

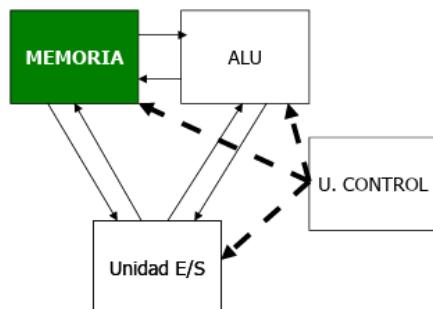
# TEMA 5 “Unidad de Memoria”

## 1. Análisis de la organización de la memoria

### 1.1. Visión global

Estructura Von Neumann:

- Unidad de Cálculo.
- Unidad de Control.
- Unidad de Memoria.
- Unidad de E/S.



### 1.2. Unidad de Memoria

Características de la Unidad de Memoria:

- Almacenamiento:
  - Programas y Datos de Usuario.
  - Programas y Datos de Sistema.

Parámetros a tener en cuenta:

- Velocidad de acceso.
- Costo por byte.
- Capacidad.

Condiciones óptimas:

- Gran capacidad (Muchos gigabytes).
- Muy rápidas (mínimo nanosegundos).
- Baratas.

## 1.3. Primera clasificación

Memoria Principal:

- Acceso directo por CPU.
- Almacena los programas que se están ejecutando actualmente.
- De tipo semiconductor:
  - De lectura/escritura (RAM) o sólo lectura (ROM).
  - Estáticas (SRAM) o dinámicas (DRAM).
- Capacidad del orden de los MB.
- Tiempos de acceso del orden de las decenas de nanosegundos.
- Costo alto y proporcional a la velocidad de acceso.

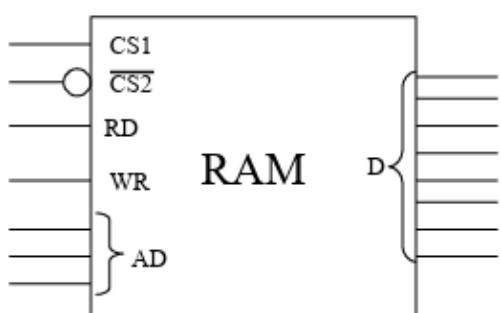
Memoria Auxiliar o Secundaria:

- Acceso a través de Interfaces.
- Almacenamiento de programas y datos no ejecutándose.
- De tipo Magnético u Óptico.
- Capacidad del orden de los GB/TB.
- Tiempo de acceso entre milisegundos y segundos.
- Baratas y proporcional a la capacidad.

## 1.4. Organización Memoria Principal

**Memoria de acceso aleatorio de Lectura/Escritura (RAM).**

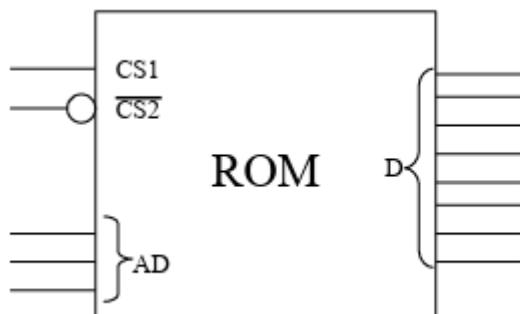
- Líneas de Dirección [AD].
  - Líneas de Datos (bidireccional) [D].
  - Líneas de Habilitación [CS1,CS2'].
  - Líneas de control de Lectura/Escritura [RD/WR].



CS1	CS2	WR	RD	Función	Bus
0	0	X	X	Inhibir	Z
0	1	X	X	Inhibir	Z
1	0	0	0	Inhibir	Z
1	0	0	1	Leer	Salida
1	0	1	X	Escribir	Entrada
1	1	X	X	Inhibir	Z

## **Memoria de acceso aleatorio de Lectura/Escritura (ROM).**

- Líneas de Dirección [AD].
  - Líneas de Datos [D].
  - Líneas de Habilitación [CS1,CS2'].

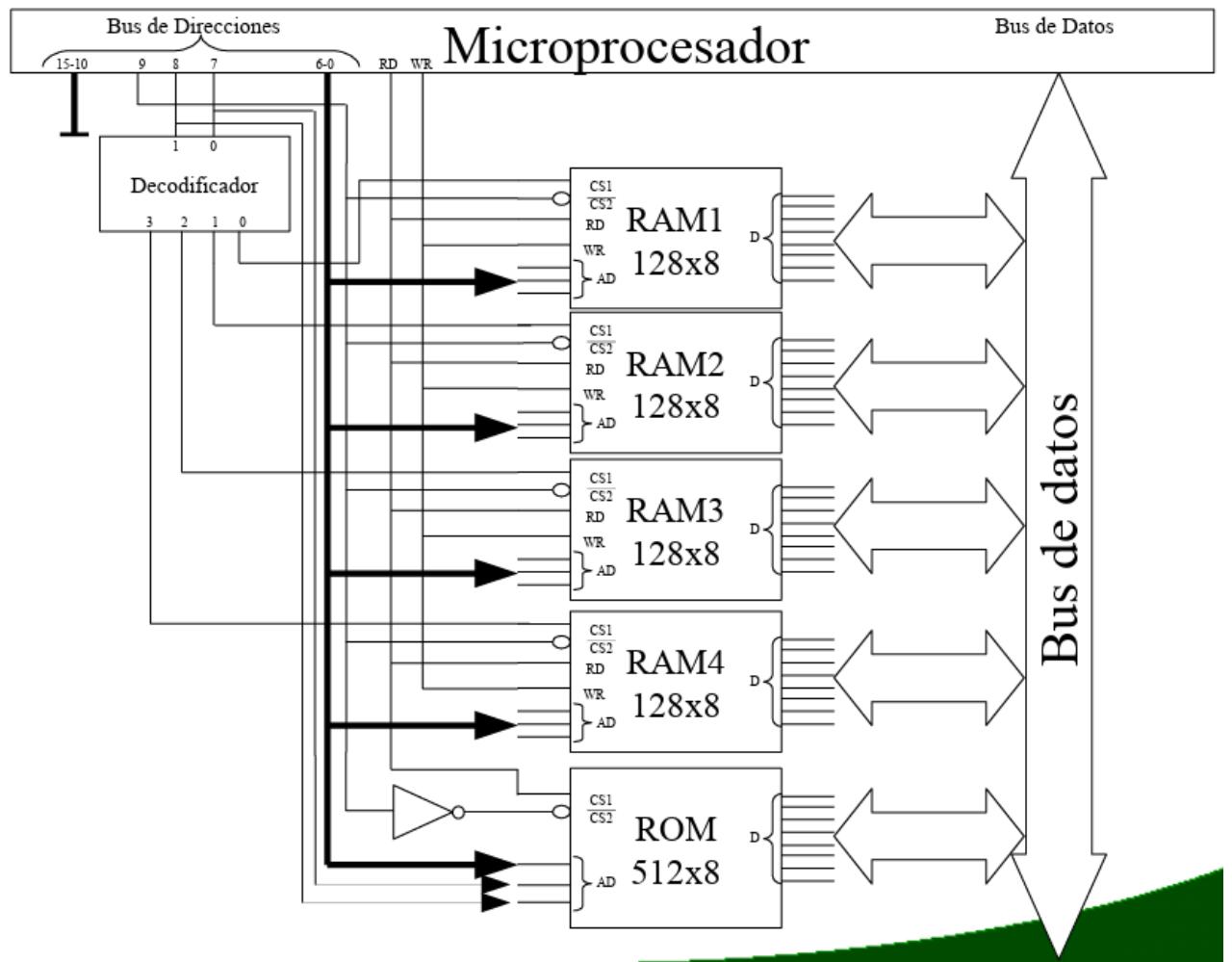


CS1	CS2	Función	Bus
0	0	Inhibir	Z
0	1	Inhibir	Z
1	0	Leer	Salida
1	1	Inhibir	Z

## Mapa de direcciones:

- Integrar en un único conjunto de direcciones varios chips de memoria RAM y/o ROM.
  - Ejemplo: Un sistema de memoria con 512 B de RAM y 512 B de ROM, a partir de pastillas de RAM de 128 B y de ROM de 512 B.

## 1.5. Organización Física de la Memoria



## 2. Jerarquía de Memoria

**Definición amplia de “Memoria”.**

- Todo componente que almacena información en una computadora.
  - Registros.
  - Memoria principal: RAM / ROM.
  - Discos Magnéticos u Ópticos.
  - Cintas USB / Memoria Flash, etc.

**Clasificación Jerárquica.**

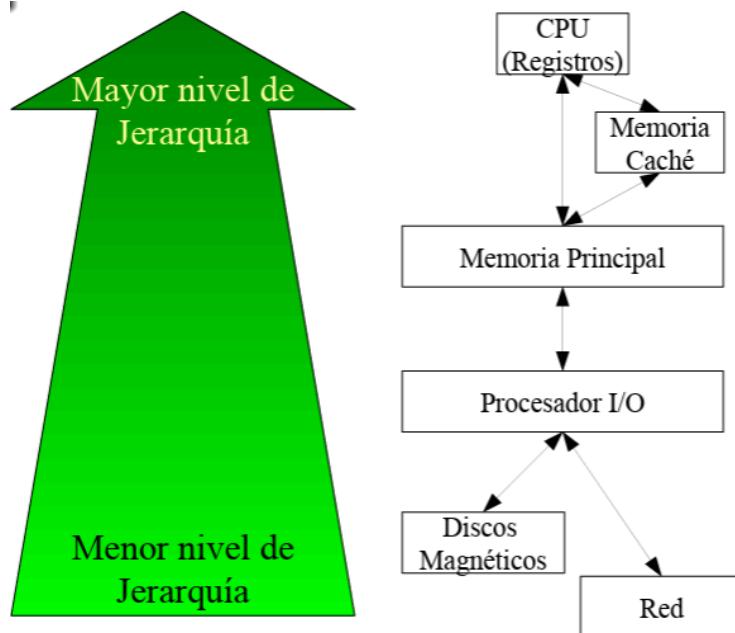
- Se tiene en cuenta características propias de cada componente para estructurarlos:
  - Valores de parámetros (Capacidad, Rapidez y Costo).
  - Forma de acceso de la CPU (Directo o a través de interfaces).

**Relación entre los parámetros de cada nivel jerárquico.**

Mayor Nivel Jerárquico => Mayor Rapidez.

Mayor Nivel Jerárquico => Mayor Costo.

Menor Nivel Jerárquico => Mayor Tamaño.



#### **Memoria de reserva (Memoria Caché).**

- Almacenar la información más utilizada.
  - Instrucciones y Datos.
- Memoria muy “rápida”.
- Al ser muy costosa, suelen ser de tamaño reducido.
- Se accede directamente desde CPU y desde Memoria Principal.

#### **Necesidad de un módulo Administrador de Memoria.**

- La amplitud obliga a su optimización.
- Programas que necesitan más memoria que la existente.
- Complejidad en Sistemas Multiusuario.

## **2.1. Memoria Asociativa**

#### **Tipo de memoria.**

- Memoria de acceso por contenido y no por dirección.
- Ejemplo: Una biblioteca: nombre libro, no su posición.

#### **Diferencias con otras memorias:**

- Acceso aleatorio por dirección.
- Acceso secuencial.
- Tipo LIFO o FIFO.

#### **Casos en los que interesa este tipo de memoria:**

- Búsquedas en bases de datos.
- Obtención de datos de una cuenta a partir del nombre.

#### **Tipo de búsqueda por contenido.**

- De forma secuencial:
  - Barrido de direcciones y comparación de contenido.
  - Lenta, aunque sencilla y económica.
    - Memoria RAM, contador y comparador.
- De forma paralela:
  - Comparación de todos los contenidos a la vez.
  - Rápida, pero compleja y costosa.

#### **Características que se pretenden:**

- Acceso de lectura por contenido.
  - Por palabra entera.
  - Por campo en una palabra.
- Muy Rápidas.
- El acceso para escritura puede ser:
  - Por dirección.
  - Se almacena en cualquier posición vacía.

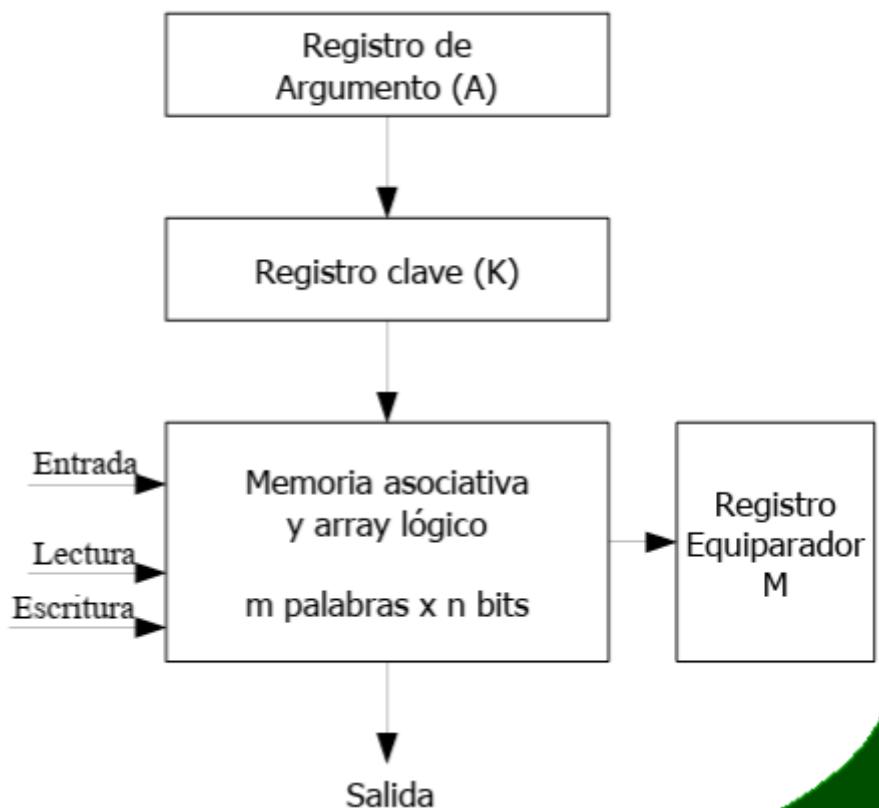
#### **2.1.1. Organización del Hardware**

Búsquedas rápidas => Circuitería compleja.

- Incluir comparaciones.
- Incluir estado de las posiciones, etc.

#### **Componentes de una Memoria Asociativa.**

- Matriz de memoria que incluye:
  - Celdas de almacenamiento ( $m \times n$ ).
  - Circuitería de comparación.
  - Registro de Argumento (A).
    - Contiene el dato o instrucción que se busca ( $n$ ).
  - Registro clave (K).
    - Para enmascarar parte del argumento de búsqueda ( $m$ ).
  - Registro Equiparador (M).
    - Indica las posiciones que tienen ese contenido ( $m$ ).

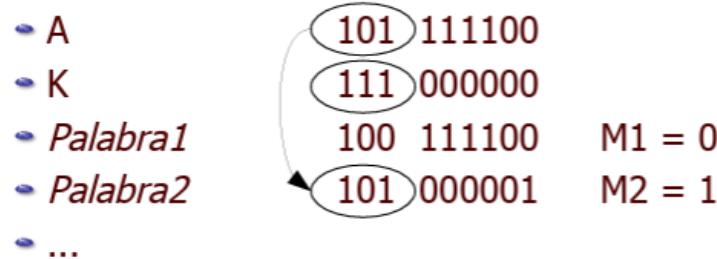


## 2.1. Memoria Asociativa

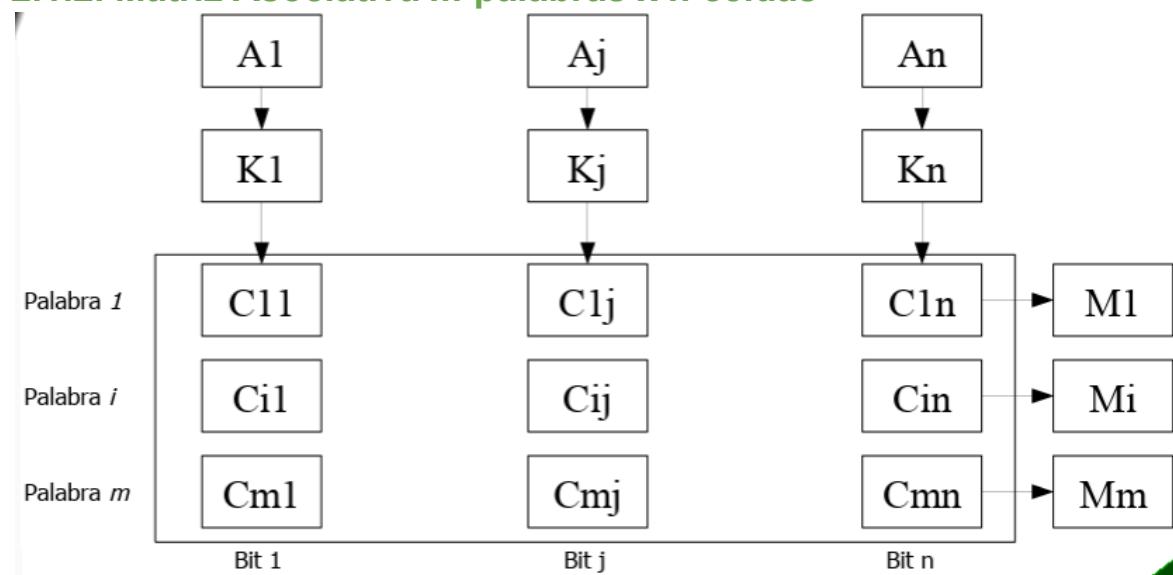
### Organización de la Matriz Asociativa.

- Conjunto de celdas asociativas ( $C_{ij}$ ).
- Registros A, K, M.

Ejemplo de búsqueda.



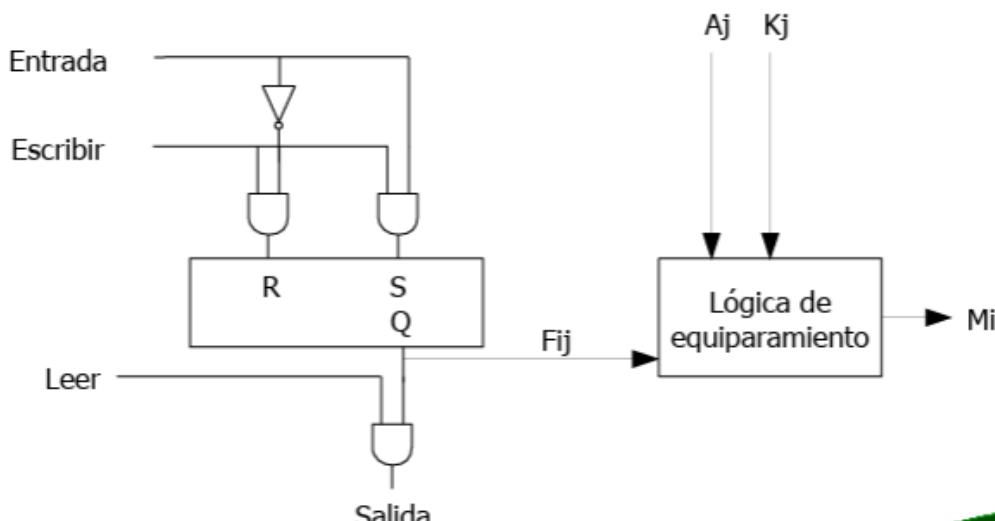
### 2.1.2. Matriz Asociativa $m$ palabras x $n$ celdas



### 2.1.3. Celda de memoria asociativa

#### Estructura de la celda básica.

- Biestable de almacenamiento.
- Lógica de equiparamiento y seguimiento.
- Lógica de escritura.



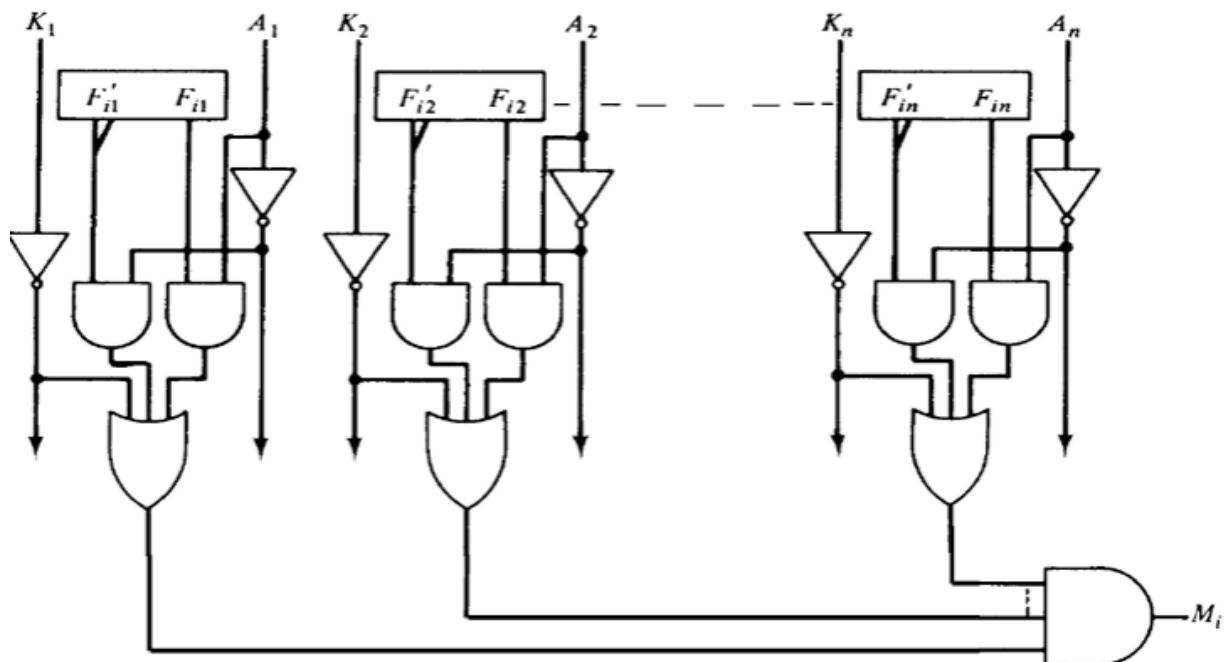
## 2.1.4. Lógica de equiparamiento

Función de igualdad:

- $x_j = A_j \cdot F_{ij} + \overline{A_j} \cdot \overline{F}_{ij}$
- $A_j$  = bit j del registro de argumento
- $F_{ij}$  = bit j de la palabra i del almacenamiento

Función de equiparación de la palabra almacenada.

- $M_i = x_1 \cdot x_2 \cdot x_3 \cdot \dots \cdot x_n$
  - $M_i$  = bit i del registro de comparación M
- Función de equiparación con enmascaramiento.
- $M_i = (x_1 + \overline{K}_1) \cdot (x_2 + \overline{K}_2) \cdot (x_3 + \overline{K}_3) \cdot \dots \cdot (x_n + \overline{K}_n)$
  - $K_i$  = bit i del registro de enmascaramiento



## 2.1.5. Lógica de Lectura

Operación de lectura en memoria asociativa.

- Se busca una información cuya etiqueta coincide con la que se presenta.
  - Información de entrada:
    - Palabra de búsqueda (n bits) en el Registro Argumento (A).
    - Palabra de máscara (k bits) en el Registro Clave (K).
      - Número de bits a comparar menor que n ( $k < n$ ).
  - Testeo de los bits del Registro M (m).
    - Si todos a 0 => Ese contenido no está en la memoria asociativa (Fallo).

- Si algún  $M_i = 1 \Rightarrow$  Ese contenido está en alguna celda de la memoria (Éxito).
  - Problema de lentitud: Lectura secuencial de las posiciones buscando  $M_i=1$ .
  - Definición de función de decisión (V).

$$V = M_1 + M_2 + M_3 + \dots + M_m$$

- Generación de la salida en la línea de Datos.

$$D_j = F_{1j} \cdot M_1 + F_{2j} \cdot M_2 + \dots + F_{mj} \cdot M_m \quad j = 1 \dots n$$

- Si  $V = 1$ , se lee  $D_j$ .
- Si  $V = 0$ , Fallo de lectura.

## 2.1.6. Lógica de Escritura

### Operación de escritura en memoria asociativa.

- Introducción de un dato en una posición que acelere la búsqueda.
- Opciones de búsqueda de una posición donde almacenar:
  - Direcciónamiento secuencial  $\Rightarrow$  Similar a un acceso aleatorio.
  - Buscando posiciones vacías:
    - Registro de Rótulos (R) de  $m$  bits (tantos como palabras).
      - $R_i = 1 \Rightarrow$  Posición ocupada.
      - $R_i = 0 \Rightarrow$  Posición libre.
    - Se testean los bits de R hasta encontrar el primer 0:
      - Se escribe la palabra.
      - Se pone el bit  $R_i$  a 1.
    - Los  $R_i$  se deben usar también como máscara:
      - Impiden la lectura de un dato no válido.

## 2.2. Memoria de Reserva (Caché)

### Localidad de Referencia.

- “Las referencias a memoria en un intervalo de tiempo ajustado suelen estar confinadas en áreas cercanas de memoria”.
  - Ejemplos:
    - Secuencialidad de los programas.
    - Bucles.
    - Subrutinas.
    - Acceso a tablas o vectores de datos.
- Acelerar la ejecución de un Proceso.
  - Colocar en una memoria rápida los datos activos e instrucciones que más habitualmente se deben ejecutar.

### Memoria Caché.

- Memoria rápida colocada entre Memoria Principal y CPU.
- Almacenamiento de Instrucciones y Datos activos.
- Tan rápida como la CPU o, al menos, más rápida que la Memoria Principal.
- Menor capacidad que la Memoria Principal.

### Funcionamiento.

- La CPU solicita una dirección  $\Rightarrow$  Se busca 1º en la Mem.
  - Caché Si está en Memoria Caché  $\Rightarrow$  Toma el contenido y continúa.

- Si no está en Memoria Caché.
  - Accede a Memoria Principal y se toma el dato.
  - Traerse de Memoria Principal a Memoria de Reserva, el bloque que contenga la dirección accedida.

### 2.2.1. Tasa de aciertos

#### Conceptos previos.

- Si la dirección está en la Caché => ACIERTO.
- Si la dirección NO está en la Caché => FALLO.

#### Tasa de Aciertos.

- Mide la cantidad media de solicitudes de direcciones con ACIERTO frente a las totales.

$$Tasa\ de\ Acierto = \frac{Aciertos}{Aciertos + Fallos}$$

#### Tiempo medio de Acceso.

- Ejemplo:
  - Memoria Principal, tiempo de acceso: 1000 ns.
  - Memoria Caché, tiempo de acceso: 100 ns.
  - Tasa de acierto = 0.9.
  - Tiempo medio de acceso =  $190\text{ ns} (100 \times 0.9 + 1000 \times 0.1)$ .

## 3. Políticas de administración de memoria

### 3.1. Mapeo de memoria

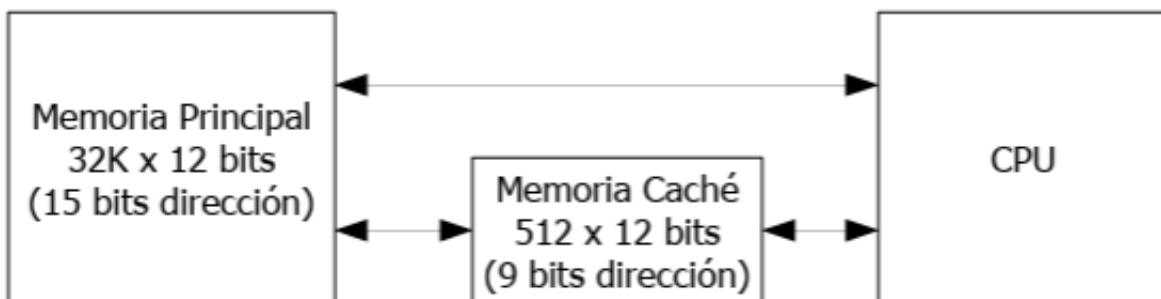
#### 3.1.1. Mapeo de Memoria Caché

Cómo relacionar las palabras de Memoria Principal y Memoria Caché.

#### Tipos:

- Mapeo Asociativo.
- Mapeo Directo.
- Mapeo Asociativo por Conjuntos.

#### Ejemplo:



### 3.1.2. Mapeo Asociativo

Implementación usando Memoria Asociativa:

- Rápida pero costosa.

Cada posición de la Memoria Caché almacena:

- Dirección.
- Datos (contenido) de dicha dirección.

Procedimiento:

- La CPU solicita una dirección de 15 bits, colocándola en el Registro de Argumento.
- Si se encuentra en la Memoria Asociativa:
  - Se proporcionan los datos.
- Si no se encuentra:
  - Acceder a Memoria Principal para traer dirección y datos.
  - Si la Caché está llena, utilizar un algoritmo de reemplazo.



### 3.1.3. Mapeo Directo

Implementada utilizando RAM de acceso aleatorio.

- Poco costosa y suficientemente rápida.
- No es demasiado eficiente.

La dirección se divide en 2 campos:

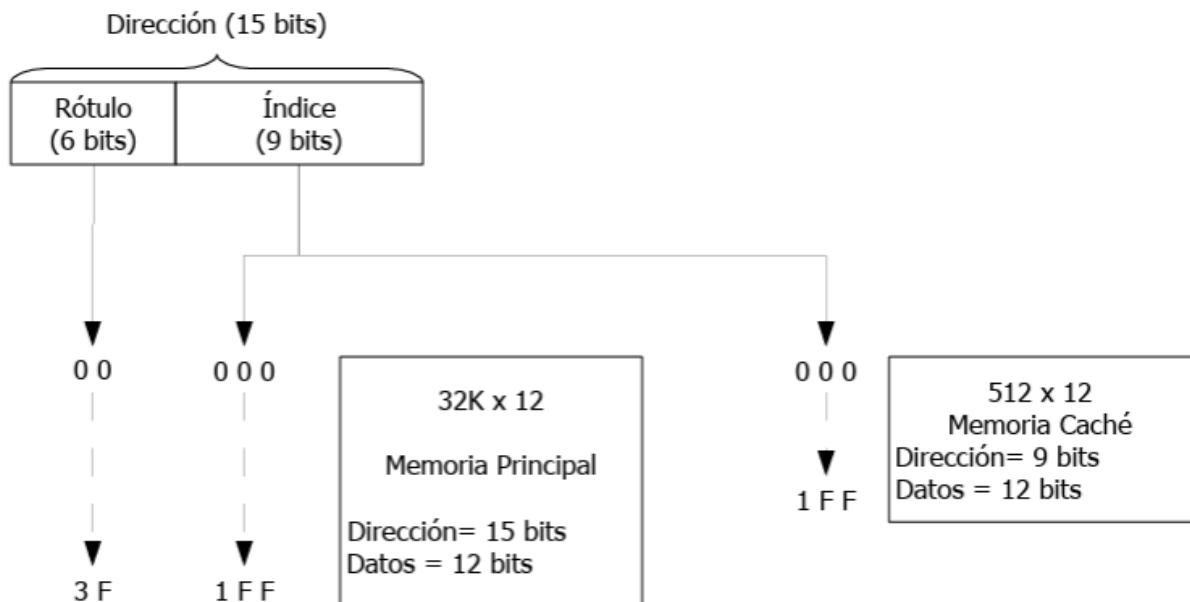
- Campo índice.
  - Número de bits que seleccionan cualquier posición de la Memoria Caché.
- Campo rótulo o de identificación.

Cada posición de la Memoria Caché.

- Campo para rótulo.
- Campo para datos.

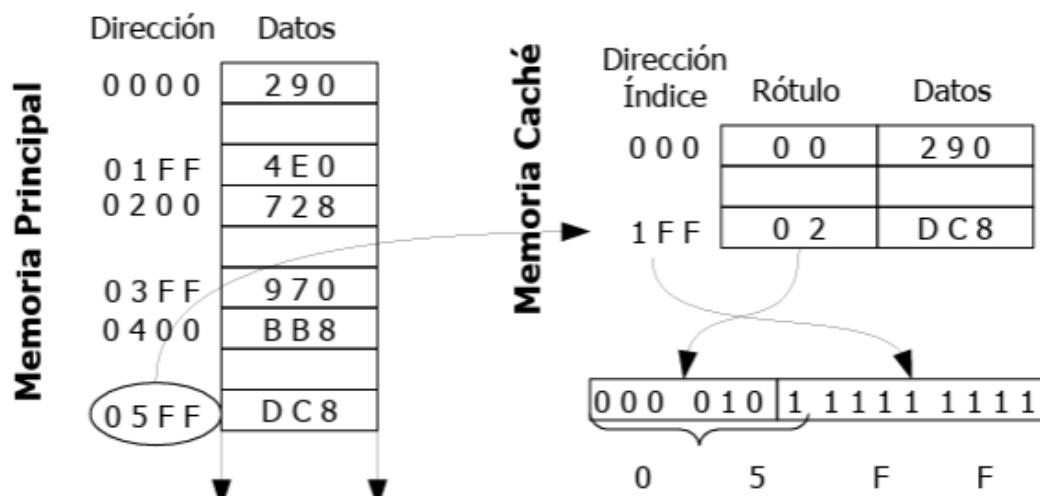
Restricciones de posicionamiento de los datos en Caché.

- La posición en Memoria Caché viene impuesta por los bits del campo Índice.
- No puede haber 2 direcciones en Caché con el mismo campo Índice.



### Búsqueda.

- Los bits de Índice seleccionan la posición en Memoria Caché.
- Se compara el rótulo de la dirección con el rótulo de la Caché.
  - Si coinciden, ACIERTO.
  - Si no coinciden, FALLO.



### Tamaño de Bloque.

- Varias palabras en un mismo bloque.
- Campos de una dirección.
  - Rótulo – Índice (Compuesto por: Bloque | Palabra dentro del bloque).
    - Bits asociados a bloque = Posición dentro de la Memoria Caché.
    - Bits asociados a palabra = Posición dentro del bloque.

### Búsqueda.

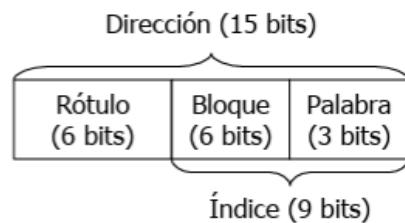
- Se busca mediante los bits de Índice.
- Se comparan los rótulos.
  - Si coinciden, se localiza la palabra.
  - Si no, se cambia el bloque completo.

### Ventajas e Inconvenientes.

- Mayor probabilidad de acierto al tener N palabras consecutivas.
- Mayor tiempo de intercambio en caso de fallo.

### Índice (9 bits).

- Bloque (6 bits): 64 bloques en M. Principal.
- Palabra (3 bits): 8 palabras por bloque.



	Índice	Rótulo	Dato	
Bloque 0	0 0 0	0 1	7 2 8	
	...			
000000xxx	0 0 7	0 1	A C 4	Contiene las direcciones desde la 0200h - 0207h
Bloque 1	0 0 8	0 6	3 3 A	
	...			
000001xxx	0 0 F	0 6	0 0 1	Contiene las direcciones desde la 0C08h - 0C0Fh
	...			
Bloque 63	1 F 8	1 1	2 3 2	
	...			
111111xxx	1 FF	1 1	0 0 1	Contiene las direcciones desde la 23F8h - 23FFh

### 3.1.4. Mapeo Asociativo por Conjuntos

Mezcla las ventajas de los dos mecanismos anteriores.

#### Definición:

- Conjunto o Vía: Agrupación de palabras con su rótulo asociados a un mismo bloque de Memoria Caché.

#### Almacenamiento:

- Como Mapeo Directo:
  - Cada bloque de Caché podrá almacenar palabras cuya dirección en Memoria Principal coincida con el Índice asociado.
- Como Mapeo Asociativo:
  - Cada bloque de Caché podrá almacenar varias palabras con rótulos distintos.

#### Búsqueda.

- Utilizando el campo Índice de la dirección física, se determina la posición en Caché.
- Mediante Memoria Asociativa, se determina si el rótulo está en dicho bloque:
  - Si está, se devuelve el dato.
  - Si no está, se lanza un algoritmo de Reemplazo.

Índice	Vía 0		Vía 1		Vía 2		Vía 3	
	Rótulo	Dato	Rótulo	Dato	Rótulo	Dato	Rótulo	Dato
000	00	728	02	BB8				
...								
1FF	11	001						

### 3.1.5. Escritura en Memoria Caché

**La Escritura en Caché necesita que se copie en Memoria Principal.**

- Falta coherencia entre lo que hay en Caché y Memoria Principal.

**Mecanismos para garantizar la coherencia de la Caché.**

- Escritura Directa.
  - Se escribe en paralelo en la Memoria Principal y en la Caché.
  - La memoria principal siempre tiene datos actualizados, a costa de enlentecer el sistema.
- Escritura diferida.
  - Se escribe únicamente en la Caché.
  - Se marca dicha palabra con un bit de “modificado”.
  - Al retirar la palabra, se mira ese bit:
    - Si está activo, copiar a M. Principal.
    - Si no, retirar.
  - Durante un cierto tiempo, hay falta de coherencia entre Caché y M. Principal.

### 3.1.6. Validez de los datos en Caché

**La Caché comienza vacía.**

- Se van rellenando las posiciones de la Caché conforme la CPU va pidiendo direcciones.
- Cuando hay que hacer reemplazos, se eliminan de la Caché.
  - Problemas de tiempo.
  - Problemas de validez del dato: ¿Con qué valor escribo?
- ¿Cómo saber si una posición está vacía?
  - Incluir un bit en cada posición, que indica la validez del dato.
  - Si se desea eliminar una posición, poner dicho bit a inválido.
- El bit de validez permite eliminar rápidamente grandes cantidades de posiciones de la Caché.
  - Procesos que finalizan su ejecución.

### 3.1.7. Sistemas de Programa Almacenado

**Sistemas de programa almacenado.**

- El programa debe estar en Memoria Principal.
  - Requisito de los programas, tamaño menor que la memoria física.
- Problemas con sistemas multiusuario y multitarea.
  - Imposibilidad de tener tanta memoria.
- En un principio, los programas se almacenan en Memoria Secundaria.

### 3.1.8. Sistemas con Memoria Virtual

**Permite la ejecución de programas de gran tamaño.**

**Los programas:**

- Se almacenan completos en Memoria Secundaria.
- Parte de las tareas se almacenan en Memoria Principal.

**La CPU:**

- Solicita datos o instrucciones utilizando direcciones virtuales.
- Requerimientos:
  - Convertir las direcciones virtuales en direcciones físicas.
  - Necesidad de Hardware específico.
- Si el dato/instrucción no está en Memoria Principal:
  - Traerlo desde Memoria Secundaria.
  - Fallos de acceso: Elevado coste temporal.

- Si no cabe, ¿qué quitar?
- Combinación Hardware/Software.
  - Software de Administración.
  - Hardware de mapeo.

### 3.1.9. Espacio de Direcciones y de Memoria

#### Espacio de Direcciones.

- Conjunto de direcciones virtuales o direcciones que genera el programa.

#### Espacio de Memoria.

- Conjunto de localizaciones o direcciones físicas de la memoria.

#### Sistemas de Programa Almacenado:

- Ambos espacios coinciden.

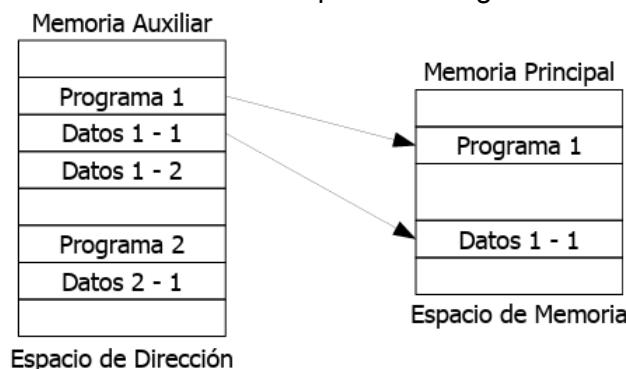
#### Sistemas de Memoria Virtual:

- Espacio de Dirección es mayor que el Espacio de Memoria.

### 3.1.10. Memoria Virtual

#### Sistema multiprogramado.

- Se cargan varios trozos de Programas o Datos.
  - No se cargan siempre en la misma posición.
  - El programa o datos no tienen que ser contiguos.



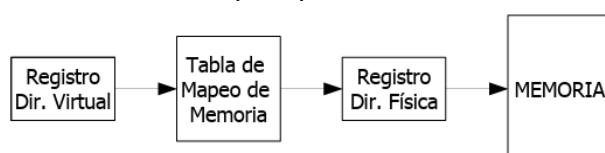
### 3.1.11. Relación entre Espacios de Memoria y de Direcciones

#### Necesidad de traducir direcciones virtuales a físicas.

- ¿Dónde se ha cargado una determinada dirección virtual en memoria principal?
- Se debe hacer automáticamente para cada dirección accedida por la CPU.

#### Tabla de Mapeo de Memoria.

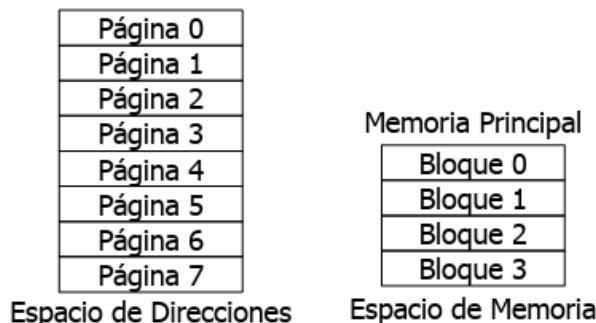
- Almacenada en memoria separada.
  - Módulo de memoria adicional necesario.
    - Tiempo de acceso extra.
- Almacenada en memoria principal.
  - Dos accesos de memoria.
    - Mucho más lenta.
- Mediante memoria Asociativa.
  - Utilización de memoria rápida para dicha traducción.



### 3.1.12. Mapeo de Dirección Virtual

#### Mapeo de Dirección.

- División del Espacio de Direcciones y de Memoria en trozos:
  - Página** para el Espacio de *Direcciones*.
  - Bloque** para el Espacio de *Memoria*.
  - Ambos con tamaño fijo e igual.
- Programas divididos también en **páginas**.
- Ej. Páginas 1KB.
  - Programa 8KB: 8 páginas => 13 bits dirección virtual.
  - Memoria Física 4KB (4 bloques) => 12 bits de dirección física.



#### 3.1.12.1. Direcciones Virtuales

##### Estructura de una dirección virtual:

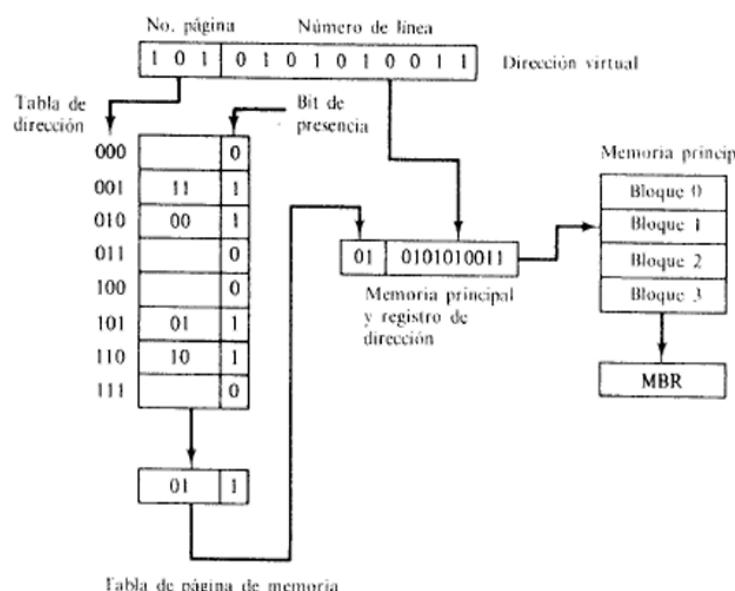
- P bits más significativos: N° de página.
- Resto de bits: N° de línea dentro de dicha página (posición).

##### Ejemplo:

- 101 0101010011.
- Página: 5.
- Línea: 339.

##### Traducción por Tabla de página de memoria.

- Tabla que relaciona número de página con número de bloque.
- Se añade un bit de presencia (está en memoria física o no).
- Número de línea igual dentro del bloque de memoria física.
- Dirección física: N° Bloque + N° Línea.



### 3.1.13. Tabla de Página de Memoria

#### Inconvenientes:

- Tamaño: Se necesitan tantas posiciones como páginas haya.
- Acceso:
  - Penalización si la página no está en Memoria Principal.
  - Acceso a Memoria Auxiliar.
  - Algoritmos de Reemplazo: ¿qué bloque quito?

#### Mejoras:

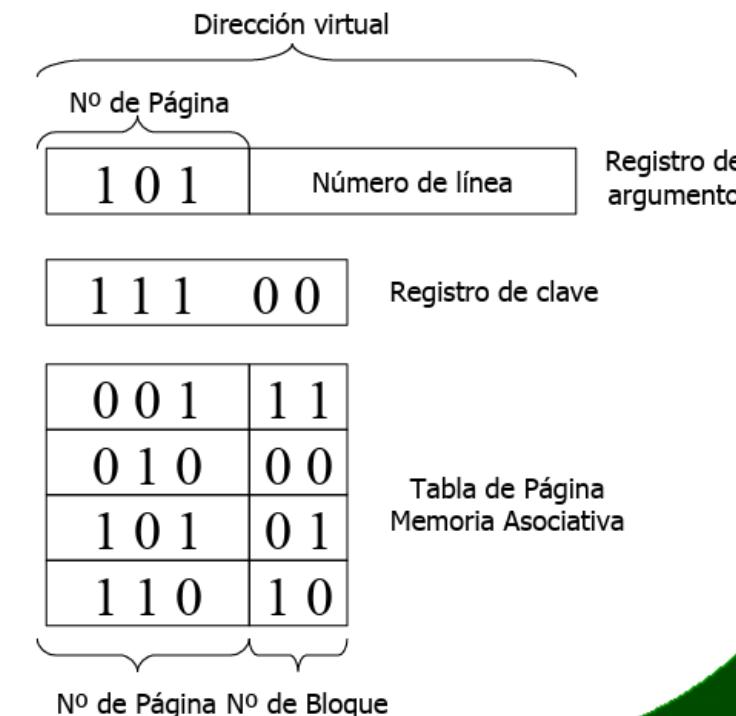
- La tabla con tantas posiciones como bloques (menor número).
- En cada posición de la tabla está el nº de página que contiene dicho bloque.
- Problemas de búsqueda:
  - Determinar si una determinada página está presente: búsqueda secuencial bloque por bloque => Lento.

### 3.1.14. Tabla de Página de Memoria Asociativa

#### Mejora en la búsqueda:

- Tabla con tantas posiciones como bloques en memoria física.
- Realizar la búsqueda por contenido (Memoria Asociativa).
  - Muy rápida.
- Cada palabra de la Tabla de Página de Memoria contiene 2 campos:
  - Nº de página.
  - Nº de bloque.
- Detección rápida si está presente una página:
  - ACIERTO => Generar dir. física:
    - Nº de bloque + Nº de línea.
  - FALLO => Reemplazo de página:
    - Algoritmos de reemplazo.

#### Mismo ejemplo anterior:



### 3.1.15. Administrador de memoria

El administrador de memoria debe decidir:

- Qué página retirar cuando hay un fallo de página.
- Cuándo se transfiere una página de Memoria Secundaria a Principal.
- En qué bloque de Memoria Principal se coloca una página.
- Cargar de manera inicial algunas páginas de un programa en Memoria Principal:
  - Indicación de la posición en la Tabla de Página.

### 3.1.16. Fallos de Página

Fallo de página:

- Se produce cuando la CPU se refiere a una página que se encuentra en Memoria Secundaria.

Acciones en caso de Fallo de Página:

- Suspender la ejecución del proceso actual (estado de espera).
- Lanzar un proceso de E/S en un procesador externo (de E/S).
  - Realizar la operación de transferencia de la página requerida a Memoria Principal.
- Al finalizar la transferencia de memoria, activar el proceso.

Transferencia de Memoria Secundaria a Principal:

- Si la Memoria Principal está llena, retirar una página.
  - Aplicar un algoritmo de reemplazo.
  - Si la página ha cambiado, copiarla a Memoria Secundaria.
- Transferencia de Página y actualización de la Tabla de Página.

### 3.1.17. Algoritmos de Reemplazo

Determinar qué página retirar de Memoria Principal.

- FIFO (First In, First Out): Elimina la página que entró primero.
  - Necesidad de una pila (FIFO) por páginas.
- LRU (Least Recently Used): Elimina la página usada menos recientemente.
  - Necesidad de un contador asociado a cada página.
    - Incremento periódico cada cierto tiempo.
    - Reseteo del contador cada vez que se accede a una página.
  - Se elige la página que tiene el valor del contador más alto.
- LFU (Least Frequently Used): Elimina la página usada menos frecuentemente.
  - Necesidad de un contador asociado a cada página.
    - Incremento cada vez que se referencia la página.
  - Se elige la página que tiene el valor del contador más bajo.

### 3.1.18. Memoria Caché vs. Memoria Virtual

Memoria Caché.

- Contiene las Instrucciones y Datos utilizados más frecuentemente.
- Acceso directo por la CPU.
- Tiempo de acceso de 5 a 10 veces más reducido.
- Tamaño de bloque pequeño (entre 4 y 16 palabras).
- Administración Hardware.

Memoria Virtual.

- Contiene (en Mem. Secundaria) las partes de código y de datos de un programa no usados en ese momento.
- Acceso a través de un proceso de E/S.
- Tiempo de acceso de 5 a 10 veces más grande.
- Tamaño de bloque grande (entre 64 y 4K palabras).

- Administración es parte Hardware y parte Software.

### 3.1.19. Unidad de Manejo de Memoria

**Módulo de Administración de Memoria.**

- Hardware de administración.
- Software de administración => Módulo del Sistema Operativo.

**Objetivos.**

- Partición y recolocación de las tareas.
- Traducción de Dir. Lógicas a Dir. Físicas.
- Compartición de tareas comunes entre usuarios.
- Protección del acceso a la información.
  - Entre usuarios.
  - Con respecto a otras funciones del Sistema.

### 3.1.20. Partición de los procesos

Trocear los procesos.

- Página como unidad de partición.
  - No tiene coherencia lógica.
    - Una rutina puede ocupar varias páginas.
    - Datos independientes pueden encontrarse en la misma página.
- Segmento como unidad de partición.
  - “Conjunto de instrucciones y/o datos que están relacionados entre sí de manera lógica”.
  - Cierta coherencia lógica.
    - Se agrupan estructuras de código o de datos (bucles, funciones, matrices, etc.).
  - Generados por el programador, por el compilador o por el S.O.

### 3.1.21. Programas Segmentados

**Dirección Lógica.**

- Dirección generada por un Programa Segmentado.
- Parecida a la Dirección Virtual.
  - Los segmentos pueden tener longitud variable, las páginas tienen longitud fija.
  - La Dirección Lógica puede ser mayor, igual o menor que la Dirección Física.

**Mapeo de Direcciones Lógicas a Direcciones Físicas.**

- Mapeo de Página Segmentada.
- Mapeo por Registro de Segmento.

### 3.1.22. Mapeo de Página Segmentada

**La longitud del segmento es variable.**

- Imposibilidad de predecir los bits necesarios de dirección.
- Se divide en Páginas que sí tienen igual tamaño.
- La longitud de un segmento se asocia a un nº de páginas.

**Campos de la Dirección Lógica.**

- Campo de Segmento (nº de segmento).
- Campo de Página (nº de página del segmento).
- Campo de Palabra (posición en la página).

**Traducción de Direcciones Lógicas a Direcciones Físicas.**

- Segmento | Página | Palabra => Bloque | Palabra.

## **Tabla de Segmento.**

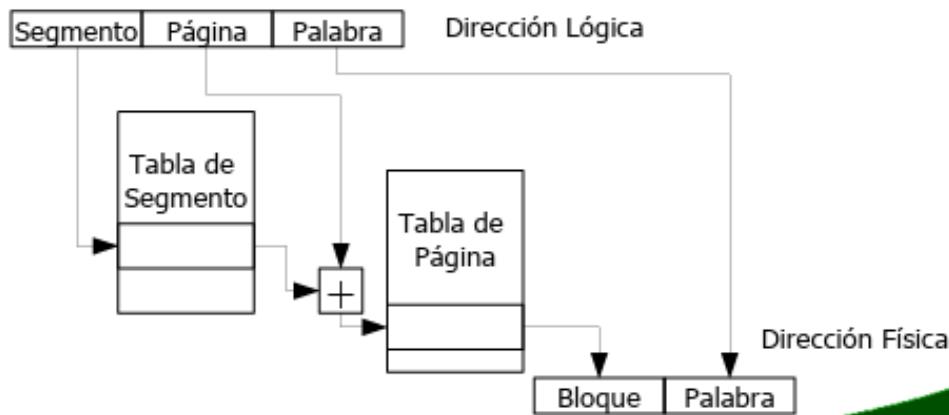
- Direccionada por nº de Segmento.
  - Determinar posición de la Página 0.
    - Sumar Página 0 del Segmento + Nº de Página de la Dirección Lógica.

## **Tabla de Página.**

- Utilizar el valor anterior (suma Pág. 0 + N° de Pág.).
  - Determinar Bloque en Memoria Principal.

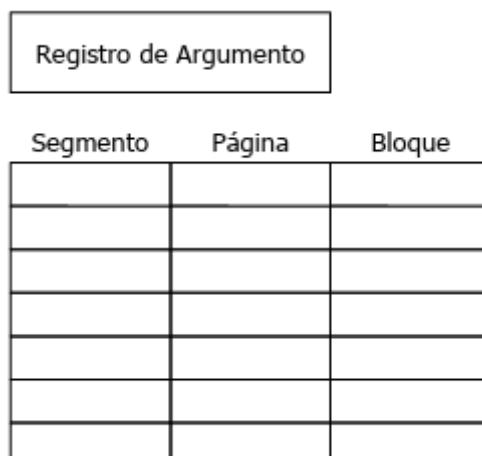
## Dirección Física.

- Concatenar al valor del Bloque anterior, la Palabra de la Dirección Lógica.



## **Ubicación de las Tablas:**

- En Memoria Aleatoria.
    - Dos memorias pequeñas adicionales.
    - En Memoria Principal.
    - Mecanismo muy lento => 3 accesos a RAM.
  - En Memoria Asociativa.
    - Como entrada de equiparación.
      - Determinar que par (Segmento ; Página) está presente en la Tabla.
      - Salidas:
        - Si está presente, se devuelve el valor del Bloque.
        - Si no, se dispara un algoritmo de reemplazo.
    - Mecanismo muy rápido => 1 único acceso a Memoria Asociativa.



## Ejemplo

### Dirección Lógica: 20 bits.

- Segmento: 4 bits (0..15).
- Página en Segmento: 8 bits (0..255).
- Palabra dentro de Página: 8 bits (0..255).
- Tamaño posible de los Segmentos.
  - Más pequeño: 1 Página (256 palabras).
  - Más grande: 256 Páginas (64 K palabras).

### Dirección Física: 20 bits.

- Bloque: 12 bits (0..4095).
- Palabra dentro de Bloque: 8 bits (0..255).

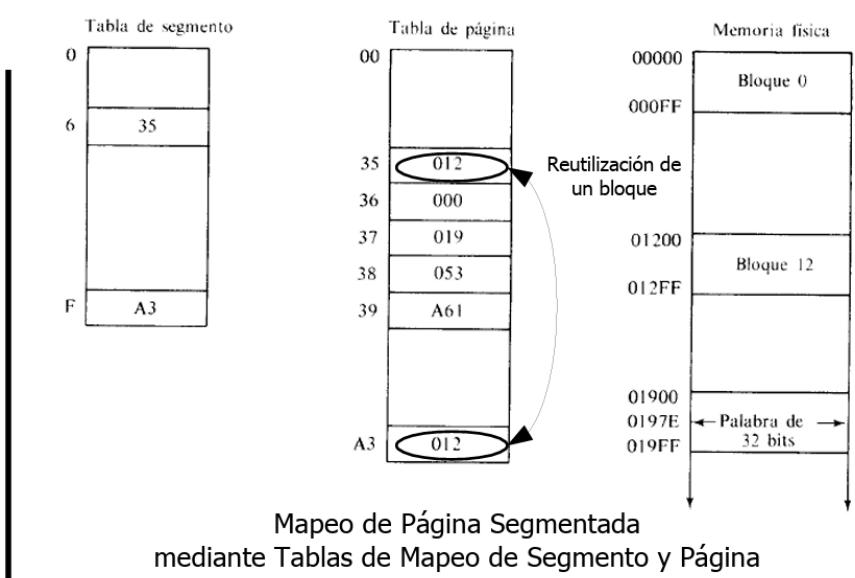


### Programa.

- Nº de Páginas: 5.
- Direcciones Lógicas:
  - 60000h – 604FFh.

Segmento	Página	Bloque
6	00	012
6	01	000
6	02	019
6	03	053
6	04	A61

Mapeo de Página Segmentada  
utilizando Memoria Asociativa



### **3.1.23. Sistema de Administración de Memoria Sistemas Segmentados**

#### **Utilizando Sistemas Segmentados:**

- Asignar un número variable de Páginas a cada uno de los Segmentos.
- Cada Página puede estar mapeada en cualquier Bloque de la Memoria Física.
- Las Páginas pueden moverse a otros Bloques según necesidades.
  - Simplemente actualizando el nº de Bloque en la Tabla de Página.
- Los Segmentos pueden crecer o decrecer sin afectar a otros.
- Segmentos diferentes pueden usar el mismo Bloque.
  - Varias instancias de un mismo programa.

## **3.2. Protección de la memoria**

**Prevenir accesos no autorizados a ciertas zonas de memoria.**

#### **¿Dónde se establece la protección?**

- En la Dirección Física.
  - Cada Bloque tiene unos bits de protección.
  - Al mover una Página se actualizan dichos bits.
- En la Dirección Lógica.
  - Se incluye información en el descriptor de Segmentos.

#### **Descriptor.**

- Contenido de cada entrada de una Tabla de Segmento o de cada Registro de Segmento.
- Estructura: Dirección base | Longitud | Protección.

### **3.2.1. Descriptor de Segmento**

#### **Contenidos habituales.**

- Dirección base.
  - Dirección de la Tabla de Páginas (para Mapeo de Página Segmentada).
  - Dirección del Bloque Base (para Mapeo por Registros de Segmentos).
- Longitud.
  - Tamaño del Segmento.
    - Comparándolo con el nº de Página, evita accesos fuera de límite.
- Protección.
  - Permisos de acceso a dicho Segmento.
    - Para Mapeo de Página Segmentada, cada Página tendrá una propia.
  - Derechos típicos:
    - Privilegios de lectura y escritura completa.
    - Sólo lectura (protección contra escritura).
    - Ejecución (protección de programa).
    - Sistema solamente (protección de Sistema Operativo).