Memoria Principal

Introducción

[1)] La memoria es un componente crucial para la operación de un sistema informático moderno. La memoria está compuesta de una gran matriz de palabras o bytes, cada uno con su propia dirección. La CPU extrae instrucciones de la memoria de acuerdo con el valor del contador de programa. Estas instrucciones pueden provocar operaciones adicionales de carga o de almacenamiento en direcciones de memoria específicas.

Un ciclo típico de ejecución de una instrucción procedería en primer lugar, por ejemplo, a extraer una instrucción de la memoria. Dicha instrucción se decodifica y puede hacer que se extraigan de memoria una serie de operandos. Después de haber ejecutado la instrucción con esos operandos, es posible que se almacenen los resultados de nuevo en memoria.

La unidad de memoria tan sólo ve un flujo de direcciones de memoria y no sabe cómo se generan esas direcciones (mediante el contador de programa, mediante indexación, indirección, direcciones literales, etc.) ni tampoco para qué se utilizan (instrucciones o datos). Por tanto, podemos ignorar el cómo genera el programa las direcciones de memoria; lo único que nos interesa es la secuencia de direcciones de memoria generadas por el programa en ejecución.

Hardware Básico

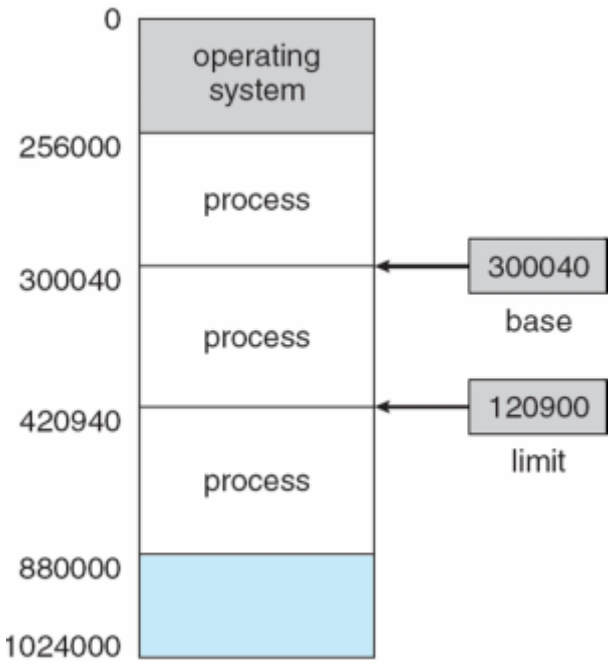
La memoria principal y los registros integrados dentro del propio procesador son las únicas áreas de almacenamiento a las que la CPU puede acceder directamente. Hay instrucciones de máquina que toman como argumentos direcciones de memoria, pero no existe ninguna instrucción que acepte direcciones de disco. Por tanto, todas las instrucciones en ejecución y los datos utilizados por esas instrucciones deberán encontrarse almacenados en uno de esos dispositivos de almacenamiento de acceso directo. Si los datos no se encuentran en memoria, deberán llevarse hasta antes de que la CPU pueda operar con ellos.

Generalmente, puede accederse a los registros integrados en la CPU en un único ciclo del reloj de procesador. La mayoría de los procesadores pueden decodificar instrucciones y realizar operaciones simples con el contenido de los registros a la velocidad de una o más operaciones por cada tic de reloj.

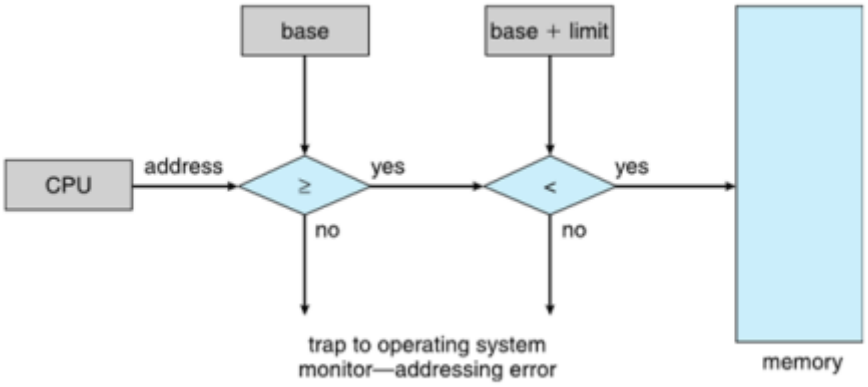
No podemos decir lo mismo de la memoria principal, a la que se accede mediante una transacción del bus de memoria. El acceso a memoria puede requerir muchos ciclos reloj del procesador para poderse completar, en cuyo caso el procesador necesitará normalmen detenerse, ya que no dispondrá de los datos requeridos para completar la instrucción que esté ejecutando. Esta situación es intolerable, debido a la gran frecuencia con la que se accede a la memoria. El remedio consiste en añadir una memoria rápida entre la CPU y la memoria principal. Un búfer de memoria utilizado para resolver la diferencia de velocidad de memoria se denomina caché.

No sólo debe preocuparnos la velocidad relativa del acceso a la memoria física, sino que también debemos garantizar una correcta operación que proteja al sistema operativo de los posibles accesos por parte de los procesos de los usuarios y que también proteja a unos procesos de usuario de otros. Esta protección debe ser proporcionada por el hardware y puede implementarse de diversas formas.

Primero tenemos que asegurarnos de que cada proceso disponga de un espacio de memoria separado. Para hacer esto, debemos poder determinar el rango de direcciones legales a las que el proceso pueda acceder y garantizar también que el proceso sólo acceda a esas direcciones legales. Podemos proporcionar esta protección utilizando dos registros, usualmente una base y un límite. El registro base almacena la dirección de memoria física legal más pequeña, mientras que el registro límite especifica el tamaño del rango. Por ejemplo, si el registro base contiene el valor 300040 y el registro límite es 120900, entonces el programa podrá acceder legalmente a todas las direcciones comprendidas entre 300040 y 420940 (incluyendo los dos extremos).



La protección del espacio de memoria se consigue haciendo que el hardware de la CPU compare todas las direcciones generadas en modo usuario con el contenido de esos registros. Cualquier intento, por parte de un programa que se esté ejecutando en modo usuario, de acceder a la memoria del sistema operativo o a la memoria de otros usuarios hará que se produzca una interrupción hacia el sistema operativo, que tratará dicho intento como un error fatal. Este esquema evita que un programa de usuario modifique (accidental o deliberadamente) el código y las estructuras de datos del sistema operativo o de otros usuarios.



Los registros base y límite sólo pueden ser cargados por el sistema operativo, que utiliza una

instrucción privilegiada especial. Puesto que las instrucciones privilegiadas sólo pueden ser ejecutadas en modo kernel y como sólo el sistema operativo se ejecuta en modo kernel, únicamente el sistema operativo podrá cargar los registros base y límite. Este esquema permite al sistema operativo modificar el valor de los registros, pero evita que los programas de usuario cambien el contenido de esos registros.

El sistema operativo, que se ejecuta en modo *kernel*, tiene acceso no restringido a la memoria tanto del sistema operativo como de los usuarios. Esto le permite cargar los programas de los usuarios en la memoria de los usuarios, volcar dichos programas en caso de error, leer y modificar parámetros de las llamadas al sistema, etc.

Reasignacion de direcciones

Usualmente, los programas residen en disco en forma de archivos ejecutables binarios. Para poder ejecutarse, un programa deberá ser cargado en memoria y colocado dentro de un proceso. Dependiendo del mecanismo de gestión de memoria que se utilice, el proceso puede desplazarse entre disco y memoria durante su ejecución. Los procesos del disco que estén esperando a ser cargados en memoria para su ejecución forman lo que se denomina **cola de entrada.**

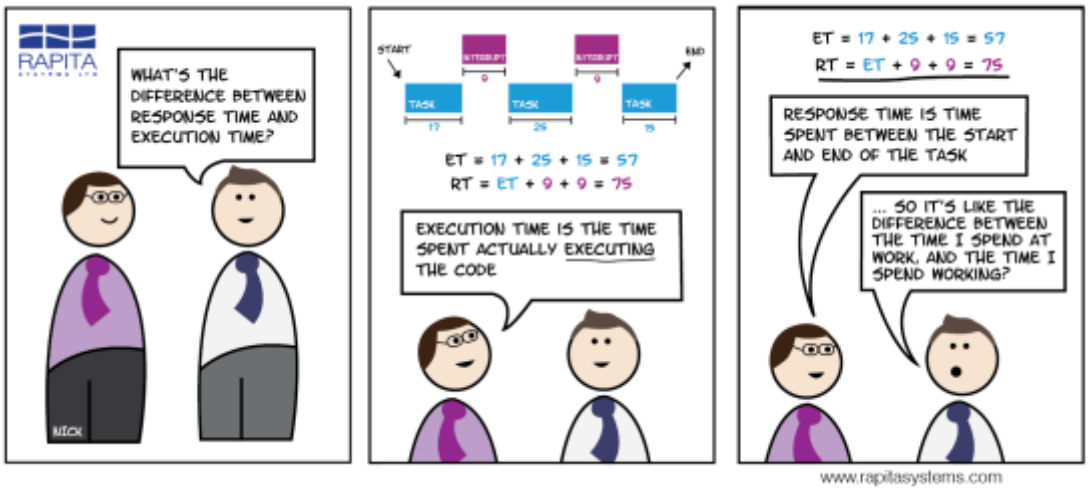
El procedimiento normal consiste en seleccionar uno de los procesos de la cola de entrada y cargar dicho proceso en memoria. A medida que se ejecuta el proceso, éste accede a las instrucciones y datos contenidos en la memoria. Eventualmente, el proceso terminará su ejecución y su espacio de memoria será declarado como disponible.

La mayoría de los sistemas permiten que un proceso de usuario resida en cualquier parte de la memoria física. Así, aunque el espacio de direcciones de la computadora comience en 00000, la primera dirección del proceso de usuario no tiene por qué ser 00000. Esta técnica afecta a las direcciones que el programa de usuario puede utilizar.

Clásicamente, la reasignación de las instrucciones y los datos a direcciones de memoria puede realizarse en cualquiera de los pasos:

- **Tiempo de compilación.** Si sabemos en el momento de realizar la compilación dónde va a residir el proceso en memoria, podremos generar código absoluto. Por ejemplo, si sabemos que un proceso de usuario va a residir en una zona de memoria que comienza en la ubicación R, el código generado por el compilador comenzará en dicha ubicación y se extenderá a partir de ahí. Si la ubicación inicial cambiase en algún instante posterior, entonces sería necesario recompilar ese código. Los programas en formato .COM de MS-DOS se acoplaban en tiempo de compilación.
- **Tiempo de carga.** Si no conocemos en tiempo de compilación dónde va a residir el proceso en memoria, el compilador deberá generar código reubicable. En este caso, se retarda la reasignación final hasta el momento de la carga. Si cambia la dirección inicial, tan sólo es necesario volver á cargar el código de usuario para incorporar el valor modificado.

- **Tiempo de ejecución.** Si el proceso puede desplazarse durante su ejecución desde un segmento de memoria a otro, entonces es necesario retardar la reasignación hasta el instante de la ejecución. Para que este esquema pueda funcionar, será preciso disponer de hardware especial. La mayoría de los sistemas operativos de propósito general utilizan este método.



Volver

(78)

1)
Extraido y resumido de Fundamentos de Sistemas Operativos - Silberschtaz

From:
http://wiki.educabit.ar/ - **Wiki Sistemas**

Permanent link:
http://wiki.educabit.ar/doku.php?id=so_memppal

Last update: **2025/09/11 22:48**