# Unidad 1: Conceptos generales de Sistemas Operativos.

## Tema 2: Estructura de los sistemas de computación.

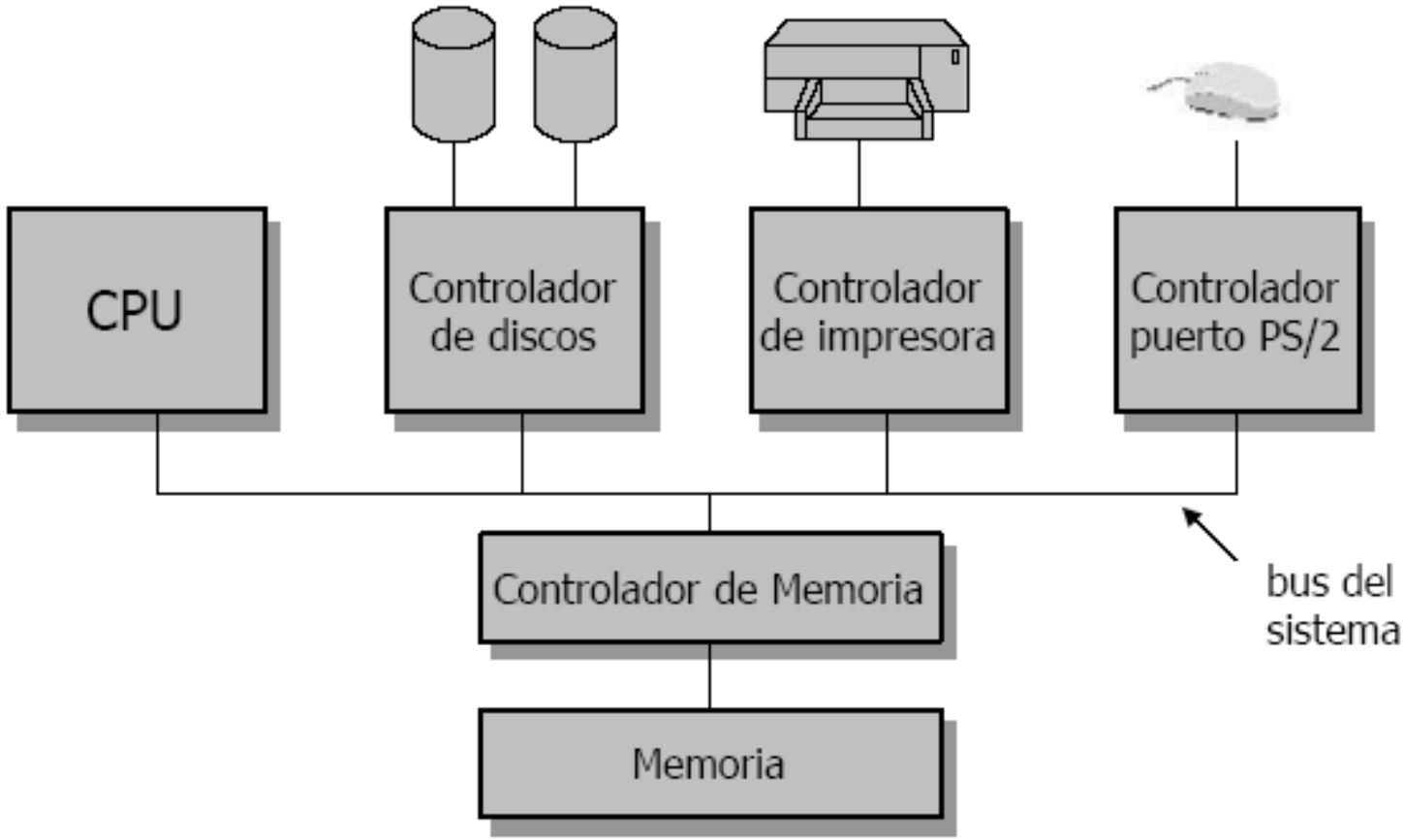
2.1 Funcionamiento de los sistemas de computación.

2.2 Ejecución de instrucciones e interrupciones y estructura de E/S.

2.3 Almacenamiento: Estructura y jerarquías.

2.4 Protección por hardware.

# Estructura de un computador moderno:



## 2.1 Funcionamiento de los sistemas de computación:

- **Programa de arranque:** necesario para que un ordenador comience a funcionar.
  - **Asignará valores iniciales** a los registros de la CPU, a los controladores de dispositivos y al contenido de la memoria.
  - Sabrá cómo **cargar el SO y comenzar a ejecutarlo**:
    - Localiza y carga en memoria el núcleo del SO.
    - Ejecuta el primer proceso **“init”**.
    - Espera la ocurrencia de algún suceso.

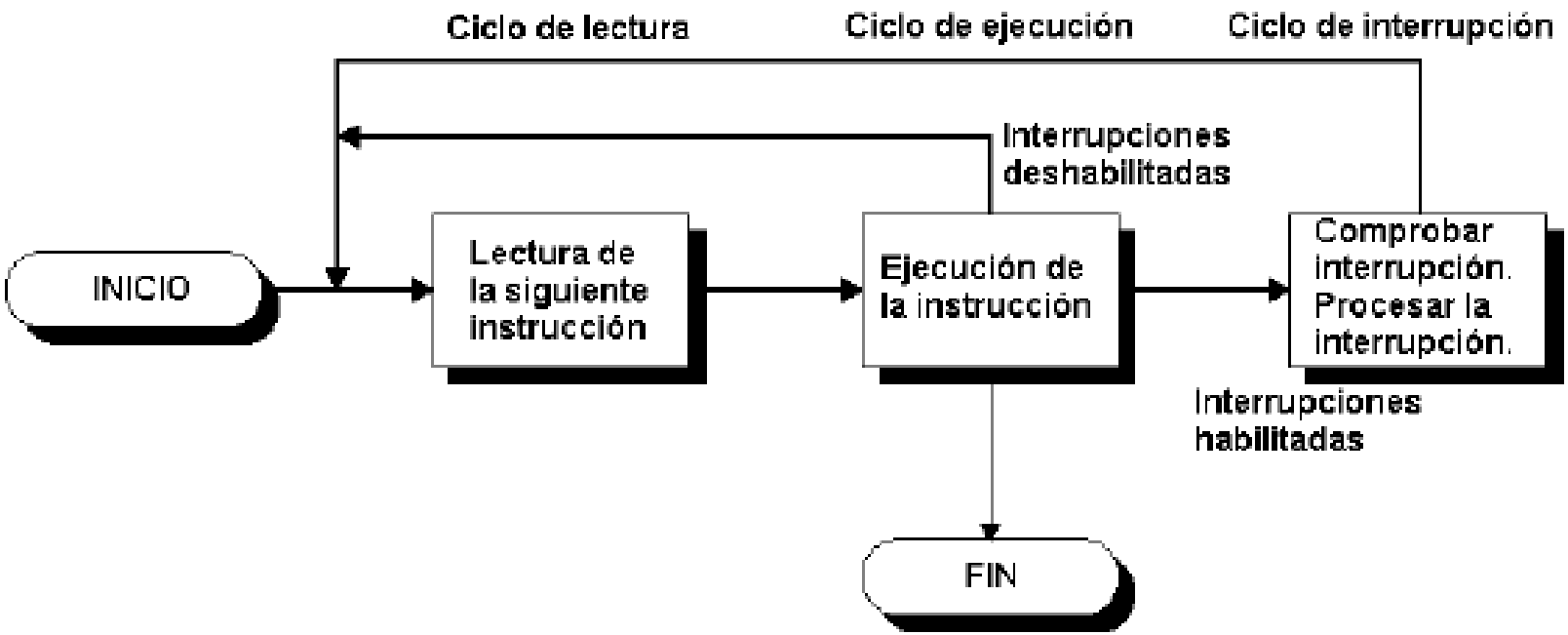
- **Interrupción:** indica la ocurrencia de un suceso.
  - El hardware **envía una señal a la CPU**.
  - El software ejecuta una **llamada al sistema**.

## 2.1 Funcionamiento de los sistemas de computación:

- **Interrupciones:**
  - Pueden generar una interrupción:
    - División entre 0, o desbordamiento (de software).
    - Acceso no válido a la memoria, error de dirección (de software).
    - La terminación de una operación de E/S (de hardware).
    - Señal externa de reloj, fallo de circuitería (hardware).
  - Existe una **rutina de servicio (RSI)** para cada una de ellas.
  - Cuando la CPU se interrumpe, se suspende la acción, se guarda el estado de la CPU, y se transfiere la ejecución a una posición fija que contiene la dirección inicial de la **rutina de servicio para esa interrupción**. Se ejecuta la rutina, y al terminar la CPU reanuda la acción que fue interrumpida.
  - Para manejar las interrupciones más rápidamente se utiliza una tabla de punteros a rutinas de interrupción **(vector de interrupciones)**.

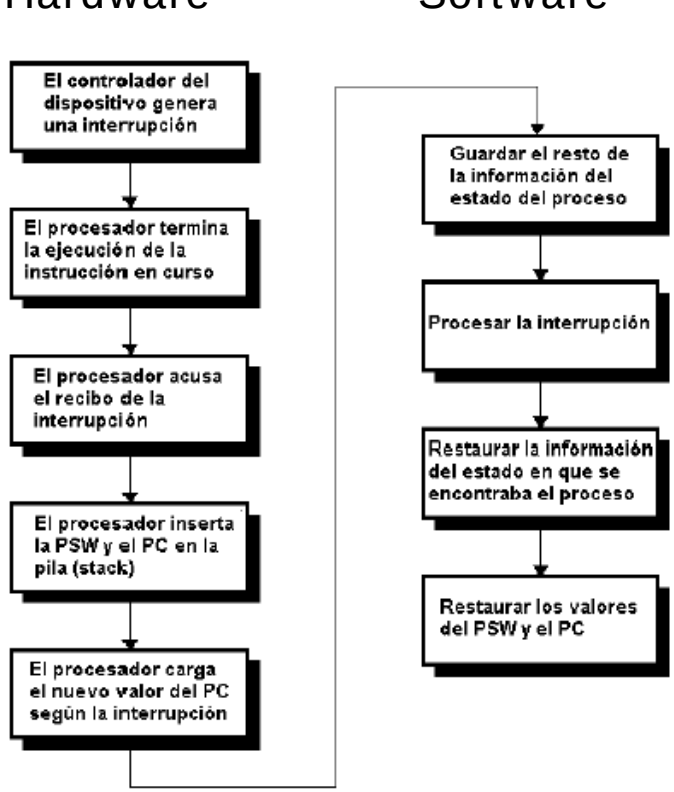
## 2.1 Funcionamiento de los sistemas de computación:

- **Ciclo de instrucción con interrupciones:**



# 2.1 Funcionamiento de los sistemas de computación:

- **Tratamiento genérico de una interrupción simple:**



## 2.1 Funcionamiento de los sistemas de computación:

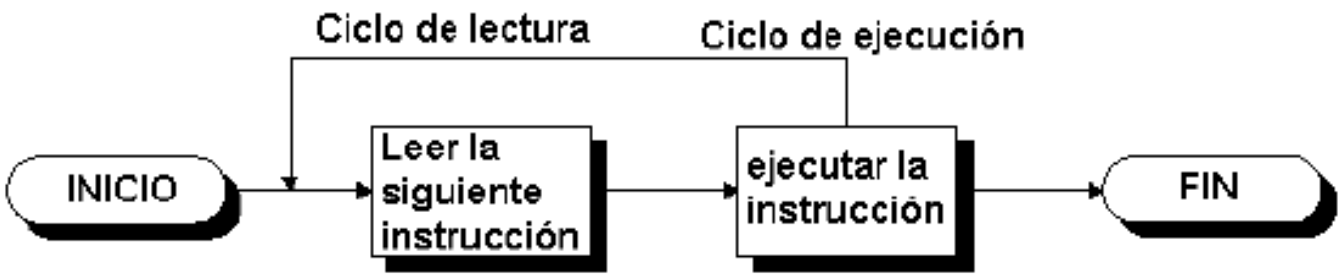
- **Interrupciones:**
  - Las demás interrupciones **se desactivan** mientras se procesa una interrupción, cuando el SO termine de atender la interrupción actual **se vuelven a activar**.
  - Nota: Algunas arquitecturas avanzadas permiten **procesar simultáneamente más de una interrupción**, siempre y cuando la nueva interrupción tenga mayor prioridad que la que se está procesando.
  - Los SO modernos son del tipo **"controlado por interrupciones"** (el SO espera tranquilamente por un **suceso**).
  - Los **sucesos** conllevan la **ocurrencia de una interrupción o una trampa** (excepción).
  - **Trampa:** interrupción generada por software debida a un error (p.e. acceso no válido a memoria) o una solicitud específica de un programa de usuario que necesita del SO. Cuando ocurren el hardware transfiere el control al SO.

# 2.2 Ejecución de instrucciones e interrupciones y estructura de E/S.

- 2.2.1 Estructura de la E/S:
  - Los dispositivos se conectan al bus a través de los controladores de E/S.
  - La CPU se comunica con los controladores a través de instrucciones especiales o de direcciones de memoria concretas.
  - Cada controlador tiene un búfer local. La CPU envía y recoge datos del búfer.
  - El controlador notifica a la CPU la finalización de una operación o la llegada de nuevos datos mediante una interrupción.

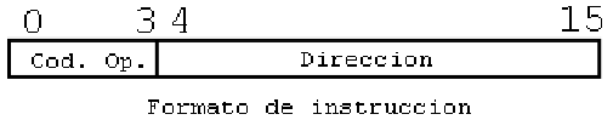
# 2.2 Ejecución de instrucciones e interrupciones y estructura de E/S.

- 2.2.2 Ejecución de instrucciones (1):
  - Se clasifican en:
    - Procesador – memoria.
    - Procesador – E/S.
    - Procesamiento de datos.
    - Control.
  - Ciclo de instrucción:



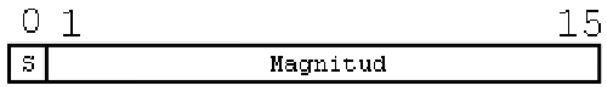
# 2.2 Ejecución de instrucciones e interrupciones y estructura de E/S.

- 2.2.2 Ejecución de instrucciones (2):
  - Características de un máquina hipotética:
    - a)



Formato de instruccion

    - b)



Formato de entero

    - c) Registros internos de CPU:
      - PC: contador de programa, contiene la dirección de la siguiente instrucción a leer.
      - IR: registro de instrucción (de la que está ejecutándose).
      - AC: acumulador, almacenamiento temporal.
    - d) Lista parcial de códigos de operación:
      - 0001 Cargar de memoria a AC.
      - 0010 Almacenar AC en memoria.
      - 0101 Sumar a AC el contenido de la memoria.

# 2.2 Ejecución de instrucciones e interrupciones y estructura de E/S.

- 2.2.3 Interrupciones de E/S:
  - Para iniciar una operación de E/S por parte de un proceso del usuario:
    - La CPU carga los registros en el controlador del dispositivo.
    - El controlador examina los registros para saber qué acción realizar.
    - Tras realizar la acción el controlador informa de ello a la CPU.
    - Esa comunicación se efectúa generando una interrupción.

# 2.2 Ejecución de instrucciones e interrupciones y estructura de E/S.

- Al iniciar una E/S hay dos alternativas:
  - **E/S síncrona:** el SO espera a que termine la E/S.
  - **E/S asíncrona:** el SO devuelve el control al programa del usuario sin esperar a que se complete la E/S.
- Funcionamiento asíncrono + multiprogramado:
  - El usuario solicita E/S mediante llamada al sistema.
  - El SO tramita la operación.
  - El SO entrega el control a otro proceso (mientras se desarrolla la E/S).
  - Cuando la E/S termina genera una interrupción que:
    - Interrumpe el proceso actual.
    - Provoca la aparición del SO, que reactiva el proceso que hizo la llamada.

# 2.2 Ejecución de instrucciones e interrupciones y estructura de E/S.

- **2.2.4 Estructura de DMA (Direct Memory Access):**
  - Ocurre que dispositivos de alta velocidad transmiten información a velocidad cercana a la de la memoria, lo que lleva a la CPU a responder continuamente a las interrupciones. Lo que implica que **no quede mucho tiempo para la ejecución de procesos**.
  - Para resolver este problema se emplea el **DMA** con los **dispositivos de E/S de alta velocidad**.
  - **Funcionamiento:**
    - El controlador del dispositivo **transfiere un bloque completo de datos** directamente de su propio buffer a la memoria, o viceversa, sin intervención de la CPU.
    - Sólo se genera **una interrupción por cada bloque**.
    - El controlador de **DMA interrumpirá a la CPU cuando se haya completado la transferencia**.

# 2.3 Almacenamiento: Estructura y jerarquías.

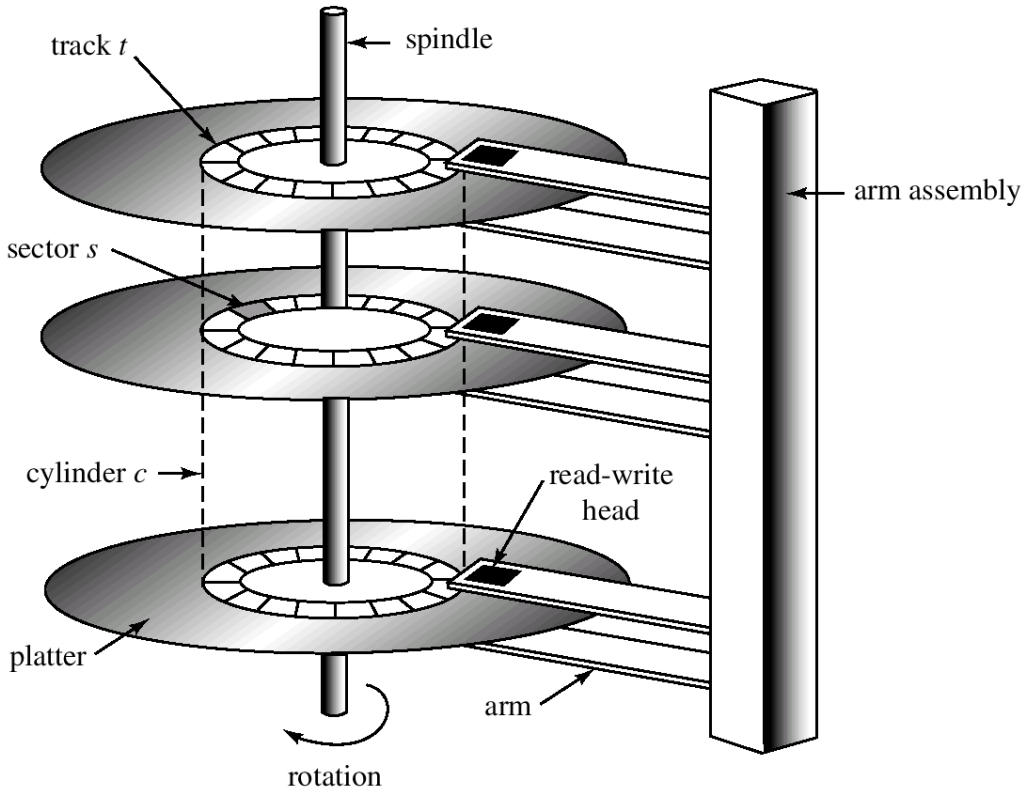
- Estructura de almacenamiento:
  - Para poder ejecutarse los **programas** deben de estar **en memoria principal**, que es el único gran área de almacenamiento al que el procesador puede acceder directamente. Es una matriz de palabras (o bytes) donde cada palabra tiene su propia dirección.
    - **Carga (load):** transfiere una palabra entre la memoria principal y la CPU.
    - **Almacenaje (store):** transfiere el contenido de un registro a la memoria.
  - Necesidad de **almacenamiento secundario**:
    - La memoria principal **no es lo suficientemente grande** para contener permanentemente todos los programas y datos.
    - La memoria principal es **volátil**.

# 2.3 Almacenamiento: Estructura y jerarquías.

- 2.3.1 Memoria principal:
  - Único almacén al que **la CPU puede acceder directamente**.
  - Los **programas y datos** deben de colocarse **en la memoria principal** para que la CPU pueda operar con ellos.
  - Método de E/S con mapa en la memoria apropiado para dispositivos con tiempos de respuesta rápidos (controladores de video, puertos serie y paralelo para conexión de módems e impresoras).

# 2.3 Almacenamiento: Estructura y jerarquías.

- 2.3.2 Discos magnéticos:

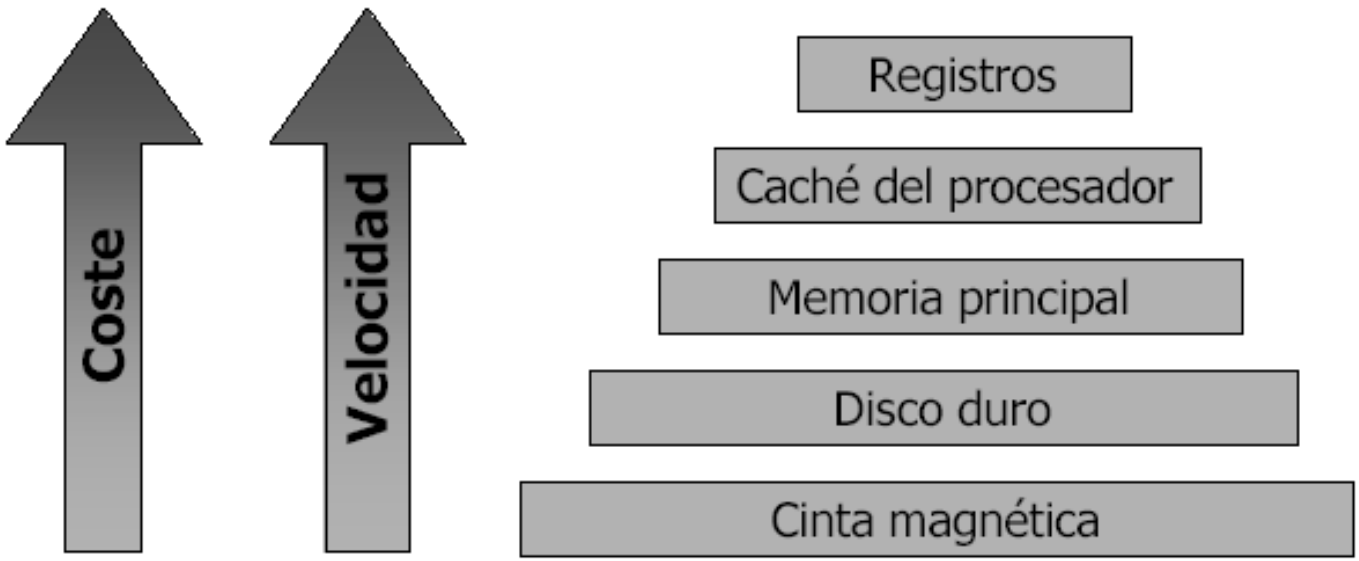


# 2.3 Almacenamiento: Estructura y jerarquías.

- 2.3.3 Cintas magnéticas:
  - **Carrete enrollable** sobre el que se pasa una **cabeza de lectura-escritura**.
  - Llegar al punto deseado de la cinta puede llevar **mucho tiempo.**
  - En desuso, el **acceso aleatorio es 100 veces menor que en un disco.**
  - **Gran capacidad de almacenamiento**.
  - **Se clasifican según su anchura**:
    - Medidas: 4 mm, 8 mm, 19 mm, ¼”, ½”.

# 2.3 Almacenamiento: Estructura y jerarquías.

- 2.3.4 Jerarquías de almacenamiento (1):
  - Organización jerárquica según **coste, velocidad y tamaño**.
  - Nota: Son **volátiles** en esta pirámide por encima de la memoria principal.



# 2.3 Almacenamiento: Estructura y jerarquías.

- 2.3.4 Jerarquías de almacenamiento (2):
  - **Uso de cachés:**
    - **Caché:** Sistema de almacenamiento temporal de **acceso muy rápido.**
    - Cuando necesitemos un elemento de información, primero miramos la caché, si está ahí utilizamos esa información directamente y si no miramos en memoria principal y colocamos una **copia en la caché si hay muchas posibilidades de que se vaya a necesitar de nuevo pronto**.
    - **Principio de caché:** guardar en la memoria más rápida la información que se usa con más frecuencia.

# 2.3 Almacenamiento: Estructura y jerarquías.

- 2.3.4 Jerarquías de almacenamiento (3):
  - **Problemas de las cachés:**
    - **Consistencia:** debemos asegurarnos que todas las réplicas se actualizan si un dato está copiado en memorias de distinto nivel. (entornos distribuidos).
    - **Coherencia:** debemos asegurarnos de que todas las réplicas se actualizan si modificamos un dato que está copiado simultáneamente en varias cachés del mismo nivel (multiprocesadores).

# 2.4 Protección por hardware.

- Con multiprogramación colocamos varios programas en memoria a la vez. Un programa con errores puede modificar el propio programa o los datos de otro programa e incluso el monitor residente.

- Para que el SO funcione adecuadamente, hay que impedir que los programas de usuario puedan realizar libremente ciertas operaciones:
  - Acceso a la memoria del SO y de otros programas.
  - Acceso directo a los dispositivos de E/S.
  - Utilizar la CPU todo el tiempo que quieran.

- Solución: Modo dual de operación.

# 2.4 Protección por hardware.

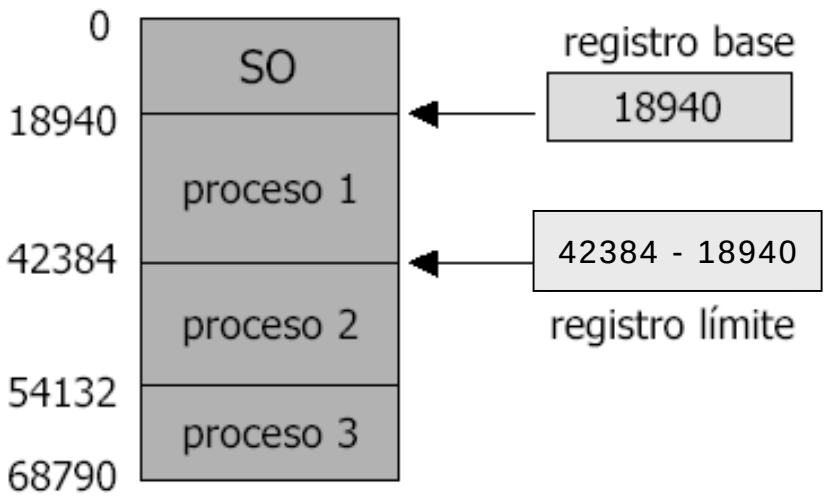
- 2.4.1. Modo dual de operación:
  - Se requiere **protección** para cualquier **recurso compartido**. Como mínimo necesito dos modos de operación distintos.
    - **Modo de usuario o no privilegiado:** si se intenta ejecutar una instrucción privilegiada la CPU interrumpe la ejecución y genera una trampa, pasando el control al SO.
    - **Modo de monitor, supervisor, de sistema o privilegiado**.
  - Al hardware se la añade un **bit de modo** para indicar en qué modo se está operando (monitor: 0, usuario: 1).
  - La **CPU arranca en modo privilegiado**, luego se carga el SO, que inicia procesos de usuario en modo usuario. Cada vez que ocurre una trampa el hw pasa del modo de usuario al modo de monitor (pone a 0 el bit de modo). El sistema cambia al modo usuario antes de transferir el control a un programa de usuario.

# 2.4 Protección por hardware.

- 2.4.2 Protección de E/S:
  - Un programa usuario puede perturbar el funcionamiento normal del sistema emitiendo **instrucciones de E/S no válidas**, accediendo a posiciones de memoria dentro del propio SO o negándose a ceder la CPU.
  - **Para evitar E/S no válidas:** definimos todas las E/S como privilegiadas, entonces, los usuarios sólo podrán emitir E/S a través del SO.
  - **Dos modos de acceso a la E/S:**
    - **Con instrucciones especiales** (in, out), que han de ser privilegiadas.
    - **A través de la memoria:** el acceso a las direcciones que usa la E/S debe de estar prohibido en modo usuario.

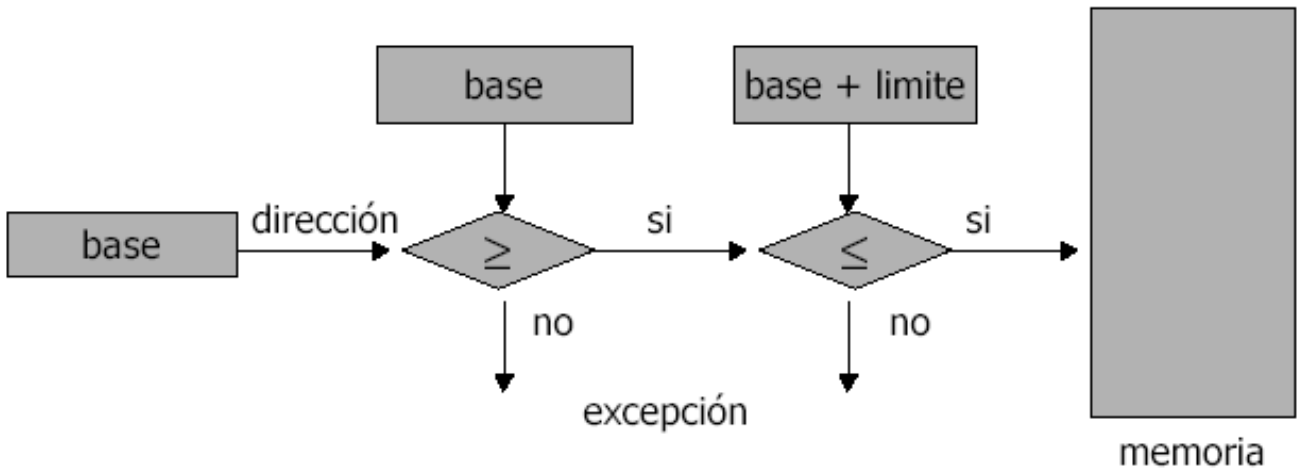
# 2.4 Protección por hardware.

- 2.4.3 Protección de memoria (1):
  - **Idea fundamental**: **delimitar el espacio de direcciones** a las que el programa puede acceder (mediante un par de registros, **base y límite**), protegiendo la memoria que no está en ese espacio.



# 2.4 Protección por hardware.

- 2.4.3 Protección de memoria (2):
  - La CPU compara con estos registros todas las direcciones generadas en modo usuario. Si un programa de modo usuario intenta acceder a la memoria del monitor o de otro usuario el control pasa a través de una trampa al monitor, que trata el intento como un error de direccionamiento.



# 2.4 Protección por hardware.

- 2.4.4 Protección de la CPU:
  - **Idea: impedir** que un programa usuario se atasque en un **ciclo infinito** (acaparando el tiempo de CPU) y nunca devuelva el control al SO.
  - **Solución: temporizador** (“timer”) que interrumpa al computador después de un periodo (fijo o variable). Genera una interrupción tras un tiempo especificado (así el SO recupera el control).
    - Contiene un **contador inicializado al valor deseado** que se decrementa con cada pulso de reloj del sistema.
    - Cuando el **contador** llega **a 0 genera una interrupción**.
  - Nota: usos de los temporizadores:
    - Implementación del tiempo compartido.
    - Cálculo de la hora actual.