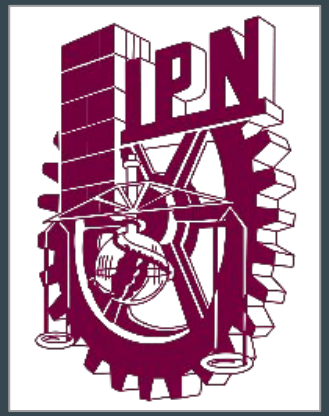


INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS OPERATIVOS

Unidad 1
Actividad 2

INTEGRANTES DEL EQUIPO:

COLIN RAMIRO JOEL
HERNÁNDEZ REYES JULIO CÉSAR
MALDONADO CERÓN CARLOS
MENDOZA GARCÍA ELIÚ EDUARDO

GRUPO: 4CM1

PROFESOR: CORTÉS GALICIA JORGE

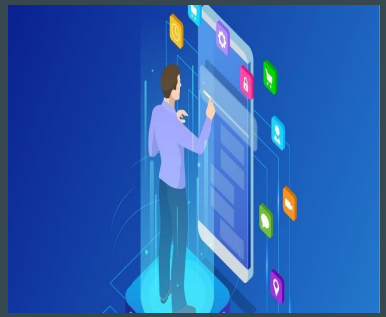
Estructura de un Sistema Operativo

Un S.O proporciona el entorno en el que se ejecuten los programas.

Podemos verlo desde varios puntos de vista. Concretamente 3:

- Servicios

- Interfaz Gráfica



- Componentes e Interconexiones

Servicios del Sistema Operativo

El SO presta ciertos servicios a los programas y a sus respectivos usuarios. Un cierto conjunto de servicios del SO proporciona funciones que resultan útiles al usuario, dentro de ellas destacan:

Interfaz de Usuario
CLI: Usa comandos de textos y algún método para introducirlos.

Ejecución de Programas
El sistema debe ser capaz de cargar un programa en memoria y ejecutarlo

Operaciones de E/S
Un programa en ejecución puede llevar a cabo operaciones de E/S dirigidas a un archivo o a un dispositivo de E/S

Manipulación de archivos
Los programas necesitan leer y escribir en archivos y directorios

Comunicaciones
Puede tener lugar entre procesos que se estén ejecutando en la misma o en otras computadoras

Detección de errores
Los errores pueden producirse en el hardware del procesador y de memoria en un dispositivo de E/S

Asignación de recursos
Cuando hay varios usuarios, o hay varios trabajos ejecutándose al mismo tiempo, debe asignarse a cada uno de ellos, los recursos necesarios

Seguridad
Los propietarios de la info almacenada en un sistema de computadoras en red o multiusuario necesitan a menudo poder controlar la info

Interfaz de Usuario del Sistema Operativo

Actualmente existen dos métodos fundamentales para que los usuarios interactúen con el S.O. :

- Intérprete de Comandos

  Consiste en proporcionar una interfaz de línea de comandos que permita a los usuarios introducir directamente comandos que el S.O pueda ejecutar.

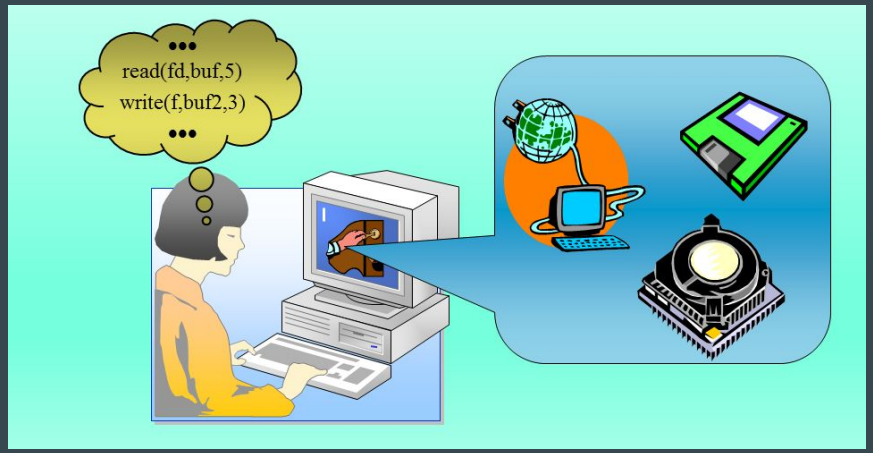
- Interfaz Gráfica de Usuario (GUI)

  Permite a los usuarios emplear un sistema de ventanas y menús controlables mediante el ratón.

Componentes e Interconexiones del Sistema Operativo

Llamadas al Sistema

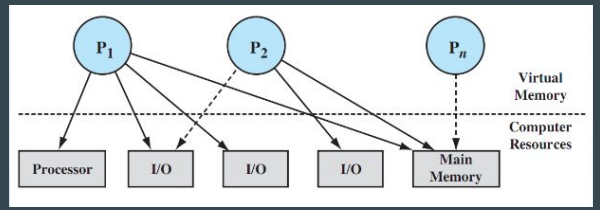
- ★ Proporcionan una interfaz con la que poder invocar los servicios que el S.O ofrece.
- ★ Permite al S.O exponer ciertas funcionalidades clave a los programas de usuario tales como:
  - Acceso al sistema de archivos
  - Creación y finalización de procesos
  - Comunicación con otros procesos
  - Asignación de memoria



Tipos de llamadas al sistema

Control de procesos

- terminar, abortar
- cargar, ejecutar
- crear y terminar procesos
- asignar y liberar memoria

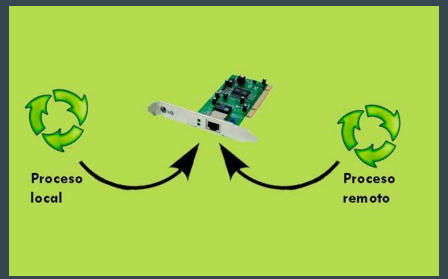


Administración de archivos

- Crear y borrar archivos
- abrir, cerrar
- leer, escribir, reposicionar

Comunicaciones

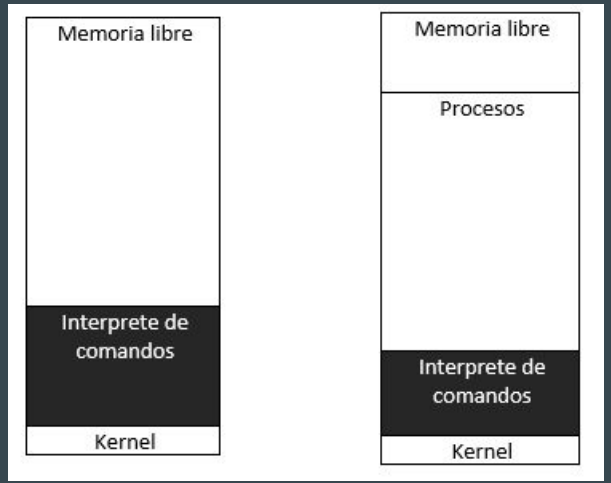
- crear, eliminar conexiones de comunicación
- enviar, recibir mensajes
- transferir información de estado
- conectar y desconectar dispositivos remotos



Si creamos un nuevo trabajo o proceso, o incluso un conjunto de trabajos o procesos, también debemos controlar su ejecución.

Una vez creados nuevos trabajos o procesos, es posible que tengamos que esperar a que terminen de ejecutarse.

Muchos sistemas operativos proporcionan un perfil de tiempo de los programas para indicar la cantidad de tiempo que el programa invierte en una determinada instrucción o conjunto de instrucciones.



Ejecución de un programa MS-DOS “Al inicio del sistema/Ejecución de un programa”

2.4.2 Administrador de archivos

FreeBSD (derivado de Berkeley UNIX) es un sistema de multitarea. cuando un usuario inicia sesión en el sistema, se ejecuta el shell elegida por el usuario.

2.4.3 Administración de dispositivos

Un proceso puede necesitar varios recursos para ejecutarse: memoria principal, unidades de disco, acceso de archivos, etc. Si los recursos están disponibles, pueden ser concebidos y el control puede devolverse al proceso de usuario. En caso contrario, el proceso tendrá que esperar hasta que haya suficientes recursos disponibles.

2.4.4 Mantenimiento de información

Muchas llamadas al sistema existen simplemente con el propósito de transferir información entre el programa de usuario y el sistema operativo. Otras llamadas pueden devolver información sobre el sistema.

2.4.5 Comunicaciones

Modelo de paso de mensajes: los procesos que se comunican intercambiando mensajes entre sí para transferir información.

Modelo de memoria compartida: los procesos usan las llamadas al sistema shared memory créate y shared memory attach para crear y obtener acceso a regiones de la memoria que son propiedad de otros procesos

2.5 Programas del sistema

Los programas del sistema proporcionan un entorno cómodo para desarrollar y ejecutar programas. Algunos de ellos son:

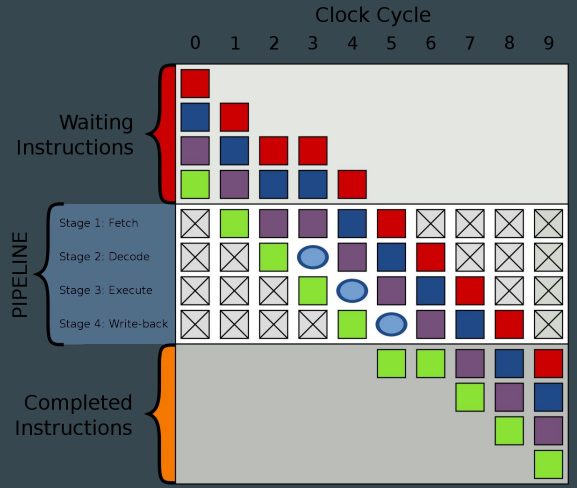
1. **Administración de archivos:** estos programas crean, borran, copian, cambian de nombre, imprimen, vuelcan, listan y de forma general manipulan archivos y directorios.
2. **Información de estado:** Algunos programas simplemente solicitan al sistema la fecha, la hora, la cantidad de memoria o espacio de disco disponible, el número de usuarios o información de estado similar.
3. **Modificación de archivos:** Puede disponerse de varios editores de texto para crear y modificar el contenido de los archivos almacenados en el disco 0 en otros dispositivos de almacenamiento.
4. **Soporte de lenguajes de programación:** Con frecuencia, con el sistema operativo se proporcionan al usuario compiladores, ensambladores, depuradores e intérpretes para los lenguajes de programación habituales, como, por ejemplo, C, C++, Java, Visual Basic y PERL.
5. **Carga y ejecución de programas:** Una vez que el programa se ha ensamblado o compilado, debe cargarse en memoria para poder ejecutarlo. El sistema puede proporcionar cargadores absolutos, cargadores reubicables, editores de montaje y cargadores de sustitución.
6. **Comunicaciones:** Estos programas proporcionan los mecanismos para crear conexiones virtuales entre procesos, usuarios y computadoras.

2.1 Pipelining

Un trabajo debe pasar por todas las etapas, y el orden es el mismo para todos los trabajos. Incluso si un plato no necesita ser raspado, debe pasar por esa etapa (uno de sus amigos estará inactivo durante ese tiempo).

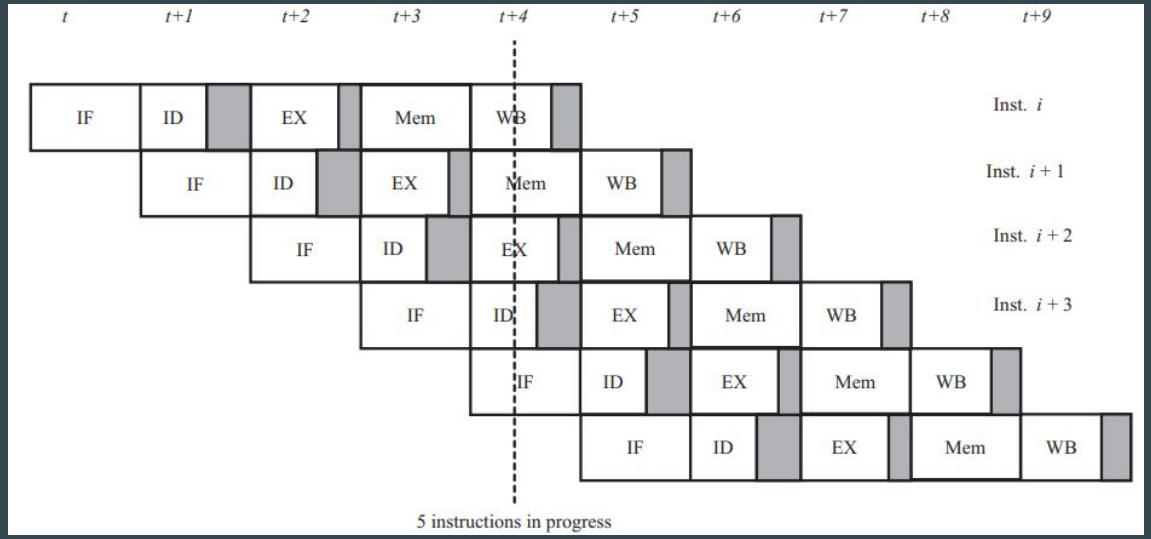
Es necesario hacer un buffering (mantener un plato en su estado actual parcialmente procesado) entre etapas, porque no todas las etapas llevan el mismo tiempo.

Cada etapa debe tener asignados todos los recursos que necesita. Por ejemplo, si sólo se dispone de un cepillo y éste se necesita una parte del tiempo, tanto por el rascador como por el lavador, se producirá cierto estancamiento en la cadena. Esta situación es una forma de peligro (estructural).



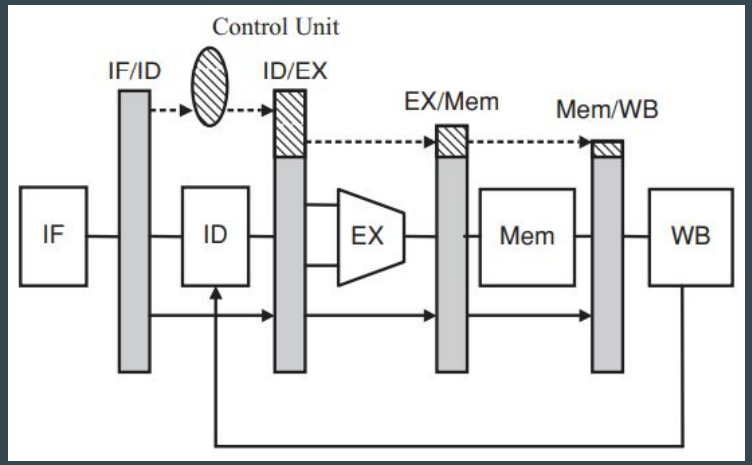
Cada paso se ejecuta en una etapa diferente, a saber:

1. **Búsqueda de la instrucción (IF).** La instrucción se obtiene de la memoria (I-cache) en la dirección la dirección indicada por el PC.
2. **Decodificación de la instrucción (ID).** Se decodifica la instrucción y se reconoce su tipo. Algunas otras tareas, como la extensión de las constantes inmediatas a 32 bits, se realizan otras tareas, como la extensión de las constantes inmediatas a 32 bits.
3. **Ejecución (EX).** En el caso de una instrucción aritmética, una ALU realiza la operación aritmética o lógica.
4. **Acceso a la memoria (Mem).** Si la instrucción es de almacenamiento, se modifica el contenido de esa se modifica el contenido de esa ubicación.
5. **Writeback (WB).** Si la instrucción no es ni una rama ni una tienda, el resultado de la operación (o de la carga) se almacena en el registro de resultados.



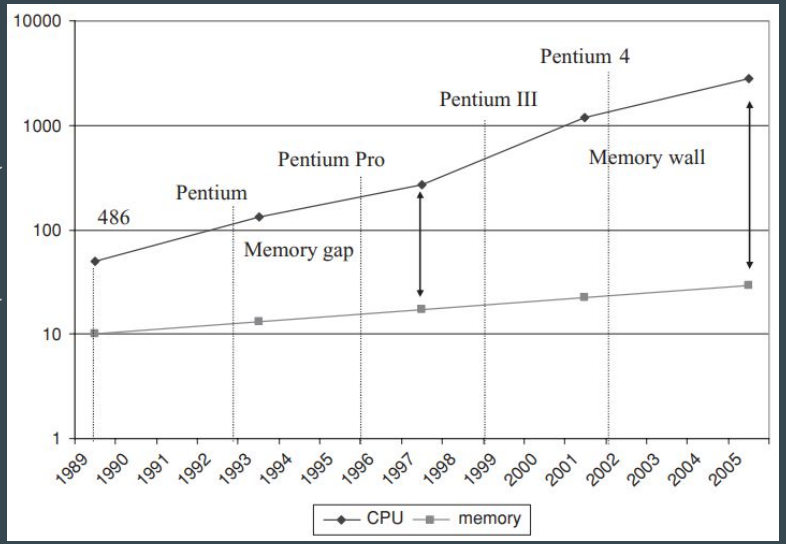
Programa secuencial en ejecución

Vista del flujo con la unidad de control



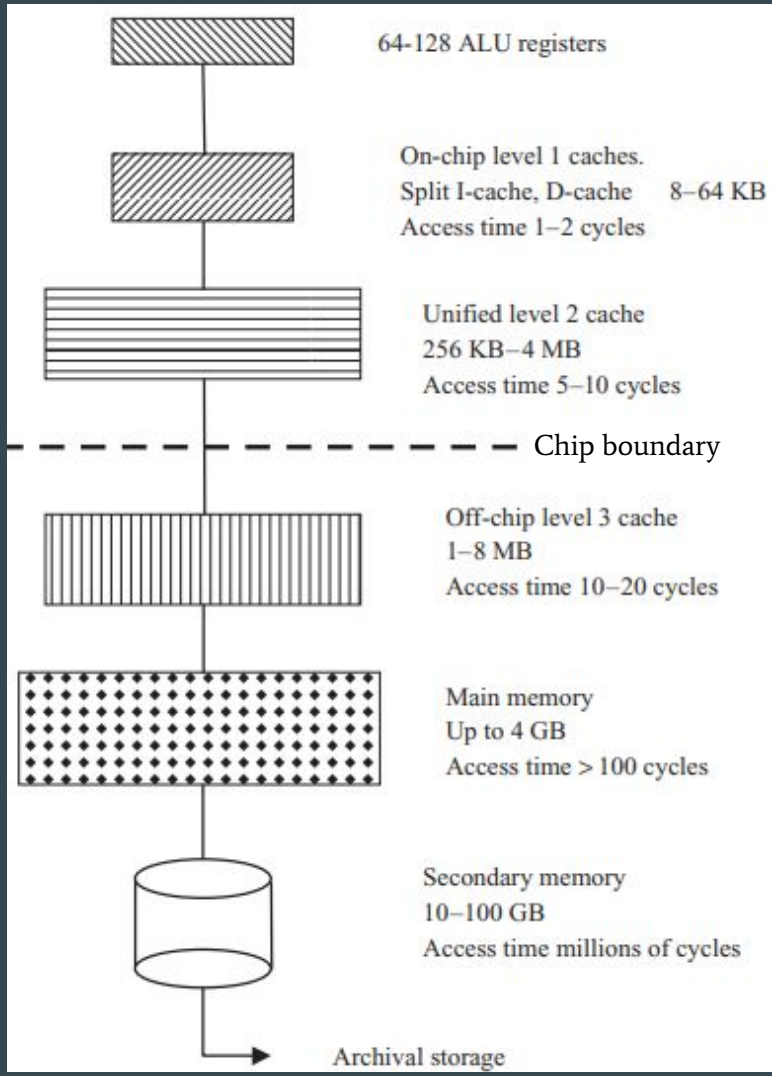
CACHÉ

Por lo años 60, se reconoció la gran brecha entre velocidades de los procesadores y la latencia de las memorias, por lo que era necesario alguna memoria “buffer” entre ambos, lo que dió paso a la creación de las memorias caché. Antes, un acceso a la memoria tomaba 100 ciclos del procesador, idealmente, la información debía ser obtenida en un solo ciclo, por lo que la caché debía ser un chip con la ALU. Sin embargo, su capacidad no podría ser tan grande, ya que el tiempo de acceso es proporcional a su tamaño, razón por la cual se tienen múltiples niveles de caché.

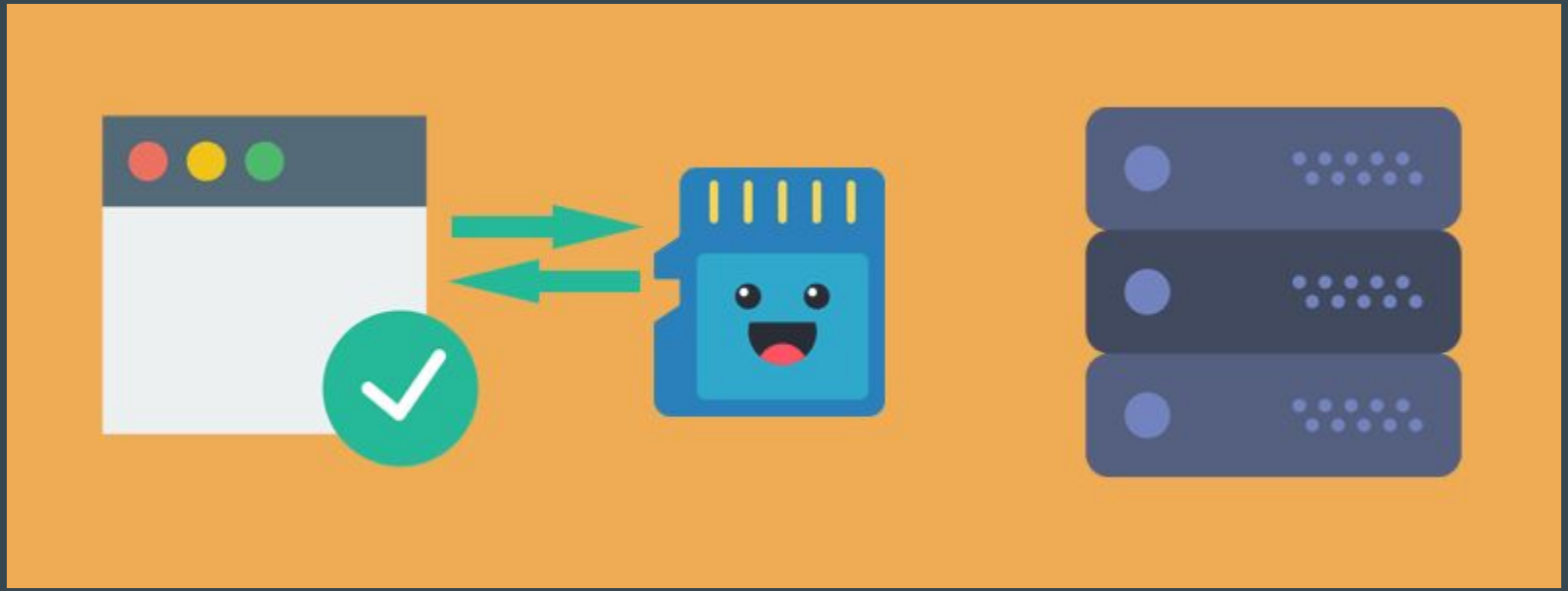


Brecha en el rendimiento entre procesador y memoria (escala logarítmica)

A la derecha se muestra una jerarquía con dos niveles de caché en chip (SRAM) y un nivel de caché no en chip (SRAM & DRAM) y la memoria principal (DRAM). El concepto de “caché” ahora se usa en una gran variedad de sistemas computacionales, discos, servidores de red, cachés en web, entre otros. Usar caché funciona gracias al “principio de localidad”. Información pasada que es probable que se reuse en un futuro cercano se guarda en la “localidad temporal”, e información relacionada con el proceso que se ejecuta en el momento está en la “localidad espacial”. La caché es mucho más pequeña que la memoria principal, por lo que en ocasiones, no puede almacenar todos los datos del programa en ejecución.



Si una localidad de memoria está mapeada en la caché, resultará en un “cache hit”, de lo contrario, se obtiene un “cache miss”



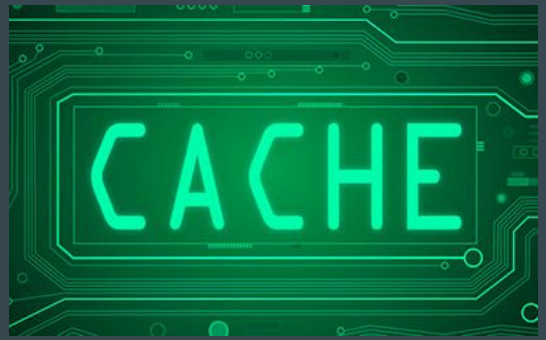
Organización de Cachés - Las 4 preguntas básicas

Las 4 preguntas básicas en el diseño de cachés:

1. ¿Cuándo mover el contenido de una localidad de memoria a la caché?
2. ¿Dónde colocarla?
3. ¿Cómo sabemos que está ahí?
4. ¿Qué pasa si la caché está llena y queremos poner información en ella? Una mejor formulación sería: ¿Qué pasa si queremos mover el contenido de una localidad de memoria que no está en caché y el lugar donde debería ir ya está ocupado?

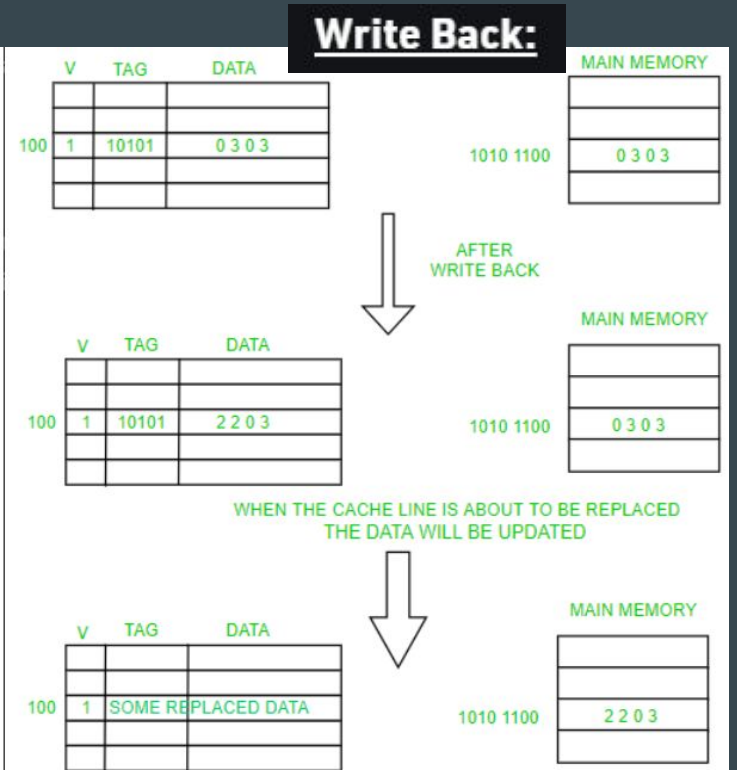
Respuestas a las 4 preguntas básicas del diseño de cachés

1. Los contenidos de una localidad de memoria son movidos a la caché cuando se pide, es decir, cuando la solicitud de la información resulta en un “cache miss”
2. Básicamente, la caché está dividida en cierto número de entradas de caché. El mapeo de una localidad de memoria a una entrada de caché específica depende en la organización de la caché
3. Cada entrada de caché contiene su nombre o tag, además del contenido. Ya sea que una referencia a memoria resulte en un “hit” o “miss”, resulta en una revisión del tag.
4. En caso de un “cache miss”, si la nueva entrada causa conflicto con una que ya está ahí, una entrada será reemplazada por aquella que ocasionó el “miss”. La decisión se deja a un algoritmo de reemplazo.



Estrategias de Escritura

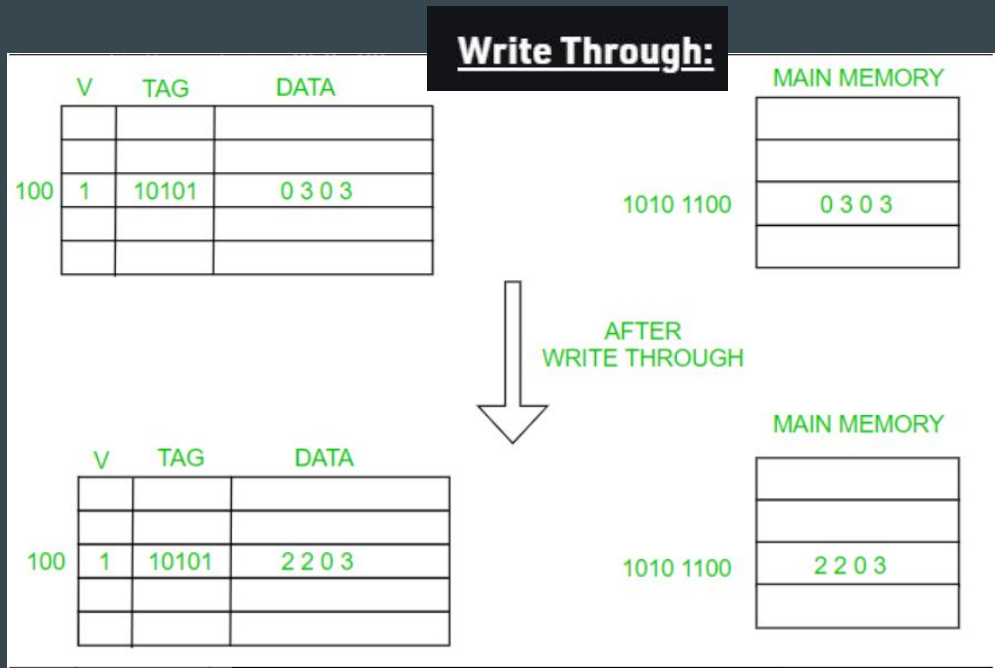
Cuando las referencias de memoria son para escribir(Consecuencia de una instrucción almacenada), nos enfrentamos con diferentes opciones:



En el caso de un Cache Hit:

1.- Write Back=> Escribir solo en la caché (Genera menos tráfico de memoria).

2.-Write-Through=> Escribir en ambas, en la caché y en el siguiente nivel de la jerarquía (Ofrece una vista consistente de la memoria).



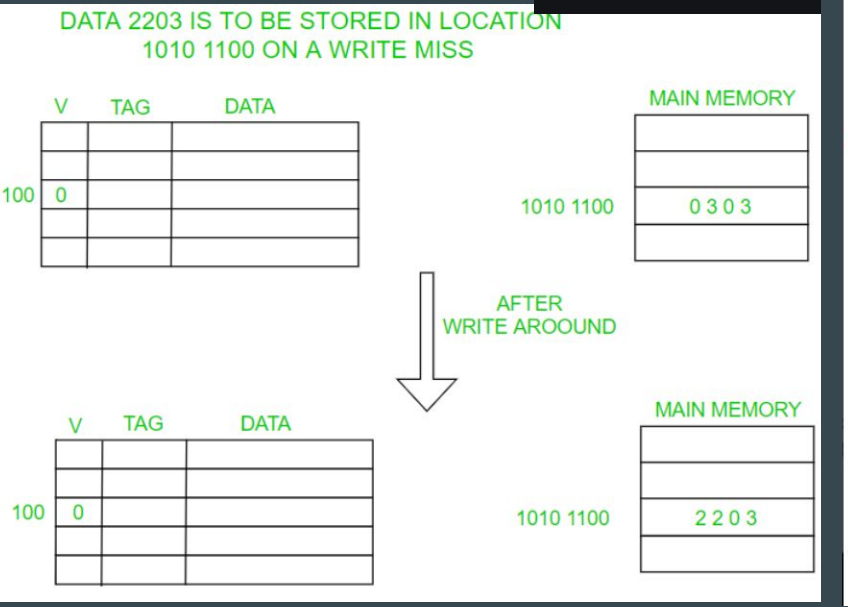
Estrategias de Escritura

En el caso de un Cache miss:.

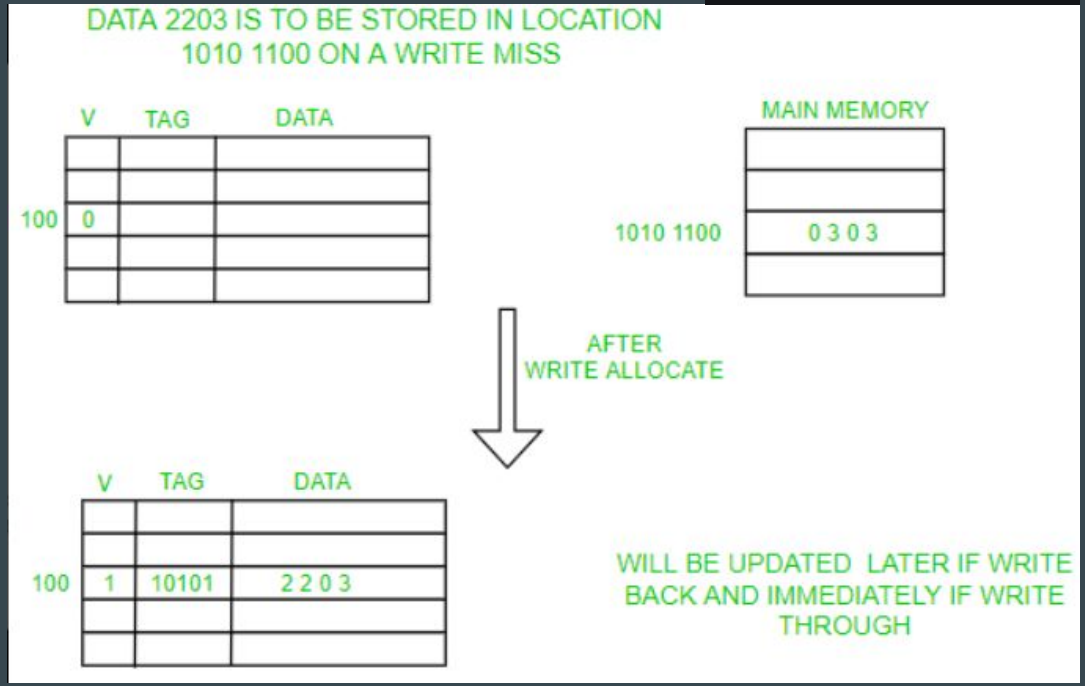
1.- Write-Allocate => es tratar el error de escritura(write miss) como un error de lectura(read miss) seguido de un éxito de escritura.(write hit).

2.-Write-Around=> es escribir solamente en el siguiente nivel de la jerarquía de memoria.

Write Allocation:



Write Around:



Virtual Memory and Paging

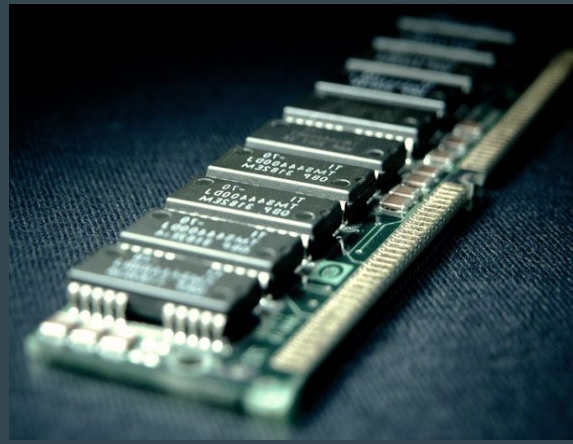
La multiprogramación se volvió norma para que más de un programa pueda estar en la memoria principal al mismo tiempo. Pero desde el punto de vista del manejo de memoria, se tienen varios retos con la multiprogramación, como:

-¿Como y donde se carga un programa en la memoria principal?

-¿Como un programa pide mas memoria si la necesita?

-¿Que previene que un programa pueda tener acceso a los datos de otro programa?

Gracias a estas preguntas a inicios de 1960, algunos científicos computacionales de la Universidad de Manchester introdujeron el término *virtual memory*(Memoria Virtual) y realizaron la primera implementación del paging system(Sistema de paginación).

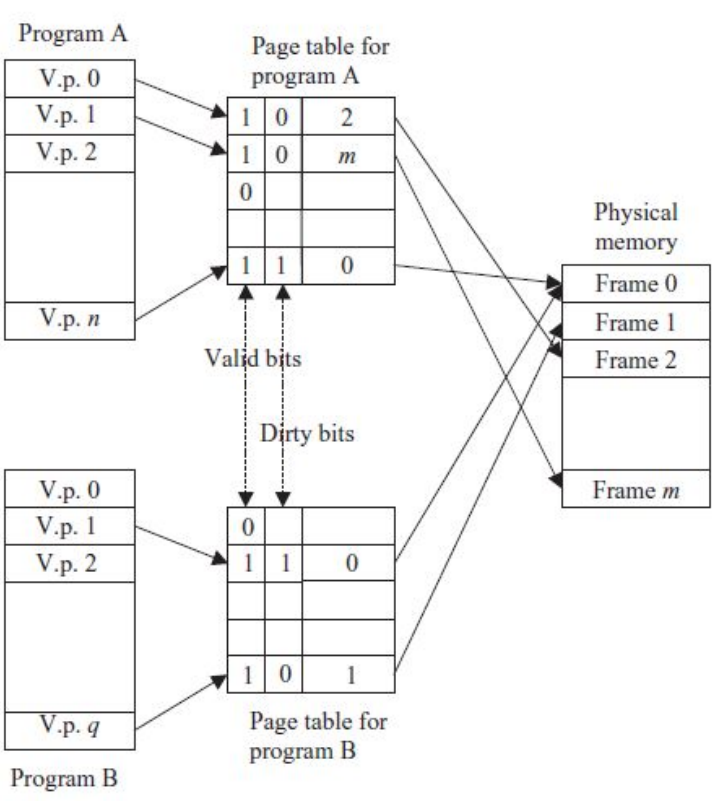


Paging Systems

La más común implementación de una memoria virtual usas la paginación. El espacio de dirección virtual está dividido en trozos del mismo tamaño llamados páginas(virtuales).

El espacio de dirección fisico tambien se divide en trozos del mismo tamaño, son usualmente llamados páginas físicas o *frames*.

El mapeo entre las páginas y los frames es totalmente asociativo, significa que cualquier página puede ser almacenada en cualquier frame.

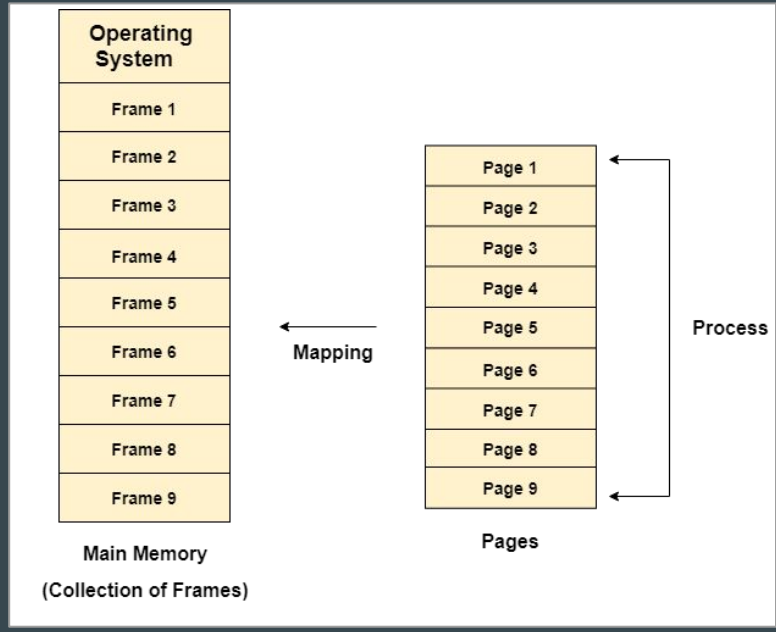


Vista general del Paging.

Entre dos programas A, B.

Funcionamiento del paging

La traducción del número de una página virtual a un número de frame físico se guarda en una *page table entry* (PTE) (Entrada de Tabla de Páginas), que en adición, contiene un *bit válido* indicando si el mapeado es concurrente o no, y también un *dirty bit* para mostrar si la pagina a sido modificada desde que se trajo a la memoria principal.



Ventajas de la memoria virtual

-La dirección del espacio virtual puede ser mucho más grande que la memoria física.

-No todo el programa y sus datos necesita están en la memoria principal en un determinado tiempo

-La memoria física puede ser compartida entre los programas (multiprogramación) sin mucha *fragmentación (La fragmentación es la porción de la memoria que está asignada y está sin usar por los espacios entre las áreas asignables)*

-Las páginas pueden ser compartidas entre los programas.