Memoria Principal

Agenda

1. Objetivos
2. Fundamentos
3. Asignación de Memoria Contigua
4. Asignación de Memoria No Contigua

UNIVERSIDAD DEL BÍO-BÍO

Objetivos

- Proporcionar una descripción detallada de las diversas formas de organizar el hardware de memoria.
- Analizar diversas técnicas de gestión de memoria, incluyendo la paginación y la segmentación.

Fundamentos

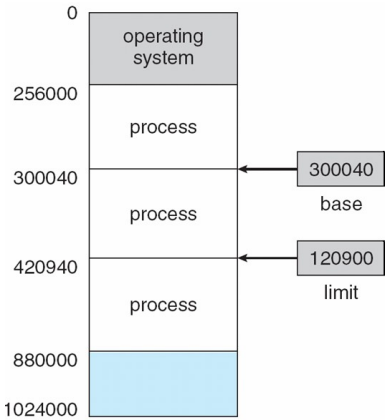
- Programas deben estar cargados en memoria principal para ser ejecutados.
- La memoria principal y los registros integrados del propio procesador son las únicas áreas de almacenamiento a las que la CPU puede acceder directamente.
- La memoria principal es un arreglo de bytes identificados por su dirección.
- Interacción con la memoria se logra por medio de una secuencia de lecturas o escrituras de direcciones específicas de la memoria.
- **Caché**, memoria rápida entre la CPU y la memoria principal utilizada para resolver el problema de las velocidades.
- Se requiere garantizar correcta operación que proteja al SO y a los procesos de usuarios de accesos indebidos (registros base y límite).

UNIVERSIDAD DEL BÍO-BÍO

Registros Base y límite

- Un par de registros **base** y **límite** definen el espacio de direcciones para cada proceso. Registro base almacena la dirección de memoria fisica legal mas pequeña. Registro límite especifica el tamaño del rango.
- Protección del espacio de memoria se logra haciendo que el HW de la CPU compare las direcciones generadas en modo usuario con el contenido de estos registros.
- Cualquier acceso ilegal (fuera de rango) generará una interrupción hacia el SO.

Registros Base y límite



Reasignación de direcciones

- Programas de usuarios deben recorrer varias etapas antes de ser ejecutados, donde las direcciones pueden representarse de diferentes formas.
  - Programa fuente, las direcciones son simbólicas (la variable "suma")
  - Luego, compilador reasignará tal dirección simbólica a reubicables (14 bytes a partir del comienzo de este módulo)
  - El cargador reasignará la dirección reubicable en absoluta (74014).

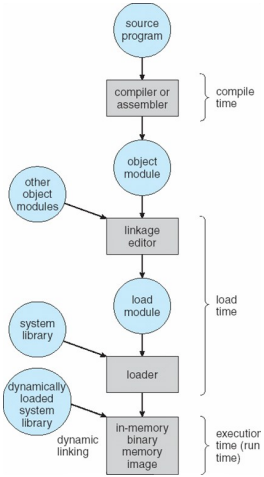
  Cada operación de reasignación constituye una relación de un espacio de direcciones a otro.

Reasignación de direcciones

- Reasignación a direcciones de memoria puede realizarse en cualquiera de las etapas:
  - **Tiempo de compilación**. Si se conoce donde va a residir el proceso en memoria al momento de compilar, el compilador puede generar **código absoluto**. Si luego cambia la ubicación, hay que recompilar (programas .COM de MS-DOS).
  - **Tiempo de carga**. Si no se conoce en tiempo de compilación donde residirá el proceso en memoria, el compilador deberá generar **código reubicable**. Se retarda la reasignación final hasta el momento de la carga. Luego, si cambia dirección inicial, basta con cargar nuevamente el código del usuario.
  - **Tiempo de ejecución**. Si proceso puede desplazarse durante su ejecución de un segmento de memoria a otro, se retarda la reasignación hasta el instante de la ejecución. Requiere HW especial (MMU - Memory Management Unit). La mayoría de los SSOO de propósito general utilizan este método.

UNIVERSIDAD DEL BÍO-BÍO

Etapas para ejecutar un programa



Espacio de direcciones lógicas y físicas

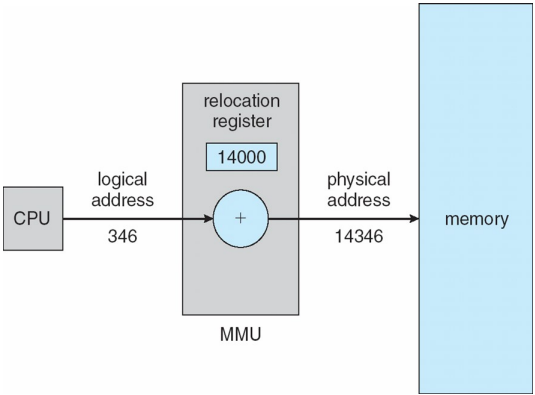
- **Dirección lógica**, dirección generada por la CPU.
- **Dirección física**, dirección recibida por la unidad de memoria y que puede ser eventualmente distinta a la dirección lógica.
- **Espacio direcciones lógicas**, conjunto de direcciones lógicas generadas por un programa.
- **Espacio direcciones físicas**, conjunto de direcciones físicas correspondientes a un conjunto de direcciones lógicas.
- Cuando la reasignación de direcciones se realiza en:
  - Tiempo de compilación o carga - direcciones lógicas y físicas son idénticas.
  - Tiempo de ejecución - direcciones lógicas (ahora llamada dirección virtual) y físicas difieren.
    - Transformación de direcciones virtuales a físicas es por HW mediante la **unidad de gestión de memoria MMU**

UNIVERSIDAD DEL BÍO-BÍO

MMU Memory-Management Unit

- Dispositivo HW que "mapea" direcciones virtuales a físicas.
- Generalización del esquema de registro Base.
  - registro base se denomina ahora **registro de reubicación**.
  - valor del registro de reubicación se *suma* a cada dirección generada por un proceso de usuario al momento de enviarlo a memoria.
  - programa de usuario trata con direcciones lógicas, nunca ve la dirección física real.

Reubicación dinámica con registro de reubicación



Carga Dinámica

- Hasta el momento hemos indicado que todo el programa y todos los datos de un proceso deben estar en memoria física para que pueda ejecutarse.
  - *Por lo tanto, el tamaño del proceso queda limitado al tamaño de la memoria física.*
- Para mejor utilización del espacio de memoria podemos usar un mecanismo de **Carga Dinámica**.
  - Programa principal se carga en memoria y se ejecuta.
  - Rutinas no se cargan hasta que son invocadas.
  - Cuando son invocadas se verifica si está ya cargada, sino, se carga en memoria.
  - Rutinas que no son utilizadas no se cargarán nunca (algunas rutinas de error).
  - Aunque el tamaño total del programa pueda ser más grande que la memoria, la porción que se utilice (que se cargue) puede ser mucho más pequeña.

UNIVERSIDAD DEL BÍO-BÍO

Carga Dinámica

- Carga dinámica no requiere de soporte especial por parte del SO.
- Responsabilidad de usuario de diseñar correctamente sus programas.
- SO puede ayudar proporcionando rutinas de bibliotecas que implementen el mecanismo de carga dinámica.

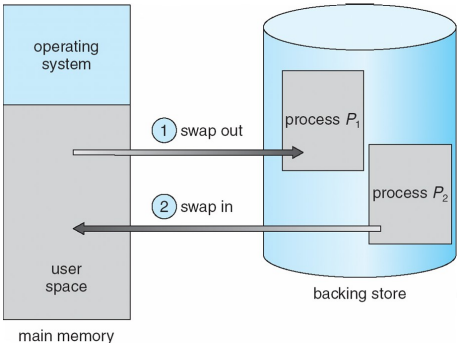
Montaje Dinámico y Bibliotecas Compartidas

- Algunos SO permiten solo **montaje estático**, bibliotecas se tratan como cualquier otro módulo del programa.
  - Cada programa que utilice estas bibliotecas, tendrá una copia dentro del ejecutable.
  - Se desperdicia espacio de disco como de memoria.
- Montaje dinámico incluye un **stub** en la imagen binaria para cada referencia a una rutina de biblioteca.
- **Stub** es un pequeño fragmento de código que indica como localizar la rutina necesaria.
- El stub es reemplazado por la dirección de la rutina y se ejecuta.
- Este mecanismo es conocido también como **Bibliotecas Compartidas**

Intercambio - Swapping

- Un proceso debe estar en memoria para ser ejecutado, sin embargo:
  - proceso puede ser **intercambiado** temporalmente, sacándolo de memoria y almacenándolo en un **almacén de respaldo** y volviendolo luego a memoria para continuar su ejecución.
- **Almacén de respaldo**, disco suficientemente rápido y grande para albergar copias de las imágenes de memoria para todos los usuarios.

Intercambio - Swapping



Asignación de Memoria Contigua

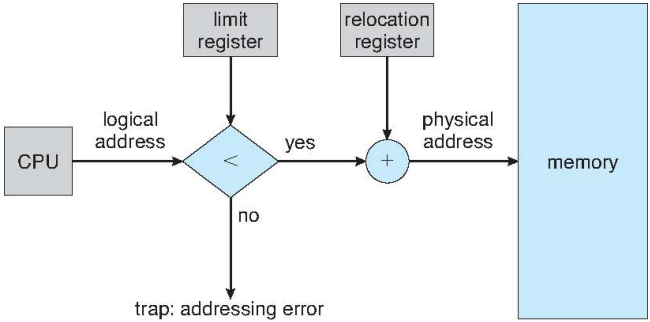
- La MP debe albergar tanto el SO como los diversos procesos de usuario, por tanto se requiere asignar las distintas partes de la memoria de la forma más eficiente posible.
- Memoria usualmente dividida en dos particiones:
  - Sistema Operativo residente
  - Procesos de Usuario
- Normalmente querremos tener varios procesos de usuario residentes en memoria al mismo tiempo.
  - Por lo tanto, se debe considerar como asignar la memoria disponible a procesos que están en cola de entrada.
  - En este esquema, cada proceso está contenido en una **única** sección contigua de memoria.

UNIVERSIDAD DEL BÍO-BÍO

Mapeo de memoria y protección

- Registro de reubicación es utilizado para proteger los procesos de usuarios de otros procesos.
  - registro reubicación contiene el valor de la dirección física mas pequeña.
  - registro límite contiene el rango de direcciones lógicas.
  - cada dirección debe ser inferior al valor del registro límite.
  - la MMU convertirá la direcciones virtuales **dinámicamente** sumándole el registro de reubicación.

Mapeo de memoria y protección



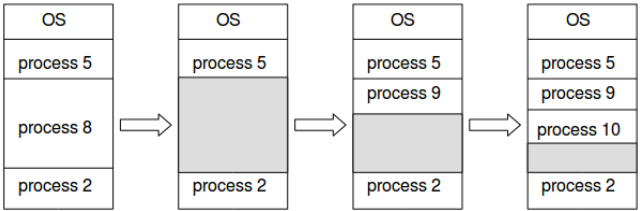
Método de Particiones Múltiples - MFT

- Fixed number of Tasks.
- Método más simple se asignación.
- Divide memoria en **particiones de tamaño fijo**.
- Cada partición puede contener exactamente un proceso.
- Grado de multiprogramación está limitado por el número de particiones disponibles.
- Sistema Operativo mantiene información de:
  - Particiones asignadas
  - Particiones libres
- Este método ya no utiliza.

Método de Particiones Múltiples - MVT

- Variable number of Tasks.
- Generalización de MFT.
- Tamaño de particiones ya no es fija.
- Existen bloques de memoria disponibles y dispersos en la memoria.
- Cuando un proceso llega, se le asigna espacio disponible lo suficientemente grande para contenerlo.

Método de Particiones Múltiples - MVT



Asignación Dinámica de espacio de almacenamiento

- ¿Cómo satisfacer una solicitud de tamaño **n** a partir de una lista de particiones libres?
- Hay tres estratégias:
  - **Primer Ajuste**, se asigna primer espacio lo suficientemente grande. Recorrido de la lista puede ser desde el principio o desde la búsqueda anterior. Búsqueda se detiene al encontrar el espacio necesario.
  - **Mejor Ajuste**, se asigna el espacio más pequeño que tenga el tamaño suficiente. Recorrido debe ser completo de la lista, a menos que esté ordenada por tamaño.
  - **Peor Ajuste**, se asigna el espacio más grande. Recorrido debe ser completo de la lista, a menos que esté ordenada por tamaño.
- Las 2 primeras tienen mejores rendimientos en términos de velocidad y utilización del almacenamiento.

UNIVERSIDAD DEL BÍO-BÍO

Fragmentación

- **Fragmentación Externa**.
  - Existe espacio total para atender solicitud, pero no es es contiguo .
  - Primer y Mejor ajuste sufren de este problema. Liberan pequeños fragmentos de memoria.
- **Fragmentación Interna**
  - Memoria asignada es demasiado grande en relación a lo solicitado. Se desperdicia espacio que no se puede asignar.

UNIVERSIDAD DEL BÍO-BÍO

Fragmentación

- **Compactación**
  - Reduce la fragmentación externa.
  - Mueve contenido de la memoria para situar la memoria libre de forma contigua, en un único bloque de gran tamaño.
  - Solo es posible si la reubicación es dinámica y se lleva a cabo en tiempo de ejecución.
- Otra solución a la fragmentación externa es permitir que espacio de direcciones lógicas de los procesos NO sea contiguo. Dos técnicas para lograr esto:
  - **Paginación**
  - **Segmentación**

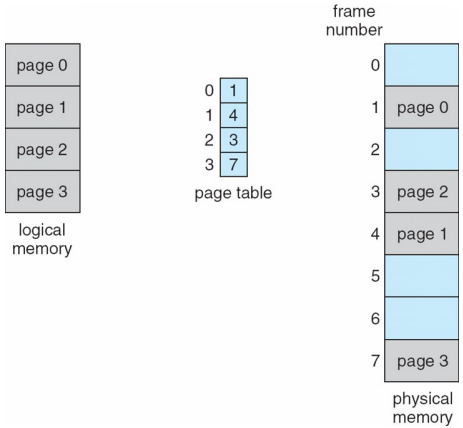
UNIVERSIDAD DEL BÍO-BÍO

Paginación

- Programas se dividen en unidades pequeñas de memoria de tamaño fijo, denominadas **páginas**.
- La memoria física se divide en unidades del mismo tamaño, denominados **marcos de página**.
- Cualquier página puede ser cargada en cualquier marco de página libre (no necesariamente contiguo).
- A los programas se les puede asignar memoria bajo demanda.
- Se requiere de una **tabla de páginas** para cada proceso (asociar página con marco de página).
- SO debe mantener información de marcos de página libres, **tabla de marcos**.
- Se requiere de MMU para la asociación eficiente página-marco en tiempo de ejecución.

UNIVERSIDAD DEL BÍO-BÍO

Paginación


page 0
page 1
page 2
page 3
logical memory
0 1
1 4
2 3
3 7
page table
frame number
0
1 page 0
2
3 page 2
4 page 1
5
6
7 page 3
physical memory

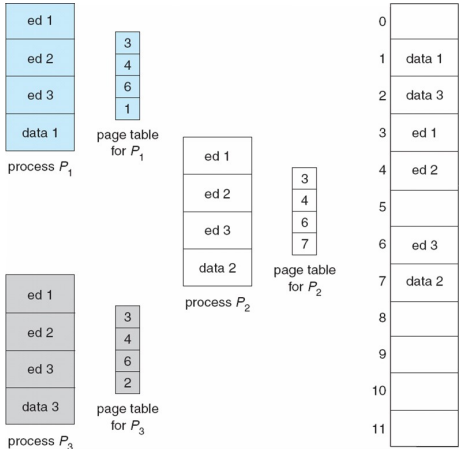
Paginación

- El usuario ve un espacio contiguo de direcciones lógicas.
- Tabla de páginas es solo manipulada por el SO (queda en el espacio físico del núcleo).
- En el cambio de contexto se debe cambiar el registro base PTBR (page-table base register) y límite PTLR, de manera de apuntar a la tabla de páginas del nuevo proceso.
- Manipulación de los registros especiales debe ser con instrucciones privilegiadas.
- Planificación de trabajos:
  - Si el trabajo requiere N páginas y existen al menos N marcos libres, entonces admitirlo.
  - Crear el proceso asignándole N marcos y definir la PT para el proceso.
  - Agregar en la PCB referencia a la PT.

Paginación

- Fragmentación:
  - Se elimina fragmentación externa, evitandose la compactación.
  - Se introduce una fragmentación interna mínima.
- Memoria caché:
  - Procesadores modernos son capaces de mantener copias de páginas más usadas en caché.
  - Memoria caché es más rápida que la principal.
  - Al referenciar página se busca primero en la cache, sino en MP.
- Compartición de páginas:
  - Paginación facilita compartición de código entre programas.
  - Varios usuarios ejecutando el mismo programa, se requiere solo una copia del código.

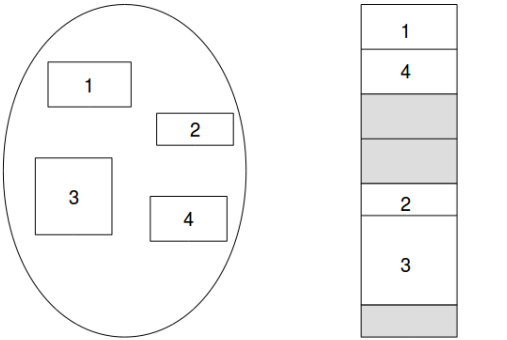
Compartición de Páginas



Segmentación

- Programa se divide normalmente en segmentos lógicos.
  - Código.
  - Datos.
  - Librerías.
  - Pilas.
  - etc.
- Compilador construye automáticamente los segmentos.
- Segmentación permite tener una visión lógica del programa que se ha segmentado.

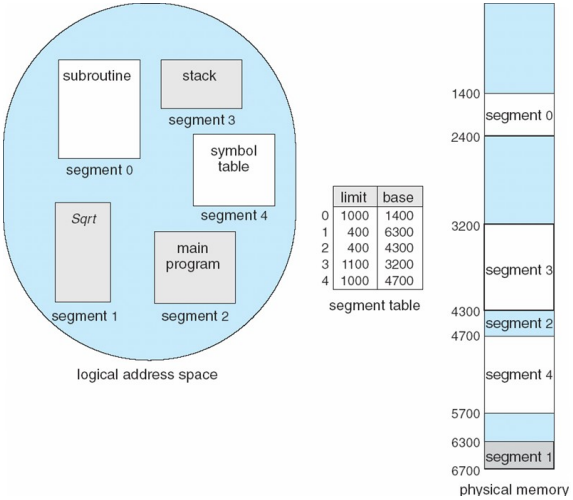
Segmentación



Segmentación

- Memoria se divide en segmentos, donde se cargan unidades lógicas de un programa.
- Un programa es un conjunto de segmentos, donde cada uno puede tener tamaño diferente.
- Se requiere una tabla de segmentos para mantener contiguidad del espacio lógico.
- Una dirección lógica es un par **(s,d)**, donde **s** es el identificador del segmento y **d** es el desplazamiento.
- Desplazamiento controlado con registros límite y base.

Segmentación



Segmentación Paginada

- Esquema muy utilizado en computadores modernos.
- Trata de obtener las ventajas de la páginación (evitar fragmentación) y de la segmentación (dividir en unidades lógicas).
- Programa se divide en segmentos y cada segmento en páginas.
- Cada dirección lógica es del tipo **(s,d)**, donde **d** se divide a su vez en **(p,d')**.

UNIVERSIDAD DEL BÍO-BÍO