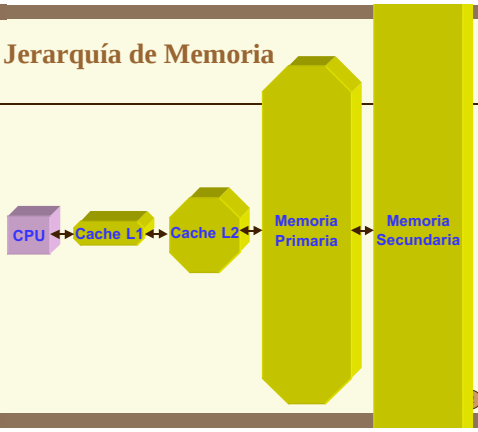
## Sistema de Memoria



## Jerarquía de Memoria






## El Cache

- ✓ El mecanismo más utilizado para compensar por la diferencia en velocidades entre el CPU y la memoria primaria (factor 1 a 10 típico)
- ✓ Es una memoria, mucho más rápida y mucho más pequeña que la memoria primaria
- ✓ Mantiene los valores de las localizaciones de memoria más frecuentemente accesados por el CPU
- ✓ Puede ser invisible para el CPU

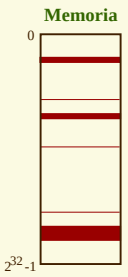


## Efectividad del Cache

- ✓ El cache es efectivo por dos razones principales:
  - *Localidad temporera - En un período de tiempo dado los accesos a memoria se concentran en un grupo relativamente pequeño de operandos.*
  - *Localidad de espacio - En un período de tiempo dado los accesos a memoria se concentran en grupos de operandos alojados en localizaciones adyacentes.*

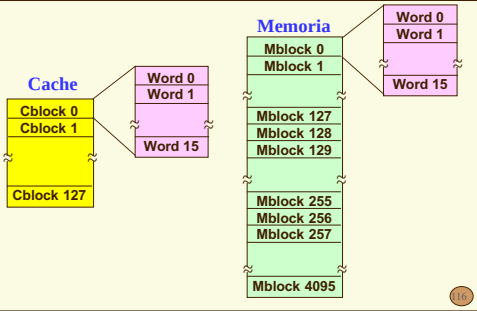


## Patrón de Accesos a Memoria Generado por Programas

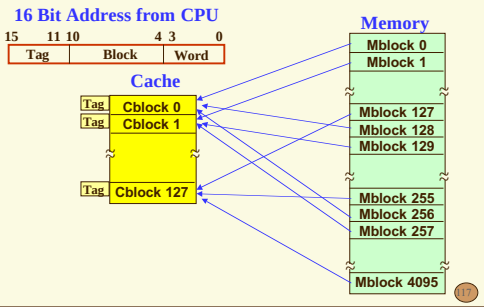





## Organización de un Cache de 2048 Words y una Memoria de 64K Words

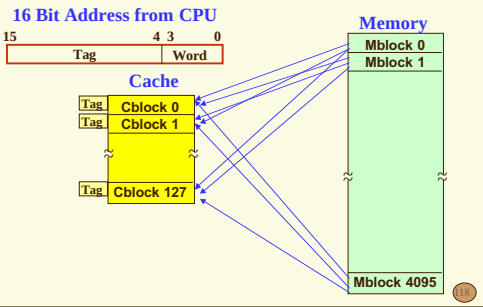


## Direct Mapped Cache



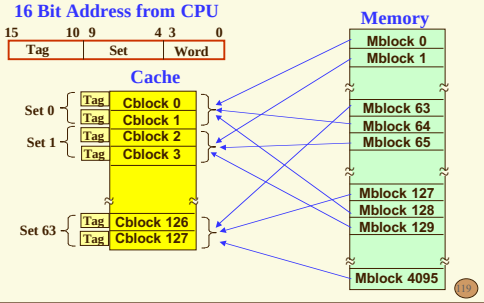

117

## Fully Associative Cache



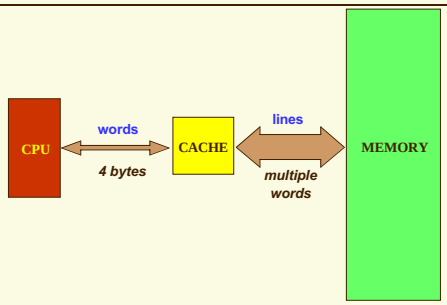

118

## Block Set Associative Cache




119

## Transferencia de Datos en Caches




120

## Cache Miss

**Cuando no existe en el cache una copia de la localización generada por el CPU**

- *Se utiliza un algoritmo de reemplazo para escoger un bloque del cache (la víctima) para alojar el bloque de memoria donde se encuentra la localización generada por el CPU.*
- *Si el bloque víctima ha sido alterado por el cache, entonces ese bloque se escribe nuevamente a la memoria primaria.*
- *Se transfiere al cache el bloque de memoria que contiene la localización generada por el CPU.*

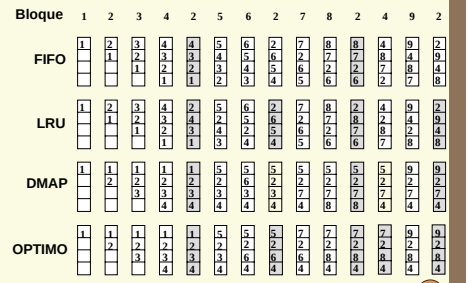
121

## Algoritmos de Reemplazo

- ✓ First-In-First-Out (FIFO) - Reemplaza el bloque que llegó primero al cache.
- ✓ Least Recently Used (LRU) - Reemplaza el bloque del cache que más tiempo lleva sin ser accesado.
- ✓ Least Frequently Used (LFU) - Reemplaza el bloque que menos veces se ha accesado.
- ✓ Random - Reemplaza un bloque al azar.
- ✓ Optimo - Reemplaza el bloque del cache que más tiempo va a pasar en ser accesado.

122

## Ejemplos de Algoritmos de Reemplazo




123

## Consistencia de la Data en Memoria

- ✓ **Debido a mecanismos como DMA la data en el cache no siempre es igual a la correspondiente data en memoria primaria (memoria y cache no son consistentes).**
- ✓ **Para mantener la consistencia de la data es necesario refrescar la memoria periódicamente con los bloques alojados en el "cache" que han sido alterados por el CPU.**

124

## Mecanismos para Mantener Consistencia entre Memoria y Cache

- ✓ **Write-back**
  - *Se refresca la memoria cuando el bloque, que ha sido alterado, es reemplazado del cache.*
  - *Más conveniente para caches en microprocesadores, pues reduce el tráfico entre memoria primaria y el microprocesador.*
- ✓ **Write-through**
  - *Se refresca la memoria cuando se altera el bloque en el cache.*
  - *Más conveniente para sistemas que permiten accesos a la memoria mediante DMA, pues mantiene la memoria consistente con el cache todo el tiempo.*

125

## Mecanismos para Mantener Consistencia entre Memoria y Cache

- ✓ **Snooping**
  - *El cache es notificado cuando alguna unidad trata de accesar directamente un bloque de memoria que tiene una copia en el cache.*
  - *El cache toma una acción dependiendo del tipo de acceso (read/write) y del tipo de snooping.*
  - *Existen dos tipos de snooping*
    - **Write- invalidate**
    - **Write-update**

126

## Snooping Write-Invalidate

- ✓ **Write Access:**
  *La otra unidad escribe en el bloque de memoria directamente y el correspondiente bloque en el cache se invalida.*
- ✓ **Read Access:**
  *Si el bloque en el cache difiere del correspondiente bloque en memoria el cache lo envía a memoria y luego la otra unidad procede a accesarlo.*

127

## Snooping Write-Update

- ✓ **Write Access:**
  *La otra unidad escribe directamente en el bloque de memoria y se envía una copia al cache.*
- ✓ **Read Access:**
  *Si el bloque en el cache difiere del correspondiente bloque en memoria el cache lo envía a memoria y luego la otra unidad procede a accesarlo.*

128

## Prefetching

**Se alojan bloques en el cache antes de que se necesiten**

- *Se puede implementar vía software mediante una instrucción de prefetch*
  - (El compilador debe insertar en el código estas instrucciones)
- *Puede tener los siguientes efectos:*
  - mejorar el hit ratio
  - resultar en transferencias innecesarias cuando los bloques no son accesados por el programa
  - reducir el hit ratio

129

## Lockup-Free Caches

- ✓ Permiten acceso al CPU aún mientras procesan uno o más misses
- ✓ El CPU no se estanca cuando ocurre un miss
- ✓ Reducen pipeline interlocks causados por accesos a memoria

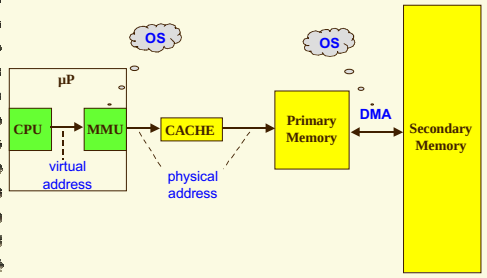
130

## Memoria Virtual

- ✓ Permite al sistema operativo administrar la transferencia de data entre memoria primaria y memoria secundaria.
- ✓ Provee un mecanismo al sistema operativo para proteger la data entre usuarios y restringir su uso.
- ✓ Permite la ejecución de programas más grandes que la memoria primaria físicamente instalada.
- ✓ Facilita el multiprocesamiento.

131

## Organización de un Sistema de Memoria con Memoria Virtual



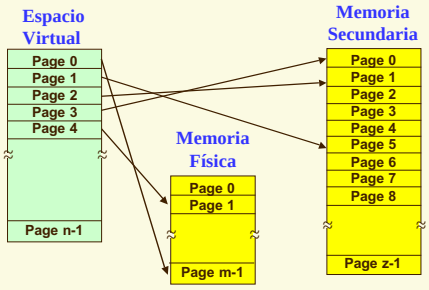

132

## Paging

- ✓ Las transferencias entre memoria primaria y memoria secundaria se hacen en bloques de words denominados páginas
- ✓ Los tamaños típicos de páginas son de 4KB a 16K (32KB a 64KB para nuevos micros)
- ✓ La páginas se transfieren a memoria primaria en demanda (cuando el programa las necesita).
- ✓ Utiliza un tabla de páginas ubicar las páginas de un programa en memoria física o en memoria secundaria.
- ✓ Cuando las páginas no se encuentran en memoria física se genera un "page fault" (un interrupt que entrega control al sistema operativo).

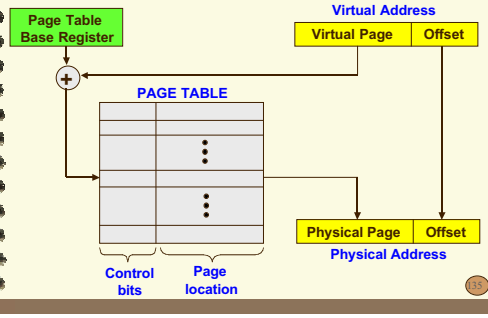
133

## Espacio Virtual y Espacio Físico




134

## Traducción de Localizaciones




135

## Page Fault

- ✓ Genera un interrupt que entrega control de ejecución al sistema operativo
- ✓ Cambia de modo de operación de usuario a modo de operación supervisor
- ✓ Realiza un procedimiento para traer a memoria física la página que contiene la localización virtual que generó el programa

136

## Procedimiento del Sistema Operativo para Atender Page Faults

1. *Detiene la ejecución del programa y pone a correr otros programas (context switch).*
2. *Selecciona mediante un algoritmo de reemplazo una página de memoria física para ubicar la página solicitada.*
3. *Si la página a ser reemplazada ha sido alterada por el CPU, instruye a un puerto para transferirla a la unidad de almacenamiento secundaria correspondiente.*
4. *Utiliza la tabla de página del programa para ubicar la página en la unidad de almacenamiento secundaria.*
5. *Instruye a un puerto a transferir la página solicitada a la página de memoria física seleccionada.*
6. *Pone a correr el programa luego que la página es transferida a memoria física.*

137

## Bits de Control

- ✓ **Valid Bit** - Indica si la página virtual se encuentra alojada en memoria física.
- ✓ **Dirty Bit** - Indica si la página virtual ha sido alterada en la memoria física.
- ✓ **Read Bit** - Indica si la página puede ser le ída por el programa que trata de tener acceso a la misma.
- ✓ **Write Bit** - Indica si la página puede ser escrita por el programa que trata de tener acceso a la misma.
- ✓ **Execute Bit** - Indica si el código alojado en la página puede ser ejecutado por el programa que intenta ejecutarlo.

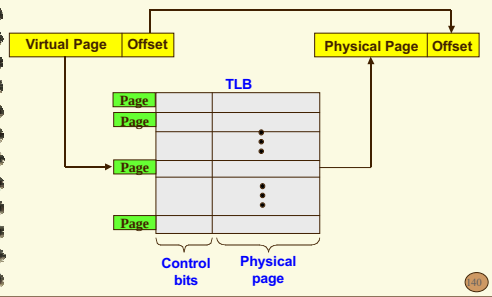
138

## Translation Lookaside Buffer (TLB)

- ✓ Es un cache "fully associative" de traducciones de páginas virtuales a físicas
- ✓ Acelera la traducción de páginas

139

## Traduciones del TLB




140

## Segmentación

- ✓ Los programas se rompen en secciones de largo variable que se pueden ubicar en cualquier lugar de memoria
- ✓ La ubicación de segmentos en memoria se puede hacer utilizando estratégias como “first fit”, “best fit” o “worst fit”
- ✓ Cada proceso tiene una tabla de segmentos
- ✓ Es propenso a fragmentación

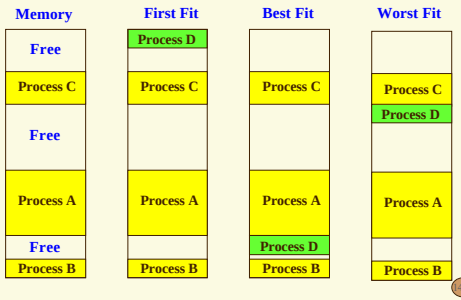
141

## Estrategias de Ubicación de Segmentos

- ✓ First Fit - ubicar el segmento en el primer espacio libre de memoria en que quepa
- ✓ Best Fit - ubicar el segmento en el espacio libre de memoria más pequeño en que quepa
- ✓ Worst Fit - ubicar el segmento en el espacio libre de memoria más grande en que quepa

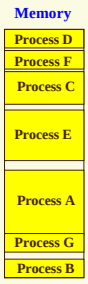
142

## Efecto de Estrategias de Ubicación de Segmentos




143

## Fragmentación




144