FACULTAD DE INGENIERÍA

Sistemas Operativos

Gestión de la Memoria

Lic. R. Alejandro Mansilla

Licenciatura en Ciencias de la Computación
Facultad de Ingeniería
Universidad Nacional de Cuyo

Intro

- Las computadoras sirven para procesar información mediante la ejecución de programas apropiados
- Tanto los programas como los datos a manipular deben coexistir durante la ejecución en la memoria RAM junto con el SO.
- El precio de la memoria RAM ha descendido en los últimos años
- El tamaño de los programas y el volumen de datos a procesar a aumentado
- La gestión de la memoria es una tarea del SO pero con la asistencia del hardware para aquellas tareas no triviales.

Funciones y operaciones

El administrador de memoria de un sistema operativo debería cumplir con las siguientes funciones:

- **Reubicación**: Sistema de memoria virtual.
- **Protección**: Los procesos no deben ser capaces de acceder al espacio de memoria de otros procesos.
- **Compartimiento:** Hay casos donde si se desea compartir memoria entre procesos sobre todo en los mecanismos de comunicación.
- **Organización lógica:** Organización que surge en cómo están estructurados internamente los procesos. Un ejemplo es la segmentación
- **Organización física:** 2 niveles: memoria principal *(rápida en el orden de los nanosegundos)* y secundaria *(más lenta, en el orden de los milisegundos)*. El gestor de memoria del SO se encarga de mover los datos de una a otra.

Jerarquía de memorias

Existe una jerarquía de memoria que nos permite hacer una categorización de velocidades de acceso y costos.

A medida que bajamos de nivel de jerarquía, vamos bajando el costo por bit, aumentando la capacidad pero también aumentando el tiempo de acceso.

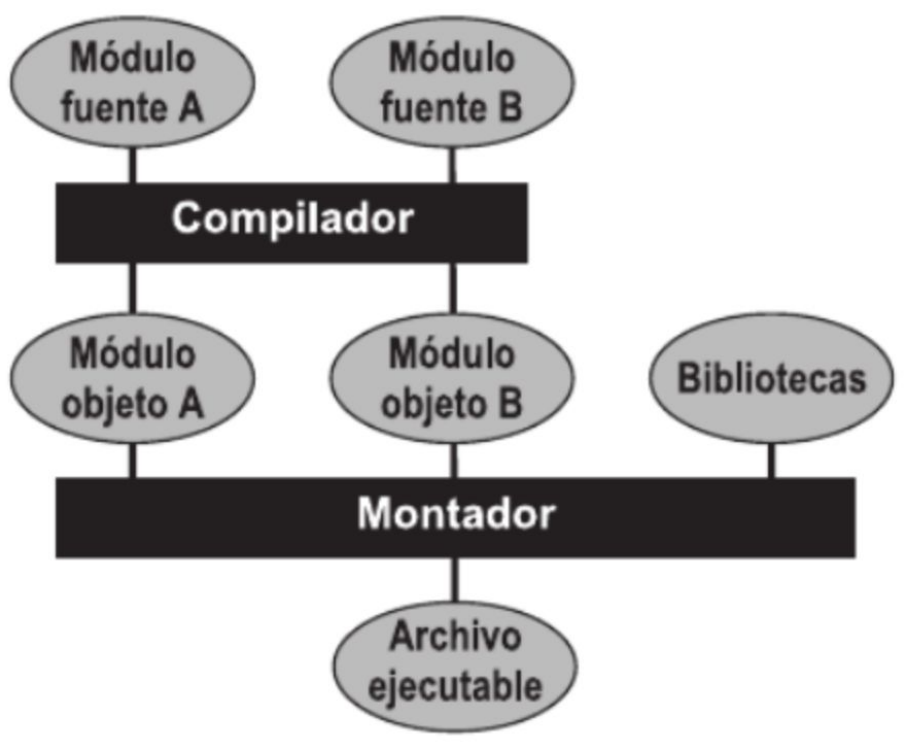
Jerarquía Tradicional	Jerarquía Moderna
Registros	Registros
Caché	Caché
Memoria Principal	Memoria Principal
Disco Magnético	Caché de disco
Cinta Magnética	Disco de estado Sólido
	Disco Magnético
	Cinta magnética/disco óptico

Tiempos de acceso de distintas memorias

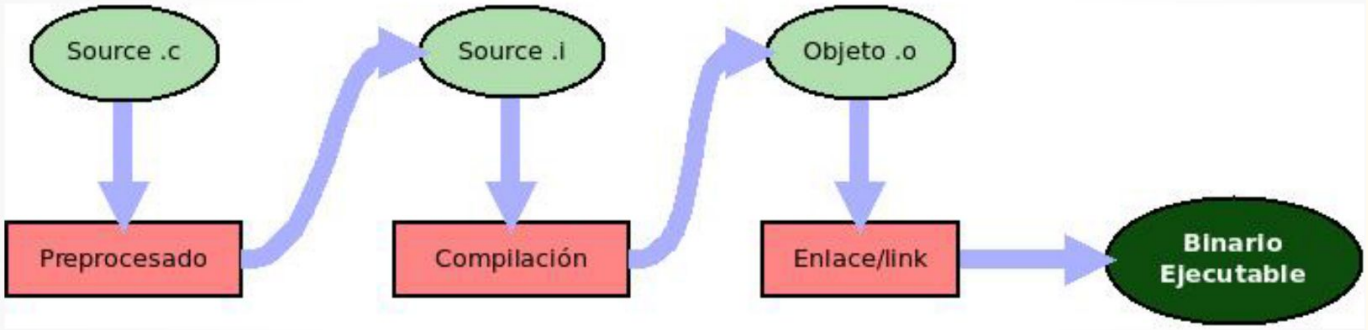
Tipo de Dispositivo	Tiempo típico de Servicio	Relativos al Segundo
Buffer del procesador	10 ns	1 seg
Memoria de acceso aleatorio	60 ns	6 seg
Llamada a procedimiento	1µs	2 min
Memoria expandida	25µs	1 hora
RPC local	100 µs	4 horas
Disco de estado sólido	1 ms	1 día
Disco "cacheado"	10 ms	12 días
Disco magnético	25 ms	4 semanas
Disco via LAN alta velocidad	27 ms	1 mes
Disco vía LAN serv.	35-50 ms	6 - 8 semanas
Disco vía WAN serv.	1-2 segundos	3 - 6 años
Disco/cinta desmontable	3 - 15 segundos	10 - 15 años

Generación de un ejecutable

- **Compilación:** generación del código de máquina correspondiente a cada módulo fuente.
- **Enlace o montaje *(link)*:** se genera el ejecutable agrupando todos los módulos objeto y resolviendo las referencias entre módulos. Este enlazado puede ser estático o dinámico. En este último caso, el módulo no es incluido sino que se enlaza en al momento de ejecutar el el programa en forma dinámica.



Generación de un ejecutable

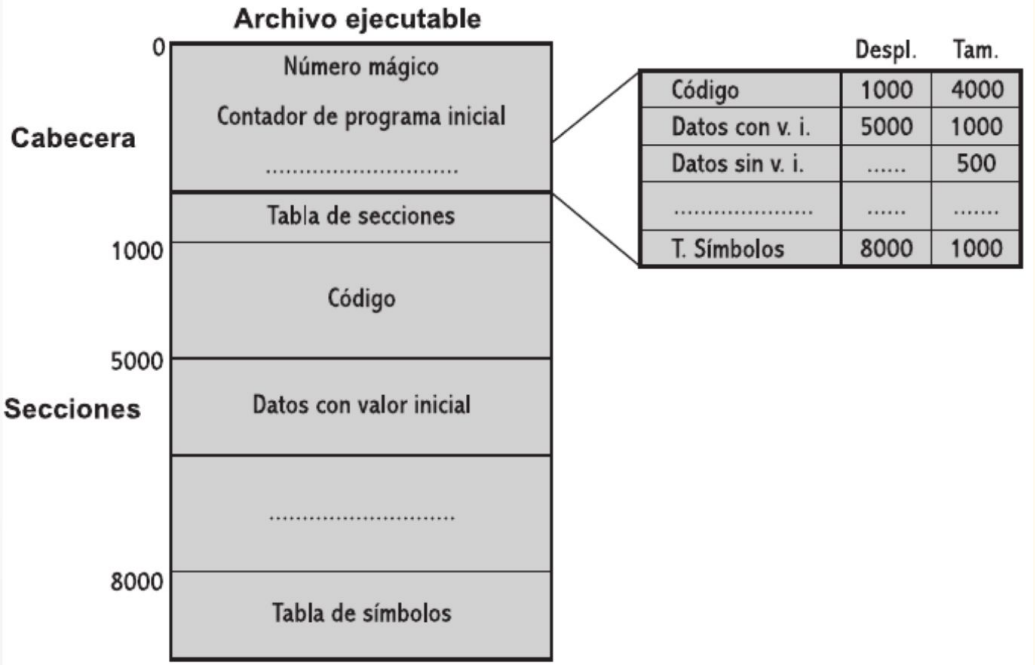


El mapa inicial de un proceso está muy vinculado con el archivo que contiene el programa ejecutable asociado al mismo, comenzaremos estudiando cómo se genera un archivo ejecutable y cuál es la estructura típica del mismo.

Formatos de un ejecutable

Como parte final del proceso de compilación y enlace se genera un archivo ejecutable que contiene el código de máquina del programa. Un ejecutable está compuesto por una cabecera y un conjunto de secciones.
Cabecera: Información de control
Secciones:

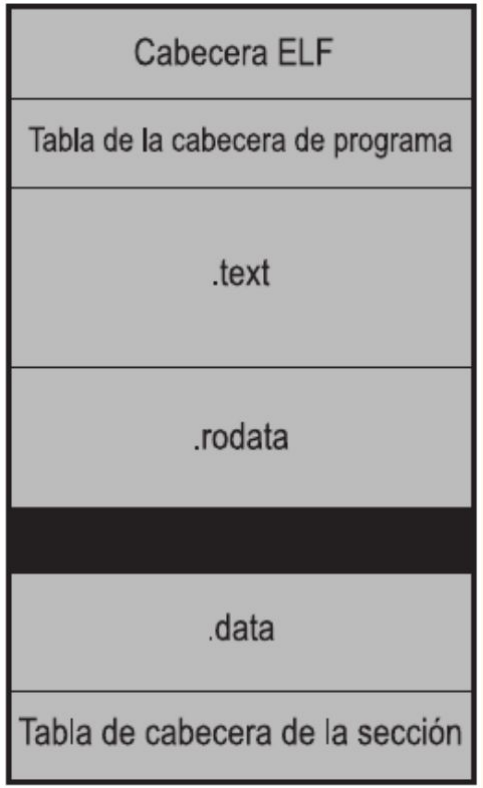
- Código
- Datos con valor inicial
- Datos sin valor inicial *(descrito en la cabecera pero no almacenado en el ejecutable porque no tienen valor)*



Formato ELF

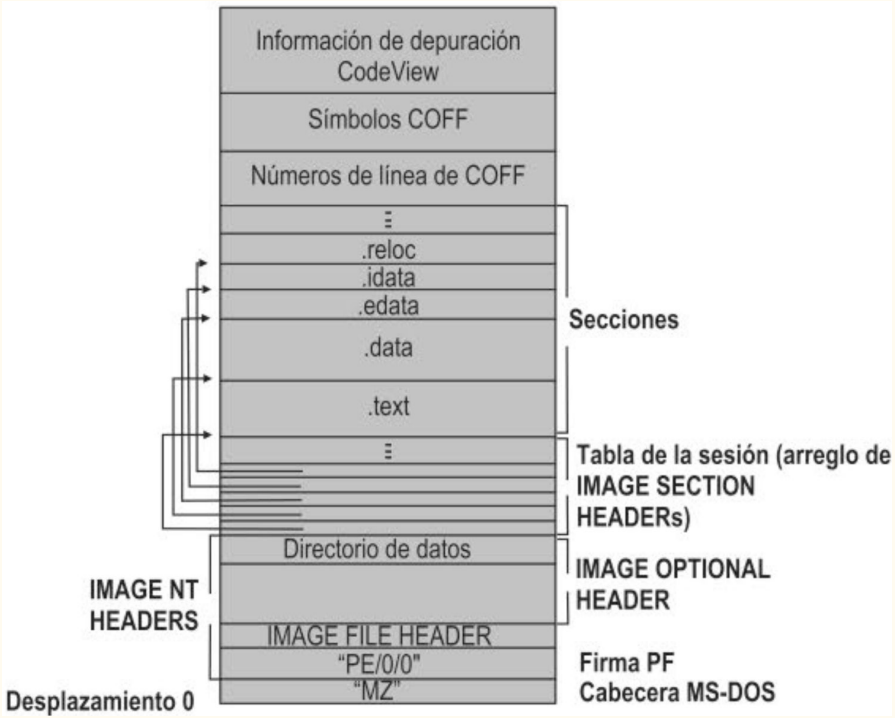
Executable and linkable Format

Lo usan archivos objeto, ejecutables, bibliotecas compartidas y volcados de memoria. No está limitado a un procesador o arquitectura. Lo utiliza Unix/Linux, Playstation Portable, Playstation 2, Playstation 3 y Wii.



Formato PE

- Parte de la especificación original Win32.
- Se utilizaba en VAX/VMS
- Se lo usa en ejecutables y DLLs
- A diferencia de ELF, que usa código independiente de la posición, PE se compila a una dirección base preferida y si no puede ser cargado en esa dirección preferida, el sistema operativo tiene que recalcular la base.



Mapa de memoria de un proceso

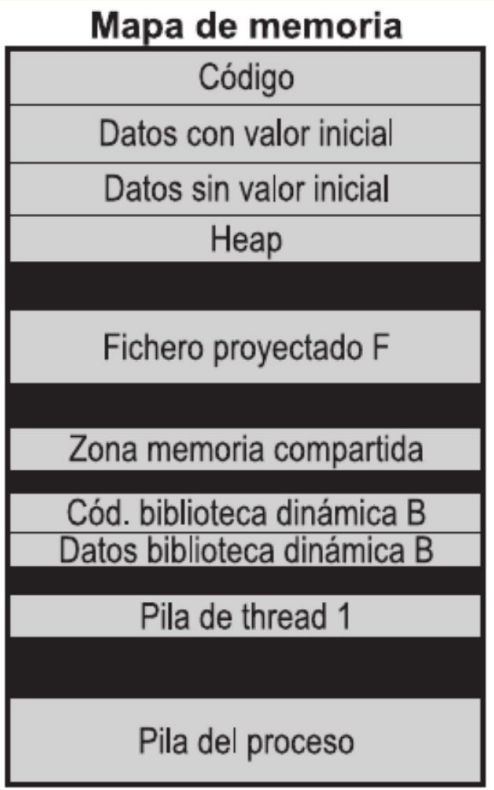
Formado por varias regiones o segmentos creadas a partir de la info del ejecutable en tiempo de ejecución.

Cada región es una zona contigua y con las siguientes características:

- Soporte de la región *(donde está almacenada la región)*
  - Soporte en archivo
  - Sin soporte. No tiene contenido inicial
- Tipo de uso compartido
  - Privado. Solo accesible por ese proceso.
  - Compartido. El cont. de la región es accesible por otros procesos.
- Protección
  - Lectura
  - Ejecución
  - Escritura
- Tamaño fijo o variable.

Mapa de memoria de un proceso*(cont)*

- Código (texto): Región compartida. Tamaño fijo. Está en el ejecutable - .text
- Datos con valor inicial: Región privada. Tamaño fijo. Con soporte en el ejecutable *(int i=7)*. Es .data
- Datos sin valor inicial. Región privada. RW y tamaño fijo indicado en la cabecera del ejecutable. *Ej int i[10]*. Es .bss
- Pila o stack: Región privada. Almacena registros de activación de las llamadas a funciones. Tamaño variable.



Mapa de memoria de un proceso *(cont.)*

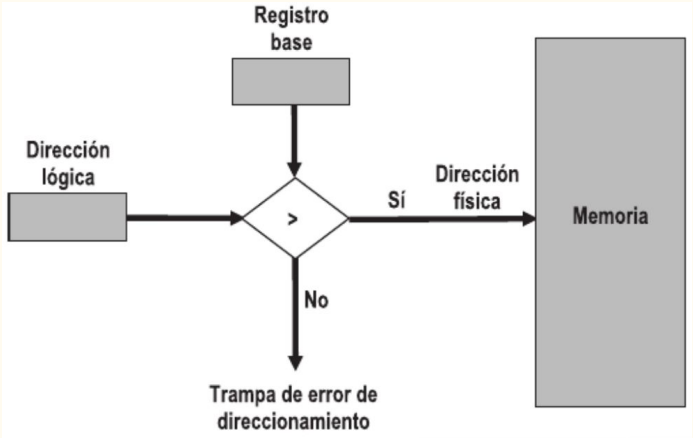
Los SO modernos ofrecen un modelo de memoria dinámico. El mapa está formado por un número variable de regiones. Pueden crearse estas:

- Montículo o heap: Memoria dinámica que reserva un proceso *(malloc)*. Región privada. Comienza luego de la región de datos sin valor inicial.
- Archivos proyectados: Cuando se proyecta un archivo, se crea una región asociada al mismo.
- Memoria compartida: Cuando se crea una zona de mc y se proyecta, se crea una región asociada.
- Pilas de threads: cada thread necesita una pila propia que se corresponde con una nueva región en el mapa. Estructura LIFO.

Esquemas de administración: Monoprogramación

Partición absoluta única

El espacio de memoria está dividido en dos: una para el SO y la otra para **UN** proceso en ejecución. Supone buen comportamiento. Los primeros DOS usaban una variante de este.



- Cuando un programa se carga, las direcciones deben corresponder con las direcciones en la partición. Pueden estar limitadas a direcciones de memoria en particular durante la compilación: **código absoluto**.
- Si el binding ocurre durante la carga es **código reubicable** se le puede agregar protección agregándole al hardware un registro base. El SO carga el RB con la dirección más baja accesible por un proceso, luego compara cada dirección generada por el proceso con el contenido del RB y las menores provocan un fallo.

Esquemas de administración - Monoprogramación

Partición reubicable única

- Contiene un registro de reubicación. Es cargado por el SO con la dirección de comienzo del proceso. Pero en lugar de comparar, sus contenidos se suman. Se opera en un espacio de direcciones lógico.
- Es compilado como si fuera a ser asignado a la memoria que comienza en la ubicación 0.
- El hardware de adm. de memoria convierte las direcciones lógicas en las direcciones físicas verdaderas.

Esquemas de administración - Monoprogramación

Superposiciones - Overlays

En los esquemas previos la cantidad de memoria de un proceso está limitada a la memoria física. Se divide el programa en partes llamadas superposiciones (overlays) .El SO se encarga de la carga y descarga pero el trabajo del programador es tedioso. El ensamblador hace dos pasadas:

- Pasada 1: Ensamblador construye tabla de símbolos.
- Pasada 2: Genera códigos
- Superposición A: tabla de símbolos, rutinas comunes y pasada 1.
- Superposición B: tabla de símbolos, rutinas comunes y pasada 2.Esta técnica puede utilizarse en los sistemas embebidos, dentro de un SoC (System on Chip) porque no tiene memoria virtual.

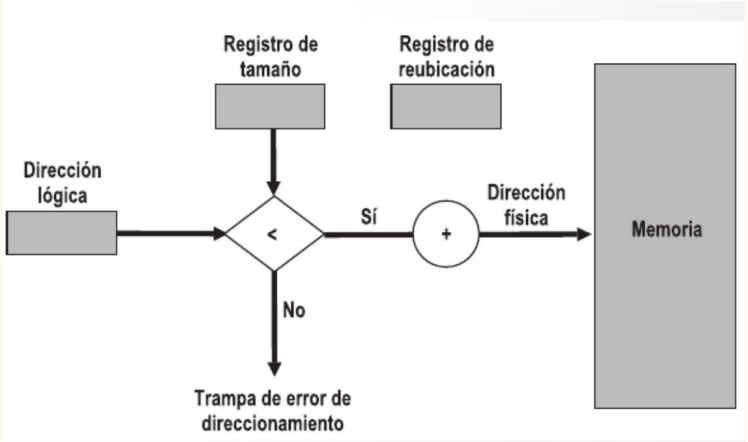
Código del paso 1	70k
Código del paso 2	80k
Tabla de símbolos	20k
Rutinas comunes a ambos pasos	30k
En total	200k

Esquemas de administración - Multiprogramación

En un ambiente de multiprogramación hay varios procesos compartiendo la memoria. El SO divide la memoria en varias particiones para que múltiples procesos queden residentes.

Múltiples particiones fijas

- Si todas las particiones son del mismo tamaño, el SO sólo necesita llevar la cuenta de cuáles particiones están asignadas a cada proceso.
- La tabla de particiones de memoria almacena o bien la dirección de comienzo para cada proceso o el número de la partición asignada.
- El espacio al final de una partición que no es usado, se desperdicia. Fragmentación interna.



Multiprogramación - múltiples particiones variables

Múltiples particiones variables

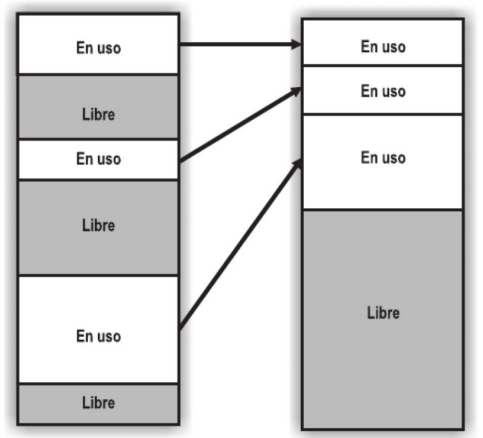
En lugar de dividir la memoria en un conjunto fijo de particiones, un SO puede elegir ubicar procesos en cualquier ubicación de memoria que esté sin usar.

A medida que los procesos se van creando y terminando, el uso de la memoria evoluciona hacia secciones alternadas de espacio asignado y sin asignar. Empiezan a generarse huecos de memoria.

A este espacio desperdiciado no asignado a ninguna partición se le llama **fragmentación externa**.

Se puede emplear compactación para hacer un uso más eficiente de la memoria.

Este movimiento puede implicar una sobrecarga.



Fragmentación externa y compactación

Algoritmo de selección de partición

- **Primer ajuste** *(first fit)*. El proceso es asignado al primer hueco encontrado que sea mayor que el tamaño del proceso.
- **Próximo ajuste** *(next fit)*. No comienza la búsqueda desde el principio sino desde el último hueco asignado.
- **Mejor ajuste** *(best fit)*. Revisa la lista completa para encontrar el hueco más pequeño.
- **Peor ajuste** *(worst fit)*. Elige el que deje el hueco remanente más grande.

Sistema de compañeras - Buddy system

- Es un compromiso entre la asignación de tamaño fijo y variable.
- La memoria se asigna en unidades que son potencia de 2.
- A un proceso se le asigna una unidad cuyo tamaño es la menor potencia de 2 pero mayor que el tamaño del proceso. Ejemplo: a un proceso de 50K se lo ubicará en una de 64K.
- Si no existe una de 64K, la asignación disponible más chica mayor que el proceso se dividirá en dos unidades “compañeras” de la mitad del tamaño que la original.
- La división continúa con una de las unidades compañeras hasta que se crea una unidad de asignación del tamaño apropiado.
- Cuando un proceso libera memoria provoca que se liberen dos unidades compañeras, las unidades se combinan para formar una unidad dos veces más grande, tratando de ese modo de minimizar la fragmentación.

Ejemplo de Buddy System

Acción	Memoria					
Comienzo	1M					
A solicita 150K	A	256K			512K	
B solicita 100K	A	B	128K		512K	
C solicita 50K	A	B	C	64K	512K	
B libera	A	128K	C	64K	512K	
D solicita 200K	A	128K	C	64K	D	256K
E solicita 60K	A	128K	C	E	D	256K
C libera	A	128K	64K	E	D	256K
A libera	256K	128K	64K	E	D	256K
E libera	512K				D	256K
D libera	1M					

Asignador de baldosas - *Slab allocator*

- Desarrollado para SunOS
- Dentro del núcleo se asigna una considerable cantidad de memoria para un conjunto finito de descriptores de archivo y estructuras comunes.
- Bonwick (1994) observó que la cantidad de tiempo requerido para inicializar un objeto ordinario excede la cantidad de tiempo requerido para asignarlo y liberarlo.
- En lugar de retornar la memoria liberada al contenedor global, la memoria sigue inicializada para el propósito previsto, reteniendo su estado entre usos.
- Se elimina así la fragmentación provocada por la asignación y liberación de memoria
- Utilizado por linux hasta la version de kernel 2.6.23

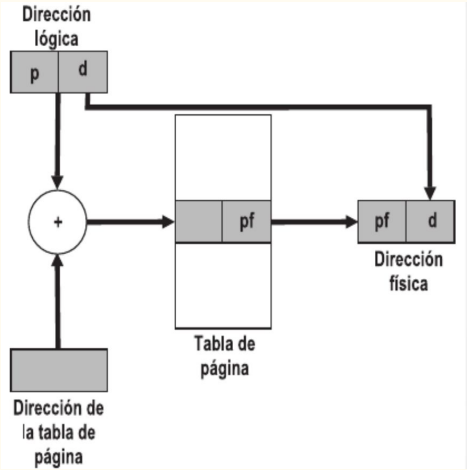
Reubicación

- En un sistema con multiprogramación no se puede conocer con antelación la posición de memoria que ocupará un proceso cuando se cargue porque dependerá del estado de ocupación de la memoria.
- Es necesario realizar un proceso de traducción o reubicación de las direcciones de memoria a las que hacen referencia las instrucciones del programa.
- Las **direcciones físicas** son los números binarios que apuntan a posiciones en la memoria física.
- Las **direcciones lógicas** son las que utiliza el proceso. El hardware convierte las lógicas en físicas.
- Las **direcciones relativas** son un caso particular de las lógicas, en las que se expresan como una posición relativa a algún punto conocido, como el principio del programa.

La Unidad de Administración de Memoria, en adelante MMU *(Memory Management Unit)*, es un componente hardware de la computadora, responsable de manejar los accesos a memoria solicitados por la CPU.

Paginación Simple

- Una solución al problema de la fragmentación externa es permitir que el espacio de direcciones lógico de un proceso sea no contiguo.
- Una forma de implementar esta solución es adoptar un esquema de paginación.
- La memoria se divide en bloques de tamaño fijo llamados **marcos** *(page frames)*. Con esto, ya no es necesario tener todo el proceso cargado en memoria.
- A los procesos se los divide en bloques llamados **páginas** *(pages)* del mismo tamaño que los marcos.
- Una **tabla de páginas** almacena el número de marco asignado a cada página de cada proceso.
- El hardware genera la dirección física añadiendo el desplazamiento de página al número de marco extraído de la tabla de páginas.
- La paginación simple, es similar al particionamiento fijo. Las diferencias son que, con la paginación, las particiones son bastante pequeñas; un programa podría ocupar más de una partición; y dichas particiones no necesitan ser contiguas.

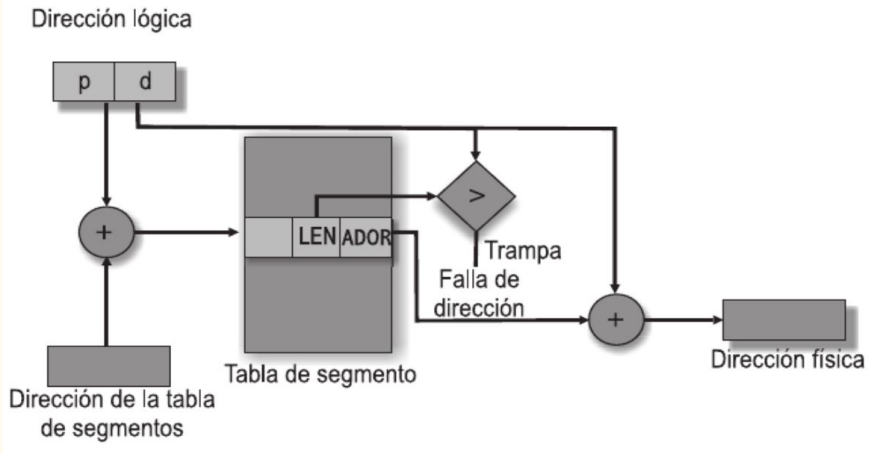


(Ampliar desde libro de Stallings pag 321)

Segmentación Simple

La visión del usuario no es igual a la memoria física real; sólo tiene una correspondencia con ella. La segmentación divide un programa en un número de segmentos, pero son de **tamaño variable** y los genera el compilador. La tabla de segmentos es similar a la tabla de particiones pero la memoria no **puede dividirse en marcos.**

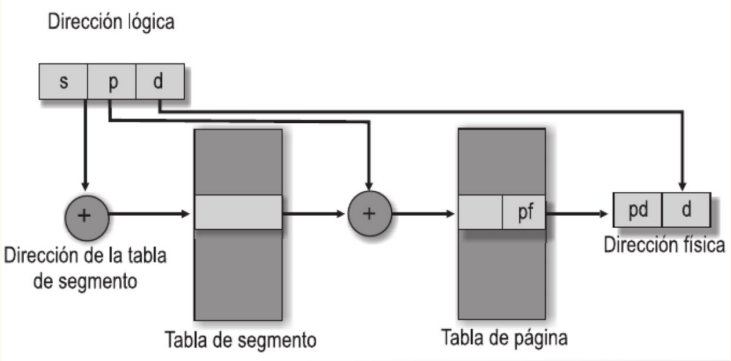
- El SO mantiene una lista de segmentos libres y los asigna.
- El compilador define segmentos de sólo lectura o compartidos.
- La dirección lógica consta de un número de segmento y un desplazamiento.
- El número de segmento sirve como índice para la tabla de segmentos.
- El desplazamiento de la dirección lógica debe estar entre 0 y LEN, sino se produce un fallo.
- Si el desplazamiento es válido se suma a ADDR, base del segmento para producir la dirección física



Segmentación con paginación

La segmentación puede combinarse con la paginación para proveer la eficiencia de esta con las posibilidades de compartir y proteger de la segmentación. La dirección lógica especifica el número de segmento y el desplazamiento dentro del segmento

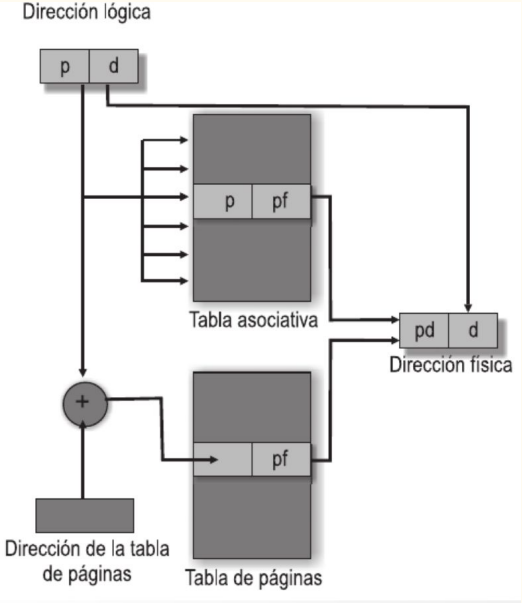
- El desplazamiento del segmento se subdivide en número de página y desplazamiento de página.
- La entrada de la tabla de segmentos contiene la dirección de la tabla de páginas de segmentos.
- El hardware agrega los bits del número de página de la dirección lógica a la dirección de la tabla de páginas para ubicar la entrada de la tabla de páginas.
- La dirección física se forma añadiendo el desplazamiento de página al número de marco de página especificado en la entrada de la tabla de páginas.



Memoria asociativa - TLB

En los sistemas que tienen tablas de páginas extremadamente grandes se puede usar la memoria asociativa *(Translation Lookaside Buffer o TLB)* que es una memoria ultra rápida de la CPU administrada por la MMU. Se la suele traducir como “buffer de traducción anticipada".

- Cada elemento en una memoria asociativa está identificado mediante un valor índice.
- A diferencia de la memoria convencional -que se basa en el conocimiento exacto de la dirección de memoria en la que se encuentra la información-, la memoria asociativa no está referenciada por una dirección, sino con un valor de búsqueda *(se le suele llamar “direccionamiento por contenido”)*.
- Se revisan todos los elementos buscando uno con un valor índice correspondiente.
- Cuando se especifica una página se busca la memoria asociativa al mismo tiempo que se accede a la tabla de páginas.Por esta forma de buscar se la denomina en inglés lookaside, que podríamos traducir como “mirando de reojo”.



Memoria asociativa - TLB *(cont.)*

Si se encuentra una entrada en la memoria asociativa *(cache hit o TLB hit)* con el número de página coincidente, se aborta el acceso a la tabla de páginas y se usa la entrada de la memoria asociativa.

Si no se encuentra una coincidencia en la memoria asociativa, se usa la entrada de la tabla de páginas *(page walk)*, lo que tardará varios ciclos más, sobre todo si la página no se encuentra la dirección buscada, se provoca un fallo de página que deberá manejar el sistema operativo.

A la misma vez que está siendo traducida la dirección lógica, la memoria asociativa reemplaza una de sus páginas, la entrada menos usada recientemente.

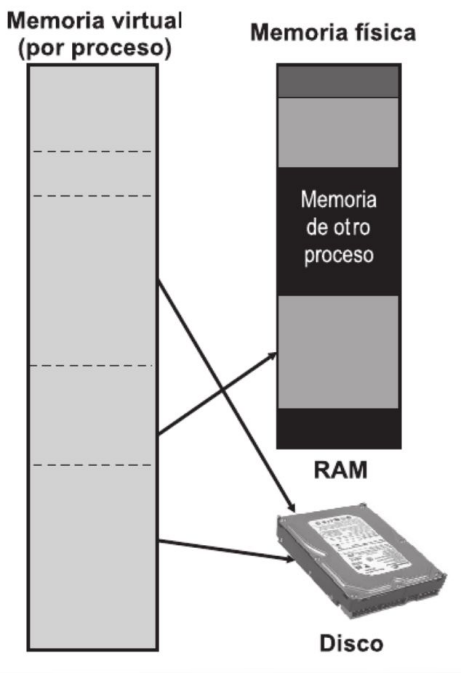
De esta manera, la memoria asociativa mantiene una copia de las entradas de las páginas más usadas recientemente.

La mayor mejora en el desempeño se logra cuando la memoria asociativa es significativamente más rápida que la búsqueda en la tabla de páginas normal y la tasa de aciertos (hit ratio) es alta. La tasa de aciertos es el cociente entre los accesos que encuentran una coincidencia en la memoria asociativa *(los exitosos)* y aquellos que no lo encuentran.

Intercambio y memoria virtual

Puede ocurrir que el SO decida que un proceso en memoria salga temporalmente, porque necesita más espacio, para incrementar el número de procesos que comparten la CPU o porque llega un proceso de mayor prioridad.

- La técnica de llevar temporalmente un proceso a memoria secundaria se llama intercambio o swapping.
- Cuando se saca de memoria, se hace swap-out; cuando se trae nuevamente, swap-in. Como esta técnica ahora se usa en combinación con el paginado, los términos son page-out y page-in.
- En Linux es habitual disponer de una partición especial para intercambio, si bien también se puede usar un archivo de intercambio. En Windows se utiliza un archivo de intercambio.
- El planificador de mediano plazo es el que lleva a cabo estas tareas.



Las alternativas de administración de memoria vistas cargan todo el proceso de manera contigua. Esto exige mayores espacios de memoria con secciones del proceso que no se utilizarán enseguida, pues la ejecución de un proceso es secuencial y se adapta al modelo de localidad

Intercambio y memoria virtual *(cont.)*

Coincidente con la evolución que tuvieron los SO y el hardware, se desprende el siguiente avance a partir de dos características de la paginación y la segmentación:

1. Todas las referencias a la memoria dentro de un proceso son direcciones lógicas que se traducen dinámicamente a direcciones físicas durante la ejecución, de tal forma que un proceso puede cargarse y descargarse de la memoria principal ocupando regiones diferentes en distintos momentos de su ejecución.
2. Un proceso puede dividirse en varias partes y no es necesario que esas partes se encuentren contiguas en la memoria principal durante su ejecución.

Fotheringham (1961) creó esta técnica y la llamó memoria virtual. La idea básica es que el tamaño del programa, los datos y la pila combinados pueden ser mayores que la memoria (real) disponible. El programador se desentiende de la cantidad de memoria real que necesitará su proceso, ya que el proceso hace uso de su propio espacio de direcciones virtuales. El SO guarda aquellas partes del programa de uso corriente en la memoria principal y el resto permanece en disco.

Paginación por demanda

La paginación por demanda combina las características de la paginación simple y las superposiciones para implementar memoria virtual.

- Cada página de un programa se almacena en forma contigua en el espacio de intercambio de paginación en almacenamiento secundario.
- A medida que se hace referencia a ubicaciones en páginas, estas páginas se copian en los marcos de página de memoria. Una vez que la página está en la memoria, se accede a ella como en la paginación simple.
- Cada entrada en la tabla de páginas tiene como mínimo dos campos: el marco de la página y el bit dentro/fuera (in/out bit).
- Cuando se genera una dirección virtual, el hardware de administración de memoria extrae el número de página de la dirección y se accede la entrada correspondiente en la tabla de páginas.
- Se comprueba el bit dentro/fuera y, si la página está en memoria, se genera la dirección física, añadiendo el desplazamiento de página al número de marco de página.
- Si la página no está en memoria, se produce una trampa *(trap)* o excepción por fallo de página *(page fault)*, transfiriendo el control a la rutina de fallos de página del sistema operativo.

Paginación por demanda (cont.)

- Sólo las páginas reemplazadas que han sido modificadas necesitan ser escritas nuevamente al espacio de intercambio.
- Cuando no hay memoria virtual, la dirección generada por el proceso se coloca directamente en el bus de memoria.
- Cuando hay memoria virtual, la dirección pasa a la MMU.
- En un esquema de paginado por demanda puro, el proceso comienza haciendo fallos de página con la primera instrucción que se ha de ejecutar, y así continúa, con una alta tasa de fallos, hasta que se estabiliza al tener todas las páginas necesarias en memoria.

(ampliar desde el libro de Silva, pag 99)

Algoritmos de reemplazo de páginas

Es el que selecciona la página para ser intercambiada a almacenamiento de intercambio. El óptimo selecciona para remover a la página que no será referenciada de nuevo en lapso más largo de instrucciones ejecutadas. Es teórico.

- **Primero en Entrar, Primero en Salir** (First In First-Out o FIFO): Selecciona la página que ha estado en memoria por más tiempo. Las entradas de la tabla de páginas deben incluir un campo para el tiempo en el que fue cargada en memoria. Cuando se carga la página, el SO graba el campo con la hora actual. La página seleccionada para reemplazo será la que tenga la hora de carga más temprana.
- **Menos Usado Recientemente** (Least Recently Used o LRU): mantiene el registro de la última vez que fue usada cada página, no cuando fue cargada en memoria. El hardware de administración de la memoria usa un contador que se incrementa durante cada referencia a memoria. Cada entrada de la tabla de páginas tiene un campo que almacena el valor del contador. Cuando se hace referencia a una página, se almacena el valor del contador en la correspondiente entrada de la tabla de páginas. Busca en la tabla una entrada con el valor de contador más bajo y selecciona esa página para reemplazo.

Algoritmos de reemplazo de páginas *(cont.)*

- **No Usado Recientemente** (Not Recently Used o NRU): es una aproximación a LRU. Además del bit sucio, cada entrada de la tabla de páginas contiene el bit de referencia. Cuando se carga una página, el SO pone el bit de referencia en cero. Cuando una página es accedida, el hardware pone el bit en 1. Además, a intervalos periódicos, el SO pone todos los bits de referencia en la tabla de páginas de vuelta en cero. Un valor de cero en el bit de referencia significa que no ha sido referenciado "recientemente" y un 1 significa que sí.
- **Envejecimiento** (Aging): es otra aproximación a LRU. Se logra al agregar un byte de referencia a las entradas de la tabla de páginas. Cuando se accede una página, el hardware pone en 1 el bit más significativo en el byte de referencia. El SO desplaza todos los bits a la derecha un lugar en los bytes de referencia. Se selecciona para eliminar a una página con el valor binario más bajo en su byte de referencia.
- **Segunda Oportunidad** (Second Chance): es una combinación de los algoritmos FIFO y NRU. Selecciona a la página más vieja como candidata para remover y la remueve si su bit de referencia es cero. Si su bit de referencia es 1, su hora de carga se restablece a la hora actual y su bit de referencia se pone en cero. El algoritmo luego se repite con la segunda página más vieja, siendo ahora la más vieja. El proceso continúa hasta que es seleccionada una página

Hiperpaginación

Aunque técnicamente es posible reducir el número de marcos asignados al mínimo, hay cierto número (mayor) de páginas que están en uso activo. Si el proceso no cuenta con este número de marcos, causará muy pronto un fallo de página. En ese momento, el proceso deberá reemplazar alguna página y, dado que todas sus páginas están en uso activo, deberá reemplazar una que volverá a necesitar de inmediato. Por lo tanto, se generará rápidamente otro fallo de página, y así sucesivamente. El proceso seguirá causando fallos, reemplazando páginas por las que entonces causará otro fallo y traerá de nuevo a la memoria

Al rendimiento degradado motivado por el intercambio excesivo se lo conoce como **hiperpaginación** *(thrashing, literalmente "paliza" o "fustigamiento")*. **Un proceso está hiperpaginando si pasa más tiempo paginando que ejecutando.**

Conjunto de trabajo

El conjunto de trabajo de un proceso en un determinado tiempo es el conjunto de páginas referenciadas en un cierto intervalo de tiempo precedente. Con frecuencia, se expresa el conjunto de trabajo usando la notación funcional

$$W(t,\Delta)$$

donde W representa al conjunto de trabajo, t representa el tiempo y Δ *representa el intervalo.*

Usualmente, es más simple medir a *t* en términos de instrucciones ejecutadas. El objetivo es elegir un valor para Δ de manera tal que el conjunto de trabajo refleje la localidad del programa.

- Si el valor de Δ es demasiado pequeño, el conjunto de trabajo no incluirá todas las páginas en la localidad actual.
- Si el valor de Δ es demasiado grande, el conjunto de trabajo incluirá páginas de una localidad previa.

Prepaginado

Si el sistema operativo conoce el conjunto de trabajo al momento de que el proceso fue intercambiado a almacenamiento secundario, puede prepaginar *(prepage)* todas las páginas en ese conjunto de trabajo cuando el proceso sea intercambiado nuevamente a memoria.

- El prepaginado previene que la referencia inicial a las páginas del conjunto de trabajo generen un fallo de página.
- Esto le ahorra al SO la sobrecarga extra de procesar esos fallos de página.
- Sin embargo, algunas de las páginas cargadas pueden nunca ser referenciadas.
- Prepaginar una página que no es referenciada, degrada el rendimiento de la entrada/salida y desperdicia marcos de página.

Segmentación

Así como la paginación simple puede modificarse para crear paginación por demanda, la segmentación también se puede modificar para crear segmentación por demanda.

- Sin embargo, la variabilidad de los tamaños de los segmentos complica muchos de los problemas encontrados en la paginación por demanda.
- En la decisión del intercambio, el tamaño de los segmentos se convierte en un factor importante para decidir qué segmentos intercambiar.
- Una forma mucho más práctica de implementar segmentación en un sistema con memoria virtual es combinarlo con paginación por demanda.

La capacidad de memoria virtual se crea por paginación por demanda de los segmentos