Sistemas Operativos I

Administración de memoria

The kernel will accept any `mem=xx' parameter you give it, and if it turns out that you lied to it, it will crash horribly sooner or later.

Linus Torvalds

2022 - Rafael Ignacio Zurita <rafa@fi.uncoma.edu.ar>
Depto. Ingeniería de Computadoras

Advertencia: Estos slides traen ejemplos.

No copiar (ctrl+c) y pegar en un shell o terminal los comandos aquí presentes.

Algunos no funcionarán, porque al copiar y pegar tambien van caracteres "ocultos" (no visibles pero que están en el pdf) que luego interfieren en el shell.

Sucedió en vivo :)

Conviene "escribirlos" manualmente al trabajar.

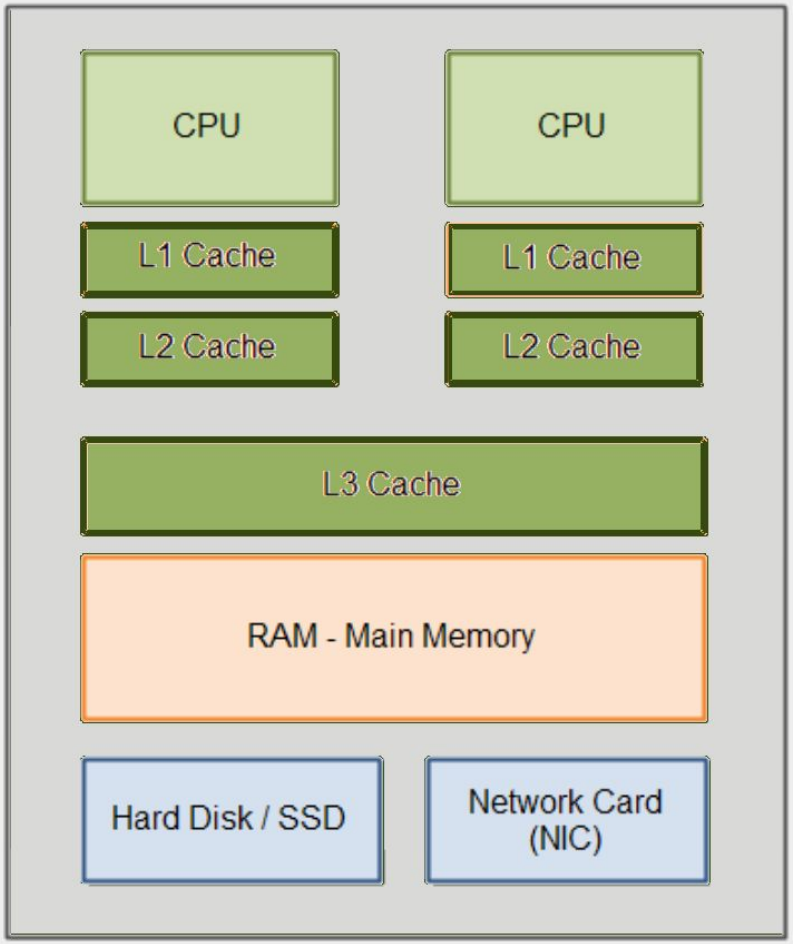
Sistemas Operativos I - Administración de memoria

Contenido:

- Repaso del hardware de memoria
- Asignación de espacio contiguo
  - Overlays
  - Algoritmos de asignación de memoria
  - Ejemplo: Xinu i386
- Swapping
- Gestión del espacio libre: mapa de bits, listas enlazadas
- Memoria Virtual:
  - Paginación a demanda con swapping
  - Fallos de página
  - Tabla de páginas
  - Hardware y Software necesarios
  - Algoritmos de Reemplazo - Trashing - working set -copy on write
  - Ejemplo: Linux amd64

Sistemas Operativos I - Administración de memoria

Repaso del hardware de memoria y segmentos de un proceso



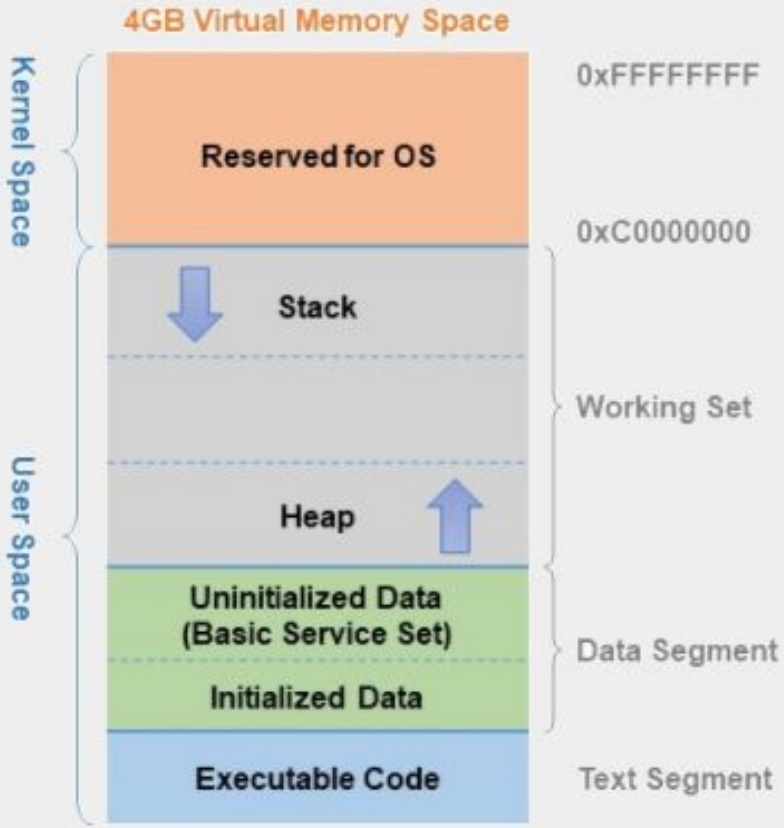
MICROPROCESADOR

MEMORIA

ALMACENAMIENTO SECUNDARIO

Sistemas Operativos I - Administración de memoria

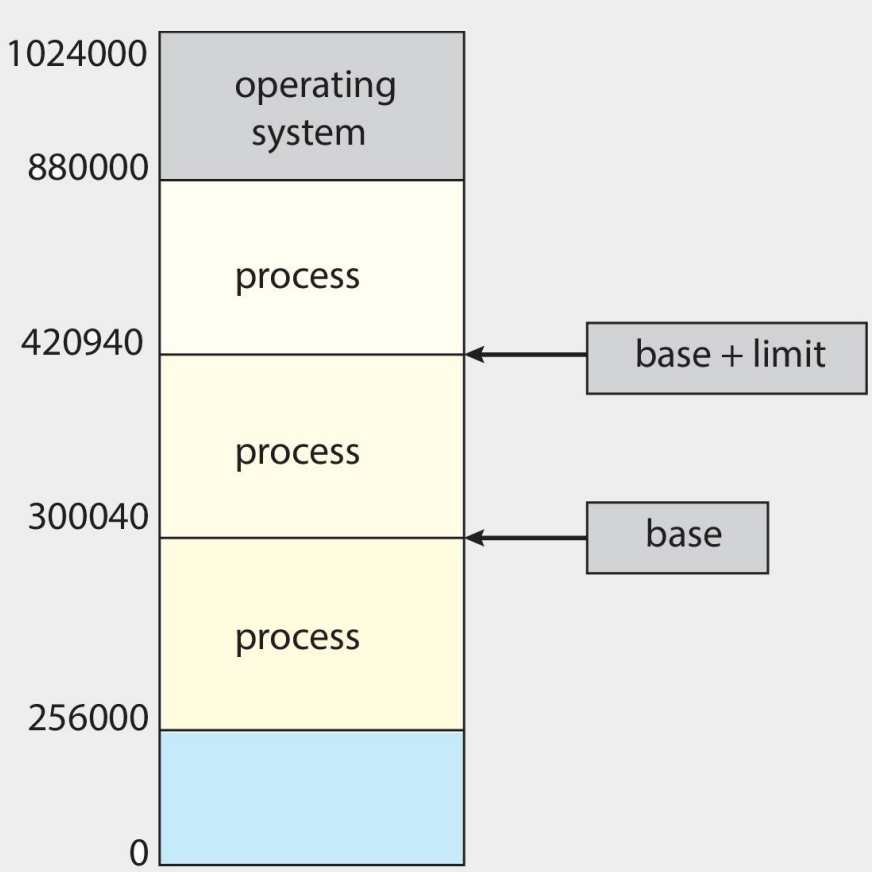
Repaso del hardware de memoria y segmentos de un proceso

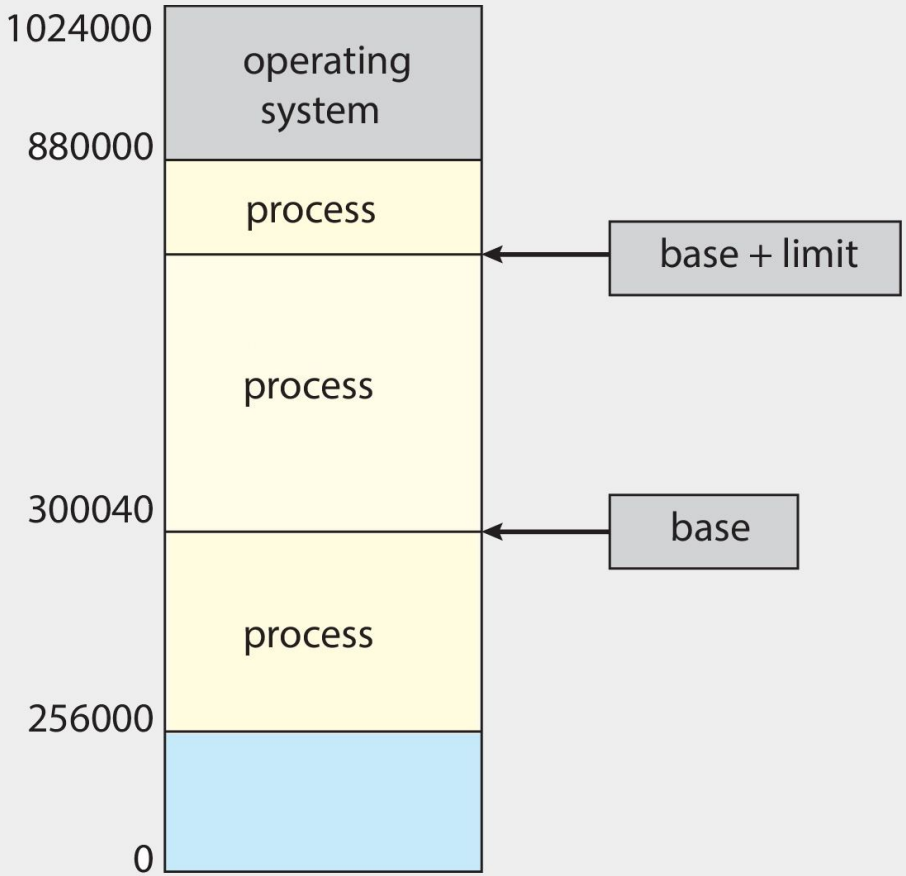



```c
#include <stdio.h>

char c;
int r = 1;

int multiplicar (int m, int n) {

    int i;
    int y=0;

    for (i=0; i < m; i++)
        y = y + n;

    return y;
}

main () {

    int a;
    int i;

    r = multiplicar(r, a);

    printf("r = %d \n", r);
}
```

Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo



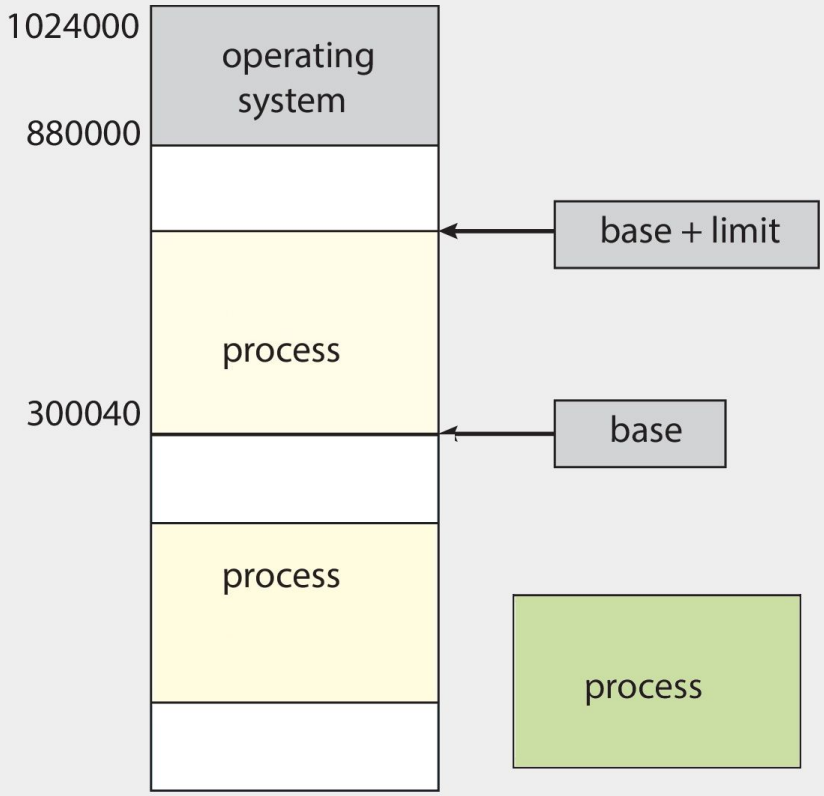


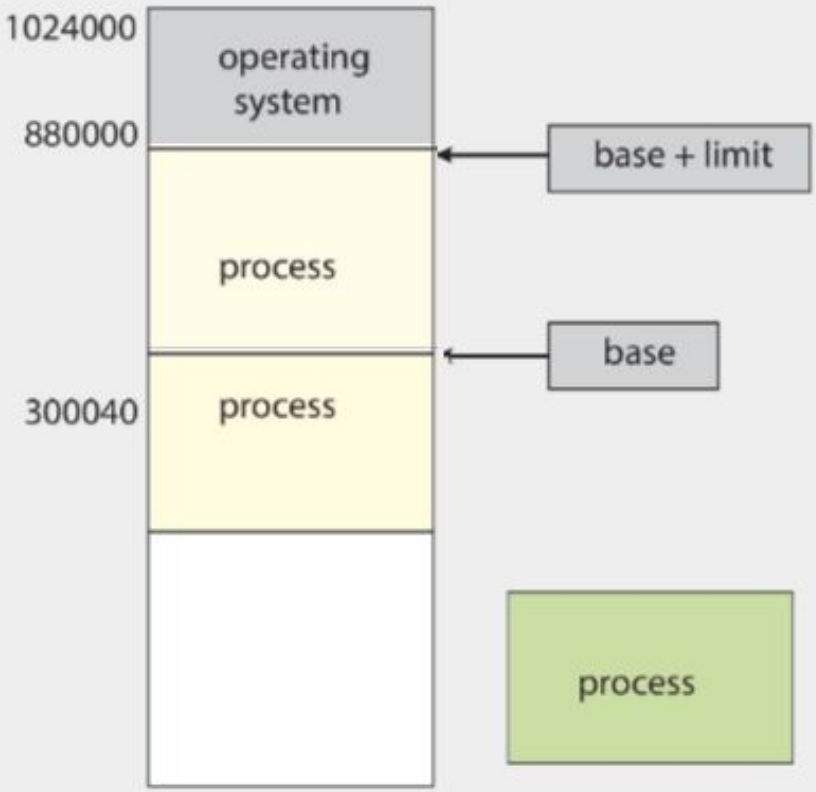
Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo

Inconveniente: Fragmentación externa

Solución: Compactación (insume mucho tiempo de CPU y memoria mover bloques de datos)

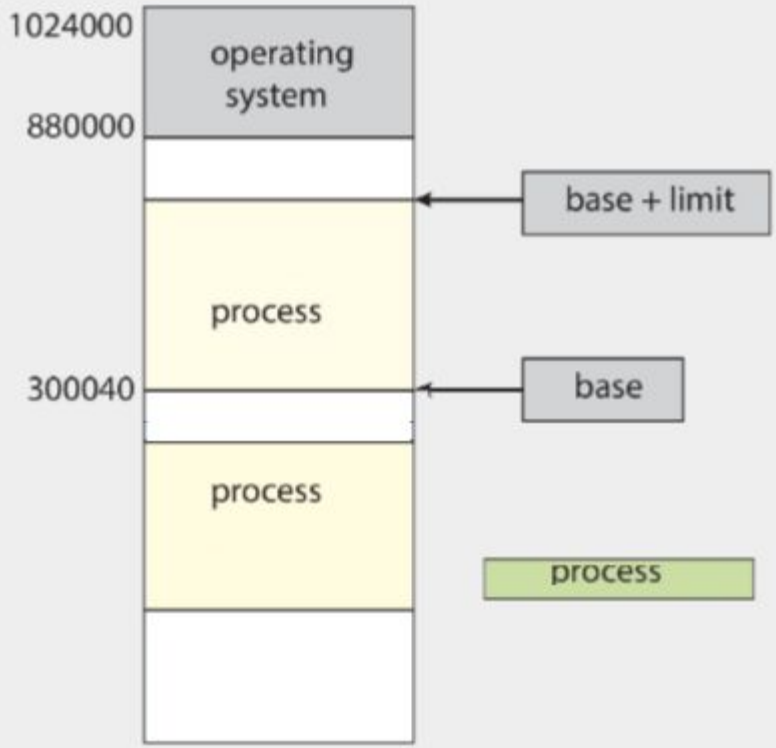




Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo.

Algoritmos de ubicación de bloque en espacio libre.



Sistemas Operativos I - Administración de memoria

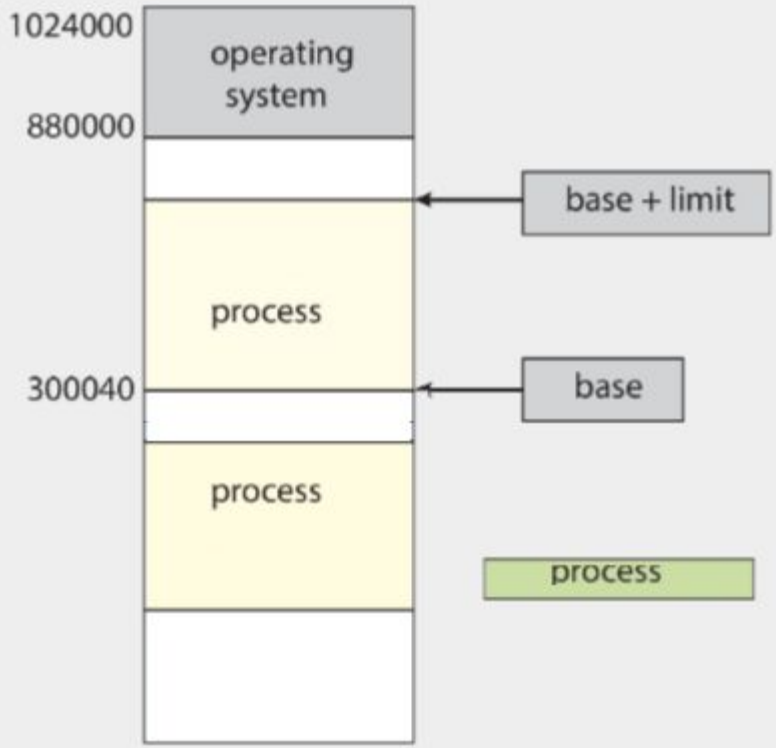
Asignación de espacio contiguo.

Algoritmos de ubicación de bloque en espacio libre.

Primer lugar libre.

Best fit.

Worst fit.



Sistemas Operativos I - Administración de memoria

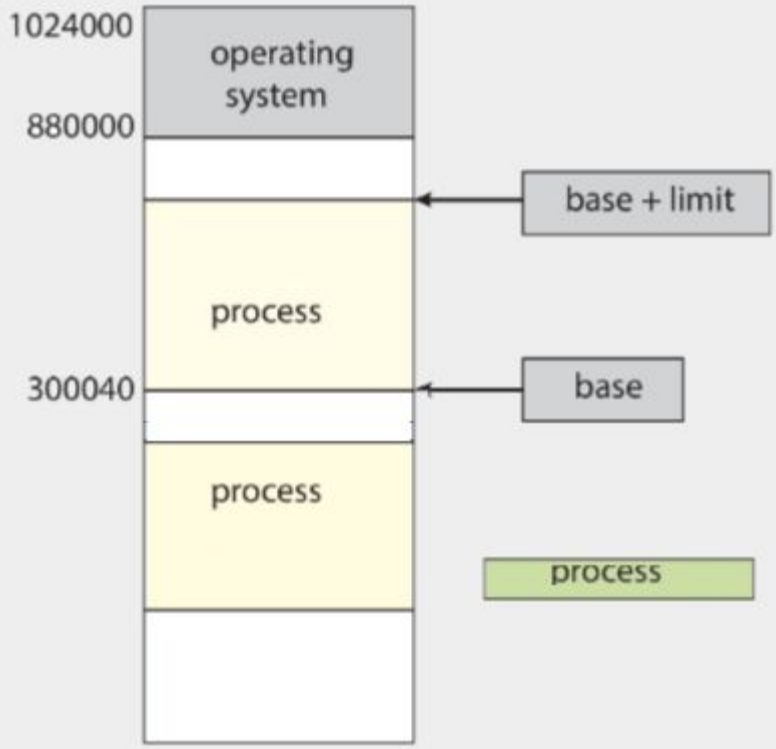
Asignación de espacio contiguo.

Algoritmos de ubicación de bloque en espacio libre.

Primer lugar libre.
Best fit.
Worst fit.

¿Cuál es mejor?

(si la lista de lugares libres no está ordenada)
(si la lista de lugares libres está ordenada)
(¿y si se tiene en cuenta la fragmentación?)



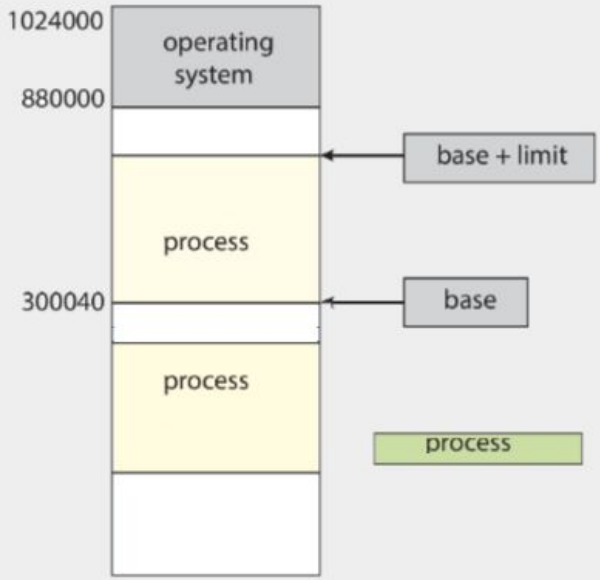
Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo.

Detalles de implementación para la gestión de bloques.

Primer lugar libre.
Best fit.
Worst fit.

¿Cómo mantener las lista de bloques libres y utilizados?



Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo.

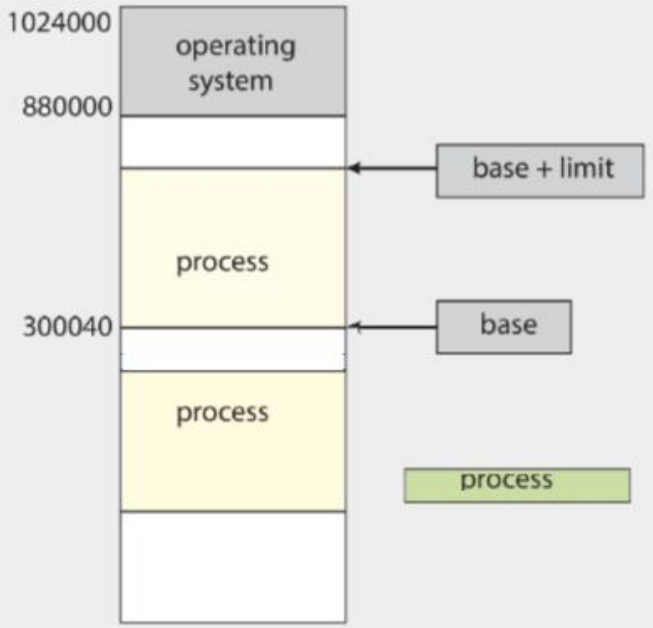
Detalles de implementación para la gestión de bloques.

Primer lugar libre.
Best fit.
Worst fit.

¿Cómo mantener las lista de bloques libres y utilizados?

Estructuras de datos del sistema operativo:

- lista enlazada o doblemente enlazada
- mapa de bits
    - ¿tamaño del mapa?



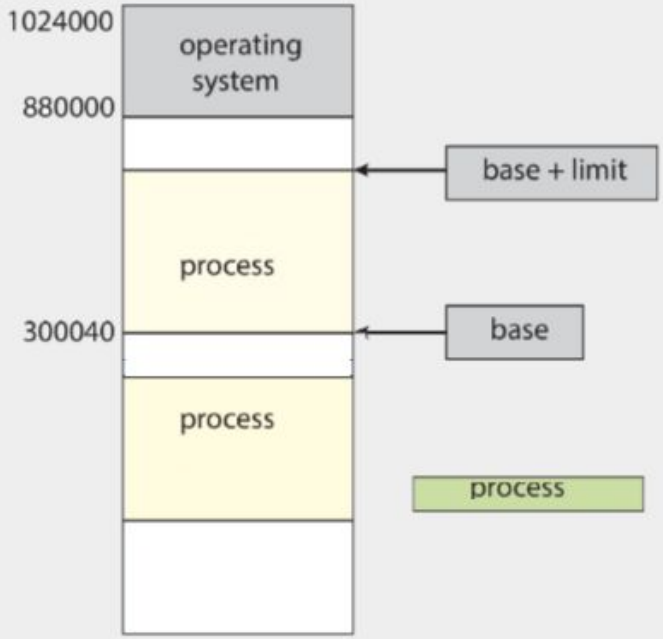
Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo.

Detalles de implementación para la gestión de bloques.

¿Cómo mantener las lista de bloques libres y utilizados?

Ejemplo: Estructura de lista en Xinu.



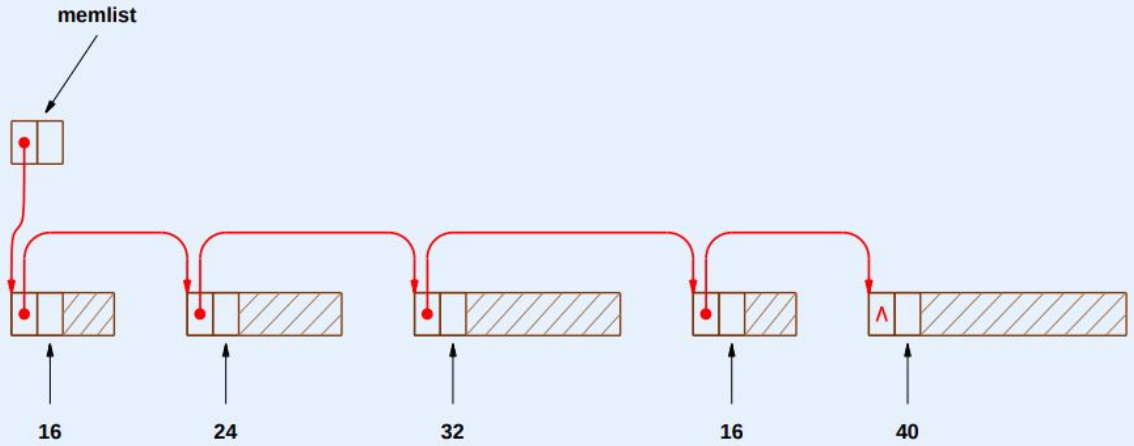
Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo.

Detalles de implementación para la gestión de bloques.

¿Cómo mantener las lista de bloques libres y utilizados?. Ejemplo: Estructura de lista en Xinu.

Illustration Of Xinu Free List



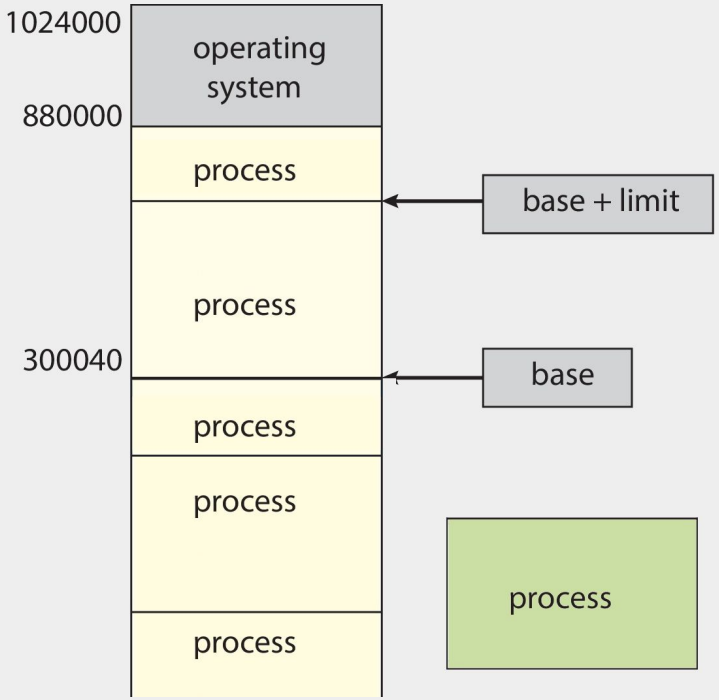
- Free memory blocks used to store list pointers
- Items on list ordered by increasing address
- All allocations rounded to size of structure *memblk*
- Length in *memlist* counts total free memory bytes

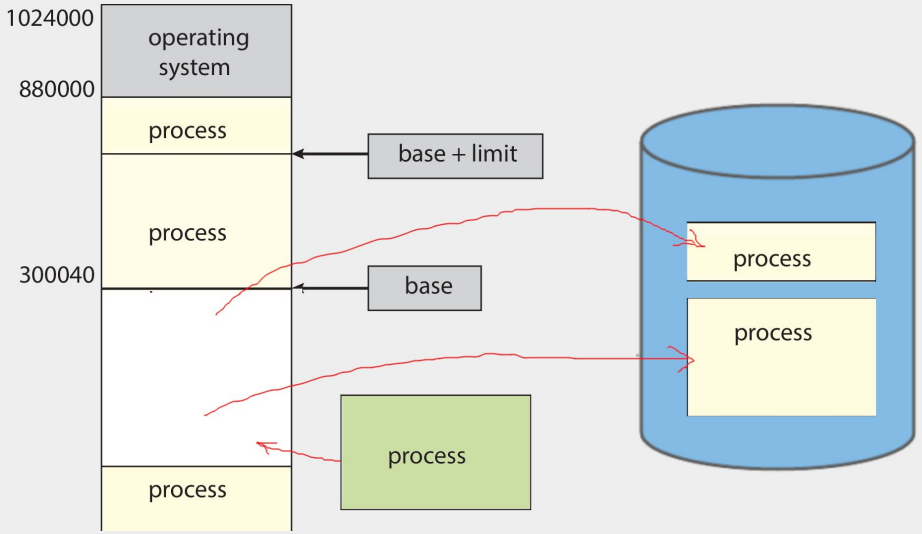
Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo.

¿Qué hacer si no hay memoria física disponible?

Swapping. (suspender uno o mas procesos)



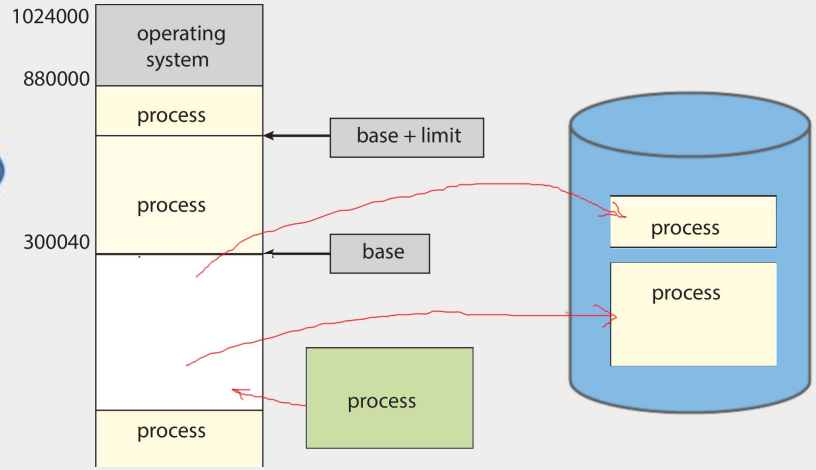


Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo.

¿Qué hacer si no hay memoria física disponible?

Swapping. (suspender uno o mas procesos)



Sistemas Operativos I - Administración de memoria

Asignación de espacio contiguo. Ejemplo: XINU

Figure 9.2 Illustration of the memory layout when Xinu begins.

Figure 9.3 Illustration of memory after three processes have been created.

9.7 Design Of The Low–level Memory Manager

A set of functions and associated data structures are used to manage free memory. The low-level memory manager provides five functions:

- *getstk* — Allocate stack space when a process is created
- *freestk* — Release a stack when a process terminates
- *getmem* — Allocate heap storage on demand
- *freemem* — Release heap storage as requested
- *meminit* — Initialize the free list at startup

Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

Terminología:

- paginación
- paginación por demanda
- dirección lógica (o virtual)
- dirección física
- páginas (pages)
- marcos (frames)
- Memoria del sistema (RAM - volátil)
- Memoria secundaria
- Fallo de página
- TLB / MMU
- Tabla de páginas
- Algoritmo de reemplazo
- Copy on write - Trashing

Sistemas Operativos I - Administración de memoria

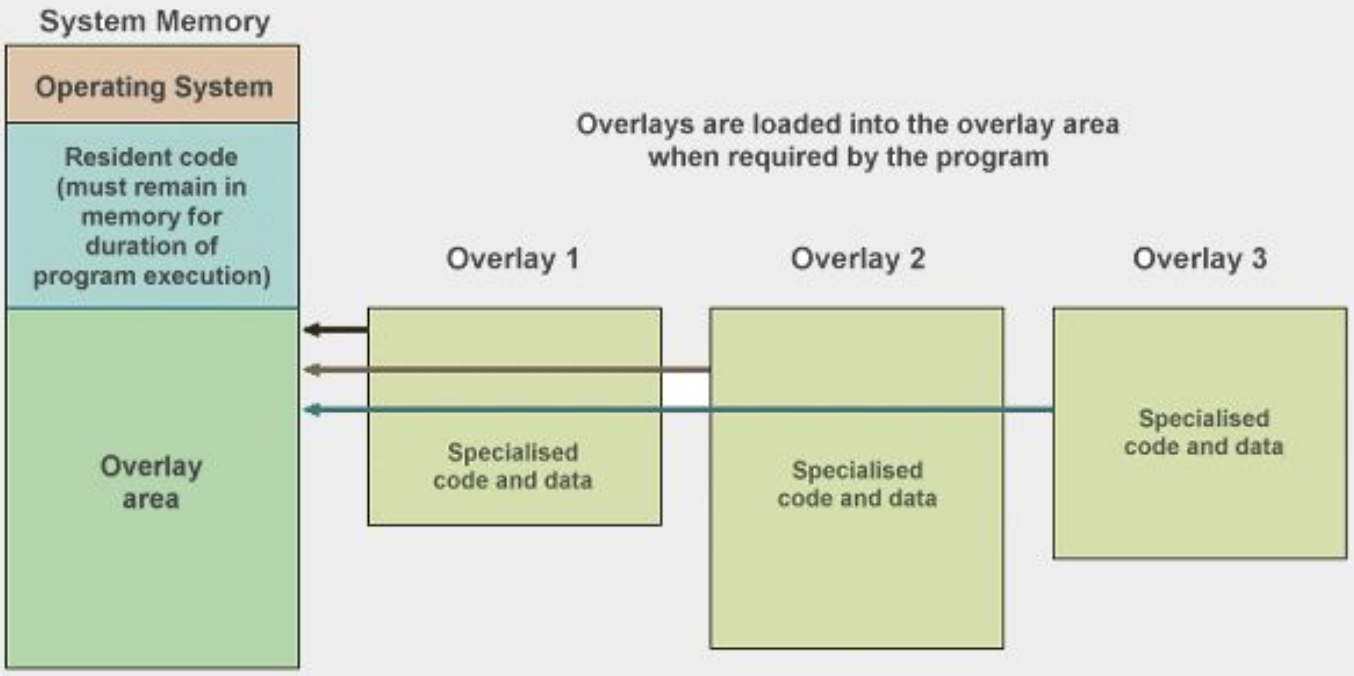
Memoria VIRTUAL

Motivación:

Aumentar la multiprogramación (mas procesos en memoria).

Mejorar o eliminar la fragmentación externa.

En el camino: OVERLAYs



Memoria VIRTUAL

Los sistemas operativos de uso diario virtualizan la memoria
y le presentan a cada aplicación una vista abstracta de la memoria.

Cada aplicación accede a un espacio de direcciones muy amplio,
que excede la poca memoria física real disponible.

El sistema operativo multiplexa la memoria física:
mueve los procesos, o partes de los mismos,
desde disco a la memoria física (y visceversa)
según las aplicaciones lo vayan necesitando.

Memoria VIRTUAL

Por lo tanto, las *motivaciones* fueron muchas:

- Aumentar la multiprogramación (más procesos en memoria).
- Mejorar o eliminar la fragmentación externa.
- Ejecutar programas que utilizan más memoria que la memoria física disponible.
- Proceso de compilación independiente del hw de memoria.
- Protección (espacios de memoria independientes).

Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

El sistema operativo debe ser capaz de mover procesos entre memoria física y disco.
Cada proceso debe utilizar un direccionamiento lógico individual.

Tres técnicas:

- swapping
- segmentación
- paginación

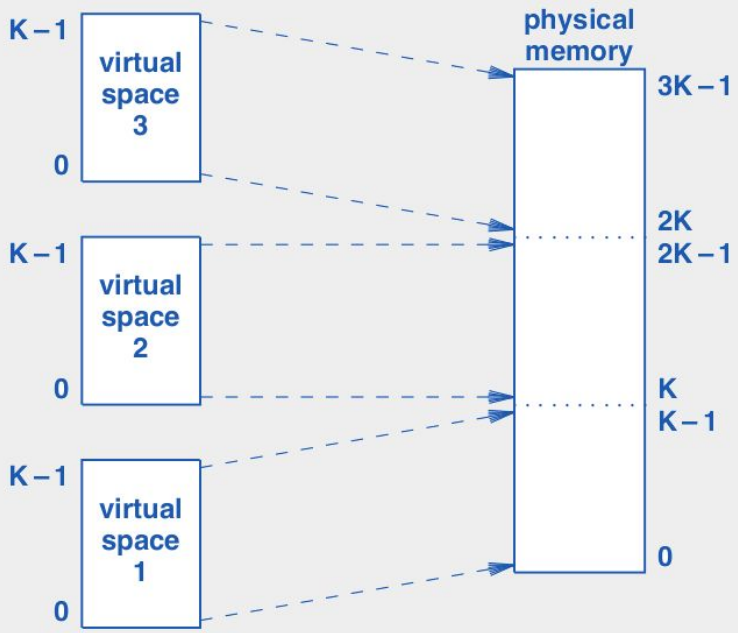
Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL: espacio de direcciones virtuales y reales

El administrador de memoria del sistema operativo debe lograr que cada proceso tenga un espacio de direcciones de memoria independiente (conjunto de k ubicaciones en memoria: numerados 0..k-1)

Utilizando soporte del hardware el OS
debe "mapear" cada espacio de direcciones virtuales
a un conjunto de ubicaciones de memoria física.

SI dos procesos acceden a sus propias direcciones cero
el SO mapeará ambas a diferentes ubicaciones reales.



Espacio de direcciones físicas (reales):
conjunto de direcciones del hardware RAM.

Espacio de direcciones virtuales (lógicas):
conjunto de direcciones disponibles para un proceso.

Memoria VIRTUAL: paginación por demanda

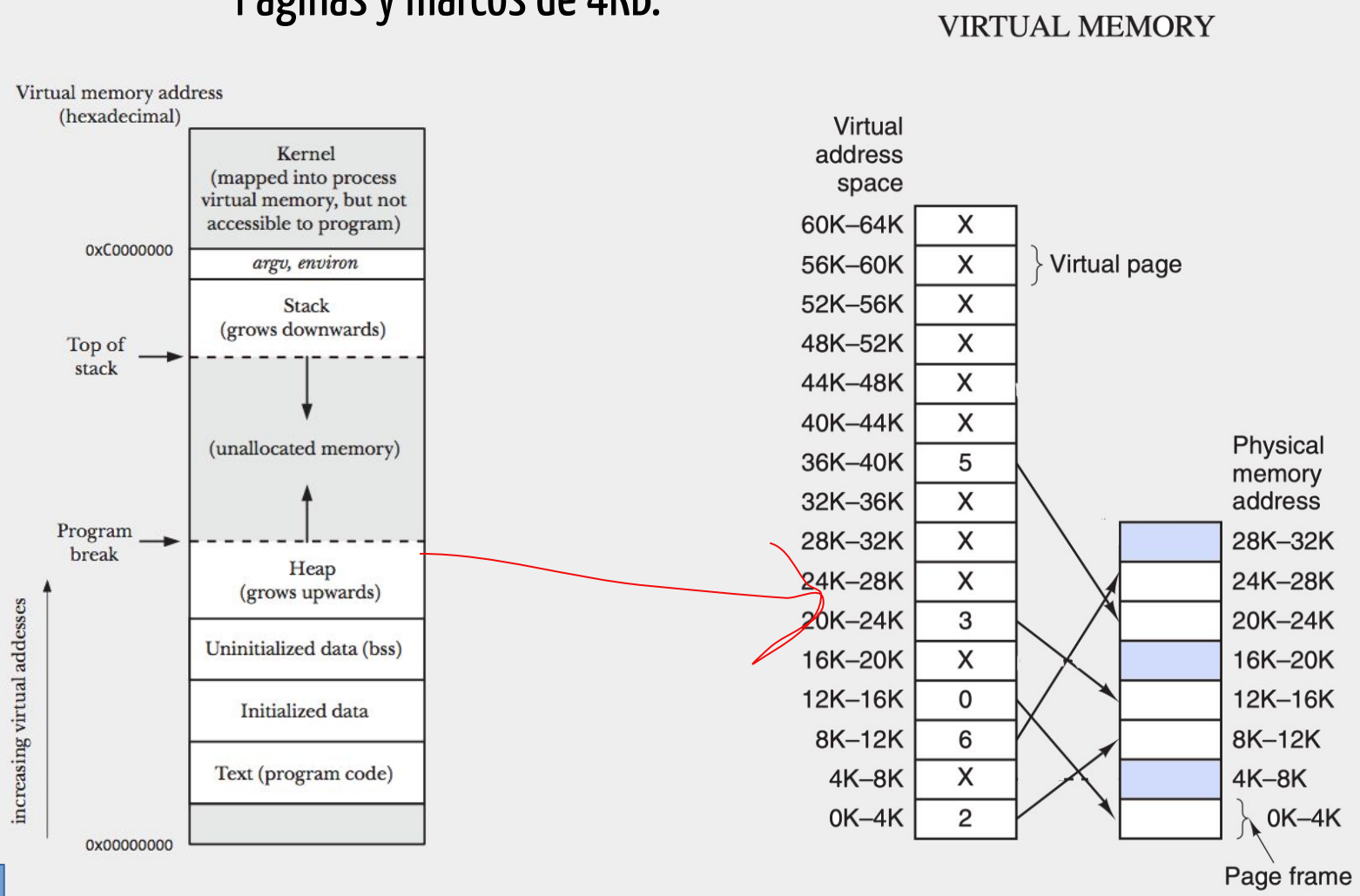
Paginación:

- La memoria física se divide en **marcos**
- Cada proceso se divide en **páginas**
- Las páginas y marcos son del mismo tamaño
- A cada página del proceso se le asigna un marco de memoria
- No existe fragmentación externa
- Existe poca **fragmentación interna**

Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL: paginación por demanda

Páginas y marcos de 4KB.



Sistemas Operativos I - Administración de memoria

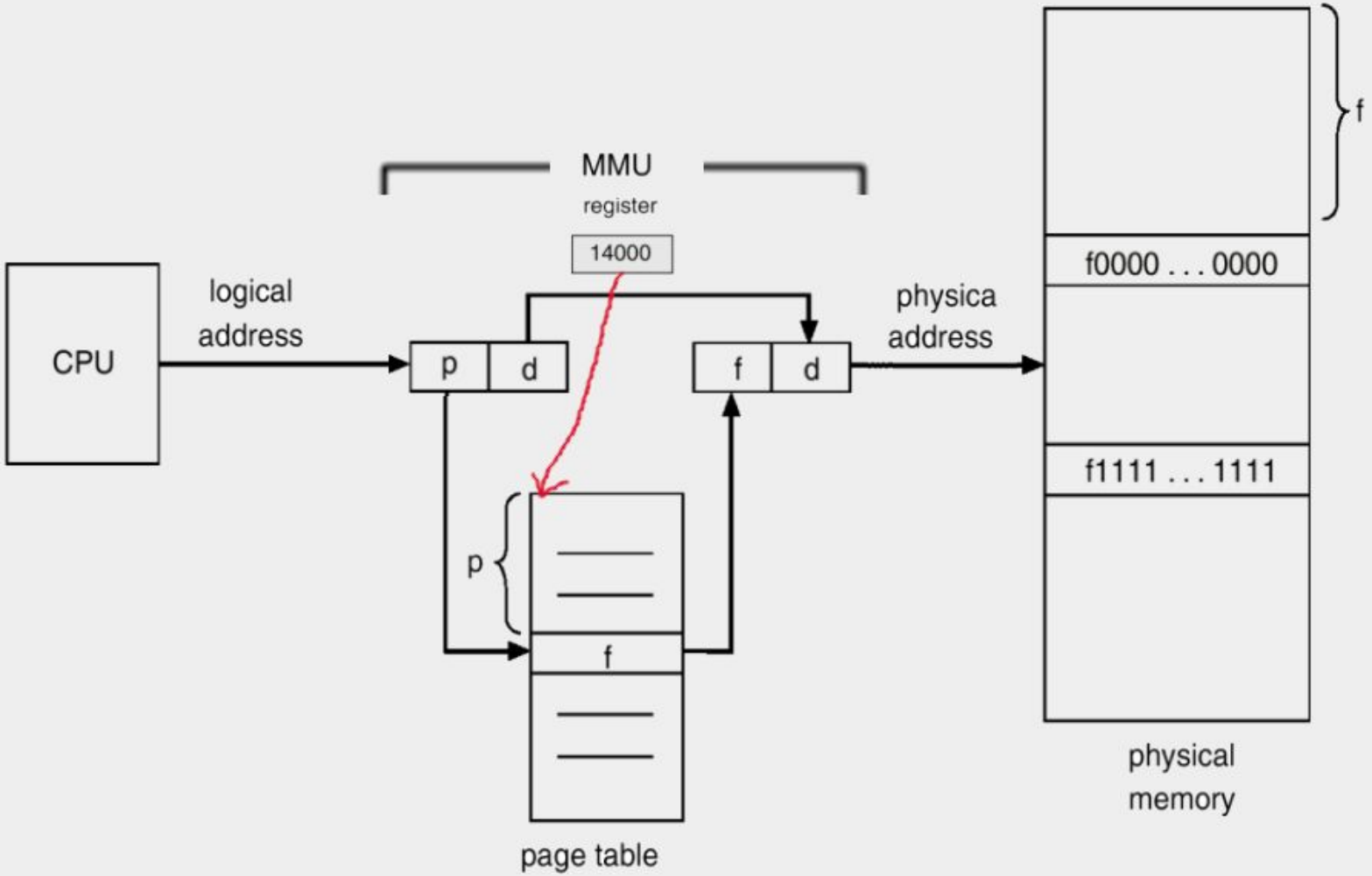
Memoria VIRTUAL: Hardware y Software para paginación por demanda

- Una unidad de traducción de direcciones (MMU)
- Una tabla de páginas por proceso
- Las tablas de páginas residen en espacio del kernel
- La MMU contiene al menos un registro que apunta a la tabla de páginas del proceso actual
  (¿Qué sucede en un cambio de contexto?)

Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL: Hardware y Software para paginación por demanda

Traducción de una dirección lógica a una dirección física

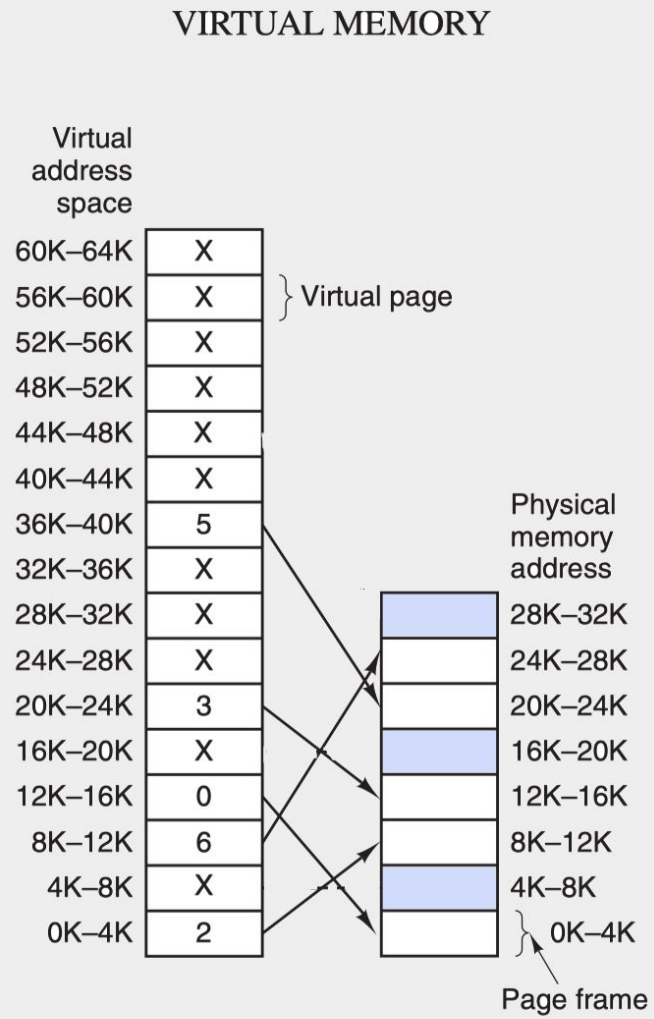


Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL: Hardware y Software para paginación por demanda

Tabla de páginas proceso p1

Marco nro.	Bit de validez	Bit Modificada	Bit de referencia
0010	1	1	1
	0	0	0
0110	1	0	1
0000	1	0	0
	0	0	0



Memoria VIRTUAL: Marcos compartidos

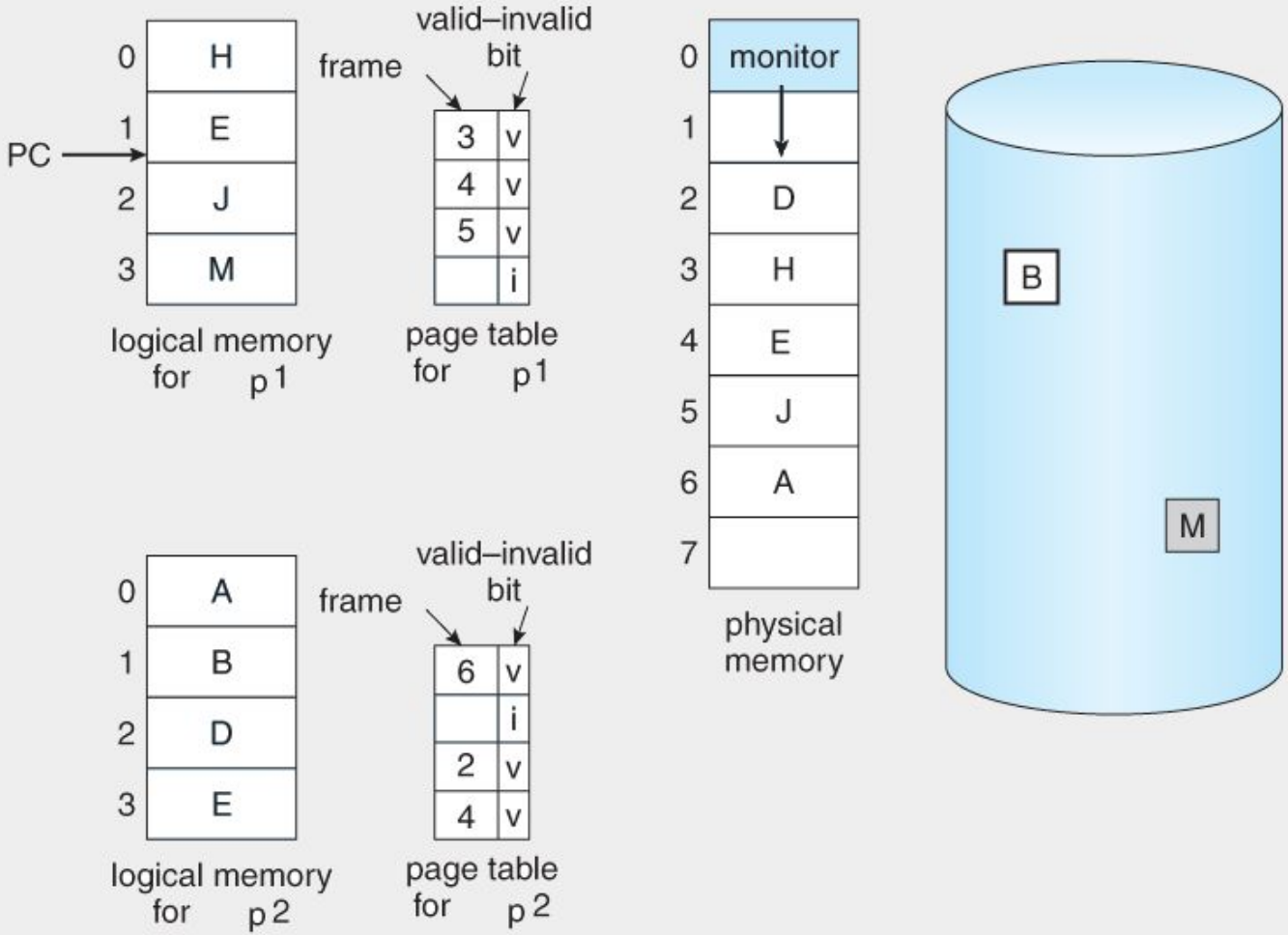
- Dos procesos en ejecución
- Cada uno con su propia tabla de páginas
- Cada uno con su propio direccionamiento virtual
- Pueden compartir páginas ¿Cuando?

Memoria VIRTUAL: Marcos compartidos

- Dos procesos en ejecución
- Cada uno con su propia tabla de páginas
- Cada uno con su propio direccionamiento virtual
- Pueden compartir páginas ¿Cuando?
    1. Código de Bibliotecas compartidas
    2. Memoria compartida (comunicación entre procesos)
    3. Proceso padre e hijo luego de un fork()

Sistemas Operativos I - Administración de memoria

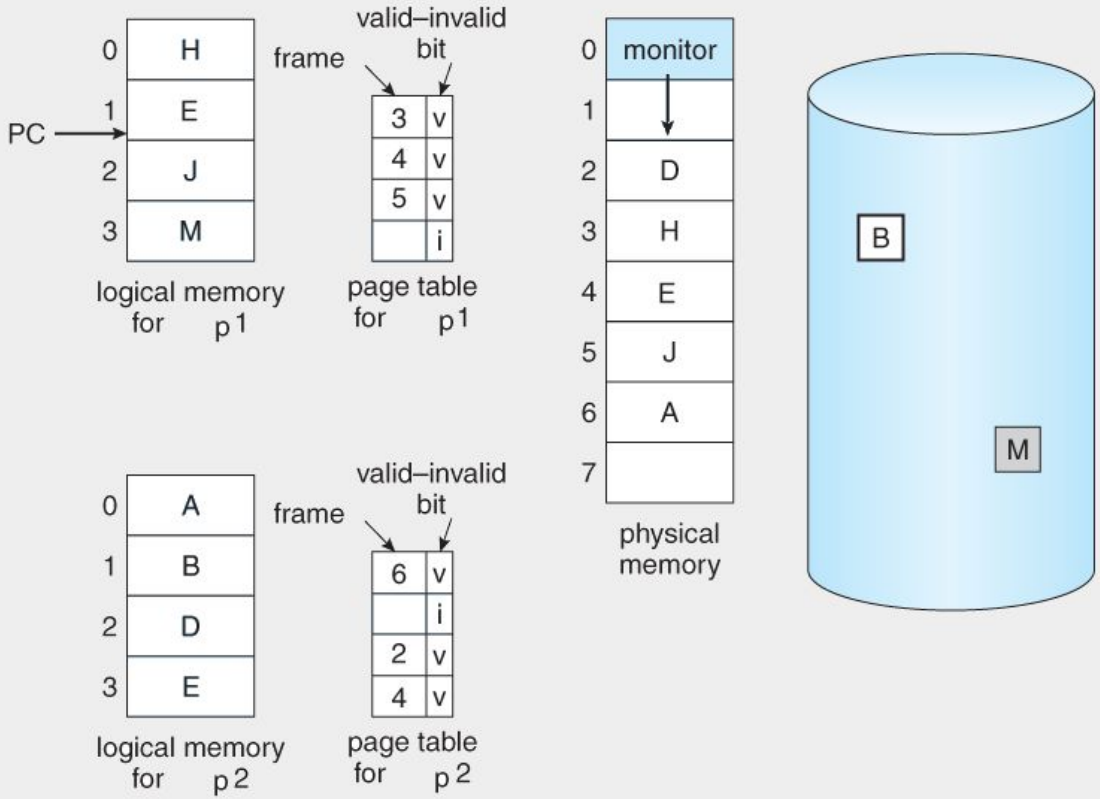
Memoria VIRTUAL:



Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

¿Qué sucede cuando una dirección de memoria virtual pertenece a una página sin marco de memoria asignado? (es decir, el marco de página no está en RAM).

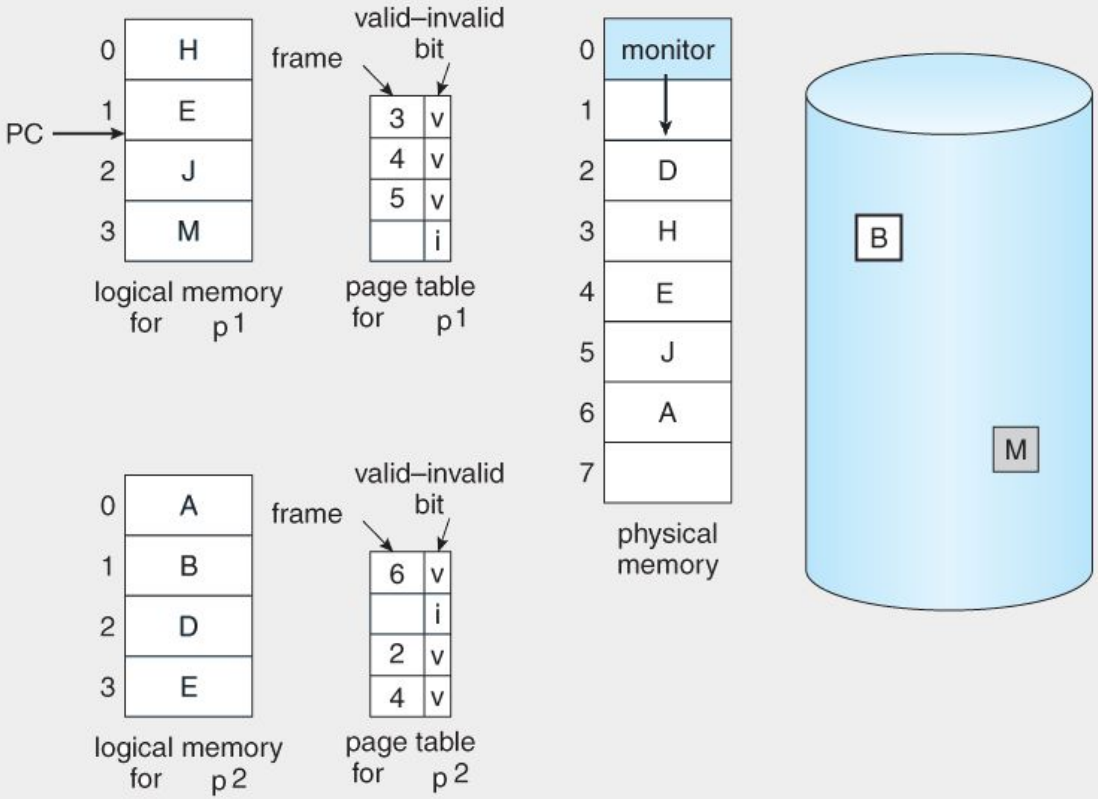


Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

¿Qué sucede cuando una dirección de memoria virtual pertenece a una página sin marco de memoria asignado? (es decir, el marco de página no está en RAM).

Se produce un FALLO DE PÁGINA



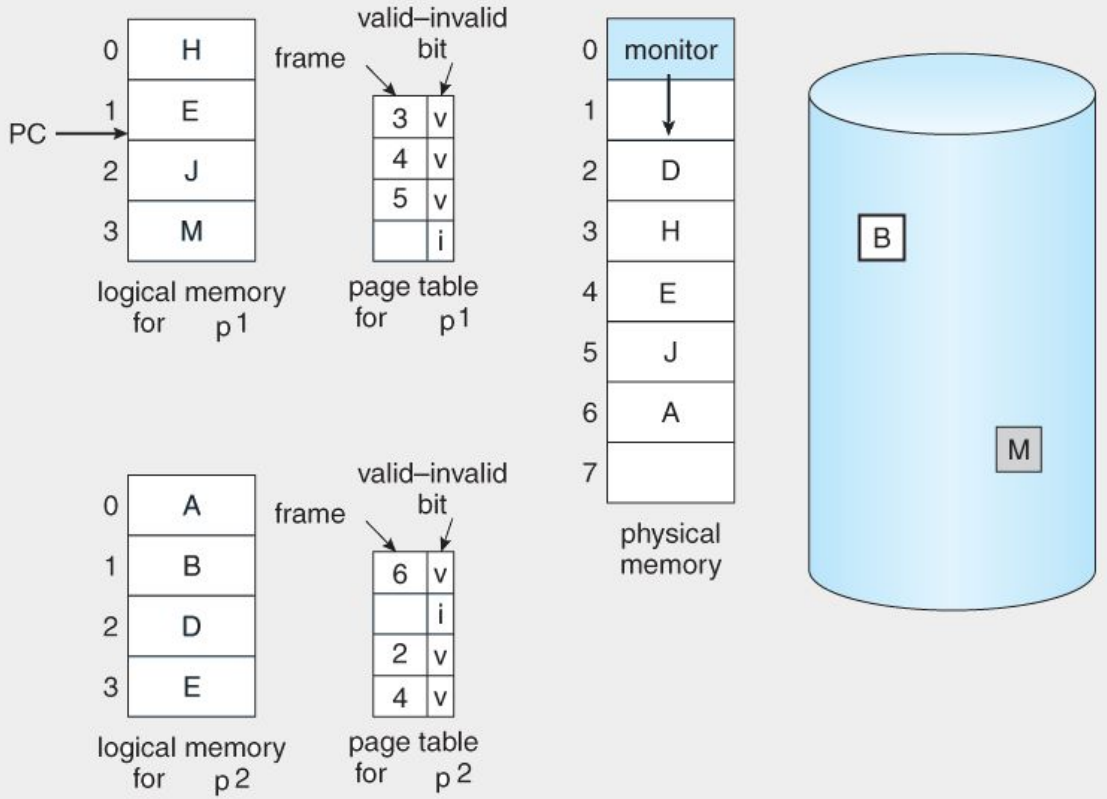
Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

¿Qué sucede cuando una dirección de memoria virtual pertenece a una página sin marco de memoria asignado? (es decir, el marco de página no está en RAM).

Se produce un FALLO DE PÁGINA

El OS podría suspender el proceso hasta que la página esté en memoria.



Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

¿Qué sucede cuando una dirección de memoria virtual pertenece a una página sin marco de memoria asignado? (es decir, el marco de página no está en RAM).

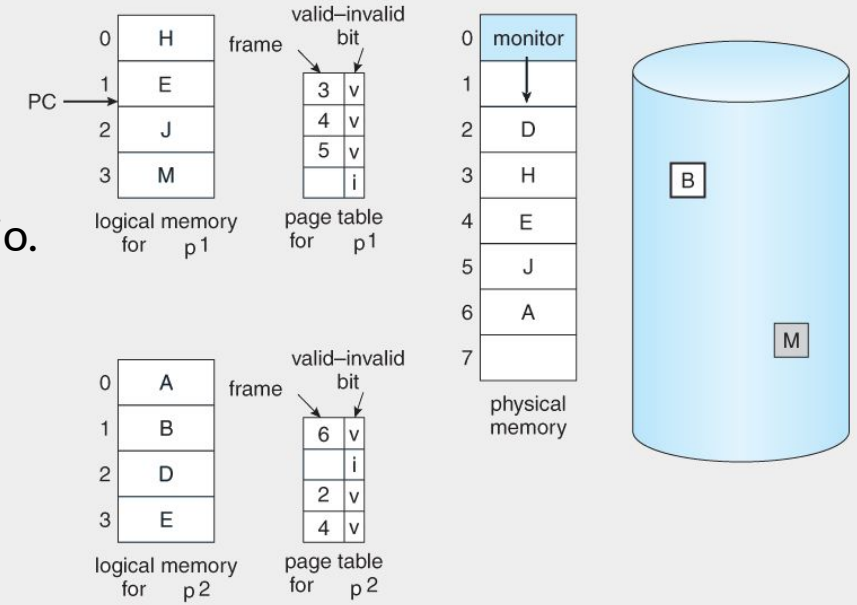
Se produce un FALLO DE PÁGINA

El SO determina la ubicación de la página referenciada en el almacenamiento secundario.

Se carga la página desde el disco a un marco vacío de la memoria.

Se establece el bit de validez en 1 en la TP.

Se reinicia la instrucción que produjo la falla de página.



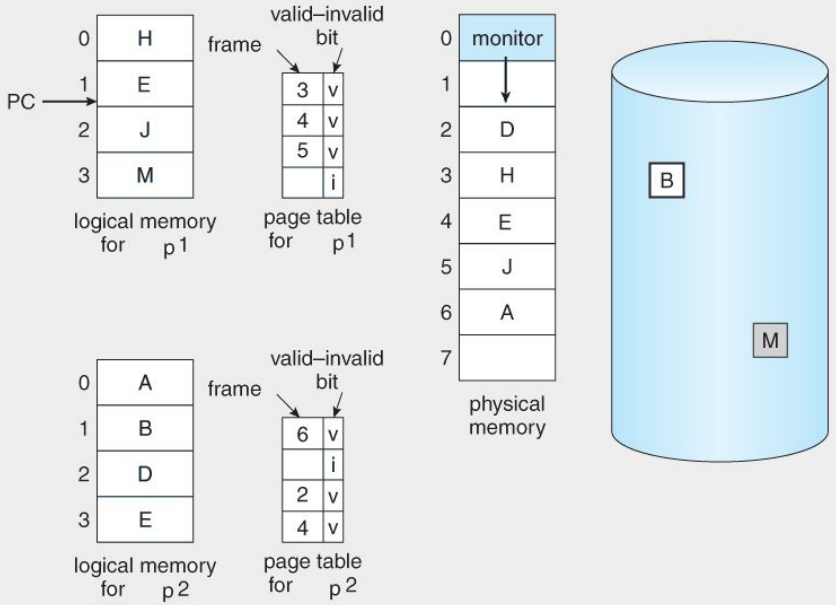
Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

¿Qué sucede cuando una dirección de memoria virtual pertenece a una página sin marco de memoria asignado? (es decir, el marco de página no está en RAM).

Se produce un FALLO DE PÁGINA

¿Y si no existen marcos libres en memoria?



Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

¿Qué sucede cuando una dirección de memoria virtual pertenece a una página sin marco de memoria asignado? (es decir, el marco de página no está en RAM).

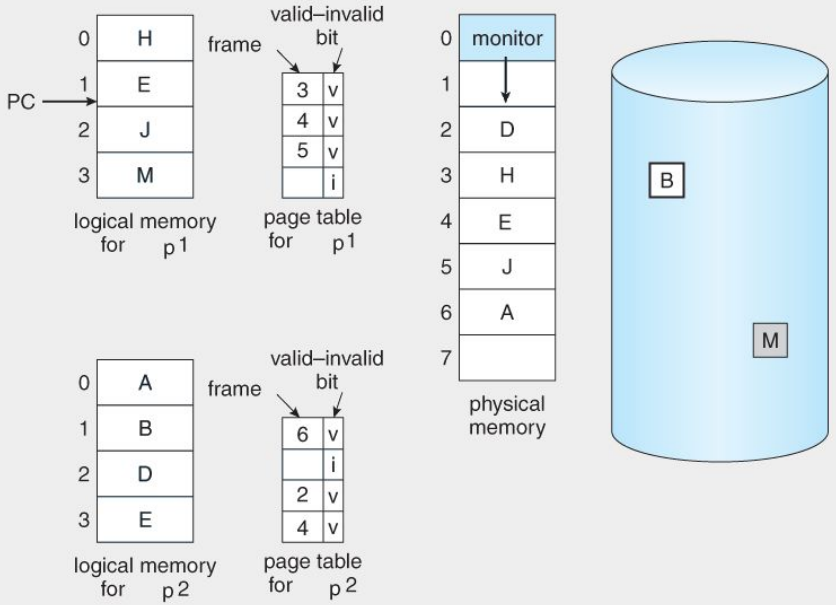
Se produce un FALLO DE PÁGINA

¿Y si no existen marcos libres en memoria?

Se seleeciona un marco para ser reemplazado.

Algoritmos de reemplazo:

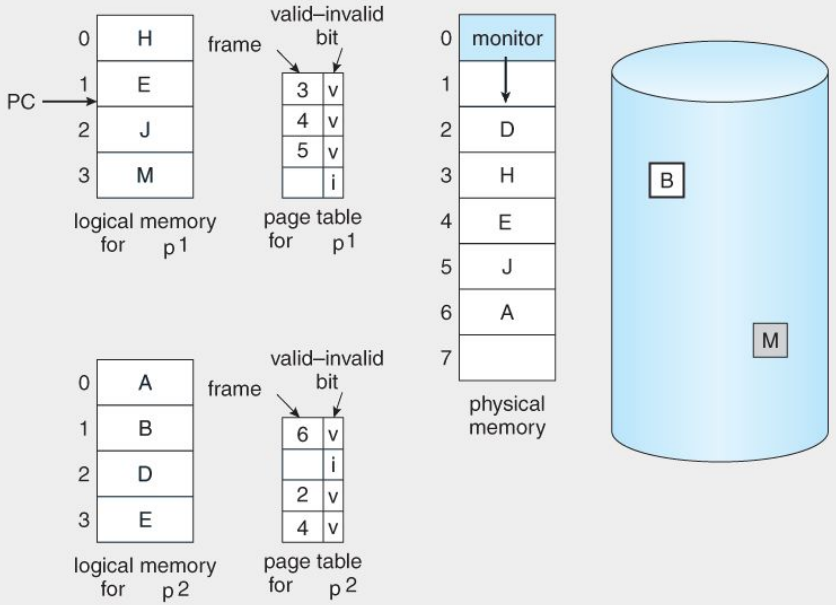
- Belady's optimal replacement algorithm
- LRU
- FIFO
- Global clock o segunda chance



Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

Cuando se ejecuta una instrucción máquina ¿Cuantos accesos a memoria son necesarios??



Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

Cuando se ejecuta una instrucción máquina ¿Cuantos accesos a memoria son necesarios??

1. Un acceso para obtener la instrucción
2. Posiblemente un acceso para acceder a un operando
3. Un acceso a la tabla de páginas para traducir la dirección virtual en 1.
4. Un acceso a la tabla de para traducir la dirección virtual de 2.

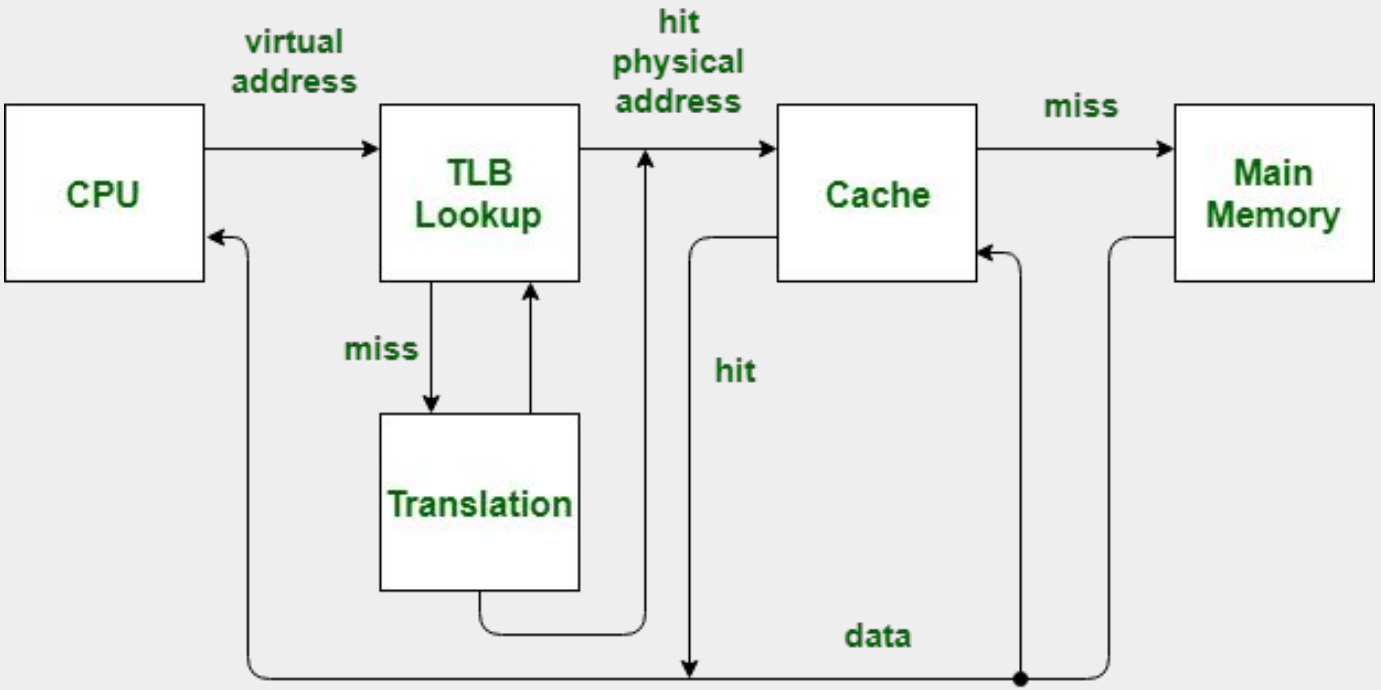
   ¿4 ACCESOS A RAM POR CADA INSTRUCCIÓN??

Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL

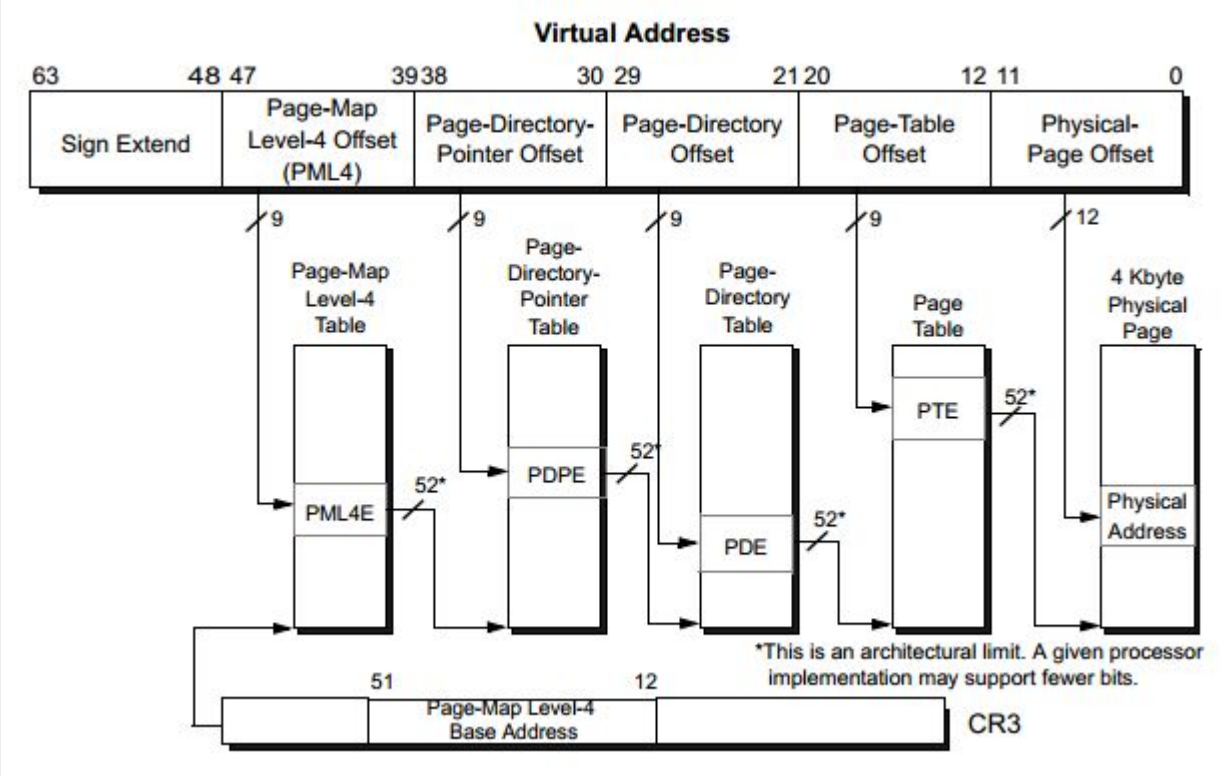
Cuando se ejecuta una instrucción máquina ¿Cuantos accesos a memoria son necesarios??

Mas hardware al rescate. TLB. Translate Lookaside Buffer



Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL: Ejemplo AMD64 (pc) linux



Sistemas Operativos I - Administración de memoria

Memoria VIRTUAL: *"te lo resumo así nomás"*

Un proceso se divide en "páginas".
Las páginas son de 4KB. O de 8KB.

La memoria física del sistema se divide en marcos (frames). Del mismo tamaño que las páginas.

La CPU ejecuta instrucciones que contienen direcciones lógicas (virtuales).
Las instrucciones ejecutadas por la CPU son producidas por el compilador. Ergo, el compilador genera todo un direccionamiento lógico (virtual).

Todas las direcciones lógicas tienen dos componentes : nro de página y desplazamiento

Cada proceso tiene su propio direccionamiento virtual. Es como si todo proceso viese toda la memoria virtual para su unico uso (como si utilizara la computadora de manera única).

Cada vez que la CPU ejecuta una instrucción se accede posiblemente 2 veces a la memoria. Una para "obtener la instrucción". Y otro acceso es para obtener un operando desde la memoria. Ambas direcciones son lógicas.

En esos dos accesos a memoria por instrucción, la CPU "convierte" cada dirección lógica en una dirección física.

Para convertir una dirección lógica en una dirección física, el sistema utiliza una TABLA DE PAGINAS. Existe una tabla de páginas por proceso. La tabla de páginas contiene por cada página, la dirección fisica de la misma. La tabla de pagina de cada proceso está en el segmento de memoria del KERNEL.

Algunos marcos de memoria física se comparten entre procesos. Ejemplo: si la biblioteca de C está en memoria, todos los programas que utilicen funciones de la biblioteca de C pueden usar esos marcos, así no se repiten marcos en memoria.
Otro ejemplo: cuando se comparte memoria (recuerden mmap y memoria compartida).
Otro ejemplo: cuando se realiza un fork() en UNIX, se ejecuta el system call clone(), el cual genera un nuevo proceso, pero ambos comparten las paginas de codigo y datos globales.

Sistemas Operativos I - Administración de memoria

Uso de memoria dinámica

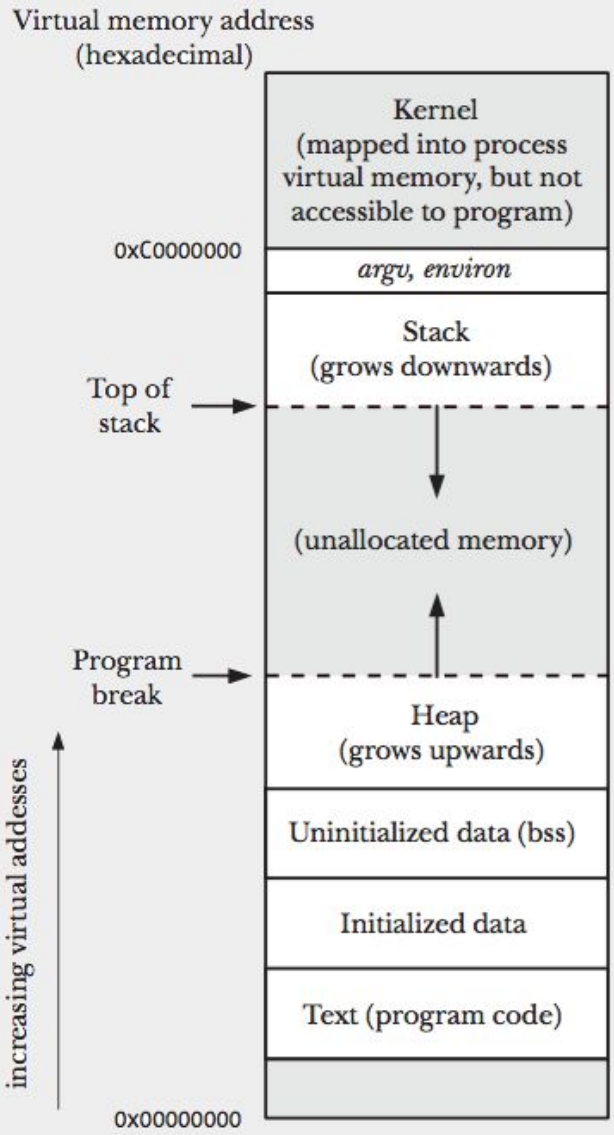
La biblioteca de C provee la función malloc para reservar memoria.

```
char * p;
p = malloc(1000); // solicita al OS 1000 bytes
```

Internamente, UNIX utiliza el system call break para reservar memoria (mas internamente Linux luego utiliza do_mmap())

```
*p = 'h';          // a lo apuntado por p le asignamos el valor 104
*(p+3) = 100;      // al cuarto byte de los mil le asignamos
                   // el valor 100
```

La función free libera la memoria reservada previamente



Sistemas Operativos I - Administración de memoria

Direcciones virtuales

En C es sencillo conocer las direcciones virtuales de las funciones y las variables.

```
void multiplicar()
{
      double j;

      printf("la dir. virtual de esta funcion es 0x%llX \n", multiplicar);
      printf("la dir. virtual de j es 0x%llX \n", &j);
}
```