Sistemas Operativos

Intercambio y asignación de memoria

Departamento de Ingeniería en Sistemas y Computación
Universidad Católica del Norte, Antofagasta.

Introducción

- Idealmente, los programadores quieren memoria
  - mucha, rápida, no volátil, barata
- Jerarquía de memorias
  - Caché: poca, rápida, cara
  - RAM: Mb, velocidad y precio medios
  - Disco: Gb, lenta y barata
- Es un problema de costes

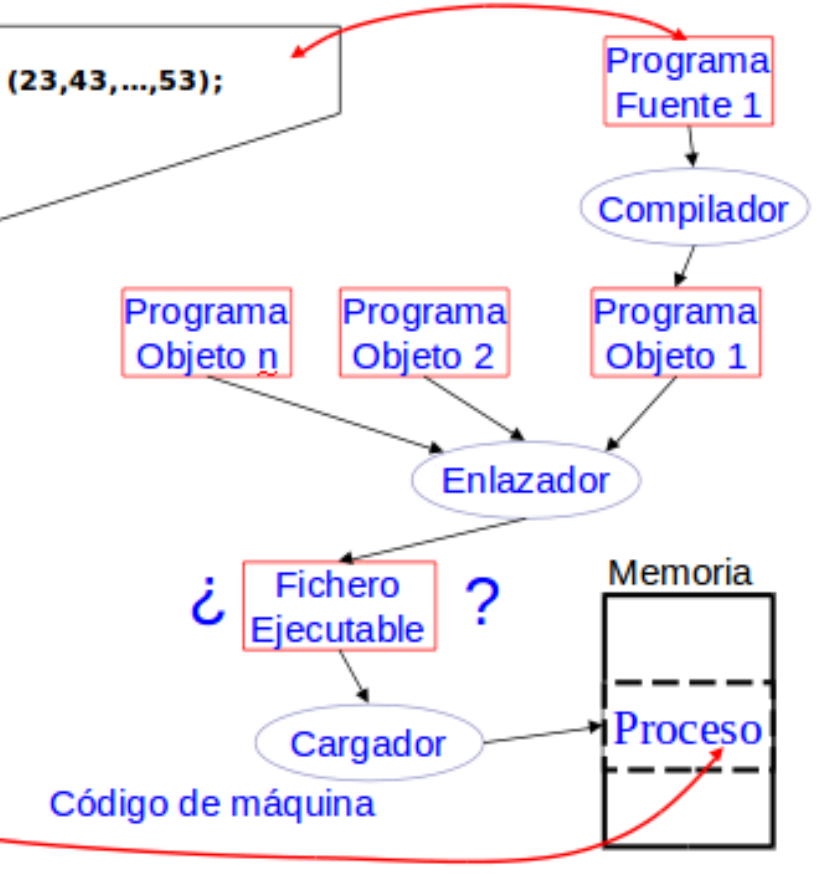
Introducción

- Trabajos del gestor de memoria
  - Qué memoria está libre/ocupada
  - Asignación/liberación de memoria a procesos
  - Intercambio RAM-disco
- Clases de gestores de memoria
  - Con intercambio
    - Swapping y paginación
  - Sin intercambio

Introducción

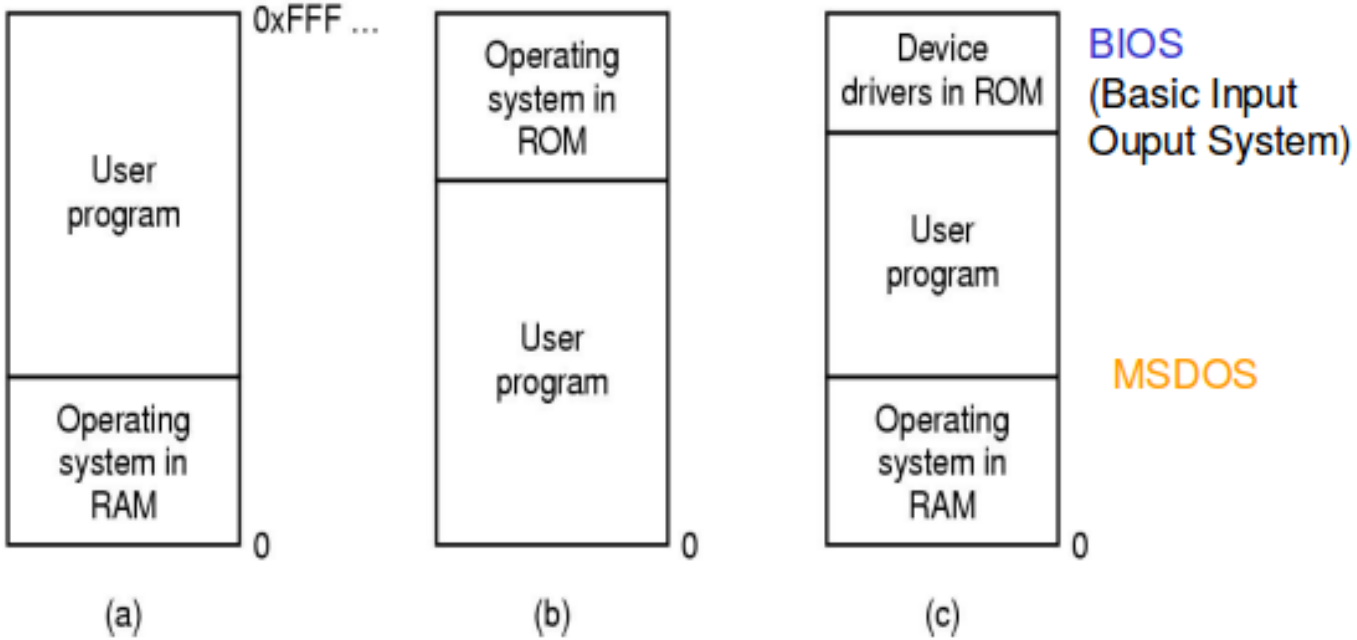
```
Program principal;
var
  Vector1: array [0..$FFF] of byte := (23,43,...,53);
  Vector2: array [0..$4FF] of byte;
begin
   ...
  if Vector1[0] = Vector2[0]
    then Subrutina;
   ...
end.
```

```
$5000    ...
$5030    cmp $9000,$A000
$5036    bnz $503E
$503A    bsr $5200
$503E    ...
$5200    (codigo de subrutina)
         ...
$9000    (valores de vector1)
         ...
$A000    (valores de vector2)
         ...
```

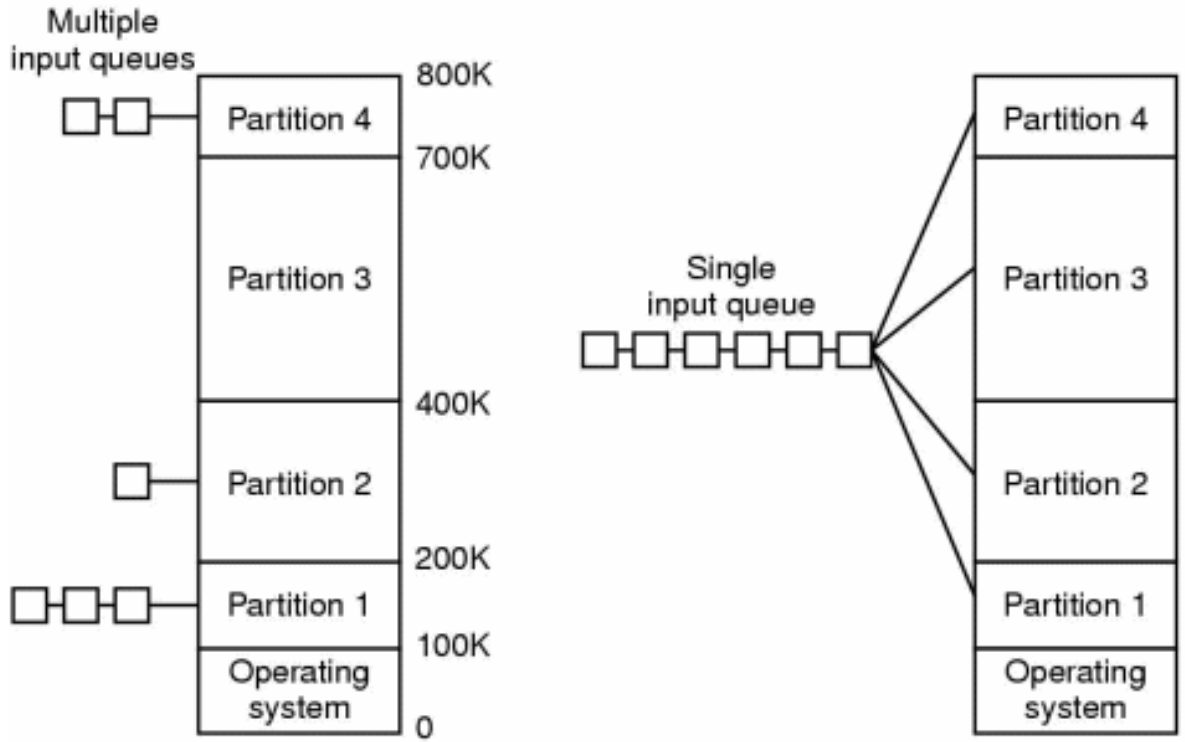


Monoprogramación sin intercambio ni paginación

- Solo un programa en memoria (junto con el SO)
- Se carga y se queda ahí hasta que acaba (CP/M)



Multiprogramación con particiones fijas

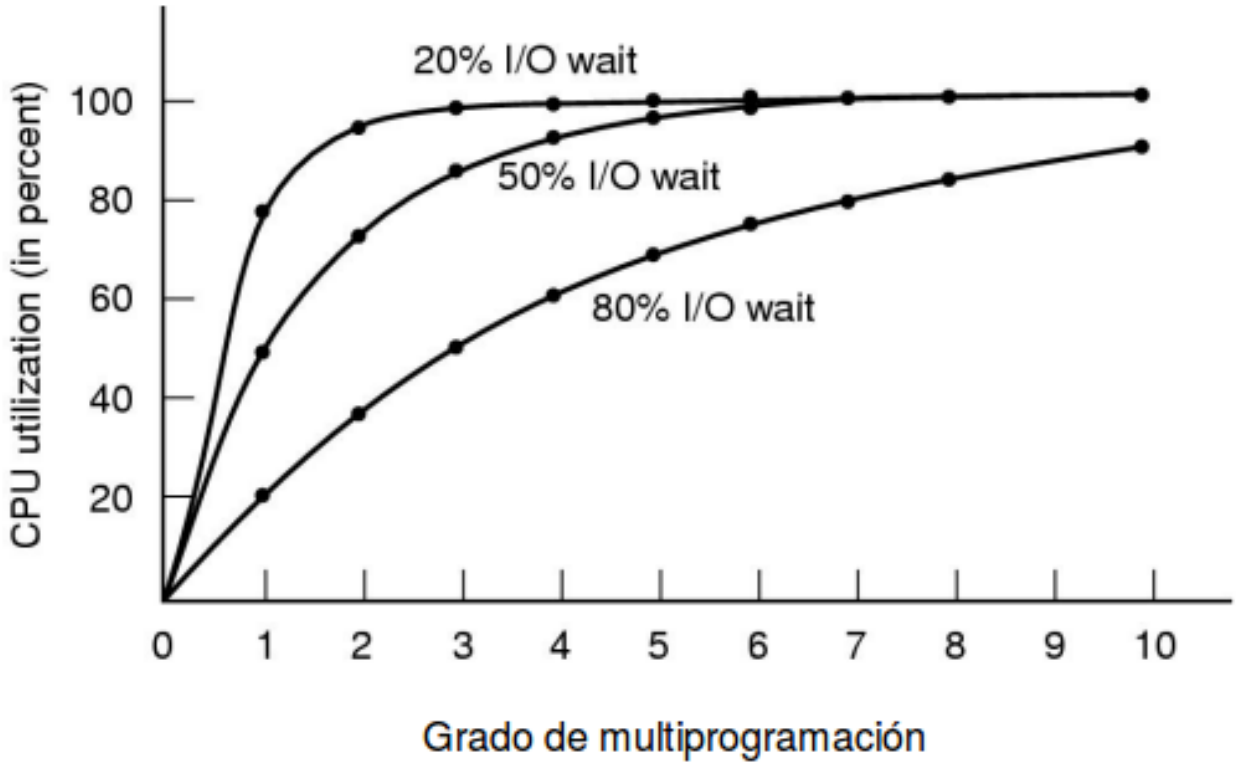


Multiprogramación con particiones fijas

Planificación: queda libre una partición, ¿qué trabajo cargar?

- El primero en la cola que quepa
  - Fragmentación interna
- El más grande de la cola que quepa
  - Se perjudica a los pequeños (y debe ser al revés)
    - Disponer de una partición pequeña
    - No retrasar un trabajo más de k veces

Modelo de multiprogramación



Reubicación y protección

- Los procesos son independientes
- n: grado de multiprogramación
- p: fracción de tiempo que un proceso está esperando una I/O

- Utilización de la CPU = $1 - p^n$

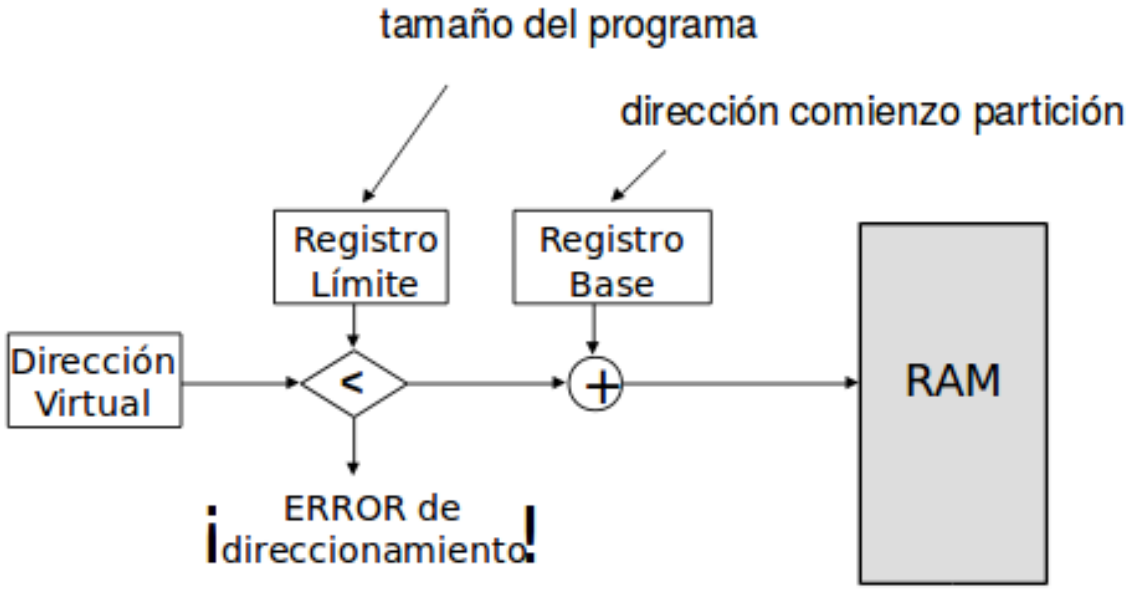
- Los procesos no son independientes: 1 sola CPU-> los otros en preparados (esperando).

Aproximación válida. Ejemplo:
32Mb memoria,
el SO ocupa 16Mb
cada proceso de usuario 4Mb => n = 4; si p=0.8, CPU = 60%

Si compramos 16Mb => n = 8; si p=0.8, CPU = 83%; ganancia: 38%
Si compramos otros 16Mb CPU=93%; ganancia: 12%

Reubicación y protección

Protección: acceso indiscriminado a cualquier área de memoria



Intercambio (swapping)

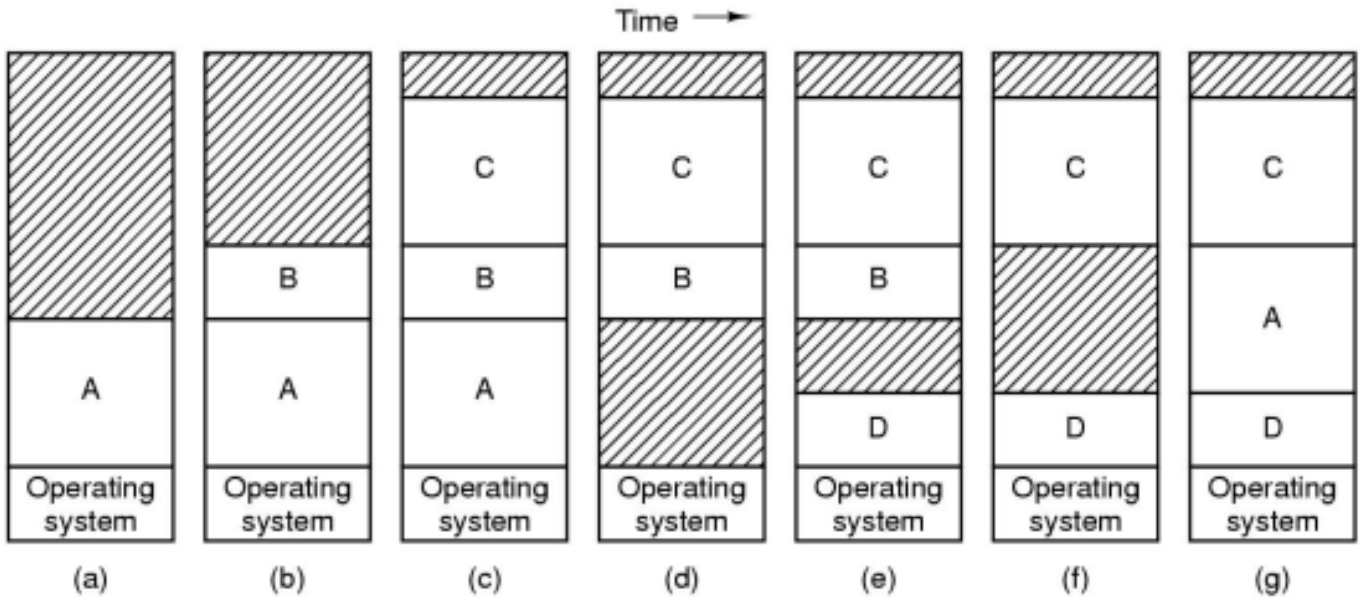
Batch ⟶ Se aceptan tantos trabajos como quepan en memoria

Tiempo Compartido ⟶ Suele haber más procesos de usuario de los que caben en memoria
¡ y hay que atenderlos a todos!

- Dos aproximaciones:
  - Intercambio (entre RAM y disco)
  - Memoria virtual (solo una parte del programa en RAM)

Intercambio (swapping)

PARTICIONES DE TAMAÑO VARIABLE
(el nº, tamaño y dirección varía con el tiempo)



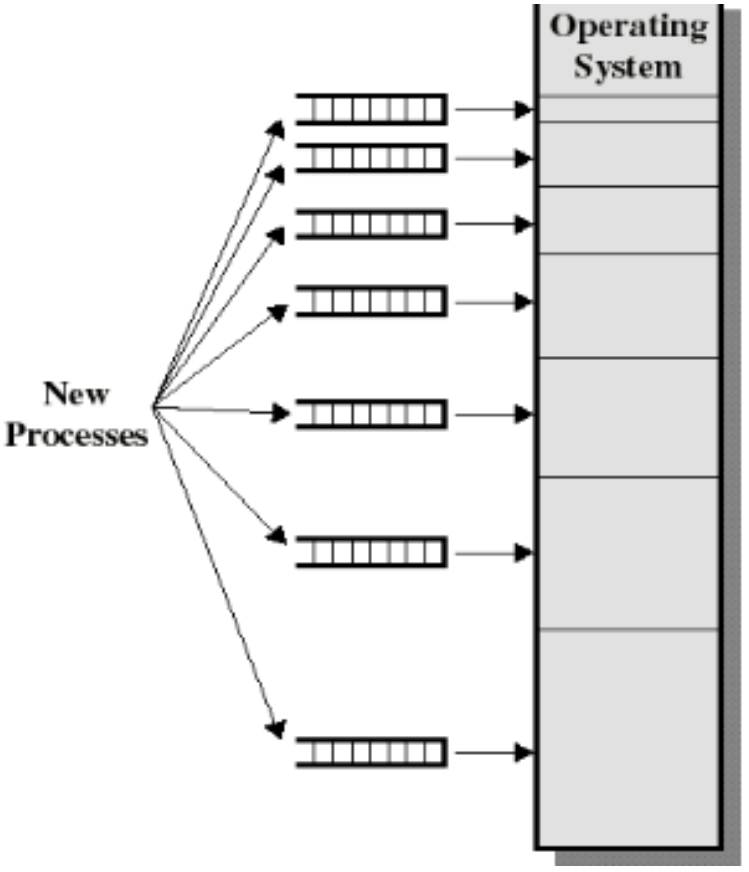
¡ Fragmentación Externa ! → COMPACTACIÓN

Algoritmo de ubicación con particiones

- Particiones del igual tamaño
  - Si hay una partición disponible, un proceso puede cargarse en esa partición
    - Debido a que todas las particiones son del mismo tamaño, no es importante cual partición se usa
  - Si todas las particiones están ocupadas por procesos que no están listos para ejecutar, escoge un proceso para sacarlo y hacer sitio para el nuevo proceso

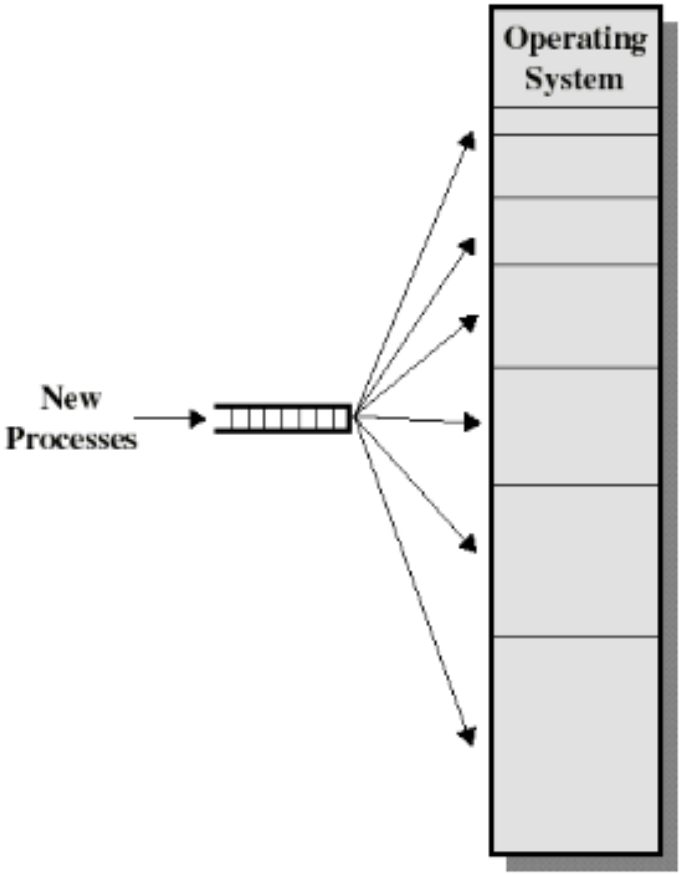
Algoritmo de ubicación con particiones

- Particiones de diferente tamaño: múltiples colas
  - Asigne cada proceso a la partición más pequeña dentro de la que quepa
  - Una cola de planificación para cada partición, que albergue los procesos expulsados, cuyo destino es dicha partición
  - Intenta minimizar la fragmentación interior
  - Problema: algunas colas estarán vacías, si ningún proceso dentro de un rango de tamaño, está presente



Algoritmo de ubicación con particiones

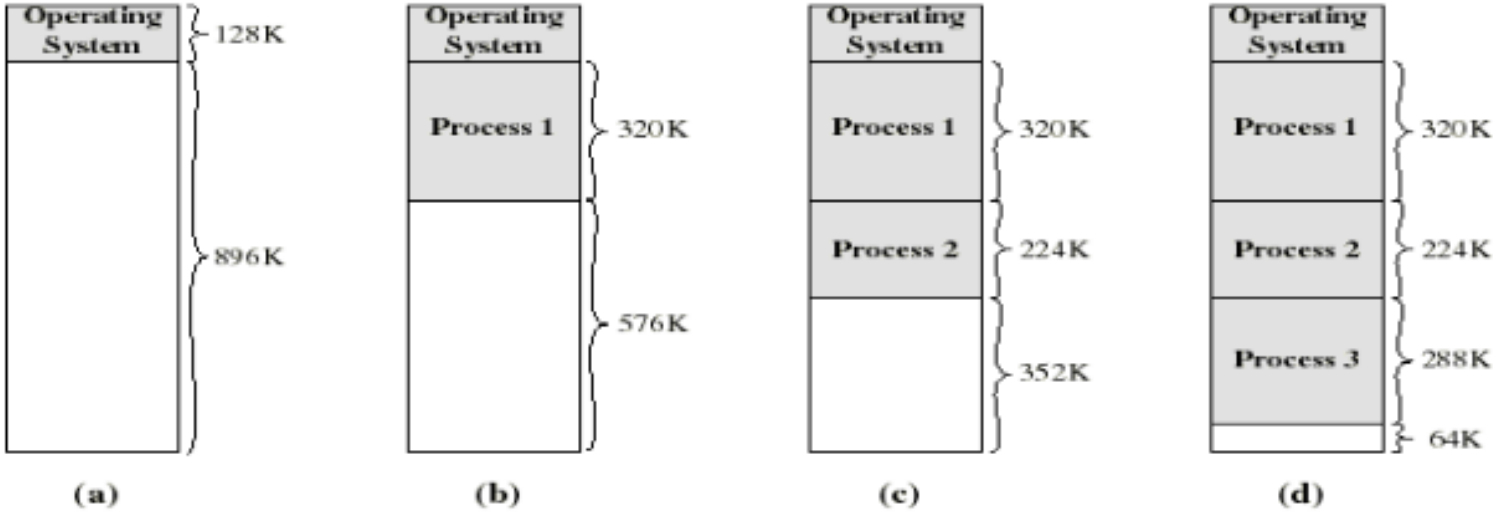
- Particiones de diferente tamaño: una sola cola
- Cuando se va a cargar un proceso en la memoria principal, la partición disponible más pequeña que pueda albergar el proceso, se selecciona
- Aumentos del nivel de multiprogramación, a expensas de la fragmentación interna



Partición dinámica

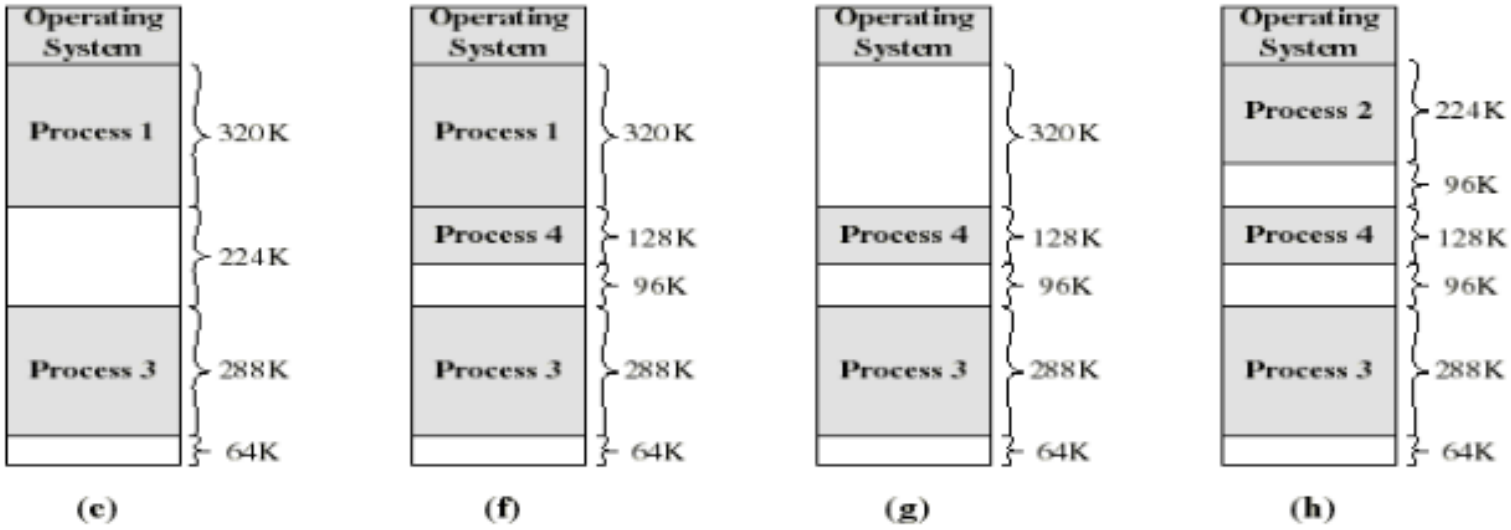
- Particiones variables en longitud y número
- A cada proceso se le asigna exactamente la memoria que éste requiere
- Eventualmente se forman agujeros en memoria principal. Esto se llama fragmentación externa
- Debe usar compactación para desplazar los procesos, para que ellos estén contiguos y toda la memoria libre quede junta en un bloque

Partición dinámica (ejemplo)



- Un hueco de 64K es dejado, después de cargar 3 procesos: no hay suficiente espacio para otro proceso
- Eventualmente cada proceso se bloquea. El OS saca el proceso 2 para traer el proceso 4

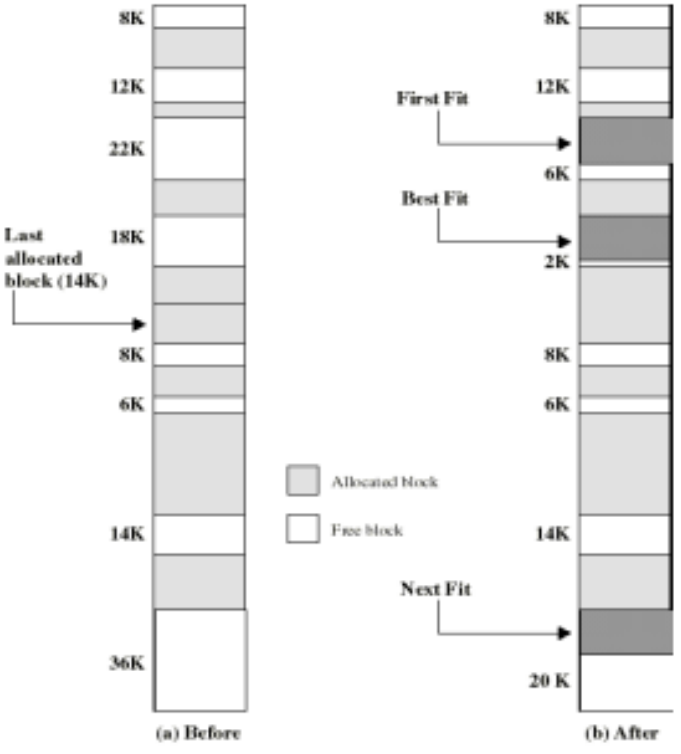
Partición dinámica (ejemplo)



- Otro hueco de 96K es creado.
- Eventualmente cada proceso se bloquea. El OS saca el proceso 1 para traer de nuevo proceso 2 y otro hueco de 96K se crea...
- La compactación produciría un solo hueco de 256K

Algoritmo de ubicación

- Usado para decidir cual bloque libre asignar a un proceso
- Objetivo: para reducir el uso de compactación (tiempo que consume)
- Posibles algoritmos:
  - El de mejor ajuste (Best-fit): escoja el hueco más pequeño
  - El Primer ajuste (First-fit): escoja el primer hueco desde desde el principio
  - Siguiente ajuste (Next-fit): escoja el primer hueco desde la última ubicación



Example Memory Configuration Before and After Allocation of 16 Kbyte Block

Algoritmo de ubicación

- Siguiente-ajuste lleva a menudo a la asignación (allocation) del bloque más grande, al final de la memoria
- Primer-ajuste favorece la asignación cerca del comienzo: tiende a crear menos fragmentación que Siguiente-ajuste
- Mejor-ajuste busca el bloque más pequeño: el fragmento dejado atrás es lo más pequeño posible
  - Memoria principal rápidamente forma huecos demasiado pequeños para mantener cualquier proceso: la compactación generalmente necesita ser hecha más a menudo

Ejercicio

Considere un sistema con intercambio, en el que la memoria posee particiones libres de tamaño fijo: 700Kb, 500Kb, 700Kb, 1300Kb, 800Kb, 1000Kb y 1300Kb. Estos huecos están dispuestos en el orden dado. Se tienen tres procesos de tamaños 1000Kb, 800Kb y 1200Kb. Determine que huecos serán asignados para los algoritmos:

- Primero en ajustarse
- Mejor en ajustarse
- Peor en ajustarse
- Siguiente en ajustarse

Ejercicio

Considere un sistema con intercambio, en el que la memoria posee particiones libres de tamaño fijo: 900Kb, 400Kb, 1800Kb, 700Kb, 900Kb, 1200Kb y 1500Kb. Estos huecos están dispuestos en el orden dado. Se tienen tres procesos de tamaños 1200Kb, 1000Kb y 900Kb. Determine que huecos serán asignados para los algoritmos:

- Primero en ajustarse
- Mejor en ajustarse
- Peor en ajustarse
- Siguiente en ajustarse

Relocalización

- Debido al intercambio y a la compactación, un proceso puede ocupar diferentes localizaciones de memoria principal durante su vida
- Debido a esto, las referencias a memoria física para un proceso, no pueden ser fijas

Tipos de dirección

- Este problema es resuelto distinguiendo diferentes tipos de dirección:
  - Dirección física (dirección absoluta).
  - Dirección lógica.
  - Dirección relativa.

Tipos de dirección

- Una dirección física (dirección absoluta) es una localización física en memoria principal
- Una dirección lógica es una referencia a una posición de memoria, independiente de la estructura/organización física de memoria
- Los compiladores producen código en el cual todas las referencias de memoria son direcciones lógicas
- Una dirección relativa es un ejemplo de dirección lógica, en la cual la dirección es expresada como una localización relativa a algún punto conocido en el programa (ejemplo: el principio)

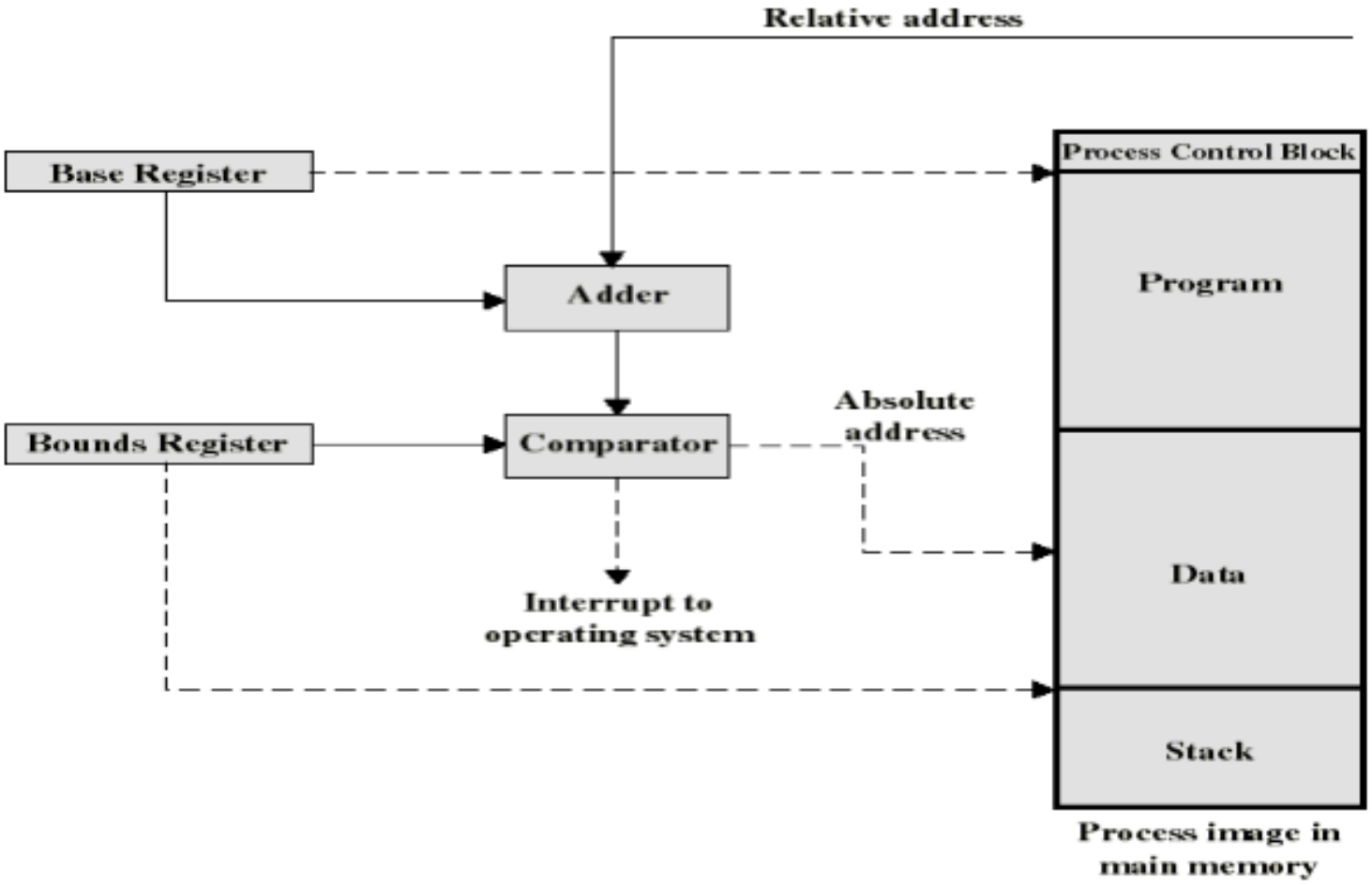
Traducción de direcciones

- El direccionamiento relativo es el tipo más frecuente de direccionamiento lógico, usado en módulos de programa (archivos ejecutables)
- Tales módulos son cargados en memoria principal, con todas las referencias de memoria en forma relativa
- Las direcciones físicas son calculadas en el camino “on the fly”, a medida que las instrucciones son ejecutadas
- Para un rendimiento adecuado, la traducción de dirección relativa a dirección física, debe ser hecho por hardware

Traducción de direcciones por hardware

- Cuando un proceso se asigna al estado de ejecución, un registro base (en CPU) es cargado con la dirección física de comienzo del proceso
- Un registro límite es cargado con la dirección física de finalización del proceso
- Cuando una dirección relativa es encontrada, ésta se suma con el contenido del registro base para obtener la dirección física, la cual es comparada con el contenido del registro limite
- Esto proporciona protección de hardware: cada proceso puede acceder sólo memoria dentro de su imagen de proceso

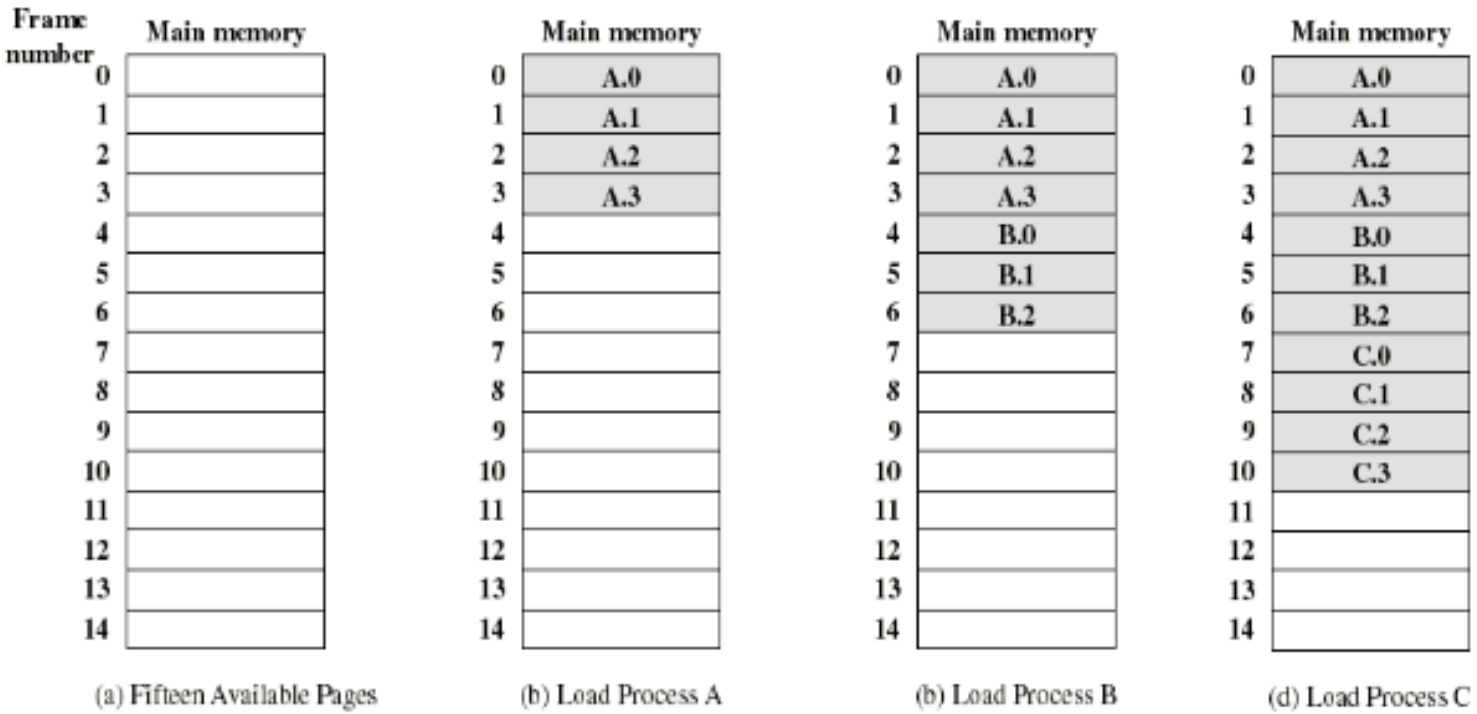
Traducción de direcciones por hardware



Paginación simple

- La memoria principal está particionada en pedazos iguales de tamaño-fijo (de tamaño relativamente pequeño)
- Truco: cada proceso también es dividido en pedazos del mismo tamaño llamado páginas
- Las páginas del proceso pueden asignarse a los pedazos disponibles en memoria principal llamado marcos (o marcos de página)
- Consecuencia: un proceso no necesita ocupar una porción contigua de memoria

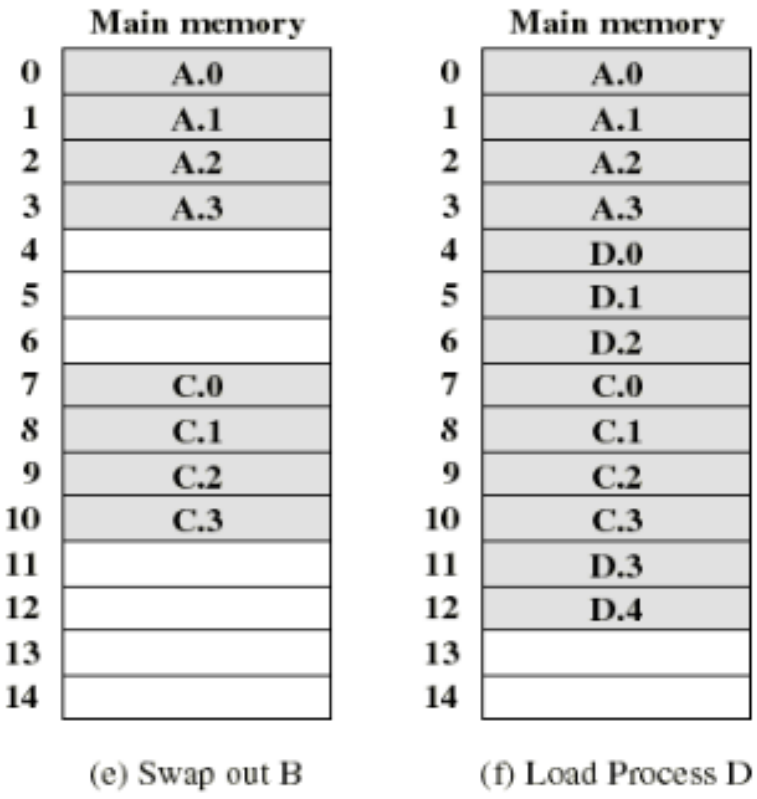
Ejemplo de carga de proceso



- Ahora suponga que el proceso B es sacado de memoria (swapped out)

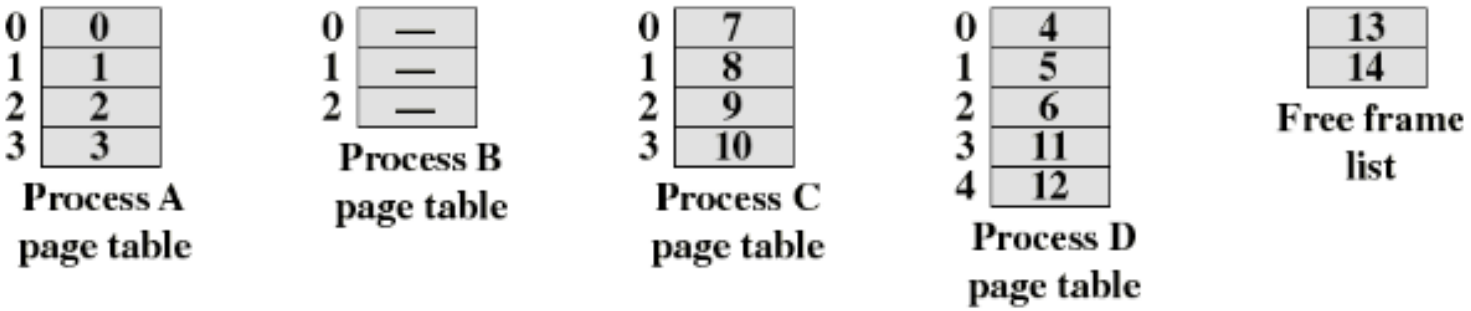
Ejemplo de carga de proceso

- Cuando los procesos A y C se bloquean, el paginador (pager) carga un nuevo proceso D que consiste de 5 páginas
- El proceso D no ocupa una porción contigua de memoria
- No hay fragmentación externa
- La fragmentación interna consiste solamente de la última página de cada proceso



(e) Swap out B (f) Load Process D

Tabla de páginas



- El OS ahora necesita mantener (en memoria principal) una tabla de página por cada proceso
- Cada entrada de la tabla de páginas, consiste del número del marco donde la página correspondiente se localiza físicamente
- La tabla de página está indexada por el número de página para obtener el número del marco
- Se mantiene una lista de marcos libres, disponible para las páginas.