etc.

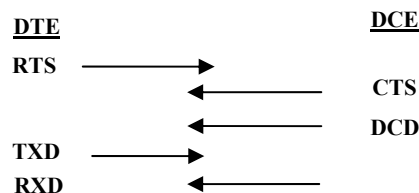


Figura 81 Señales entre DTE y DCE

El DCE va a ser denominado MODEM, acrónimo de Modulador-Demodulador.

Las señales digitales son transformadas a señales analógicas por medio del Modulador y las señales analógicas son pasadas a digitales por medio del Demodulador. Las técnicas se explicarán mas adelante.

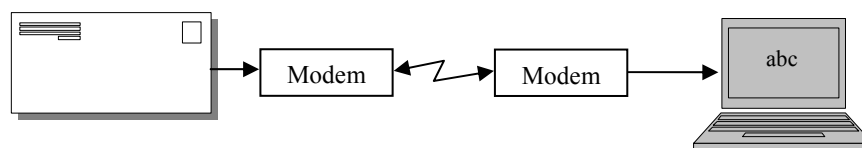


Figura 82 Conexión entre Ordenadores

Si el Terminal y el Computador están lo suficientemente cerca, no será necesario el MODEM, podrán conectarse directamente, teniendo en cuenta que habrá que cruzar los hilos de TxD y RxD (2 con 3 y 3 con 2).

Los datos son generados por la CPU y serializados por un dispositivo denominado UART o USART (Universal Synchronous Asynchronous Receiver Transmitter). Cada Byte es convertido a una ristra de bits, se trata de un conversor Paralelo/Serie - Serie/Paralelo, añadiéndose bits con el significado siguiente:

- **Bit de Start** : Comienzo de carácter, pasa de 1 a 0.
- **Bit de Paridad** : Paridad Par o Impar, Puede no haber Paridad (no hay bit).
- **Bit de Stop** : Indica fin de carácter.

3.4 Ratones

Hasta la aparición de los ratones, el usuario introducía comandos, vía teclado, para realizar acciones sobre el programa. Estos comandos eran poco amigables para un usuario no experto. Al introducir aspectos mas gráficos de cara al usuario, se introdujo este dispositivo que permitía moverse por la pantalla (aplicación) de manera mucho mas grata, realizando acciones sobre líneas de comandos de manera mas intuitiva.

La funcionalidad del ratón consiste en transmitir al computador, coordenadas (x,y) al desplazarse sobre una superficie plana. Estas coordenadas se transmiten por medio de un código serializado junto con una señal de reloj de validación de datos, el reloj lo transmite el ratón. En este código serializado también se incorpora el estado de los pulsadores que incorpora el dispositivo. El protocolo de comunicaciones es el PS/2 que admite comunicación “bidireccional” y “síncrona”. Cada byte se transmite con: “bit de Start”-“8 bits de Datos”-“bit de Paridad”-“bit de Stop”.

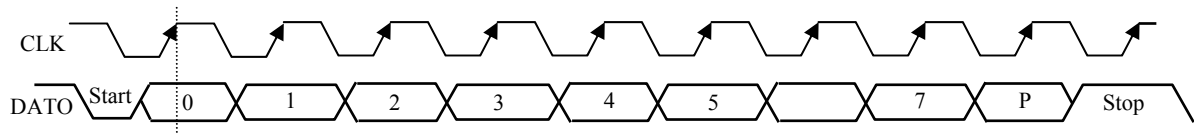


Figura 83 Cronograma de Señales en un Ratón

Cada coordenada puede ir codificada en uno o dos bytes y el estado de los pulsadores en otro byte. Esta información es presentada en pantalla en forma de flecha y la acción de los pulsadores, tendrá diferentes resultados según si es zona sensible (menú) o no (uso de botón derecho o central).

Formato de los tres bytes del mensaje								
	7	6	5	4	3	2	1	0
Byte 1	Overflow(Δy)	Overflow(Δx)	Signo(Δy)	Signo(Δx)	1	Activ.botón medio	Activ.botón derecho	Activ.botón izquierdo
Byte 2	Δx							
Byte 3	Δy							

- Overflow(Δx), Overflow(Δy): “1” si el valor de Δx o Δy excede del rango [-256,255] (Δx y Δy tomarán el valor correspondiente al límite), “0” si no hay overflow.
- Signo(Δx), Signo(Δy): “1” si el valor de Δx o Δy es negativo, “0” si es positivo.
- Activación de botones: “1” si está presionado, “0” no presionado.

Figura 84 Formato del Paquete de Datos de un Ratón

Se han desarrollado tres tecnologías:

- **Mecánicos:** Ruedas de caucho que mueven dos ejes, horizontal y vertical, estos a su vez mueven una resistencia variable. La diferencia de tensión indica el movimiento.
- **Ópticos:** Un Led emite luz y esta es reflejada sobre una superficie de cuadrícula muy fina, detectándose las transiciones en un fotodetector.
- **Optomecánicos:** Una bola de caucho mueve dos ejes, horizontal y vertical, que a su vez mueven dos ruedas con orificios en su periferia, un diodo Led emite luz y esta es detectada por un

fotodetector cuando pasa por la ventana de la rueda, la transición de luz a oscuridad y viceversa, va a codificarse como un dato proporcional al movimiento.

3.5 Impresoras

Periférico sobre el que se va a volcar determinada información. Esta información podrá provenir del nivel del Sistema Operativo o bien del nivel del programa de usuario.

Las impresoras pueden ser Monocromáticas o de Color.

3.5.1 Impresoras Monocromáticas: Dentro de este apartado, se presentan distintas tecnologías:

- **Matriciales:** El mecanismo de impresión consiste en una cabeza móvil que consta de entre 7 y 24 agujas que forman el núcleo de un electroimán. Actúan por excitación del electroimán. Con las cabezas de 7 agujas, los caracteres se componen con una matriz de 5x7 puntos. Las impresoras se suelen presentar en dos formatos 80 y 120 caracteres por línea; existen otros formatos más pequeños pero no tienen usos ofimáticos, normalmente en transacciones comerciales con el gran público. Su velocidad se mide en caracteres por segundo.

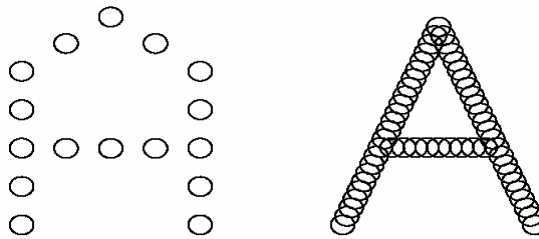


Figura 85 Muestras de impresión por Agujas

La impresión se consigue por existir una cinta entintada entre las agujas y el papel, produciéndose la transferencia de tinta por impacto de la aguja. El aumento de la calidad de la impresión suele consistir en realizar dos pasadas, además de superponer puntos entre los puntos impresos en la primera pasada. Esto provoca una operación mas lenta de este periférico.

Ventajas: Son muy económicas en su mantenimiento. Permiten realizar copias por el uso de papel de carbón. Abanico amplio de precios (para gran público y empresarial).

Inconvenientes: Son lentas y su uso está orientado a la impresión de caracteres. Los gráficos son de baja calidad. Ruidosas.

En impresoras de pequeño formato, las impresoras matriciales, han sido substituidas por las de tecnología térmica.

- **Inyección de Tinta:** Consiste en una cabeza móvil provista de un cartucho de tinta negra, la impresión se produce por expulsión de gotas, a través de boquillas o conductos, de tinta sobre el papel.

La técnica consiste introducir una gota de tinta en una cámara de expansión, calentar la gota de tinta, llevándola al punto de ebullición, en este momento explota (burbuja generada) y sale expulsada por la boquilla. Posteriormente, el enfriamiento provoca un vacío que absorbe otra gota de tinta y así continúa el proceso.

Las densidades de impresión mas comunes pueden ser 300 – 720 – 1440 dpi (dots per inch). La velocidad está supeditada al ciclo “ebullición/enfriamiento”. La calidad va a depender de la cantidad de tinta expulsada por el inyector, llegándose a p.ej. 5picolitros (5×10^{-12} litros).

Ventajas: Impresión de calidad. Gráficos de calidad. Silenciosas. Abanico amplio de precios (para gran público y empresarial).

Inconvenientes: Caras en su mantenimiento, los cartuchos son caros. Lentas.

- **Láser:** Básicamente consiste en transferir una línea completa de puntos (ancho de la página) al papel. Básicamente es la misma tecnología que las de las fotocopiadoras y facsímiles.

La descripción completa del proceso es la siguiente:

- Un cilindro de precisión es cargado electrostáticamente con una tensión entorno a los 1000Vcc.
- Un haz de luz láser incide sobre el rodillo, recorriéndolo horizontalmente. Si se modula el haz de luz, de tal manera que sobre unas zonas incide mas luz que en otras, ahí donde incide luz, pierde carga eléctrica. Esta operación crea una imagen eléctrica de la línea original.
- El haz de luz es guiado por un espejo octogonal giratorio.
- Posteriormente al girar el cilindro y pasar por la zona donde existe un polvo negro ionizable (toner), se adhiere a las áreas cargadas. Esta operación crea una imagen real de la línea original.
- Posteriormente el cilindro pasa por el papel, presionándolo, y le transfiere el polvo negro.
- El papel pasa por unos rodillos calientes que fijan (funden) el polvo al papel de manera permanente.
- Una vez que el rodillo abandona el papel, pasa por un descargador de corriente electrostática.

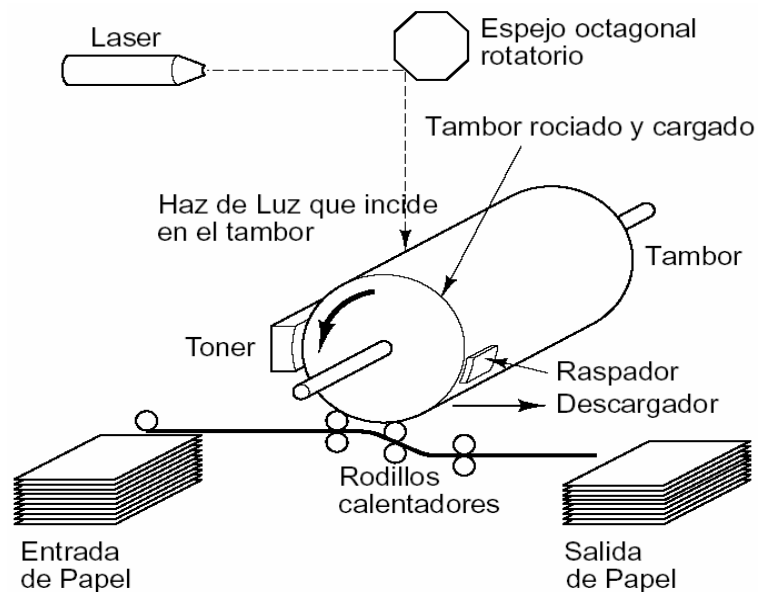


Figura 86 Esquema de una Impresora Láser

Las impresoras láser incorporan CPU y memoria en el orden de los MB para poder capturar una imagen completa (mapa de bits) de cada página y distintos tipos de letras enviados por el computador. Incorporan, además, distintos tipos de letras.

Aceptan comandos especiales para el manejo de las páginas, como por ejemplo los lenguajes PCL de HP y PostScript de Adobe.

Si se quiere imprimir imágenes, y estas presentan matices de grises, hay que recurrir a una argucia técnica similar a la empleada en los carteles comerciales. Esta se basa en el empleo de los medios tonos. El matiz de gris se consigue por cambiar la densidad de puntos negros en una matriz, cuantos menos puntos negros mas clara la percibirá el ojo humano, cuantos mas puntos negros mas oscura se percibirá.

De esta manera, la imagen se trabaja por zonas, las zonas se trabajan por matrices de 6x6, insertando celdas oscuras desde al centro hacia la periferia, esto reduce la densidad de la impresora ya que si tenemos una densidad de 600 dpi, al utilizar 6 celdas por zona, la densidad útil será 100 dpi.

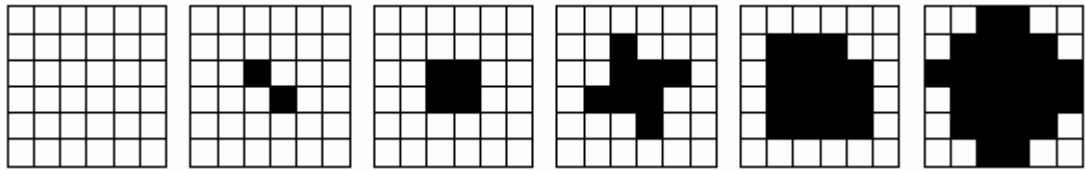


Figura 87 Densidad de puntos para escala de Grises

Ventajas: Impresión de alta calidad. Gráficos de alta calidad. Silenciosas. Relativamente rápidas. Alta fiabilidad. Precios adecuados para el entorno empresarial.

Inconvenientes: Caras en su mantenimiento, los cartuchos son caros. Caras para el gran público.

3.5.2 Impresoras de Color: En este tipo de impresoras, el color se crea de forma “sustractiva”, al contrario que en los CRT (aditivo: RGB). Los colores básicos son el Cian, Magenta y Amarillo (CMY). Además se incorpora un cuarto color, el Negro (K: black) porque con la aportación total de CMY (absorción total de los colores), no se consigue un verdadero negro. Esta impresión se denomina CYMK.

Dentro de este apartado, se presentan distintas tecnologías:

- **Inyección de Tinta (CYMB):** El funcionamiento es idéntico a las monocromáticas, solo que incorporan cuatro cartuchos.

Ventajas Generales: Impresión de calidad. Gráficos de calidad. Calidad fotográfica aceptable. Silenciosas. Abanico amplio de precios (para gran público y empresarial).

Inconvenientes Generales: Caras en su mantenimiento, los cartuchos son caros. Lentas.

Hay dos tipos de tintas con las que se obtiene una impresión de calidad:

- **Tinta de Colorantes:** Tintas basadas en colorantes, consiste en tintes disueltos en un portador.

Ventajas: Impresión de calidad. Colores brillantes. Silenciosas. Abanico amplio de precios (para gran público y empresarial).

Inconvenientes: Se altera el color por exposición a la luz ultravioleta (sol). Caras en su mantenimiento, los cartuchos son caros. Lentas.

- **Pigmentos:** Tintas basadas en pigmentos, consiste en partículas de pigmentos, suspendidas en un portador; este se evapora en el papel, dejando como residuo el pigmento.

Ventajas: Impresión de calidad. No se altera el color por exposición a la luz ultravioleta (sol). Buena calidad fotográfica. Silenciosas. Abanico amplio de precios (para gran público y empresarial).

Inconvenientes: Colores menos brillantes que la anterior. Los pigmentos tienden a taponar las boquillas. Requieren un papel especial para fotografía (retenedor de pigmentos). Caras en su mantenimiento, los cartuchos son caros. Lentas.

- ❑ **Tinta Sólida:** Formada por cuatro bloques de cera de tinta, que es fundida para ser introducida en un depósito de tinta caliente. El tiempo de preparación de la impresora está en torno a los 10 minutos.

La tinta se rocía sobre el papel, solidificándose y fusionándose con el papel al pasar entre dos rodillos que la comprimen.

- ❑ **Láser:** Funcionamiento idéntico a la monocromática, solo que incorpora las tres deposiciones necesarias para el C, M y Y. Además dispone de la K. Necesita cuatro toner distintos.

Requieren de una gran cantidad de memoria para poder almacenar una página, esto las hace más costosas, por lo demás su calidad es alta.

- ❑ **Cera:** Disponen de una cinta ancha de cera que contiene los cuatro colores. La cinta de cera está segmentada en bandas. La anchura de la banda es la de la hoja.

Hay miles de elementos calefactores que funden la cera y la depositan sobre el papel a su paso. Estas deposiciones conforman los píxel.

Este tipo de impresoras están siendo substituidas por las anteriores.

- ❑ **Sublimación:** Por sublimación se entiende el paso de “fase sólida a fase gas” sin pasar por fase líquida. Un portador de los cuatro colores CMYK pasa por una cabeza térmica que calienta los colorantes y estos pasan a fase gas siendo absorbidos por el papel.

Cada elemento calefactor puede producir 256 temperaturas distintas, cuanto más alta es la temperatura, más colorante se deposita y más intenso será el color. A diferencia de las anteriores, no es necesaria la técnica de los medios tonos por ser casi posible obtener colores continuos por cada píxel.

Esta es la tecnología utilizada para las impresoras fotográficas de calidad. El papel es caro.

3.6 Modems

Como se ha comentado anteriormente, un MODEM es un dispositivo que sirve para poner en comunicación dos equipos separados físicamente por una gran distancia. La conexión se apoya en la línea telefónica.

Un computador, a través de su conexión RS 232 C, transmite y recibe datos de manera digital, bien por cambio de tensión de p.ej -12Vcc a +12Vcc (“1” a “0”) o bien 0Vcc a 5Vcc (“0” a “1”). Si se transmitiesen estas señales directamente por la línea telefónica, estas sufrirían una gran distorsión, debido a la resistencia, capacidad y autoinducción de las líneas.

Si se transmite una onda senoidal, esta sufre una muy baja distorsión si nos movemos entre 1000 Hz y 3000 Hz. (aproximadamente). Esta señal va a recibir el nombre de “Portadora”. Una onda senoidal dada una frecuencia en el rango de las anteriores, no transmite información alguna. En cambio si alteramos esta onda en **frecuencia**, **amplitud** o **fase**, sí se podrá obtener información de estos cambios.

Este proceso se conoce como “**Modulación**”. Veamos como se efectúa este proceso:

- **Modulación en Amplitud:** Se emplean dos niveles de voltaje, el nivel alto va a indicar un “1” y el nivel bajo un “0”.
- **Modulación en Frecuencia:** El nivel de voltaje es constante durante toda la transmisión. Para diferenciar “0” de “1”, hay transición de frecuencia baja a frecuencia alta. Esta técnica se conoce por Modulación por Desplazamiento de Frecuencia FSK (Frequency Shift Keying).
- **Modulación en Fase:** La amplitud y la frecuencia no cambian, cambia la fase de la portadora, esta se invierte 180° cuando se pasa de “1” a “0” o viceversa.

Fijándonos en esta técnica, si se consigue realizar cambios de fase en ángulos diferentes a 180°, podrían transmitirse mas bits de información por unidad de tiempo indivisible. Si se desplazara la fase de la portadora de manera abrupta, en 45, 135 (45+90), 225 (135+90) o 315 (225+90), se podrían transmitir dos bits por cada transición, es decir, 45° “00”, 135° “01”, 225° “10” y 315° “11”. Esto se conoce como “codificación por fase **dibit**”. Hay técnicas que aumentan el nº de bits por periodo.

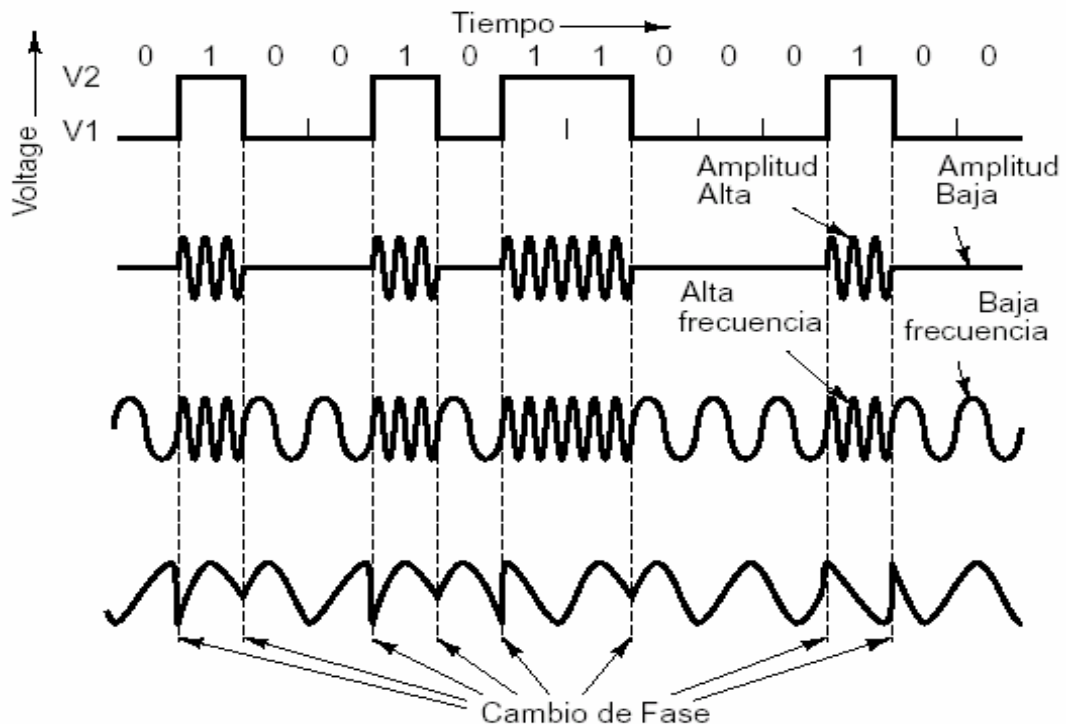


Figura 88 Modulación en Amplitud – Frecuencia - Fase

La velocidad de comunicación se mide en “baudios” (baud rate), e indica el nº de “símbolos” que se envía en la unidad de tiempo, estos símbolos pueden representar un único bit (caso en el que coinciden bits por seg. con los baudios) o varios bits donde ya es diferente la tasa de bits por segundo de los baudios.

El equipo que realiza estas modulaciones es el MODEM, que a su vez, detecta en su recepción esta modulación y “Demodula” la señal, obteniendo la información digital. La información enviada, normalmente se codifica en 8 bits por carácter mas un bit de comienzo “Start” y un bit de paro “Stop”, además puede haber un bit de paridad “Parity” (par o impar). En sistemas más antiguos (lentos), el nº de bits de Stop podía ser de 2 o 1 ½.

Las velocidades de modems, sin modulación de fase dividida, mas comúnmente manejadas son: 150 – 300 – 600 – 1200 – 2400 – 4800 – 9600 – 19200 – 38400 bits/s. Los modems que manejan velocidades superiores, p.ej. 57600 bps, emplean modulaciones combinadas.

Los equipos, según su capacidad de transmisión y/o recepción simultanea, se dividen en tres categorías:

- **Full Duplex** : Transmiten y reciben simultáneamente.
- **Half Duplex** : Transmiten o Reciben, pero no simultáneamente.
- **Simplex** : Solo transmiten o solo reciben.

3.7 Código de Caracteres

Por código de caracteres se entiende la correspondencia que existe entre el símbolo que queremos representar y el nº entero que podemos asignar en la máquina. Si nuestro computador va a trabajar con nuestros propios computadores o periféricos, podremos utilizar un código de caracteres inventado por nosotros, pero normalmente necesitamos conectarlos y trabajar con equipos diferentes. Por ello se hace necesario que exista una normalización en cuanto a la representación de los caracteres comúnmente utilizados.

Para resolver esta normalización, se crean los códigos ASCII y UNICODE.

3.7.1 ASCII

El significado del acrónimo es, Código Estándar Americano para Intercambio de Información (American Standard Code for Information Interchange).

El código ASCII inicial, representa 128 caracteres sobre 7 bits. Se representa en Octal, Hexadecimal y en Decimal. Podemos dividir los caracteres en dos grupos, según su significado:

- **Caracteres de Representación Gráfica:** Son todos aquellos símbolos que representan letras mayúsculas o minúsculas, números, símbolos de operación algebraica, etc., son los caracteres imprimibles. P.ej., la “A” se representa por 41H, “+” se representa por 2BH, etc..
- **Caracteres de Control:** Son aquellos símbolos que indican a los computadores el comienzo de una acción, el resultado de ella, etc. .P.ej., SOH (Start Of Head) significa “Comienzo de Cabecera”, EOT (End Of Transmisión) significa “Fin de Transmisión”, ACK, NACK, etc..

Posteriormente se amplió la representación de 7 bits a 8, denominándose ASCII extendido, pudiendo representarse 256 símbolos. Ya se permiten caracteres de determinadas lenguas, algunos códigos gráficos para poder realizar marcos de ventanas sobre la pantalla o impresora, etc. Esta extensión se denominó **Latín-1**, norma IS 646. Posteriormente y sobre este mismo marco de 256 códigos, se crean normalizaciones para diferentes idiomas, con el inconveniente de tener diferente significado el código según el idioma.

A continuación se representa el código ASCII en forma Hexadecimal y Decimal.

▪ **HEXADECIMAL - Character**

00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A NL	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 \$	25 %	26 &	27 '
28 (29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C	7D }	7E ~	7F DEL

▪ **DECIMAL - Character**

0 NUL	1 SOH	2 STX	3 ETX	4 EOT	5 ENQ	6 ACK	7 BEL
8 BS	9 HT	10 NL	11 VT	12 NP	13 CR	14 SO	15 SI
16 DLE	17 DC1	18 DC2	19 DC3	20 DC4	21 NAK	22 SYN	23 ETB
24 CAN	25 EM	26 SUB	27 ESC	28 FS	29 GS	30 RS	31 US
32 SP	33 !	34 "	35 #	36 \$	37 %	38 &	39 '
40 (41)	42 *	43 +	44 ,	45 -	46 .	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7
56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [92 \	93]	94 ^	95 _
96 `	97 a	98 b	99 c	100 d	101 e	102 f	103 g
104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w
120 x	121 y	122 z	123 {	124	125 }	126 ~	127 DEL

3.7.2 UNICODE

La realización de la extensión del código ASCII, no era suficiente para incorporar todos los idiomas que se sumaban al mundo de la informática. Es por ello que se forma un consorcio de compañías informáticas y crean el estándar IS 10646 que se corresponde con el código UNICODE. Este estándar ha ido pasando por distintas versiones estando en estos momentos en la 4.0.1.

La representación es más ambiciosa y se realiza en 16 bits, lo que permite incorporar 65,536 códigos, donde los 256 primeros se corresponden con el ASCII extendido (Latín-1) facilitando la conversión entre códigos.

UNICODE asigna un único código a cada símbolo existente, de esta manera se evita el que el computador o el usuario, tenga que cambiar la página de código según la situación. Esta asignación se denomina **Punto de Código**. Además, las letras con signos “diacríticos”, como p.ej. en Alemán “ü”, diferencia la letra del signo y deja al software que realice la composición de caracteres como este o nuevos que deban ser incorporados.

La estructuración de este código es como sigue:

- Cada alfabeto internacional, dispone de una zona consecutiva, p.ej. Latin (336), griego (144), etc. . Cada alfabeto tiene asignados mas puntos que los necesarios. Caso de mayúsculas y minúsculas, etc.
- Zona de código a signos diacríticos (112).
- Zona de puntuación (112).
- Zona de subíndices y superíndices (48).

- Zona de símbolos de divisas (48).
- Zona de símbolos matemáticos (256).
- Zona de formas geométricas (96).
- Zona de ornamentos tipográficos (192).

Además de estas zonas hay otras dedicadas a los símbolos del Chino, Japonés y Coreano:

- Zona de 1024 símbolos fonéticos (Katakana y bopomofo).
- Zona de 20,992 ideogramas “Han” unificados (Chino y Japonés).
- Zona de 11,156 sílabas Hangul Coreanas

Existe otra zona de uso general:

- Zona de 6400 códigos de uso local

Nota: Bopomofo ocupa de 3100H – 312FH, son 37 símbolos derivados de caracteres Chinos. Símbolos que representan sonidos del Mandarín en unión con caracteres Chinos.

TEMA 4: LENGUAJE MÁQUINA Y ENSAMBLADOR

4.1 Introducción

Un procesador entiende solamente códigos escritos como ristas de "Unos" y "Ceros", trabaja en el nivel mas bajo, es el nivel "Máquina". Un programador no puede plantearse el conocer los códigos máquina que componen todo el repertorio de instrucciones de un procesador. Por ello se recurre a crear un lenguaje "Mnemónico" que nos acerque al significado final de la instrucción. Este primer lenguaje mnemónico se denomina "Lenguaje Ensamblador (*assembly language*)". Representa "Simbólicamente" la codificación binaria de la máquina (*machine language*)

Se trata del primer nivel de acercamiento al lenguaje máquina y al programa escrito en este código se denomina "Programa Fuente". Se trata de un lenguaje de "bajo nivel" a diferencia de p.ej. "C" que es un lenguaje de "alto nivel", pero que en ambos casos, los programas escritos en estos códigos, se denominan "Fuente". Este programa no es directamente entendible (interpretable) por el procesador.

Cuando se escribe un programa en ensamblador, este tiene una significación directa con respeto al código máquina pero esto necesita de una traducción previa a lenguaje máquina. La traducción la realiza un programa denominado "Ensamblador (*assembler*)". Es decir, este programa realiza una lectura del mnemónico y genera su significado máquina en forma de "unos" y "ceros" (expresión binaria, octal o hexadecimal).

El programa generado por esta traducción, se denomina "Programa Objeto" o "Programa Ejecutable".

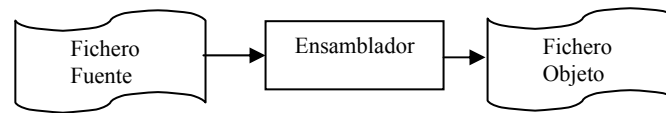


Figura 89 Secuencia de tratamiento

Es importante resaltar la diferencia entre "interprete" y "traductor". Un programa interprete ejecuta las instrucciones según las va leyendo, es decir, va interpretando el código leído y va generando los comandos adecuados para el procesador. Esto en un principio sucedía con los primeros lenguajes Basic. Actualmente este nivel también se da con Java.

Un programa traductor, realiza una o varias lecturas del programa hasta que genera una información coherente de cara al procesador, en este momento genera el programa objeto que puede ser cargado en la memoria principal del computador y ser ejecutada directamente por el procesador. Un programa escrito en lenguaje Ensamblador, Fortran, Pascal, Algol, Basic (últimas generaciones), C, C++, Ada, etc., será traducido y generado su ejecutable.

El traductor de un lenguaje mnemónico, como el lenguaje ensamblador, se denomina "Ensamblador". El traductor de un lenguaje de alto nivel se denomina "Compilador". En ocasiones los Compiladores pueden generar un programa objeto mnemónico.

4.2 Lenguaje Ensamblador

Como se ha comentado, este lenguaje es la expresión mnemónica de un código máquina escrito en binario, octal o hexadecimal. Su significación es directa, por lo tanto podríamos escribir en p.ej. hexadecimal directamente, pero esto exige un nivel de abstracción muy fuerte y nos haría perder la visión de lo que escribimos, además de ser muy dudoso que podamos aprender de memoria todos o casi todos los códigos.

Parece claro que es más fácil recordar que sumar es ADD que 24H, o que la instrucción de salto es JMP que su código 80H.

Otro aspecto importante es que si el programa debe realizar el salto a una dirección, resulta muy complicado el conocer cual es la dirección en términos absolutos y su expresión en hexadecimal. Por ello es mejor recurrir a una expresión simbólica de la dirección, denominada “Etiqueta”, mediante la cual podamos reconocer los puntos de programa donde realiza el salto.

Es el Ensamblador quien realiza la interpretación de los códigos mnemónicos y los convierte a código máquina, además reconoce las etiquetas y las asigna un valor binario correspondiente a la dirección física de la memoria donde se realizará el salto.

Un aspecto negativo del lenguaje ensamblador es su baja portabilidad entre máquinas, un procesador admite solamente su propio ensamblador. Si pretendemos llevar (instalar) un programa realizado para el ensamblador de una máquina hacia otra, nos encontraremos que no va a funcionar.

Esta portabilidad normalmente se da entre procesadores de la misma familia pero en sentido ascendente, es decir, el ensamblador realizado para el procesador mas bajo de la familia funcionará en los procesadores más altos (mayores prestaciones) pero no a la inversa. Esto es así porque cada nueva generación de procesadores incorpora nuevas instrucciones orientadas a mejorar el rendimiento con las nuevas prestaciones.

El problema de la portabilidad está mejor resuelto con los lenguajes de alto nivel como C (estándar) o Java, ya que el compilador (o traductor en caso de Java) va a realizar la adaptación al código máquina específico (o a la *Java Virtual Machine* caso de Java).

4.2.1 Uso del Lenguaje

Un punto adicional y que suele tener sus controversias es el de si un programa escrito en ensamblador es mas eficiente que uno escrito en p.ej. C. Normalmente se aduce que en ensamblador podemos acudir a bits de status del procesador y que esto no es posible en alto nivel. Además, se dice que al realizar la compilación de un programa en alto nivel, el compilador genera mas instrucciones que las necesarias (las que se supone que escribiría un experto programador). Esto era así en algunos compiladores, hoy en día los compiladores tienen la opción de optimización del código, de tal manera que podrían ser mas eficaces que si lo realizara un programador experto, además y volviendo a la primera crítica, el repertorio de instrucciones permiten acceder a registros internos de la máquina.

Resulta evidente que si se crearon los lenguajes de alto nivel fue para poder facilitar la expresión de conceptos y desarrollos de programa de cara al programador y que terceros programadores pudieran interpretar mejor lo que implementaba el programa. Por lo tanto, si es mas fácil programar en alto nivel y resulta que hay compiladores optimizadores de código ¿porqué debemos seguir utilizando el lenguaje ensamblador?.

Pues bien, la respuesta está en volver a interpretar el primer párrafo de este apartado. Hemos dicho que podemos tener acceso a registros internos y que el compilador optimiza, pues bien si tenemos dudas o es claro que esto no se corresponde con nuestra realidad y necesitamos una optimización eficaz, será necesario que esa parte de código crítica, se realice en el ensamblador de la máquina.

Como criterios de revisión de un programa escrito en alto nivel, puede apuntarse el someter al programa a un analizador de tareas/tiempo (*profiling*) durante un periodo largo y revisar que tareas del programa se ejecutan mas veces y si son las que consumen mas tiempo de CPU que otras. En este caso podría plantearse el rescribir esa parte de código en ensamblador, evaluando previamente el tiempo necesario para llevarlo a cabo. Si este tiempo no fuera asumible por costos o tiempo, no deberá realizarse ningún cambio.

Por concluir este tipo de disquisiciones comentar que los mayores apologistas del uso del ensamblador como recurso de optimización, dentro de un programa de alto nivel, se dieron a

finales de los 60 y principios de los 70. Realizaron estudios de comparación y optimización llegando a conclusiones lapidarias en favor del uso del ensamblador. Téngase en cuenta que se está trabajando con Fortran y comienzos de Algol y Pascal. El desarrollo de nuevos compiladores redujo dramáticamente estas diferencias.

Por último decir que desde un punto de vista académico, resulta IMPRESCINDIBLE el iniciarse en el uso del lenguaje ensamblador por tener mejor vivenciada la arquitectura interna de un procesador.

4.2.2 Formato de las Instrucciones

Se ha comentado que los ensambladores de los diferentes procesadores no son compatibles, pero esto no implica que no tengan un parecido muy grande. Pueden diferir los mnemónicos en alguna letra, aparecer nuevos mnemónicos, disponer de operandos ampliados, etc, pero seguirán teniendo un gran parecido.

Fundamentalmente la estructura que presenta un programa en lenguaje ensamblador es la siguiente:

Etiqueta	Cod. Operación	Operandos	Comentarios
ACCIÓN:	MOV	A, #55H	;Mueve valor inmediato
	ADI	#AAH	;Suma val. Inm. a reg. A
	JZ	ACCION	;Instr. de Salto si Z=1
	ADI	#01H	;Suma val. Inm. a reg. A
	JZ	ACCION	;Instr. de Salto si Z=1
VARIABLE	DW	#AA55H	;Reserva espacio a este valor

Tabla 8 Estructura de un programa en Lenguaje Ensamblador con un µP8085

Este ejemplo está expresado en instrucciones del microprocesador 8085 de Intel.

Etiqueta	Cod. Operación	Operandos	Comentarios
ACCIÓN:	MOV	AL,55H	;Mueve valor inmediato a AL
	ADD	AL,AAH	;Suma val. Inm. a reg. AL
	JZ	ACCION	;Instr. de Salto si Z=1
	ADD	AL,1	;Suma val. Inm. a reg. AL
	JZ	ACCION	;Instr. de Salto si Z=1
VARIABLE	DW	AA55H	;Reserva espacio a este valor

Tabla 9 Ejemplo con un µP8088, 8086, ..., 80846

Este ejemplo está expresado en instrucciones del microprocesador 8088, 8086, 80286, 80386, 80486, Pentium, Pentium Pro y II de Intel.

Etiqueta	Cod. Operación	Operandos	Comentarios
ACCIÓN	MOVE	#55H,D1	;Mueve valor inmediato a D1
	ADDI.B	#AAH,D1	;Suma val. Inm. a reg. D1
	BEQ	ACCION	;Instr. de Salto si Z=1
	ADDI.B	#1,D1	;Suma val. Inm. a reg. D1
	BEQ	ACCION	;Instr. de Salto si Z=1
VARIABLE	DC.W	\$AA55H	;Reserva espacio a este valor

Tabla 10 Ejemplo con un µP68000, ..., 68040

Este ejemplo está expresado en instrucciones del microprocesador 68000, 68010, 68020, 68030 y 68040 de Motorola.

Vemos que los programas se enuncian, o se estructuran, en cuatro columnas o campos:

- **Columna nº 1:** Dedicada a la definición de las “Etiquetas”. Puede requerir de los “:” o no, dependiendo del procesador. Para un compilador es mas fácil distinguir una etiqueta con los “:” en caso de estar ocupando una sola línea (no confusión con posible mnemónico ocupando una sola línea).

También esta columna se utiliza para denominar una variable a la que se va a reservar espacio.

- **Columna nº 2:** Dedicada a los “Códigos de Operación”. Difieren entre procesadores, p.ej. para “mover” contenido de Memoria a Registro y viceversa, Intel usa MOV, Motorola MOVE y Sparc LD para lo primero y ST para lo segundo.

También dedicada a “Reservar” espacio, p.ej. DW reserva una palabra (16 bits).

- **Columna nº 3:** Dedicada a los “Operandos” necesarios del código de operación. En el caso de haber realizado reserva de espacio, esta columna se utiliza para dar un valor inicial a la variable. Según los distintos procesadores, hay diferencias entre **Destino ← Origen** u **Origen → Destino**.
- **Columna nº 4:** Dedicada a los “Comentarios”. Aquí se expresará con la mayor claridad posible, las acciones que se van a realizar.

4.2.3 Pseudoinstrucciones

Estamos acostumbrados a ver instrucciones orientadas al procesador, es decir, que acciones va a realizar; pero también pueden existir instrucciones orientadas al Ensamblador, estas le van a pedir al ensamblador que realice determinadas acciones, como p.ej., asignar espacio en la memoria. Estas instrucciones se denominan “Pseudoinstrucciones” o también “Directivas” del Ensamblador.

Veamos como son algunas de las pseudoinstrucciones que podemos encontrar en los ensambladores:

EQU: asigna un nombre simbólico a un valor, a un registro o a una expresión. No se puede redefinir.

AÑO	EQU	2003	;En vez de usar 2003, se utilizará AÑO
ENERO	EQU	1	
FEBRERO	EQU	ENERO+1	;Admite expresiones aritméticas

SET: asigna un nombre simbólico a un valor, a un registro o a una expresión. Se puede redefinir.

LIMITE	SET	120	;Asigna un valor numérico
LIMITE	SET	80	;Asigna un valor numérico
VALOR	SET	R2	;Asigna el valor del registro R2

END: terminar el programa ensamblador.

DB: reserva uno o mas bytes de memoria inicializados. Símbolo, Cadena de Caracteres o Expresión.

VALOR1	DB	#01H	;Reserva un byte
VALOR2	DB	'ESPERA'	;Reserva 6 bytes

DW: reserva una o mas palabras de memoria inicializadas. Símbolo, Cadena de Caracteres o Expresión.

VALOR2	DW	#1234H
--------	----	--------

ORG: asigna dirección, en memoria, de comienzo de un bloque de programa.

ORG	100H
ORG	START

PUBLIC: indica que esta variable puede ser utilizada en todos los archivos de programas que componen el programa total.

```
VARI1 PUBLIC
```

EXTERN: variable definida en otro módulo de programa y que se está utilizando en este módulo. Avisa al Ensamblador para que no de alarma.

```
VARI2 EXTERN
```

BIT, CODE, DATA, IDATA, XDATA: Direcciones de Bits, Datos o Código

```
SIMBOLO1 BIT 20H
SIMBOLO2 CODE 1E00H ;Código 0000H .. FFFFH
SIMBOLO3 DATA 126D ;0..127, 128..255 (SFR)
SIMBOLO4 IDATA 8AH ;0 .. 255
SIMBOLO5 XDATA 1E00H ;Datos 0000H .. FFFFH
```

4.3 Macroinstrucciones (Macros)

Durante el proceso de escritura de un programa, es muy normal el tener que escribir el mismo código varias veces, esto puede resultar tedioso siempre y cuando no sea un código de pocas líneas. Ante esta situación, pueden plantearse varias estrategias:

- Llamada a un Procedimiento o Subrutina: Presenta un pequeño inconveniente, cada llamada necesita de dos instrucciones adicionales, la llamada y el retorno. Ralentiza la ejecución del programa. Problema ante tiempos de ejecución críticos.
- Uso de Macroinstrucciones: Se asigna un “Nombre” que se define como “Macro” y es donde va el cuerpo de instrucciones que se repiten. En el resto de lugares del programa se escribe el nombre de la macroinstrucción que va a suplir al código de programa repetido.

Las “Macros” se definen, normalmente de la siguiente manera (válido para 8085, 8051, 8088, 8086, 80286, .. , Pentium II):

```
MIMACRO MACRO ;Cabecera de definición
Instrucción1 ;Cuerpo de la macro
Instrucción2
.....
InstrucciónN
ENDM ;Pseudoinstrucción de finalización
```

Por lo tanto, el programa se escribirá de la siguiente manera:

```
InstrucciónA
InstrucciónB
MIMACRO
InstrucciónC
InstrucciónD
.....
MIMACRO
InstrucciónJ
.....
END
```

Veamos como trata el Ensamblador este tipo

- a. El Ensamblador realiza una lectura del texto y cuando encuentra la definición de una Macro, guarda en una tabla el cuerpo de la definición, para usarla posteriormente.
- b. Cuando encuentra el nombre de la macro como código de operación, es decir, cuando se usa, substituirá el nombre por el cuerpo en el código a ejecutar. Este proceso se denomina “Expansión” de la Macro.

Es importante realizar las siguientes precisiones:

- La expansión de la Macro se efectúa durante el proceso de Ensamblado y no durante el proceso de Ejecución del programa.
- Antes de “usar” la Macro, debe estar definida.

La diferencia fundamental con el uso de Procedimientos o Subrutinas, es que en estas, el cuerpo de instrucciones invocado, solamente se escribe una vez en el código y cuando es invocado este procedimiento, el programa realiza un salto a la posición donde se ejecuta el procedimiento y al finalizar vuelve el programa a la instrucción siguiente a la que invocaba al procedimiento. Requiere guardar la dirección de retorno del procedimiento.

Como generalidad, nos podemos encontrar que las instrucciones requeridas en una Macro, pueden necesitar de parámetros con diferente valor según el punto de ejecución. Esto se resuelve dotando a la definición de la Macro de un conjunto de parámetros que van a permitir el paso de valores diferentes. En la definición se introducen los Parámetros Formales y en la llamada los Parámetros Reales.

Cada Ensamblador dispone de un número máximo de parámetros. Van separados por comas.

```

MIMACRO    MACRO A,B, ...,N    ;Cabecera y parámetros "Formales"
           Instrucción1        ;Cuerpo de la macro
           .....
           InstrucciónN
           ENDM                ;Pseudoinstrucción de finalización
           .....
MIMACRO    A1,B1, ...,N1      ;Llamada y Parámetros "Reales"
           InstrucciónD
           .....
MIMACRO    A2,B2, ...,N2      ;Llamada y Parámetros "Reales"
           InstrucciónJ
           .....
           END
    
```

Dentro de una Macro, podemos encontrarnos "pseudoinstrucciones" como por ejemplo REPT cuyo objetivo es repetir una o varias instrucciones "N" veces.

```

MIMACRO    MACRO              ;Cabecera de definición
           Instrucción1        ;Cuerpo de la macro
           REPT N
           Instrucción2
           .....
           ENDM                ;Finalización de Repetición
           .....
           InstrucciónN
           ENDM                ;Pseudoinstrucción de finalización
    
```

Otra pseudoinstrucción importante es EXITM, cuando el Ensamblador encuentra esta pseudoinstrucción, termina la expansión de la Macro. Se utiliza en ensamblado condicional.

```

MIMACRO    MACRO A            ;Cabecera de definición
           IF NUL A           ;Condición sobre parámetro A
           EXITM
           ENDIF
           REPT A
           NOP                 ;Repite "A" veces
           ENDM               ;Finalización de Repetición
           ENDM               ;Pseudoinstrucción de finalización
    
```

En una Macro se admiten "etiquetas" para poder realizar lazos. Estas etiquetas deben ser declaradas como "Locales" para que no existan problemas de duplicidad en nombres y no puedan ser accedidas por zonas de programa externas a la Macro.

```

MIMACRO    MACRO ITER         ;Cabecera de definición y parámetro
           LOCAL LAZO         ;Declaración etiqueta Local
           MOV R7,#ITER
           DCR R7
           JZ LAZO
           ENDM               ;Pseudoinstrucción de finalización
LAZO:
    
```

En una Macroinstrucción, pueden definirse más Macros, es decir, pueden anidarse mas Macroinstrucciones, definidas las etiquetas de las Macros interiores como Locales.

Las Macroinstrucciones pueden ser recursivas, es decir, llamarse a si mismas N veces. El número de llamadas es diferente por cada Ensamblador. Una cifra orientativa es 9.

4.4 Ensamblador

Desde un punto de vista general, el proceso de ensamblado se encarga de realizar las siguientes tareas:

- ❑ Traducir todas las instrucciones en lenguaje ensamblador al código máquina del procesador objetivo.
- ❑ Resolver todos los saltos a “etiquetas”, es decir, asignar las direcciones “simbólicas” de salto.
- ❑ Expandir las “macros” que aparecen en el programa.
- ❑ Interpretar las Directivas como p.ej. DB, asignando espacio.

La realización de estos procesos se lleva a cabo en lo que se conoce como “**Ensamblado de dos Pasadas**”. La idea fundamental que transmite este proceso es que mediante dos pasadas (dos lecturas completas del programa fuente), resuelve todos los puntos enumerados.

4.4.1 Primera Pasada

En este proceso se tratan los siguientes puntos:

- Se identifican las “etiquetas” que constituyen los saltos, para resolver el problema de “**las referencias adelante**”.

```

MOV    A,1EAFH
DCR    A
JZ     FIN
..
FIN:   MOV    B,#55H
        END
    
```

- Se identifican los símbolos que constituyen las pseudoinstrucciones.

```

AÑO    EQU    2003
    
```

- Se identifican los símbolos que constituyen las “variables” que van a estar almacenadas en memoria de datos.

```

STA    VALOR1
    
```

- Se identifican las Macroinstrucciones.

```

MIMACRO    MACRO
MOV    R7,#AÑO
MOV    A,R7
ADI    #01H
MOV    R7,A
ENDM
    
```

Para realizar una identificación completa de los procesos mencionados, se crea la “**Tabla de Símbolos**”, donde se incorporan todos los símbolos encontrados junto con un valor numérico, que identifica su posición en el programa fuente; este valor numérico se conoce como “**Contador de Posición de Instrucciones**” (ILC: *Instruction Location Counter*). Esta variable comienza con valor “0” y se incrementa en el valor necesitado por cada instrucción leída. P.ej. si encuentra:

```

LAZO1    SUI    #01H    ;Resta 1 al reg. A
          JNZ    LAZO1    ;Salta si Z=0
    
```

Incrementa en “2” por ser este el espacio necesitado por la instrucción SUI
Incrementa en “3” por ser este el espacio necesitado por la instrucción JNZ

Código de Operación (SUI) + Operando (01)

Código de Operación (JMP) + Dirección L + Dirección H

Nota: Puede suceder que haya símbolos que no se encuentran en este fichero de programa, estos símbolos se denominan “**Referencias no resueltas**”, esto normalmente es debido a que están contemplados en otro fichero de programa, como se mencionará mas adelante. Se resuelven con el Enlazador (*Linker*).

En la primera pasada, se crean las siguientes tablas: Tabla de Símbolos, Tabla de Pseudoinstrucciones y Tabla de Códigos de Operación.

- Tabla de Símbolos: Donde están ubicados los símbolos (etiquetas) y su posición de línea en el módulo.
- Tabla de Pseudoinstrucciones: Contiene los símbolos y su valor asignado. Indicará si los símbolos son Locales o Globales (accesibilidad). Se indica el tamaño si es necesario.
- Tabla de Código: Donde se incluye el mnemónico, operandos, código máquina (Hex) y la longitud de la instrucción.

4.4.2 Segunda Pasada

El objetivo de la segunda pasada es el generar el Programa Objeto. En esta operación, el ensamblador debe generar información adicional que será útil al programa denominado “Enlazador”. Este necesitará información para poder unir varios programas objeto y generar un único Programa Ejecutable.

Se leen las líneas de código, generando el código máquina y se asigna el valor de nº de línea de los símbolos referenciados, esto último se toma de la tabla de símbolos. Estos símbolos tendrán una “dirección relativa” al módulo, teniéndose que reubicar en el “enlazado”.

En esta segunda pasada, es donde se notifican los errores, estos pueden ser los siguientes:

- a. Se utilizó un símbolo pero no se definió
- b. Se definió un símbolo mas de una vez
- c. El mnemónico ubicado en el campo de código de operación no se corresponde con el repertorio de instrucciones
- d. Faltan operandos a un código de operación
- e. Un número Octal vale 8 o mas.
- f. Un número Hexadecimal difiere de 0..9, A..F
- g. Uso incorrecto de registros
- h. Falta la instrucción END

4.4.3 Estructura de un Módulo Objeto

Resumiendo lo visto hasta este momento, podemos decir que la estructura que crea el compilador, después de la segunda pasada, presenta el siguiente aspecto:

Fin de Módulo
Diccionario de Reubicación
Instrucciones Máquina y Constantes (La única que se cargará en Memoria)
Tabla de Referencias Externas
Tablas de Puntos de Ingreso
Identificación

Tabla 11 Estructura de un Módulo Objeto

- **Identificación:** Contiene el Nombre del módulo y datos auxiliares (fecha ensamblado, long de las partes del módulo, etc..).
- **Puntos de Ingreso:** Los símbolos declarados como PUBLIC (pseudoinstrucción) y su dirección además puede ser el nombre del módulo si este es una subrutina o procedimiento y su dirección.
- **Referencias Externas:** Símbolos utilizados en este módulo pero que estarán declarados en otro (si no “error”). Declarados como EXTERN (pseudoinstrucción). Esta tabla junto con la anterior pueden ser la misma.
- **Instrucciones Máquina y Constantes:** Contiene el código Objeto del programa.
- **Diccionario de Reubicación:** Suma constantes de reubicación a las instrucciones que disponen de direcciones de memoria.
- **Fin de Módulo:** Puede contener un Código de verificación de errores y la dirección absoluta de comienzo de ejecución del módulo

4.5 Enlazado y Carga

Es muy normal que un programa no se escriba en un solo fichero sino que se escriba en varios, según la funcionalidad requerida. Estos ficheros se denominan “Módulos” o también “Tareas”.

Cada módulo va a realizar una serie de funciones específicas y diferenciadas del resto de módulos. El conjunto de módulos constituye la funcional global del programa, por lo tanto, para que sean operativos, es necesario que un programa, denominado “Enlazador” (*Linker*) realice la unión de todos los módulos para ser cargados en memoria como un programa único.

También se requerirá el uso del enlazador, en el caso de que se invoque a una función de librería, trayéndola de esta e incorporándola al programa objeto a ser cargado en la memoria.

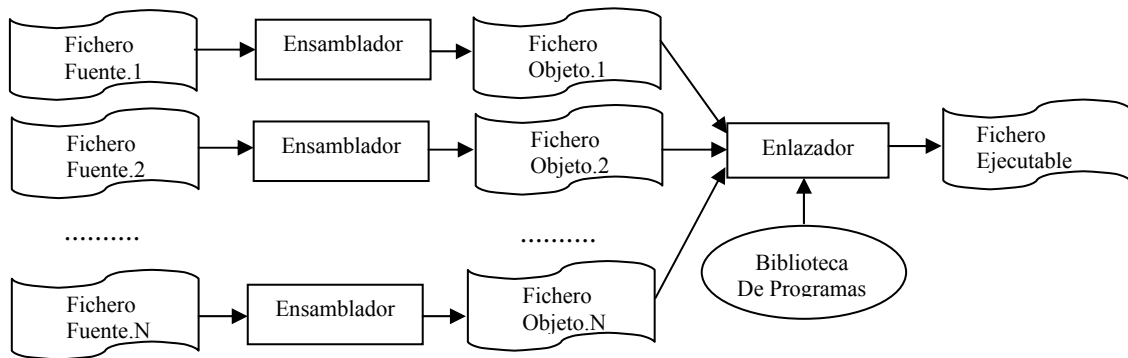


Figura 90 Esquema de Ensamblado – Enlazado - Carga

Cuando se realice un cambio en un programa fuente, solo será necesario ensamblar este módulo ya que el enlazador trabajará con los ficheros objeto no modificados y el nuevo fichero objeto.

Por lo tanto, la actividad desarrollada por el Enlazador, se resume en los siguientes puntos:

- Búsqueda en la Biblioteca de programas (Librería) para añadir al programa final la función de librería referenciada.
- Adjudicar las direcciones de memoria que van a ocupar todos los módulos de programa y “reubica” sus instrucciones ajustando las referencias absolutas.
- Resuelve las referencias entre ficheros. Adjudica la dirección de las referencias.

Por situar el problema, supongamos que partimos de cuatro módulos realizados en lenguaje ensamblador:

Ejemplo: Cuatro Módulos Objeto

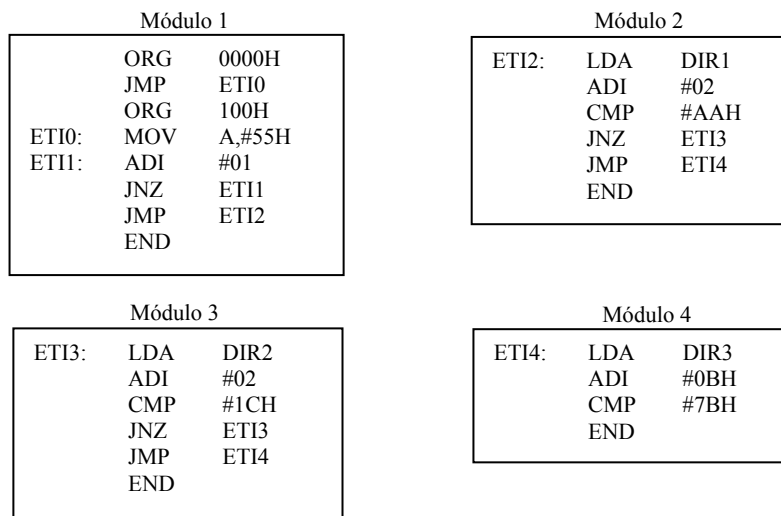


Figura 91 Ejemplo con cuatro Módulos Objeto

El compilador va a crear cuatro módulos con comienzo, cada uno, en la posición “0000”.

Módulo 1		Módulo 2		Módulo 3		Módulo 4	
12	ETI2H	12	ETI4H	12	ETI4H	6	7BH
11	ETI2L	11	ETI4L	11	ETI4L	5	CMP
10	JMP	10	JMP	10	JMP	4	0BH
9	ETI1H	9	ETI3H	9	ETI3H	3	ADI
8	ETI1L	8	ETI3L	8	ETI3L	2	DIR3H
7	JNZ	7	JNZ	7	JNZ	1	DIR3L
6	1	6	AAH	6	1CH	*0	LDA
*5	ADI	5	CMP	5	CMP		
4	55H	4	2	4	2		
*3	MOV	3	ADI	3	ADI		
2	DIR0H	2	DIR1H	2	DIR2H		
1	DIR0L	1	DIR1L	1	DIR2L		
0	JMP	*0	LDA	*0	LDA		


Tabla 12 Cont. Ejemplo con cuatro Módulos Objeto

Los Enlazadores requieren normalmente dos pasadas. La actividad desarrollada en cada pasada se resume en lo siguiente:

- Primera pasada:
 - Se leen todos los módulos Objeto
 - Se construye una tabla de nombres y longitudes de módulos
 - Se construye una tabla de símbolos global
- Segunda pasada:
 - Se leen los módulos Objeto
 - Se reubican
 - Se enlazan uno a uno, dejando, normalmente, un espacio de direcciones lineal.

Ejemplo: Cuatro Módulos Objeto Enlazados

142	7BH
141	CMP
140	0BH
139	ADI
138	DIR3H
137	DIR3L
*136	LDA
135	136H – 00H
134	136L – 88H
133	JMP
132	123H – 00H
131	123L – 7BH
130	JNZ
129	1CH
128	CMP
127	2
126	ADI
125	DIR2H
124	DIR2L
*123	LDA
122	136H – 00H
121	136L – 88H
120	JMP
119	123H – 00H
118	123L – 7BH
117	JNZ
116	AAH
115	CMP
114	2
113	ADI
112	DIR1H
111	DIR1L
*110	LDA
109	110H – 00H
108	110L – 6EH
107	JMP
106	102H – 00H
105	102L – 66H
104	JNZ
103	1
*102	ADI
101	55H
*100	MOV
....
2	100H -- 00H
1	100L --64H
0	JMP



142	7BH
141	CMP
140	0BH
139	ADI
138	DIR3H
137	DIR3L
*136	LDA
135	00H
134	88H
133	JMP
132	00H
131	7BH
130	JNZ
129	1CH
128	CMP
127	2
126	ADI
125	DIR2H
124	DIR2L
*123	LDA
122	00H
121	88H
120	JMP
119	00H
118	7BH
117	JNZ
116	AAH
115	CMP
114	2
113	ADI
112	DIR1H
111	DIR1L
*110	LDA
109	00H
108	6EH
107	JMP
106	00H
105	66H
104	JNZ
103	1
*102	ADI
101	55H
*100	MOV
....
2	00H
1	64H
0	JMP

Una vez que se han dispuesto de manera contigua los módulos, el enlazador resuelve las referencias, dando direcciones absolutas.

Los valores de DIR1, DIR2 y DIR3, los asigna el enlazador del espacio de memoria de datos.

Tabla 13 Cont. Ejemplo con cuatro Módulos Objeto

La segunda columna es la que sería cargada a memoria por el “Cargador” de programas.

4.4.4 Enlazado Dinámico

Todo lo visto anteriormente se trata de un enlazado que se realiza antes de cargarse el programa a memoria y queda fija la estructura hasta que sea cambiada por actualización voluntaria por parte del programador. Podemos verlo como un enlazado estático.

Se abre una posibilidad de realizar otra estrategia que se denomina “Enlazado Dinámico”. Consiste en enlazar un módulo cuando este es invocado por otro módulo que ya está residente en Memoria Principal y se está ejecutando. De esta manera, si disponemos de un programa muy extenso y partido en muchos módulos, realizaremos el enlazado de los módulos básicos para el arranque de la aplicación, dejando el resto a la dinámica del propio programa.

Una de las ventajas que trae este tipo de estrategia es que sacamos mas rendimiento de la memoria virtual y solamente enlazaremos módulos cuando estos sean invocados. Puede haber módulos que rara vez deban de ser utilizados, como por ejemplo ciertas llamadas de error.

Recurriendo a un Sistema Operativo muy extendido en el mundo de la Ofimática, como Windows, diremos que emplea un enlazado dinámico con un formato de archivo especial denominado **DLL** (*Dynamic Link Library*) Biblioteca de Enlace Dinámico. Su extensión es **.dll**, **.drv** (*drivers*) o **.fon** (tipos de letras).

Los archivos .dll pueden contener: Procedimientos o Datos de Librería. La configuración mas usual de una .dll consiste en una librería que contiene varios procedimientos que pueden ser cargados en memoria a la vez. Varios programas ejecutándose en memoria pueden requerir de esos procedimientos indistintamente.

Cada vez que se invoca a una DLL (un proceso se liga), se realiza un proceso de adjudicación de memoria y cuando se deja de utilizar (un proceso se desliga) se libera memoria.

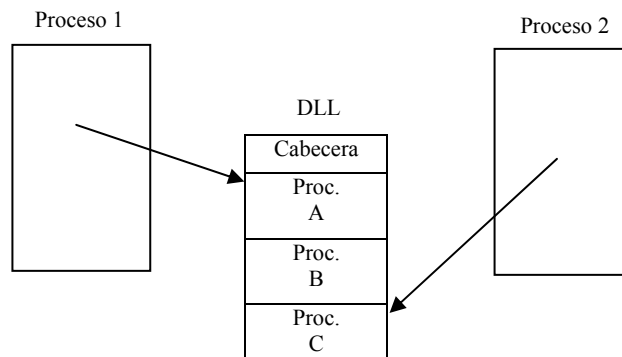


Figura 92 Llamadas a DLLs

Una DLL no puede ejecutarse sola, a diferencia de un programa binario ejecutable. No puede iniciarse ni ejecutarse sola ya que no dispone de programa principal.

Los programas se ligan a las dll de las dos maneras siguientes:

- Enlazado Implícito: Un archivo especial denominado Biblioteca de Importación, permite que el programa de usuario, pueda acceder a la DLL. El programa de usuario se enlaza estáticamente con la biblioteca de importación.

Un programa de usuario cargado en memoria que sigue el procedimiento implícito, realiza una inspección de las DLL que son necesarias y si están en memoria; las que no estén son cargadas inmediatamente. Los procedimientos que están en las DLL, se preparan para ser direccionados en el espacio de memoria virtual. Ya pueden ser llamadas como si hubieran sido ligadas estáticamente.

- Enlazado Explícito: No requiere de bibliotecas de importación, no requiriéndose que se carguen las DLL al mismo tiempo que el programa de usuario. El programa de usuario invoca a la DLL para ligarse dinámicamente y posteriormente realiza llamadas adicionales para obtener las direcciones de los procedimientos necesarios. Cuando ha finalizado el último proceso, el programa realiza una llamada para desligarse de la DLL, liberándose memoria.

TEMA 5: ARQUITECTURA DE COMPUTACIÓN EN PARALELO

5.1 Introducción

La cuestión fundamental que se plantea es como conseguir mayor capacidad de cómputo dentro de una inversión razonable. Unos resolverán la cuestión con la adquisición de un computador mas rápido que el anterior pero para otros esto nunca será la solución porque el nivel de cálculo necesitado siempre superará a la mejor de las expectativas de evolución del mercado en aspectos de velocidad. Por ello es necesario invertir en esfuerzos para, por una parte mejorar el diseño de los programas, optimizando su ejecución o aislando tareas independientes o “quasi” independientes y por otra conseguir crear una arquitectura de procesadores trabajando en “paralelo”, de tal manera que cada procesador realice la tarea específica y comparta resultados con el resto de procesadores; esto en el caso de tratarse de tareas “quasi” independientes. Podemos tener computación en paralelo de tareas que son absolutamente independientes.

Veamos un ejemplo sencillo de Procesamiento Paralelo:

$$\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} + \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} = \begin{pmatrix} a_1 + a_2 & b_1 + b_2 \\ c_1 + c_2 & d_1 + d_2 \end{pmatrix}$$

Si disponemos de 4 procesadores, cada procesador realizará la suma de la fila-columna correspondiente, entregando el resultado a uno de ellos. Una situación real es mas compleja, hay un mayor nº de Datos, uno de los procesadores recogerá los Datos de entrada repartiéndolos al resto de procesadores según el algoritmo de distribución y el resultado será recibido por uno de los procesadores.

La computación en paralelo presenta un inconveniente añadido que es que el programador debe conocer, en alguna medida, la arquitectura de su multicomputador. Aspecto este que con un único procesador, es el compilador sobre quien recae la labor.

Para la computación en paralelo se necesitan Procesadores y Memorias. Cada Procesador dispondrá de una Memoria de Datos propia o bien dispondrá de una Memoria de Datos compartida por los Procesadores. Pueden existir mas soluciones dependiendo de arquitecturas.

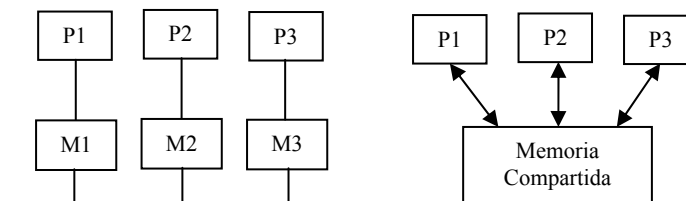


Figura 93 Esquemas de Arquitecturas de Procesamiento Paralelo

Por lo tanto, el diseño de una arquitectura de multiprocesamiento, requiere plantearse las siguientes cuestiones:

- ¿Cómo se comparten los Datos?
- ¿Cómo se coordinan los Datos?
- ¿Cuántos Procesadores tiene?

Una solución a la compartición de memoria se basa en disponer memorias tipo caché para poder trabajar a alta velocidad, disponiéndose de 2 a 4 niveles de caché.

Los procesadores con memoria compartida presentan tres variantes:

- Tiempo de acceso a memoria igual para todos los procesadores y todas las palabras. Estas máquinas se denominan UMA (acceso uniforme a memoria), dentro de esta categoría están los SMP (multiprocesadores simétricos).

- Algunos accesos a memoria son más rápidos que otros dependiendo que procesador haga la referencia y a que palabra. Se denominan NUMA (acceso no uniforme a memoria). El acceso a memoria local es mas rápido que a la remota.
- Cuando solo hay acceso a memoria caché. Estas máquinas se denominan COMA (*Caché Only Memory Access*)

Las máquinas NUMA presentan un mejor rendimiento y son más escalables (mejor crecimiento). La tendencia comercial es hacia esta solución.

El modelo alternativo al de memoria compartida es el de “paso de mensajes”. En este caso se realizan intercambios de mensajes entre los procesadores. Esto es necesario cuando la memoria es “privada” para cada procesador. Para ello deben implementarse rutinas o directivas que realicen la función de “enviar” y “recibir” datos. Esto lo podemos ver en computadores conectados a través de una red local como ejemplo extremo de paso de mensajes entre procesadores.

La coordinación de los Datos requiere de mecanismos que garanticen la validez de ellos en todo momento, es decir si un procesador modifica un dato y estamos trabajando con memoria compartida, se debe asegurar que el procesador que realiza el cambio es el único que accede a memoria y una vez realizado el cambio este es comunicado a los demás procesadores. Esta técnica se denomina de “Bloqueo” (*Lock*). El mecanismo se controla por semaforización con acceso indivisible (uso de instrucción “*swap*”).

Cuando se trata de memorias propias, el cambio es notificado con SEND o RECEIVE. Un procesador sabe cuando envía y cuando recibe por lo tanto el acceso a memoria propia se realiza de manera voluntaria y con total control.

Hemos visto los dos tipos de comunicación entre procesadores, ahora podemos contemplar el tipo de organización en la interconexión que puede existir entre los procesadores.

5.2 Modos de Interconexión

Recogiendo las ideas anteriores, podemos distinguir dos tipos de configuraciones de interconexión:

- Multiprocesadores conectados por un solo Bus
- Multiprocesadores conectados por una Red

5.2.1 Multiprocesadores conectados por un solo Bus

Los microprocesadores de alto rendimiento y bajo coste renovaron el interés por los multiprocesadores en los años 80. Existen varias razones para poder optar a este tipo de conexión por medio de un Bus:

- Los microprocesadores son mas pequeños que un procesador multichip (conf. *Bitslice*), por lo tanto pueden colocarse mas procesadores en el Bus.
- Las cachés pueden reducir el tráfico del Bus
- Existen mecanismos para mantener la coherencia de caché y memoria para multiprocesamiento. Esto simplifica la programación.

La Figura 94 nos da una idea de la arquitectura.

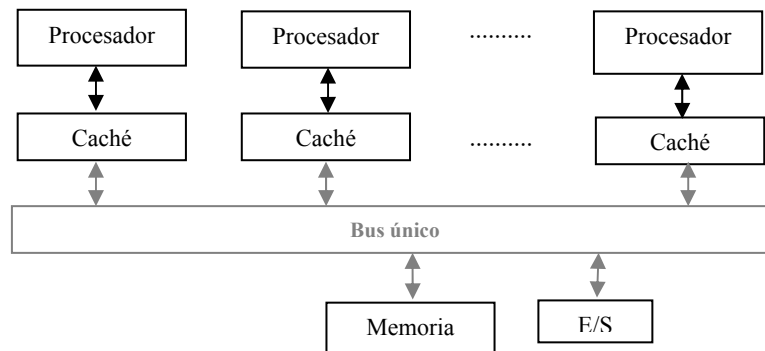


Figura 94 Multiprocesadores conectados por un solo Bus

El tráfico por procesador y el ancho de banda del bus determinan el número de procesadores útiles en este sistema multiprocesador. Una configuración comercial como la anterior maneja entre 2 y 36 procesadores.

El modo en como un procesador trata el movimiento de datos de o hacia memoria es con instrucciones LOAD y STORE (o equivalentes), no es necesario nada más.

Las cachés replican los datos en sus memorias tanto para reducir la latencia de acceso a los datos como para reducir el tráfico de datos con la memoria en el Bus. Esto da lugar a los “Protocolos de Coherencia de Caché”.

Protocolo “Snooping” (husmear o espiar): Mediante este protocolo todos los controladores de las cachés vigilan el tráfico en el Bus de tal manera que determinan si tienen una copia del bloque compartido. El esquema de la Figura 94 puede expandirse de la siguiente manera:

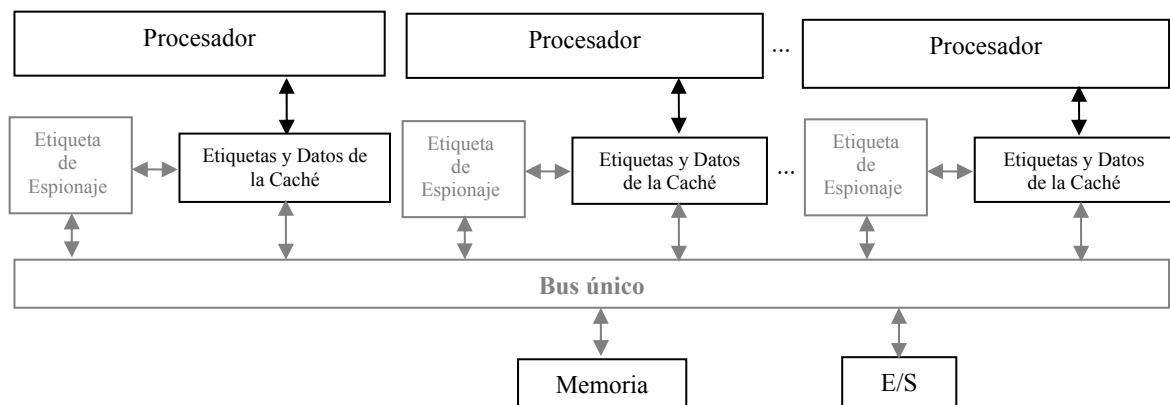


Figura 95 Multiprocesadores con Protocolo Snooping

La coherencia implica Lectura y Escritura. Copias múltiples en caso de lectura no presenta ningún problema, pero la escritura de una palabra en un procesador necesita un acceso exclusivo a la palabra. El resto de procesadores debe tener acceso a esa palabra en el momento de ser actualizada. Por lo tanto una escritura en una palabra requiere la actualización de todas las palabras compartidas en todas las cachés o bien la invalidación de todas las palabras después de la escritura. El protocolo debe conocer qué cachés comparten el dato. En las etiquetas de “espionaje” están replicadas las etiquetas de direcciones de la caché (no la caché al completo).

Los bits de estado ya presentes en la caché se expanden para acomodar el protocolo de “espionaje”.

Se dan dos tipos de protocolos de “espionaje” según como se traten las escrituras:

- **Invalidación por escritura:** El procesador que realiza la escritura, invalida previamente las palabras de las copias de las otras cachés y después actualiza los datos locales. El procesador que escribe envía una señal de invalidación al bus y todas las cachés comprueban si tienen una copia, si la tienen invalidan el bloque compartido que la contiene.
- **Actualización por escritura:** El procesador que escribe, envía los nuevos datos a todos los procesadores, de esta manera las copias se actualizan con el nuevo valor. Esto se conoce como “Escritura Difundida” (*Write Broadcast*).

La escritura por invalidación es similar a la “escritura diferida” (recuérdese la memoria caché). Esta escritura usa el bus una única vez ya que se envía la invalidación una única vez. Reduce el tráfico del Bus por lo tanto reduce el ancho del Bus. Permite disponer de mas procesadores para un mismo ancho de Bus. Prácticamente todas las máquinas comerciales utilizan el protocolo de escritura por invalidación.

La escritura por actualización es similar a la “escritura a través” (recuérdese la memoria caché) porque todas la escrituras son enviadas por el bus para actualizar las copias de los datos compartidos.

Un ejemplo de protocolo de **Invalidación por escritura**, es el MESI (Pentium Pro y Power PC), basado en un protocolo anterior denominado ”Protocolo de Escritura Única”. Consta de cuatro estados que son los que dan el nombre al acrónimo.

1. **Invalid (No Válida):** La entrada de caché no contiene datos válidos.
2. **Shared (Compartida):** Varias cachés podrían contener la línea (memoria actualizada)
3. **Exclusive (Exclusiva):** Ninguna caché contiene la línea (memoria actualizada)
4. **Modified (Modificada):** La entrada es válida, la memoria no es válida (no existen copias)

Retomando el tema central de este apartado, conexión a través de un solo Bus, se puede realizar la consideración final siguiente, y es que el diseño de sistemas multiprocesador con un solo bus presenta limitaciones debido a que las tres características deseables “Gran Ancho de Banda”, “Latencia Pequeña” y “Gran Longitud”, son incompatibles. Existe una limitación en el ancho de banda de los módulos de memoria conectados al bus. Esto hace que hoy en día el número de procesadores conectados vaya disminuyendo (los 36 mencionados anteriormente va decreciendo). La ampliación se consigue acudiendo a la conexión por red, como se verá mas adelante.

Esta estructura de datos compartidos presenta una “localidad” espacial y temporal menor que el resto de tipos de datos. Los fallos de caché deciden la conducta de la caché aunque el acceso a datos esté entre un 10% y un 40%. El tamaño del bloque también es un factor a tener en cuenta.

5.2.1.1 Sincronización

Como se ha comentado en la introducción, debe existir un método de coordinación para poder acceder a la memoria compartida por parte de un único procesador, debe garantizarse un proceso de exclusión mutua. El método garantizará un “bloqueo” de la memoria por medio de la conveniente “semaforización”. Se leerá/escribirá sobre la “variable de bloqueo”.

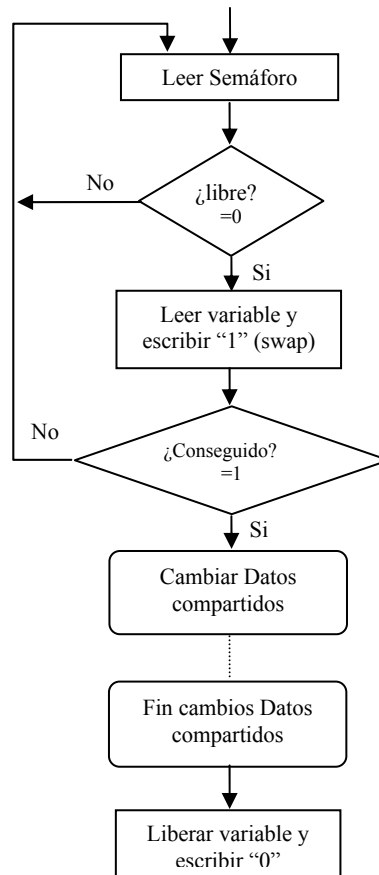


Figura 96 Sincronización de Memoria

5.2.2 Multiprocesadores conectados por una Red

Mediante esta configuración se consigue aumentar la longitud y el ancho de banda. Hemos añadido mas buses para el movimiento de datos a memoria, cada memoria está conectada a un procesador de tal manera que el bus se utiliza para accesos a memoria por cada procesador pero la red solo es utilizada por los procesadores, no se requiere acceso a memoria. Las memorias son privadas.

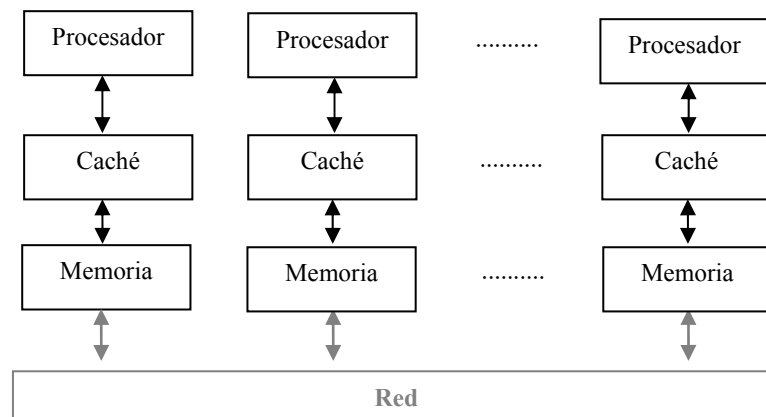


Figura 97 Multiprocesadores conectados por una Red

Podemos ver esta arquitectura como una Memoria Compartida “Distribuida” (DSM), es decir, una memoria dividida en páginas de direcciones virtuales donde cada página es asignada a un procesador o de múltiples memorias “privadas”, donde la comunicación se realiza por medio de directivas del Sistema Operativo SEND (enviar) y RECEIVE (recibir), en contraposición a las instrucciones LOAD y STORE vistas en arquitecturas de un solo Bus. Realmente es mejor definir esta memoria como “Memoria Compartida Distribuida” (DSM), ya que cuando un procesador necesite un dato y no lo tenga realizará una petición por medio de la red.

Si la memoria es distribuida, cuando un procesador haga referencia a una zona de memoria ubicada en el espacio de otro procesador, el Sistema Operativo ejecutará una “trap” (trampa) y buscará en el procesador adecuado la memoria requerida (SEND y RECEIVE). Si no es distribuida, las directivas “enviar-recibir” serán realizadas por el programa con llamada al S.O.

Los multiprocesadores conectados en Red, deben presentar también una coherencia de Caché, en este caso uno de los protocolos mas importantes es relacionado con el manejo de un directorio donde están relatadas las líneas de caché utilizadas.

Protocolo de Directorios: Se mantiene una Base de Datos que indica donde está cada línea de caché y cual es su estado. Cuando se referencia a una línea de caché, se consulta este directorio para averiguar donde está y si está limpia o sucia (modificada). El hardware debe ser de alta velocidad ya que este directorio se consulta con cada instrucción. La **Figura 99** muestra esta arquitectura.

Cuando un Procesador emite una dirección, el MMU la traduce a una dirección física, dividiendo la información entres campos, un primer campo indicará en que Nodo está el dato requerido, un segundo campo indicará que línea (bloque) de caché es la requerida y un tercer campo indicará la distancia (palabra requerida).

Nodo	Bloque	Distancia
------	--------	-----------

Figura 98 Dirección en el Bus

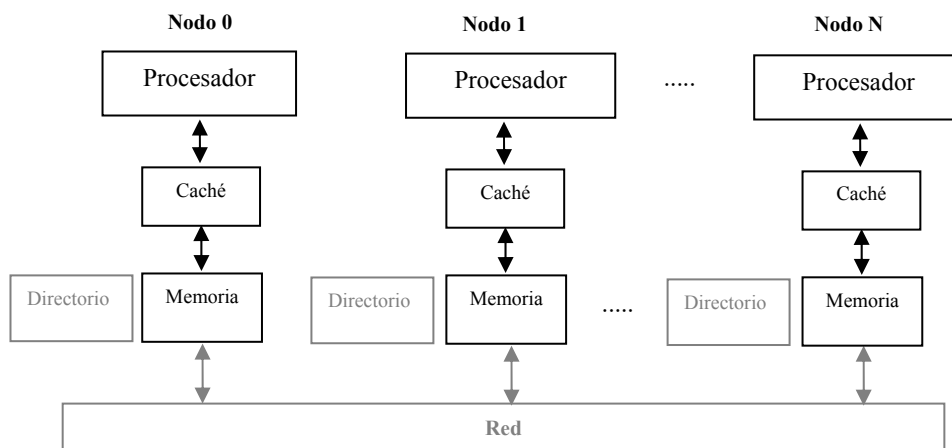


Figura 99 Multiprocesadores con Protocolo de Directorios

La memoria se reparte entre todos los procesadores, de tal manera que cada procesador dispone de un área reservada exclusivamente para el.

El controlador del directorio envía comandos explícitos a los nodos (procesadores) que tienen copia de los datos. Como se aprecia, no está constantemente revisado el tráfico de la Red para ver si hay peticiones que afectan al nodo.

5.2.3 Comparativa de Arquitecturas

La Figura 100 muestra unas gráficas comparativas entre los dos tipos de arquitecturas (datos obtenidos en el año 1997):

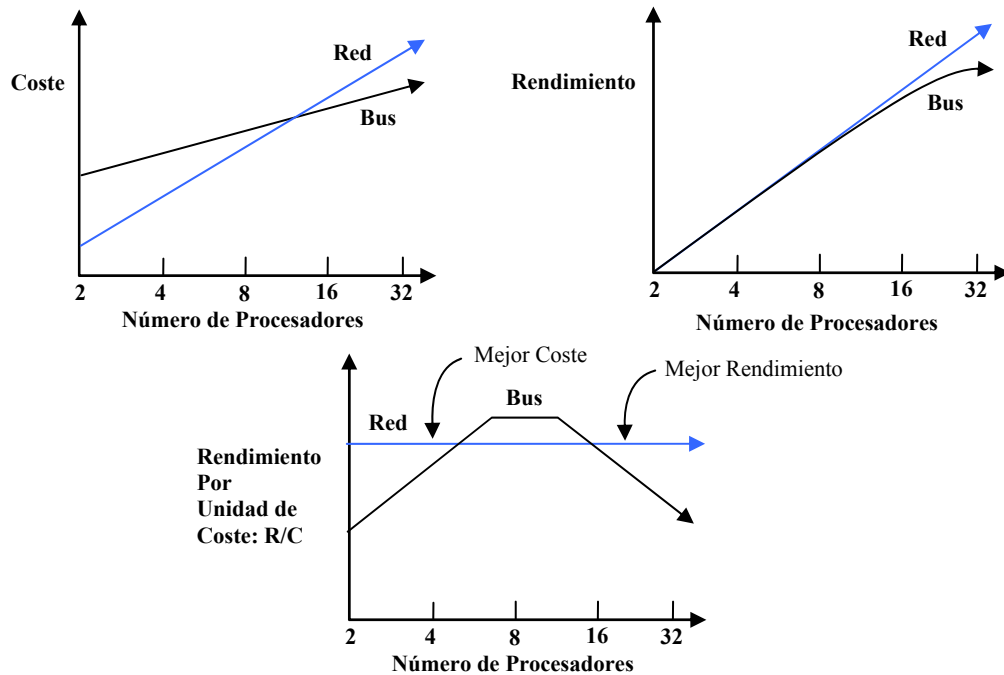


Figura 100 Coste y Rendimiento según arquitecturas

El diseñador decidirá la zona óptima de trabajo del Bus.

5.3 Clusters

Se dan otro tipo de aplicaciones para grandes computadores, como Bases de Datos, Servidores, Procesamiento por Lotes, etc., que pueden ser ejecutados en máquinas “menos acopladas” que las máquinas con arquitecturas como las anteriores con coherencia de caché. Estas máquinas necesitan una disponibilidad permanente, lo que implica cierta tolerancia a fallos y reparabilidad. Si nos basamos en las redes de área local de alta velocidad, los conmutadores de alta velocidad y los ordenadores de sobremesa, la arquitectura para un procesamiento a gran escala se construirá en torno a agrupaciones de los elementos mencionados, creándose “Clusters” (agrupaciones) de computación. Es decir, los “clusters” son agrupaciones de computadores (normalmente PCs) libremente acopladas. Están conectados por medio de redes locales de características no propietarias. Un usuario verá al cluster como una única máquina.

Desventaja: Una desventaja de los clusters es que el coste de administrar N máquinas es similar al coste de administrar N máquinas independientes, mientras que el coste de administración de un sistema multiprocesador con un espacio de direcciones compartido por N procesadores es similar a administrar una sola máquina.

Ventaja: Teniendo en cuenta que un cluster se construye con computadores (no con procesadores), cuando se deba reemplazar una máquina, esto será una tarea mucho más fácil que en una arquitectura NUMA, no será necesario el tener que parar todo el sistema. Permite una alta disponibilidad y una expansión rápida.

Una evolución actual de los clusters es el realizar arquitecturas mixtas, es decir, cluster formados por computadores conectados en red y los computadores con varios procesadores (arquitectura multiprocesador) UMA.

Podemos ver una arquitectura denominada híbrida y como ejemplos podemos tomar un cluster de 32 procesadores que puede ser construido con 8 procesadores UMA con 4 procesadores cada UMA, o bien, 4 procesadores UMA con 8 procesadores cada UMA. A esta agrupación también se la conoce como “cluster de memoria compartida”.

Por último comentar que los clusters se clasifican en “centralizados” y “descentralizados”; en los primeros, se encuentran en la misma sala todas sus unidades mientras que en los segundos se encuentran sus unidades distribuidas en el edificio o edificios.

5.4 Topologías de Red

En este apartado se van a describir las diferentes topologías que afectan tanto a Procesadores como a Computadores (Clusters), es decir, los modos de interconexión aplican a ambos tipos de configuración ya sea Multiprocesador conectado por Red como Multicomputador conectado por Red.

5.4.1 Introducción

Dado que no se considera operativo el que todos los Procesadores (Computadores) estén conectados con todos a la vez, se introducen redes o retículas de interconexión donde cada nodo tiene conectado un Procesador (Computador) por medio de un dispositivo que envía o recibe mensajes de o hacia la red. Este dispositivo recibe el nombre de conmutador.

Las Redes se representan como “grafos”, los arcos (o aristas) del grafo representan los enlaces de la red de comunicación y que unen los nodos, en los nodos se disponen los “conmutadores”. Todos los enlaces vamos a considerarlos “bidireccionales”, es decir, la información puede viajar en ambos sentidos. Además todas las redes están basadas en el uso de “conmutadores” que son los que unen los nodos “procesador-memoria” con los otros enlaces (a conmutadores) y son los encargados de “encaminar” los datos.

Hay que introducir una medida que nos permita realizar comparaciones de velocidad entre diferentes clusters, esta es la de “ancho de banda total de la red”, esta se define como el ancho de banda de cada enlace multiplicado por el número de enlaces.

Esta medida es la óptima pero no se adecua al peor de los casos, para ello se define el “ancho de banda de la bisección”. El cálculo es como sigue, se divide la máquina en dos partes, a cada lado se deja el mismo número de nodos y se suma el ancho de banda de los enlaces que atraviesan esta mitad imaginaria. La idea primordial es escoger la división imaginaria que de peor medida. Los programas paralelos están limitados por el enlace mas débil.

Otra medida introducida es “diámetro de una red”, es el nº de arcos o aristas que hay entre los nodos mas alejados, cuanto menor sea el diámetro menor será el retraso de los mensajes o lo que es lo mismo, mejor será su rendimiento.

5.4.2 Topologías

Veamos a continuación varias topologías de Red:

- **Red en Anillo:** Arquitectura 1-D. Los mensajes van de O – E o de E – O. Hay nodos que no están conectados entre si por lo tanto sus mensajes deben pasar por otros nodos intermedios hasta llegar a su destino.

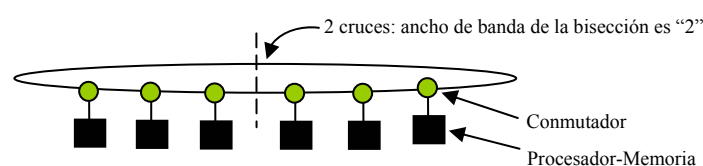


Figura 101 Red en Anillo

A diferencia de un Bus, el anillo permite varias transferencias en la misma unidad de tiempo.

El ancho de banda total de la red, si hay P procesadores, es P veces el ancho de banda de cada enlace. El ancho de banda de la bisección es dos veces el ancho de banda del enlace.

- **Red en Cuadrícula:** Arquitectura 2-D. Los mensajes van de O – E o de E – O o de N – S o S - N. Si hay N procesadores por lado, requiere de N^2 nodos.

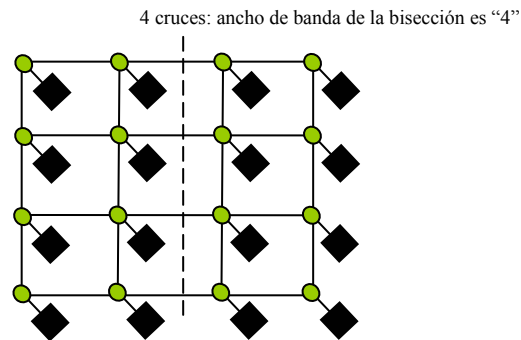


Figura 102 Red en Cuadrícula

- **Red en Cubo:** Arquitectura N-D. Es una arquitectura n-cubo con 2^N nodos, que requiere N enlaces por conmutador mas uno para el procesador. Dispone de N enlaces vecinos.

La Figura 103 ilustra el caso de N=3.

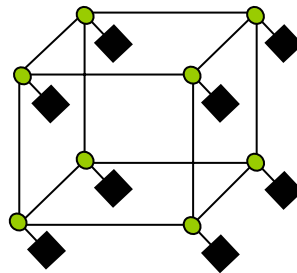


Figura 103 Red en Cubo con N=3

Con el fin de aumentar el rendimiento y fiabilidad de las redes en cuadrícula o cubo, suelen añadirse arcos que unen nodos extremos, de tal manera que disminuye el camino de enrutamiento. Esto lo podemos ver en la **Figura 104**:

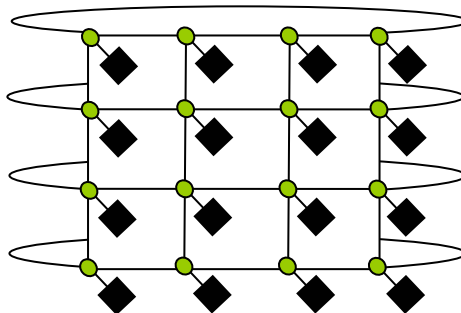


Figura 104 Red en Cuadrícula Toroide

Para aumentar la eficiencia de un cluster pueden realizarse topologías totalmente conectadas “Interconexión Total”, es decir, todos los nodos disponen de un enlace entre si. El número de enlaces requerido es $k(k-1)/2$, siendo k el nº de nodos. El valor obtenido puede resultar inmanejable cuando crece el nº de nodos.

Ancho de banda total : $P \times (P-1)/2$
 Ancho de banda de la bisección : $(P/2)^2$

siendo P el Nº de Procesadores

5.4.3 Otras Topologías - Redes Multietapa

Las topologías de redes totalmente conectadas que acabamos de ver, presentan un gran rendimiento pero a costa de un incremento considerable en su coste. Una alternativa a colocar un Procesador (Computador) en cada nodo, consiste en disponer en los nodos de solamente el conmutador o bien en algunos nodos. Los conmutadores son más pequeños que el nodo procesador-memoria-conmutador y puede aumentarse la densidad de empaquetamiento de estas soluciones. La consecuencia directa es que se acortan las distancias y se incrementa el rendimiento. Estas redes se denominan “redes multietapa”. Se dan dos arquitecturas:

- **Crossbar o Totalmente Conectada:** Todos los procesadores están conectados con los otros por medio de una malla (crossbar) que une sus caminos con conmutadores. El nº de conmutadores necesarios es N^2 , siendo N el nº de procesadores. Se consiguen mejores empaquetamientos del conjunto procesador-conmutador.

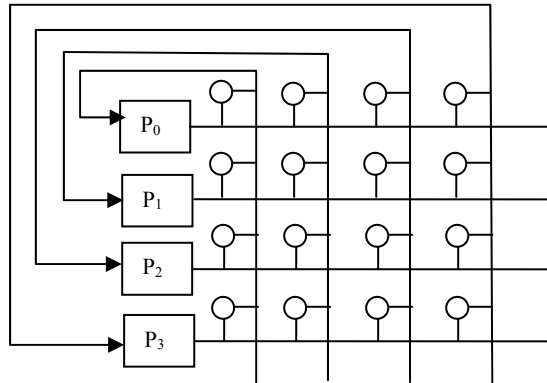


Figura 105 Red Crossbar

- **Red Omega:** Utiliza menos circuitería que la *crossbar*, $(N/2)\log_2 N$.

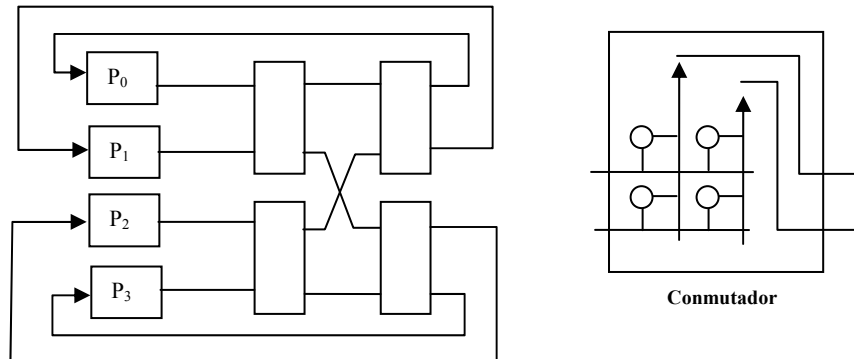


Figura 106 Red Omega

Tanto la red *Crossbar* como la Omega pueden presentar conexiones con memorias en vez de con los procesadores. Como ejemplo podemos ver la primera conectada a memorias:

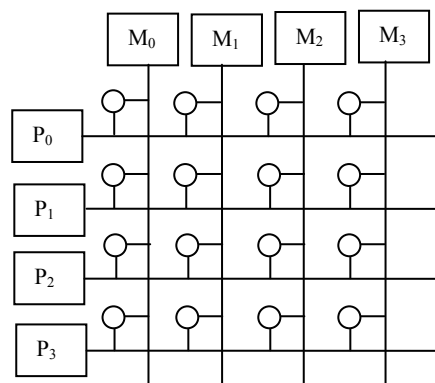


Figura 107 Red Crossbar con Memoria

5.4.4 Conmutadores

Como ya se ha comentado, los conmutadores son los dispositivos encargados de encaminar los paquetes de información entre nodos. Además pueden realizar un almacenamiento del mensaje con el fin de descargar al procesador enviante y mantener un protocolo de reenvíos en caso de errores de comunicación. El almacenamiento es FIFO.

Cuando el conmutador recibe un mensaje, revisa el encabezamiento del mensaje con el fin de decidir que ruta va a tomar el mensaje ya que el conmutador “conoce” la conexión de todos los nodos (procesador conectado a nodo). Además si el canal del conmutador está ocupado por otro mensaje, lo mantendrá en la FIFO hasta que se libere o bien decidirá si lo envía por otra ruta alternativa.

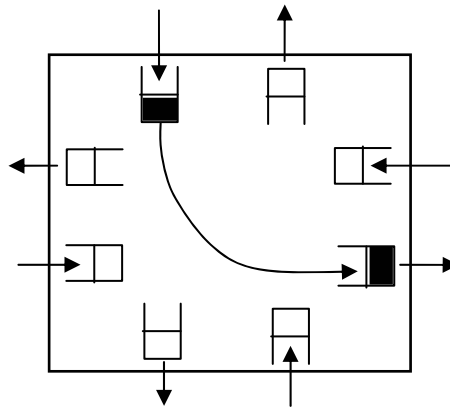


Figura 108 Conmutador

Según como sea la arquitectura del conmutador y su algoritmo de movimiento de mensajes internamente, un mensaje podría pasar a la FIFO del canal de salida adecuado y esperar a su emisión, dejando libre el canal de entrada para posible encolamiento de otro mensaje entrante.

5.5 Ley de Amdahl

En la introducción de este tema se comentó que la incorporación de técnicas de multiprocesamiento hacía indispensable el efectuar optimizaciones en la programación, para poder alcanzar mejores rendimientos.

En todos los programas se da una parte de código en forma “secuencial” y otra parte de código en forma “paralelizable”. La parte “secuencial” está normalmente asociada a tareas como “Iniciación” – “Lectura de Datos” – “Agrupación de Resultados”. La ley de Amdahl, aplicada al multiprocesamiento, dice lo siguiente: “*Para conseguir tasas altas de procesamiento paralelo, hay que conseguir tasas de procesamiento secuencial de parecida magnitud*”.

La expresión matemática de esta ley es de la siguiente forma:

$$T_{\text{ejecución después de la mejora}} = \frac{T_{\text{ejecución afectado por la mejora}}}{\text{Cantidad de mejora}} + T_{\text{ejecución no afectado}}$$

Otra manera de expresar esta ley es mediante el razonamiento siguiente:

Si disponemos de un programa que se ejecuta en un tiempo T , donde una fracción “ f ” está dedicada al tratamiento secuencial, tendremos que “ $1-f$ ” es la fracción del tiempo dedicado al programa paralelizable. Si ahora introducimos “ n ” procesadores paralelos, tendremos que la fracción paralelizable queda reducida a “ $(1-f)/n$ ”. Por lo tanto el nuevo tiempo de ejecución será el siguiente:

$$f \cdot T + (1 - f) \cdot \frac{T}{n}$$

La aceleración obtenida se expresa de la siguiente forma:

$$Aceleración = \frac{T_{inicial}}{T_{nuevo}} = \frac{T}{f \cdot T + (1-f) \cdot \frac{T}{n}} = \frac{n}{1 + (n-1) \cdot f}$$

Vemos, por tanto, que la aceleración va a depender del tiempo de tratamiento secuencial.

5.6 Perspectiva Histórica – Clasificación de los Computadores

Como cierre de tema dedicado al multiprocesamiento, realizaremos una revisión histórica de cómo se han clasificado los computadores según su arquitectura.

Es en ¿1966 o 1972? cuando Flynn observando el nº de instrucciones paralelas y los flujos de datos, etiqueta a las computadoras del siguiente modo:

1. Flujo único de Instrucciones, único flujo de Datos : SISD
2. Flujo único de Instrucciones, múltiple flujo de Datos : SIMD
3. Flujo múltiple de Instrucciones, único flujo de Datos : MISD
4. Flujo múltiple de Instrucciones, múltiple flujo de Datos : MIMD

Existen máquinas híbridas entre estas categorías pero se utiliza el modelo de Flynn por ser sencillo y fácil de entender.

- **SISD**: Sistema monoprocesador clásico de flujo secuencial (arquitectura Von Newman) sobre el que no se comentará nada en este capítulo
- **SIMD**: Este sistema opera con “Vectores” de Datos. Como solo dispone de un único flujo de Instrucciones, solo ejecuta una instrucción por unidad de tiempo pero esta instrucción opera sobre un conjunto de Datos a la vez.

Como ejemplo supongamos que una instrucción suma 64 números + 64 números. Envía 64 flujos de Datos a 64 ALUs para realizar 64 sumas en un solo ciclo.

Las instrucciones son mezcla de SISD y SIMD, es decir, instrucciones secuenciales e instrucciones paralelizables. El máximo rendimiento se consigue con bucles “for” y el peor con “case” o “switch”.

Una variante de SIMD lo constituyen los “Procesadores Vectoriales”. Trabajan con vectores de números y disponen de “Unidades funcionales” segmentadas que operan con unos pocos elementos del vector. Un SIMD opera con todos los elementos a la vez.

Hemos visto una arquitectura de un procesador con múltiples ALUs, en esta misma clasificación se da la arquitectura de “Arrays” de procesadores que trabajan con la misma instrucción y sobre vectores de datos. Una Unidad de Control es la encargada de alimentar a los procesadores con la instrucción.

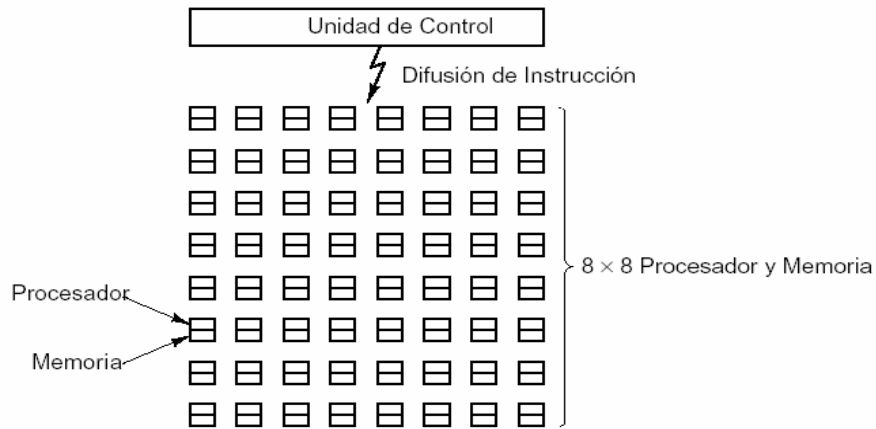


Figura 109 Array de Procesadores ejecutando la misma instrucción

- **MISD:** Difícil imaginar un “Multiflujo de Instrucciones” con un solo Dato. No hay aplicaciones conocidas. ¿Procesadores Sistólicos?, no presenta interés.
- **MIMD:** Aquí se engloban las máquinas multiprocesadoras actuales. Hoy en día, los programadores tienden a escribir un único programa que será ejecutado en todos los procesadores. Se evita escribir programas distintos para los diferentes procesadores.

Uno de los primeros proyectos mejor conocidos se realizó en los años 70 en la Cornegie-Mellon University. Constaba de 16 PDP-11 (Digital Equipmen Corporation) conectados por un “crossbar” a 16 unidades de memoria; este sistema se denominó C.m.m.p. Una evolución de esta máquina, la Cm*, incorporó multiprocesadores con “clusters” con memoria distribuida y tiempo de acceso no uniforme. No disponía de cachés.

La evolución posterior se marcó según los siguientes hitos:

- 1984: Primer multiprocesador (Synapse N+1) conectado por Bus y con Cachés con protocolo de “espionaje”.
- 1985: “Cosmic Cube”, reúne las dos direcciones de trabajo, “multicomputadores con paso de mensaje” y “multiprocesadores escalables con memoria compartida”, estructurando en “mallas” e “hipercubos”. Esta arquitectura redujo el coste de las interconexiones.
- 1986: Protocolos alternativos de coherencia de Cachés.

FIN