Main Memory

Chapter 8

8.1 Background

La memoria consiste de listas de bytes con direcciones. El CPU jala las instrucciones de memoria según el program counter. Un ciclo basico de ejecucion toma las instrucciones de la memoria, la decodifica, y manda a traer los comandos a memoria. La memoria solo ve direcciones.

8.1.1 Basic Hardware

La memoria principal y los registros están construidos dentro del procesador, y solo el almacenamiento de propósito general puede accesarlos. SI la data no está en memoria se necesita mover antes de que el CPU trabaje con ella. Poder ingresar a memoria puede tomar varios ciclos del reloj del CPU. En tal caso el procesador espera porque no tiene la data necesaria para completar la instrucción que le piden.

8.1.2 Address Binding

Un programa reside en el disco como un ejecutable. Para que el proceso se ejecute se necesita que el programa se traiga a la memoria principal. Dependiendo de la cantidad que necesite, la data se estará moviendo entre disco y memoria durante su ejecución. Normalmente el binding de las instrucciones y data a direcciones de memoria se puede hacer en cualquier paso: tiempo de compilación, load time o execution time.

8.1.3 Logical Versus Physical Address Space

An address generated by the CPU is commonly referred to as a logical address, whereas an address seen by the memory unit - that is, the one loaded into the memory-address register of the memory - is commonly referred to as a physical address. The set of all physical addresses corresponding to these logical addresses is a physical address space. The set of all logical addresses generated by a program is a logical address space.

The run-time mapping from virtual to physical addresses is done by a hardware device called the memory-management unit (MMU). The user program never sees the real physical addresses.

8.1.4 Dynamic Loading

Para usar mejor la memoria se usa el dynamic loading. Lo que hace él es no cargar totalmente la rutina a memoria hasta que se llame. La ventaja de dynamic loading es que la rutina es cargada solo cuando se necesita, esto es muy provechoso para situaciones con una gran cantidad de data.

8.1.5 Dynamic Linking and Shared Libraries

Librerías dynamically linked son sistemas de librerías que le ayudan al usuario a correr applicaciones. Dynamic linking es similar al dynamic loading, pues es pospuesto hasta que se le llame, facilitando las llamadas por el mismo recurso usado constantemente.

8.2 Swapping

Un proceso puede ser intercambiado temporalmente fuera de memoria y traído de regreso, el intercambio o swapping hace posible que las memorias físicas no se llenen de mala manera.

8.2.1 Standard Swapping

Swapping estándar: El intercambio estándar implica mover procesos entre la memoria principal y un espacio de respaldo. Debe ser lo suficientemente grande para acomodar copias de todas las imágenes de memoria para todos los usuarios, y debe proporcionar acceso directo a estas imágenes de memoria. El sistema mantiene una cola lista que consta de todos los procesos cuyas imágenes de memoria se encuentran en el almacén de respaldo o en la memoria y están listas para ejecutarse. Swapping se limita por factores como que el proceso tiene que estar totalmente dormido (idle).

8.3 Contiguous Memory Allocation

En la asignación de memoria contigua, cada proceso está contenido en una sola sección de la memoria que es contigua a la sección que contiene el siguiente proceso.

8.3.1 Memory Protection

Podemos evitar que un proceso acceda a la memoria que no posee mediante la combinación de dos ideas discutidas previamente.

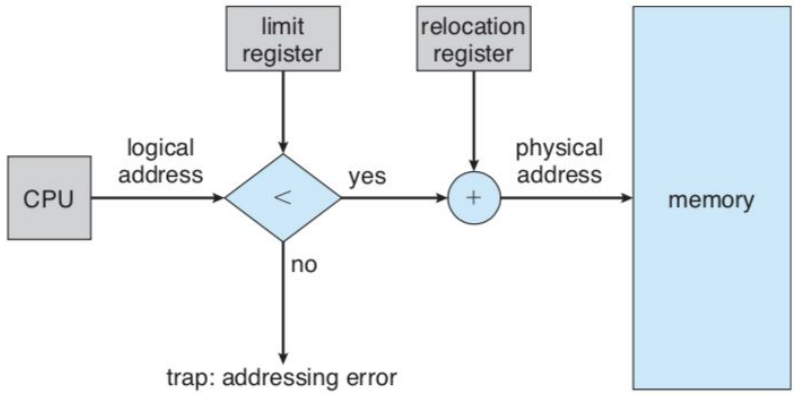

Figure 8.6 Hardware support for relocation and limit registers.

8.3.2 Memory Allocation

Uno de los métodos más simples para asignar memoria es dividir la memoria en varias particiones de tamaño fijo. Cada partición puede contener exactamente un proceso. Por lo tanto, el grado de multiprogramación está limitado por el número de particiones. En este método de partición múltiple, cuando una partición está libre, se selecciona un proceso de la cola de entrada y se carga en la partición libre. Cuando el proceso termina, la partición queda disponible para otro proceso.

El problema de asignación de almacenamiento dinámico se refiere a cómo satisfacer una solicitud de tamaño n de una lista de agujeros libres. Hay muchas soluciones a este problema. Las estrategias de primer ajuste, mejor ajuste y peor ajuste son las que se utilizan más comúnmente para seleccionar un agujero libre del conjunto de agujeros disponibles.

Primer ajuste Asigne el primer agujero que sea lo suficientemente grande. La búsqueda puede comenzar ya sea al principio del conjunto de agujeros o en la ubicación donde terminó la búsqueda de primer ajuste anterior. Podemos dejar de buscar tan pronto como encontremos un agujero libre que sea lo suficientemente grande.
Mejor ajuste. Asigne el agujero más pequeño que sea lo suficientemente grande. Debemos buscar en la lista completa, a menos que la lista esté ordenada por tamaño. Esta estrategia produce el agujero de sobra más pequeño.
El peor ajuste. Asignar el agujero más grande. De nuevo, debemos buscar en la lista completa, a menos que esté ordenada por tamaño. Esta estrategia produce el orificio sobrante más grande, que puede ser más útil que el orificio sobrante más pequeño desde un enfoque de mejor ajuste.

8.3.3 Fragmentation

A medida que los procesos se cargan y se eliminan de la memoria, el espacio libre en la memoria se divide en partes pequeñas. La fragmentación externa existe cuando hay suficiente espacio total en la memoria para satisfacer una solicitud, pero los espacios disponibles no son contiguos: el almacenamiento se fragmenta en una gran cantidad de agujeros pequeños. Una solución al problema de la fragmentación externa es la compactación. El objetivo es mezclar los contenidos de la memoria para colocar toda la memoria libre en un bloque grande.

8.4 Segmentation

8.4.1 Basic Method

Cada segmento tiene un nombre y una longitud. Las direcciones especifican tanto el nombre del segmento como el desplazamiento dentro del segmento. Por lo tanto, el programador especifica cada dirección en dos cantidades: un nombre de segmento y un desplazamiento. Una dirección lógica consta de dos tuplas: <número de segmento, desplazamiento>.

8.4.2 Segmentation Hardware

Una dirección lógica consta de dos partes: un número de segmento, s, y un desplazamiento en ese segmento, d. El número de segmento se utiliza como un índice para la tabla de segmentos. El desplazamiento d de la dirección lógica debe estar entre 0 y el límite de segmento. Si no lo es, capturamos el sistema operativo (intento de direccionamiento lógico más allá del final del segmento).

8.5 Paging

La segmentación permite que el espacio de direcciones físicas de un proceso no sea contiguo. La paginación es otro esquema de administración de memoria que ofrece esta ventaja. Sin embargo, la paginación evita la fragmentación externa y la necesidad de compactación, mientras que la segmentación no lo hace. También resuelve el problema considerable de colocar trozos de memoria de diferentes tamaños en la tienda de respaldo.

8.5.1 Basic Method

Cada dirección generada por la CPU se divide en dos partes: un número de página (p) y un desplazamiento de página (d). El número de página se utiliza como un índice en una tabla de páginas. Cuando usamos un esquema de paginación, no tenemos una fragmentación externa: cualquier marco libre puede asignarse a un proceso que lo necesite. Sin embargo, podemos tener alguna fragmentación interna. Observe que los marcos se asignan como unidades. Si los

requisitos de memoria de un proceso no coinciden con los límites de la página, es posible que el último cuadro asignado no esté completamente lleno.

Cuando llega un proceso al sistema que se va a ejecutar, se examina su tamaño, expresado en páginas. Cada página del proceso necesita un marco. Por lo tanto, si el proceso requiere n páginas, al menos n marcos deben estar disponibles en la memoria. Si hay n marcos disponibles, se asignan a este proceso de llegada. La primera página del proceso se carga en uno de los marcos asignados, y el número de marco se coloca en la tabla de páginas para este proceso. La siguiente página se carga en otro marco, su número de marco se coloca en la tabla de páginas, y así sucesivamente.

8.5.2 Hardware Support

La implementación del hardware de la tabla en paginación se puede hacer de varias maneras. En el caso más simple, la tabla de páginas se implementa como un conjunto de registros dedicados. Estos registros deben construirse con una lógica de muy alta velocidad para que la traducción de la dirección de paginación sea eficiente. Cada acceso a la memoria debe pasar por el mapa de paginación, por lo que la eficiencia es una consideración importante. El despachador de la CPU vuelve a cargar estos registros, al igual que vuelve a cargar los otros registros. Las instrucciones para cargar o modificar los registros de la tabla de páginas son, por supuesto, privilegiadas, de modo que solo el sistema operativo puede cambiar el mapa de memoria.

8.5.3 Protection

La protección de la memoria en un entorno paginado se logra mediante los bits de protección asociados con cada trama. Normalmente, estos bits se guardan en la tabla de páginas. Generalmente se adjunta un bit adicional a cada entrada en la tabla de páginas: un bit válido-no válido. Cuando este bit se establece como válido, la página asociada se encuentra en el espacio de direcciones lógicas del proceso y, por lo tanto, es una página legal (o válida). Cuando el bit se establece como no válido, la página no se encuentra en el espacio de direcciones lógicas del proceso.

8.6 Structure of the Page Table

8.6.1 Hierarchical Paging

Una dirección lógica se divide en un número de página que consta de 20 bits y un desplazamiento de página que consta de 12 bits. Debido a que la página de la página de la página, el número de la página se divide aún más. Debido a que la traducción de direcciones funciona desde la tabla de páginas externa hacia adentro, este esquema también se conoce como una tabla de páginas asignada hacia adelante.

8.6.2 Hashed Page Tables

Un enfoque común para manejar espacios de direcciones de más de 32 bits es usar una tabla de páginas con hash, con el valor de hash como el número de página virtual. Cada entrada en la tabla hash contiene una lista enlazada de elementos que contienen hash en la misma ubicación (para manejar las colisiones). Cada elemento consta de tres campos: (1) el número de página virtual, (2) el valor del marco de página asignado y (3) un puntero al siguiente elemento en la lista enlazada.

Se ha propuesto una variación de este esquema que es útil para espacios de direcciones de 64 bits. Esta variación utiliza tablas de páginas agrupadas, que son similares a las tablas de páginas con hash, excepto que cada entrada en la tabla de hash hace referencia a varias páginas (como 16) en lugar de una sola página. Por lo tanto, una sola entrada de la tabla de páginas puede almacenar las asignaciones para varios marcos de páginas físicas. Las tablas de páginas agrupadas son particularmente útiles para espacios de direcciones dispersos, donde las referencias de memoria no son contiguas y están dispersas por todo el espacio de direcciones.