Accesos a memoria:

Escritura: Transfiere datos de la CPU a la memoria, es una operación de almacenamiento (store).

Lectura: Transfiere datos de la memoria a la CPU, es una operación de carga (load).

Un bus es una colección de cables que llevan direcciones, datos, y señales de control.

En una lectura, la CPU ubica la dirección en el bus de memoria, la memoria lee la dirección del bus, lee el dato y lo ubica el bus, la CPU lee el dato del bus y lo copia en el registro.

En una escritura, CPU ubica la dirección en el bus de memoria, la memoria la lee y espera el dato en el bus, la CPU ubica el dato en el bus, la memoria lee el dato del bus, y lo copia en la memoria previamente leída.

**RAM (Random Access Memory):** Decimos que es la **Memoria Principal**. Permite la lectura y escritura de cualquier dato sin importar su ubicación en la memoria, es empaquetada en chips, el almacenamiento básico es una celda con un bit por celda, se compone de muchos chips.

**SRAM (Static):** Los bits se almacenan como estados estables, 6 transistores por bit, mantiene los bits mientras tenga energía. Caches L1, L2, L3.

**DRAM (Dinámica):** Los bits se almacenan como carga en un capacitor, 1 transistor y 1 capacitor por bit. Memoria Principal.

**Almacenamientos no volátiles:** Mantienen la información incluso si se los desenchufa, 100 a 1000 veces más rápidos que los discos magnéticos. Por lo general, tienen un uso más específico. SSD, caches de otros discos mas lentos.

Cache: Es un dispositivo de almacenamiento de datos más rápido y de menor capacidad que actúa como staging área de un subconjunto de datos almacenados en un dispositivo mas lento y de mayor capacidad. Utiliza la idea fundamental de la **jerarquía de memorias**, la cual dice, para cada k, el dispositivo de menor capacidad y de mayor velocidad en el nivel k (Lk) sirve de cache para el dispositivo en el nivel k + 1 (Lk+1), es decir, el software tiende a acceder con mayor frecuencia a los datos del nivel k que a los del nivel k + 1.

Bloque o línea: Unidad de copiado, puede abarcar varias palabras.

Hit: Dato pedido esta presente en el nivel superior

Miss: El dato no se encuentra.

Hit ratio = hits/accesos

Miss ratio = 1 – hit ratio

Tiempo de acceso: Tiempo que tarda el procesador en obtener una línea de cache.

Penalización por fallo: Tiempo adicional requerido debido a un miss.

Principio de Localidad: Los programas tienden a acceder a una parte reducida de su espacio de memoria en un tiempo acotado, es decir, utilizan instrucciones o datos en direcciones cercanas o iguales a las usadas recientemente.

**Temporal:** Elementos recientemente accedidos sean reutilizados (direcciones iguales).

**Espacial:** Elementos que se encuentran cerca de uno accedido recientemente, suele ser utilizado (direcciones cercanas).

Cache Hit: Se pide un dato de la memoria, el dato esta en el bloque de la cache, entonces no se accede a la memoria.

Cache Miss: Se pide un dato de la memoria, el dato NO esta en el bloque de la cache, entonces se le pide el boque a la memoria.

Almacenamiento en Cache: Ocurren dos cosas: Al bloque se lo ubica en la memoria cache, y lo que estaba antes se lo reemplaza. Para esto se necesita de una **Política de ubicación (placement policy):** Determina donde se ubica el bloque dentro de las opciones. **Política de reemplazo (replacement policy):** Determina que bloque se reemplaza.

Tipos de Misses en la cache:

**Cold Misses:** Ocurren porque la cache esta vacía, y se da la primera vez que se referencia un bloque.

**Fallos por capacidad:** Ocurren cuando el conjunto de bloques activos en la cache (**working set**) es mayor que la cache.

**Fallos por conflictos:** Ocurren cuando, siendo la cache lo suficientemente grande, mas de un bloque se quiere ubicar en la misma posición.

¿Como se calcula el tamaño de la cache?

**C** (tamaño de la cache) = **S** (numero de sets) \* **E** (líneas por sets) \* **B** (bytes por bloque)

Dirección de una palabra:

|  |  |  |
| --- | --- | --- |
| t bits (tag) | s bits (set index) | b bits (block offset) |

Proceso de Lectura:

1. Ubicar el set
2. Comparar tags
3. Si hay tag coincidente es Hit, sino hay que ubicar el dato.

Proceso de Escritura:

Políticas cuando hay hit:

**Escritura inmediata (write through):** Escribe el dato en la memoria en el momento.

**Post escritura (write back):** Escribe el dato en memoria cuando se desaloja el bloque (requiere un bit extra, el dirty bit).

Políticas cuando hay Miss:

**Write allocate:** Se carga la línea en la cache, y se escribe en la cache.

**No write allocate:** Se escribe directamente en la memoria.

Cualquier combinación es posible, pero normalmente se hace write through / no write allocate o write back / write allocate.

Hay que tener en cuenta que si se utiliza write back / write allocate se va a necesitar un bit más, este bit es para el dirty bit, que indica si el bloque fue modificado (es decir, si se escribió en el), si el bloque fue modificado, a este lo escribimos en la memoria antes de desalojarlo de la cache.

Código que sea amigable con la cache:

Hacer rápido el caso común, hacer foco en los ciclos de las funciones principales.

Minimizar los misses en los ciclos internos, esto puede ser referenciado variables repetidamente (localidad temporal), o usando el patrón de acceso de 1 paso (localidad espacial).

**Throughtput de lectura**: Cantidad de bytes leídos de la memoria por segundo (MB/s)

**Row-Major-Order:** Las columnas de una fila están ubicadas en posiciones contiguas, y las filas son bloques de memoria contiguos.

**Blocking:** Es una técnica que se utiliza para aprovechar la localidad de la información en los ciclos anidados. En vez de operar sobre filas o columnas completas de un arreglo, los algoritmos bloqueados trabajan con submatrices o bloques, de forma tal que los datos cargados en los niveles mas rápido de la jerarquía de memoria son reutilizados.

Memoria Virtual:

**Direccionamiento físico:** El chip de la CPU está conectado directamente con la memoria principal, se usa en sistemas como los embebidos.

**Direccionamiento virtual:** El chip de la CPU se conecta con el **MMU (memory management unit)**, el cual se conecta con la memoria principal. Se usa para sistemas mas complejos, como computadoras. Se considera una de las grandes ideas en la computación.

Espacios de direcciones lineal: Conjunto de direcciones dadas por una secuencia de números enteros no negativos. Hay **espacios de direcciones físicas (PAS)** y **espacios de direcciones virtuales (VAS)**.

¿Porque usar la memoria virtual?

**Uso eficiente de la memoria principal**, se usa la DRAM como cache de parte del VAS.

**Simplifica la gestión de la memoria**, cada proceso tiene un espacio de direcciones lineal.

**Aísla los espacios de direcciones**, un proceso no puede interferir con la memoria de otro, también, un programa de usuario no puede acceder a información y código privilegiado del kernel.

Memoria virtual como herramienta de cacheo

Conceptualmente la memoria virtual es un arreglo de N bytes almacenados en forma continua en el disco, este arreglo es cacheado en memoria física, más precisamente en la DRAM, los bloques de esta cache se llaman páginas.

DRAM: Es 10 veces más lenta que la SRAM. Los discos son 10000 veces mas lentos que la DRAM. Se tarda realmente mucho en cargar un bloque del disco, es por eso que los tamaños de las paginas son grandes. Es fully asociative, con esto logramos que cualquier pagina virtual se puede ubicar en cualquier pagina física, pero a su vez requiere una función de mapeo grande. También, los algoritmos de reemplazo son altamente sofisticados y costosos. Se utiliza el write back.

**Tablas de paginación:** Esta formada por **PTE (page table entries)** que mapean paginas virtuales a paginas físicas. La table de paginación es una estructura de datos que el kernel almacena por proceso en al DRAM.

**Page Hit:** Se referencia un dato usando memoria virtual correspondiente a una página física que está en la memoria principal (es un acierto en la cache DRAM).

**Page Fault:** Se referencia un dato usando memoria virtual correspondiente a una página física que no está en la memoria principal (fallo en la cache DRAM). En caso de haber un page fault, es la MMU quien se encarga de lanzar una excepción. Lo maneja el handler.

**Localidad al rescate:** La memoria virtual parece muy ineficiente, pero funciona por el principio de localidad. Los programas tienden a acceder a un conjunto activo de páginas llamado working sets, los programas con buena localidad mantienen conjuntos de trabajos reducidos.

Si el conjunto de trabajo (working set) < memoria principal tiene un desempeño bueno para el proceso, después de los fallos en frio.

Si conjunto de trabajo > memoria principal ocurre thrashing, el desempeño cae abruptamente porque las paginas se copian continuamente entre disco y DRAM.

Si se ejecutan varios procesos en simultaneo, el conjunto de trabajo total es la suma de los conjuntos.

Memoria virtual para gestionar la memoria

Cada proceso tiene su propio espacio de direcciones virtuales, permite ver la memoria como un arreglo de bytes, la función de mapeo se encarga de distribuir las direcciones en la memoria física.

Simplifica la reserva de memoria, cada pagina virtual se puede mapear a cualquier pagina física, en diferentes momentos una misma pagina virtual se puede almacenar en páginas física diferentes.

Se puede compartir código y datos entre procesos, se hace mapeando paginas virtuales distintas a una misma página física.

Enlazado y Carga

Enlazado: Cada programa tiene un VAS similar al resto. Las secciones de código, datos, y el heap siempre comienzan en los mismos lugares.

Carga: La función execve reserva paginas virtuales, y crea PTEs marcados como inválidos para las secciones .text y .data.

Memoria virtual como herramienta de protección

Se extienden las PTEs con bits de permisos, la MMU comprueba estos bits en cada acceso, estos permisos pueden ser SUP, READ, WRITE, EXEC, por ejemplo, para acceder a SUP, se requiere estar en modo kernel.

Traducción de direcciones

VPO: Offset de la página virtual

VPN: Numero de la página virtual

PPO: Offset de la página física, PPO y VPO son iguales.

PPN: Numero de la página física.

Page Hit:

1. El procesador envía la dirección virtual a la MMU.

2-3. La MMU obtiene la PTE de la tabla de paginación en memoria (DRAM).

1. La MMU envía la dirección física a la memoria (DRAM).
2. La cache/memoria envía el dato al procesador.

Page Fault:

1. El procesador envía la dirección virtual a la MMU.
   1. La MMU obtiene la PTE de la tabla de paginación en memoria (DRAM).
2. El bit de validez es cero y la MMU lanza la excepción (page fault), y se lo delega al handler.
3. El handler elije la victima (si es necesario la escribe al disco).
4. El handler carga la nueva pagina y actualiza el PTE en memoria (DRAM).
5. El handler vuelve al proceso original y reinicia la instrucción.

Aceleración de la traducción usando un TLB

Integración de la memoria cache y la memoria virtual: Las entradas de la tabla de paginación (PTEs) se guarda en la cache L1 como cualquier otro dato, las PTEs podrían ser desalojadas por cualquier otra referencia, los hits todavía tienen el retardo de la memoria cache.

Solución: Uso de la **TLB (Translation Lookaside Buffer)**, es una cache asociativa por conjuntos, en hardware, en la MMU. Mapea números de páginas virtuales con números de páginas físicas. Contiene PTEs completas para un subconjunto pequeño de páginas.

TLBI: TLB index.

TLBT: TLB tag.

La MMU usa la parte VPN (virtual page number) de la VA (virtual address) para acceder a la TLB, funciona como una cache de las ya vistas, pero no requiere del cache offset porque solo almacena PTEs.

El uso de la TLB, cuando es un acierto elimina un acceso de memoria. Pero cuando es miss se le agrega un acceso a memoria, pero los misses en la TLB son poco comunes.

Tabla de paginación de múltiples niveles: Suponiendo que solo hay dos niveles, podemos decir que las tablas de nivel 1 son las tablas que tienen como PTEs que apuntan a las tablas de nivel 2, y estas tablas son las que realmente tienen PTEs que apuntan a una página.