# SISTEMAS OPERATIVOS I

## INTRODUCCIÓN

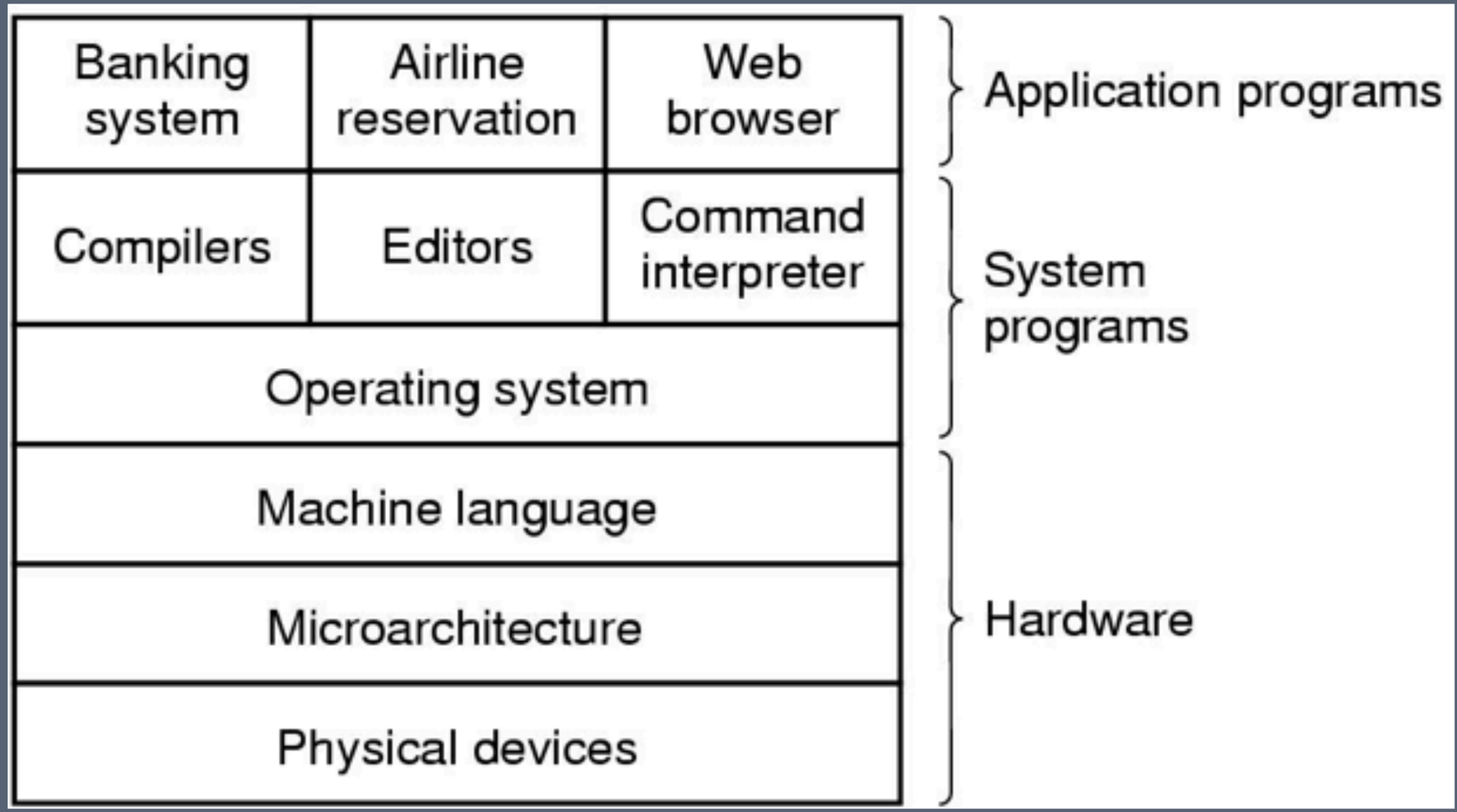
- QUE ES UN SO?
- HISTORIA DE LOS SO
- VARIEDAD DE SO
- REVISIÓN DEL HARDWARE
- CONCEPTOS DE SO
- LLAMADAS AL SISTEMA
- ESTRUCTURA DE LOS SO

# ¿QUÉ ES UN SISTEMA OPERATIVO?

# SISTEMA OPERATIVO

- ES UNA MAQUINA EXTENDIDA
- OCULTA LOS DETALLES TEDIOSOS Y COMPLEJOS QUE SE DEBEN REALIZAR
- PRESENTA AL USUARIO UNA "MÁQUINA VIRTUAL" QUE ES MUCHO MÁS SIMPLE DE USAR
- ES UN ADMINISTRADOR DE RECURSOS
- CADA PROGRAMA OBTIENE TIEMPO DE USO DE RECURSOS
- CADA PROGRAMA OBTIENE ESPACIO DE UTILIZACIÓN DE UN RECURSO-

# COMPONENTES DE UN SISTEMA

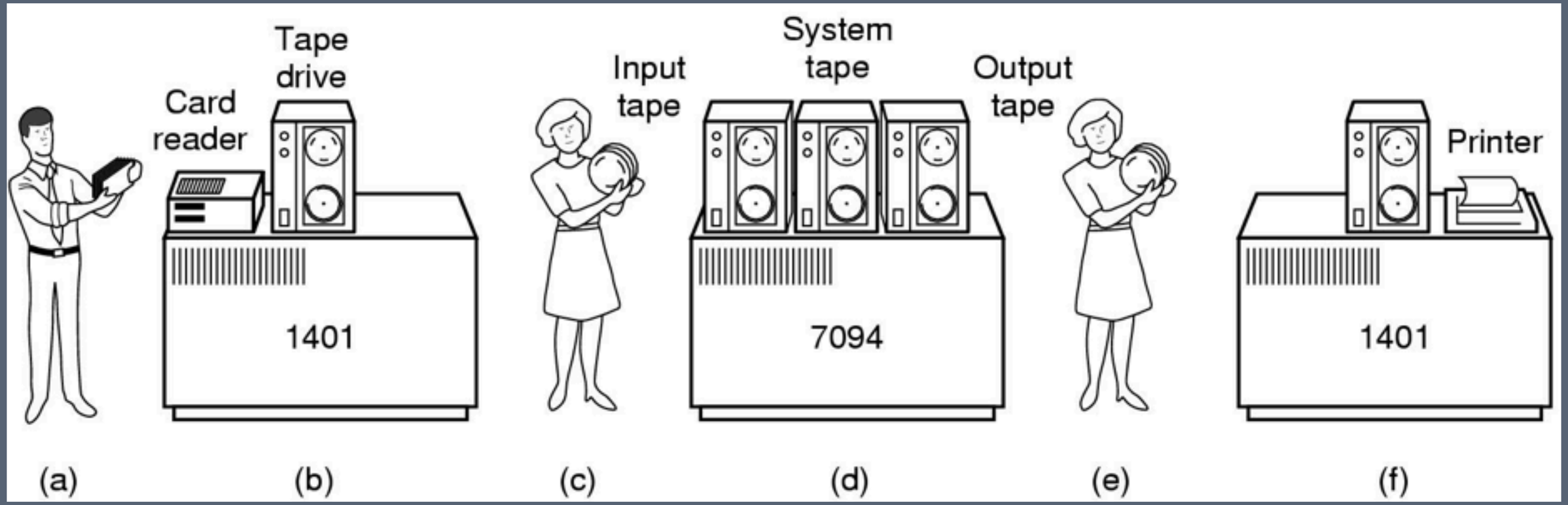


# HISTORIA

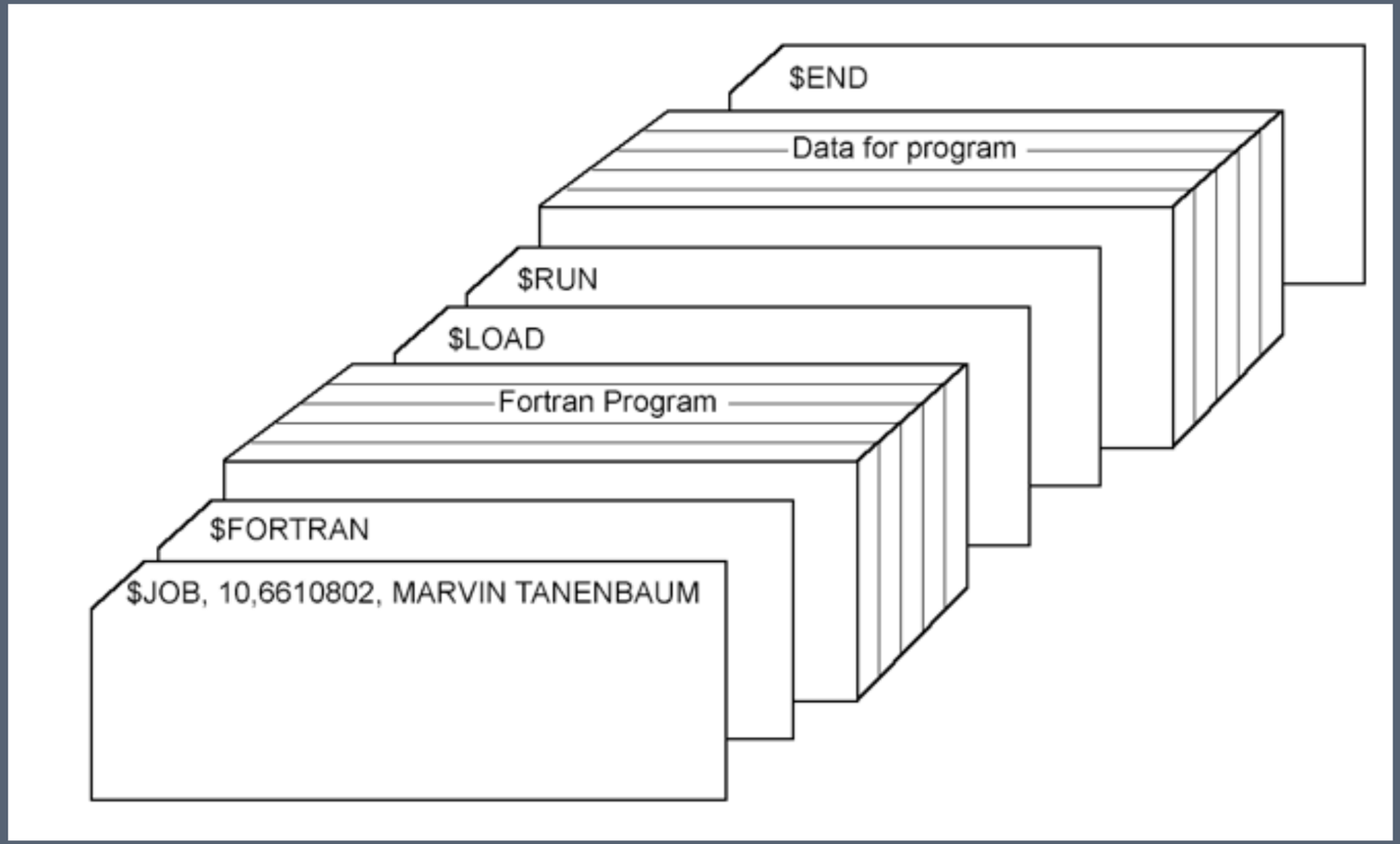
# GENERACIONES

- PRIMERA GENERACION 1945 - 1955: VALVULAS, TABLEROS
- SEGUNDA GENERACIÓN 1955 - 1965: TRANSISTORES, SISTEMAS BATCH
- TERCERA GENERACIÓN 1965 - 1980: CIRCUITOS INTEGRADOS, MULTIPROGRAMACIÓN
- CUARTA GENERACIÓN 1980 - PRESENTE: MICROPROCESADORES, COMPUTADORAS PERSONALES

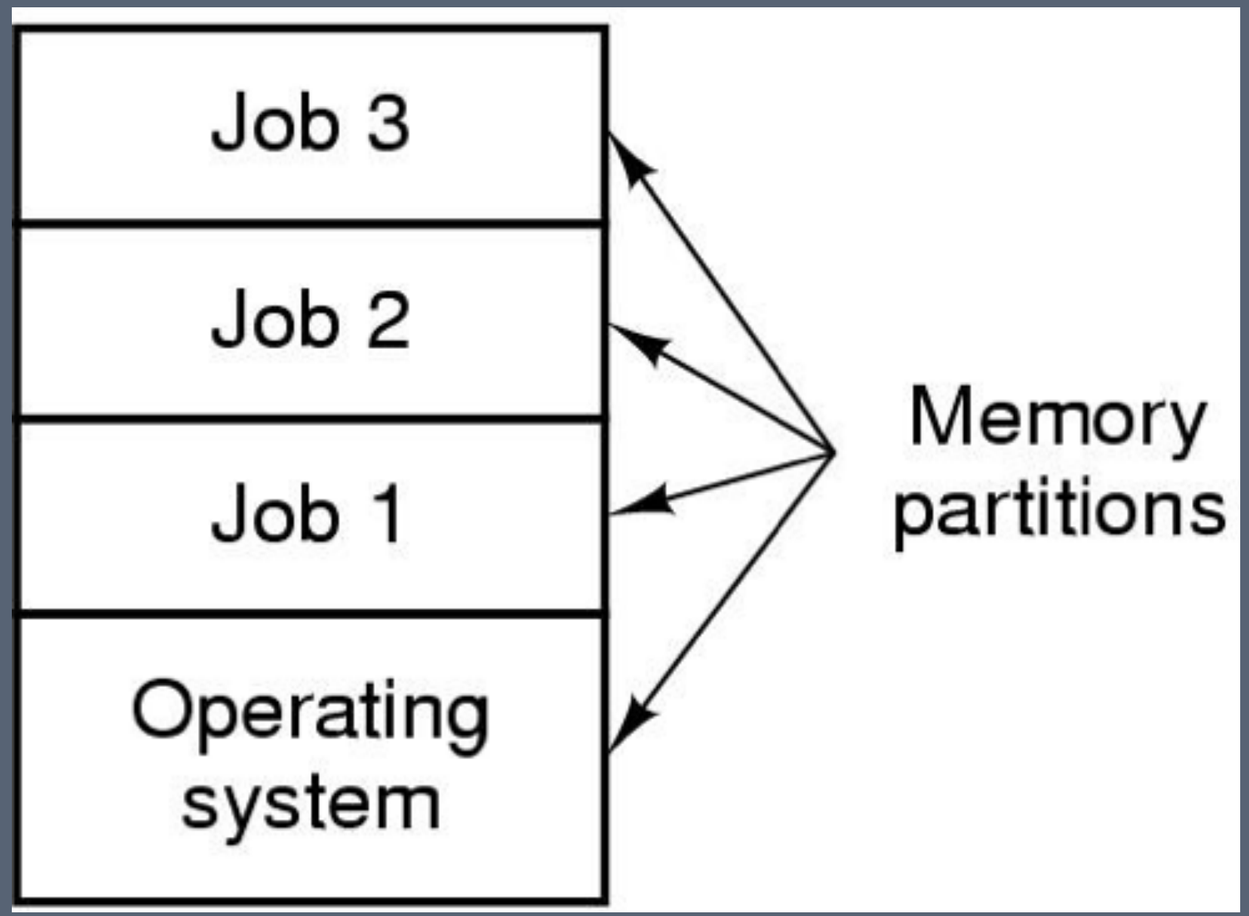
# SISTEMAS BATCH



# SISTEMAS BATCH



# MULTIPROGRAMACIÓN

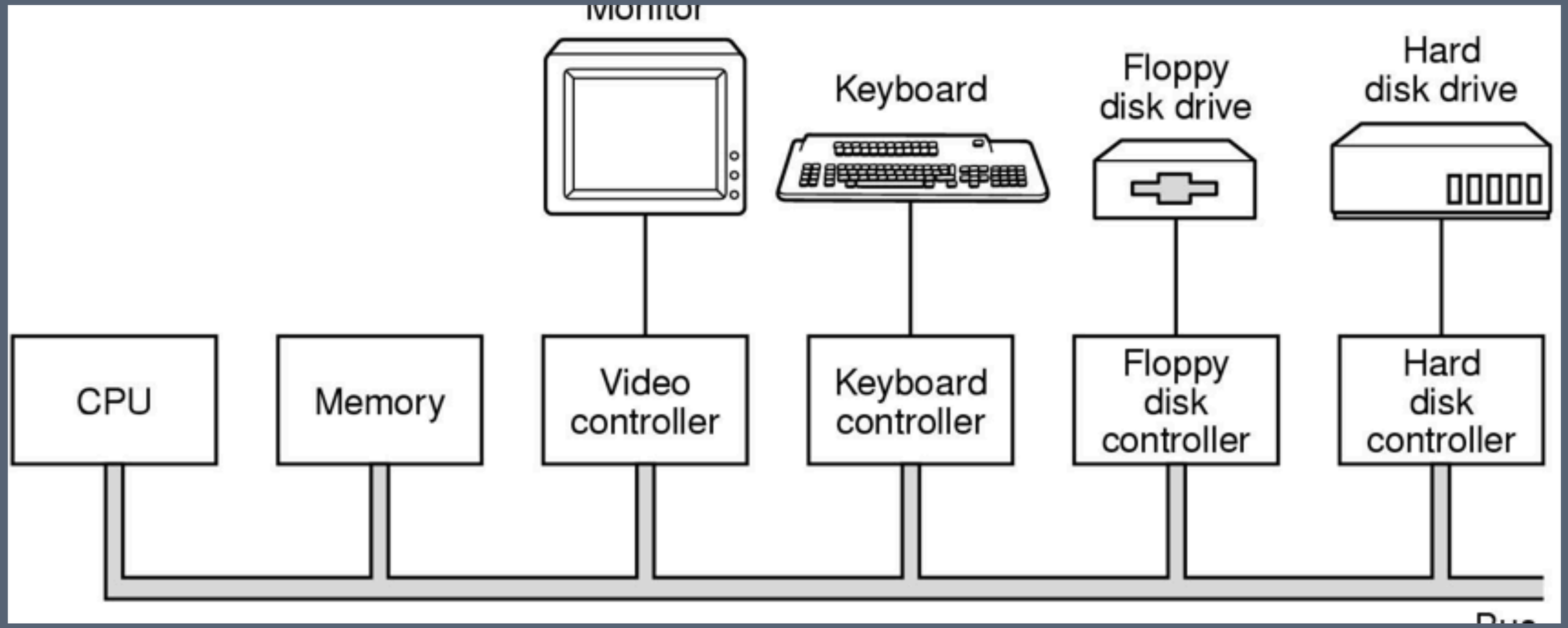


# DIVERSIDAD DE SO

- MAINFRAMES
- SERVIDORES
- MULTIPROCESADOR
- COMPUTADORA PERSONAL
- TIEMPO REAL
- EMBEBIDOS

# REVISIÓN DEL HARDWARE

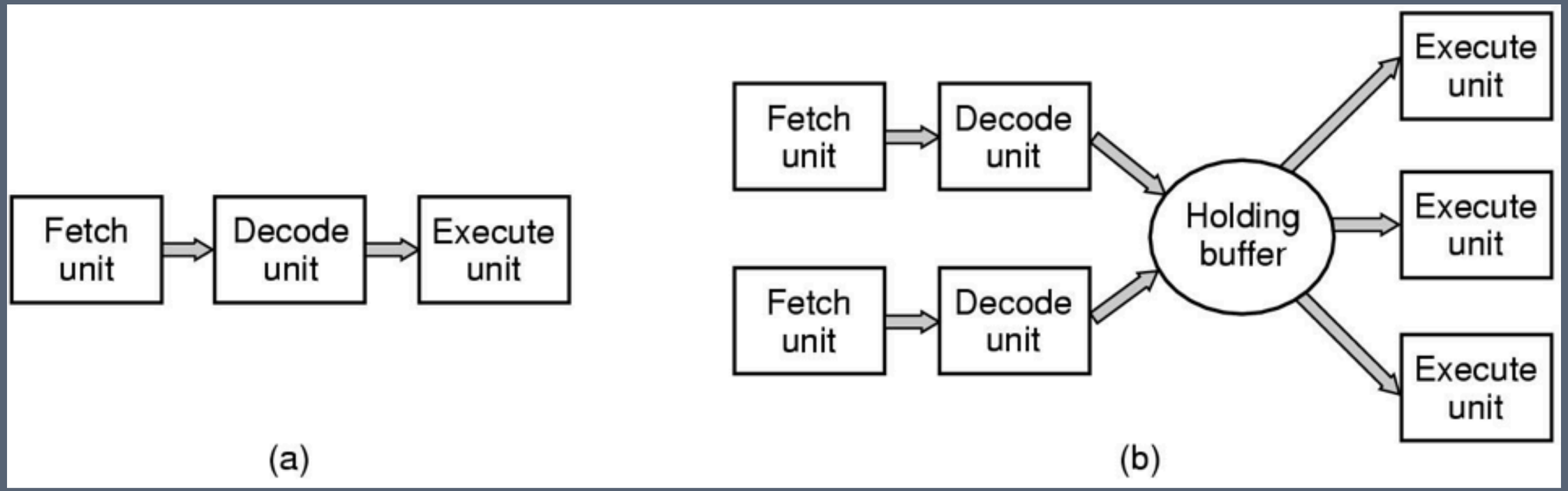
# COMPONENTES DE UNA COMPUTADORA



# CPU

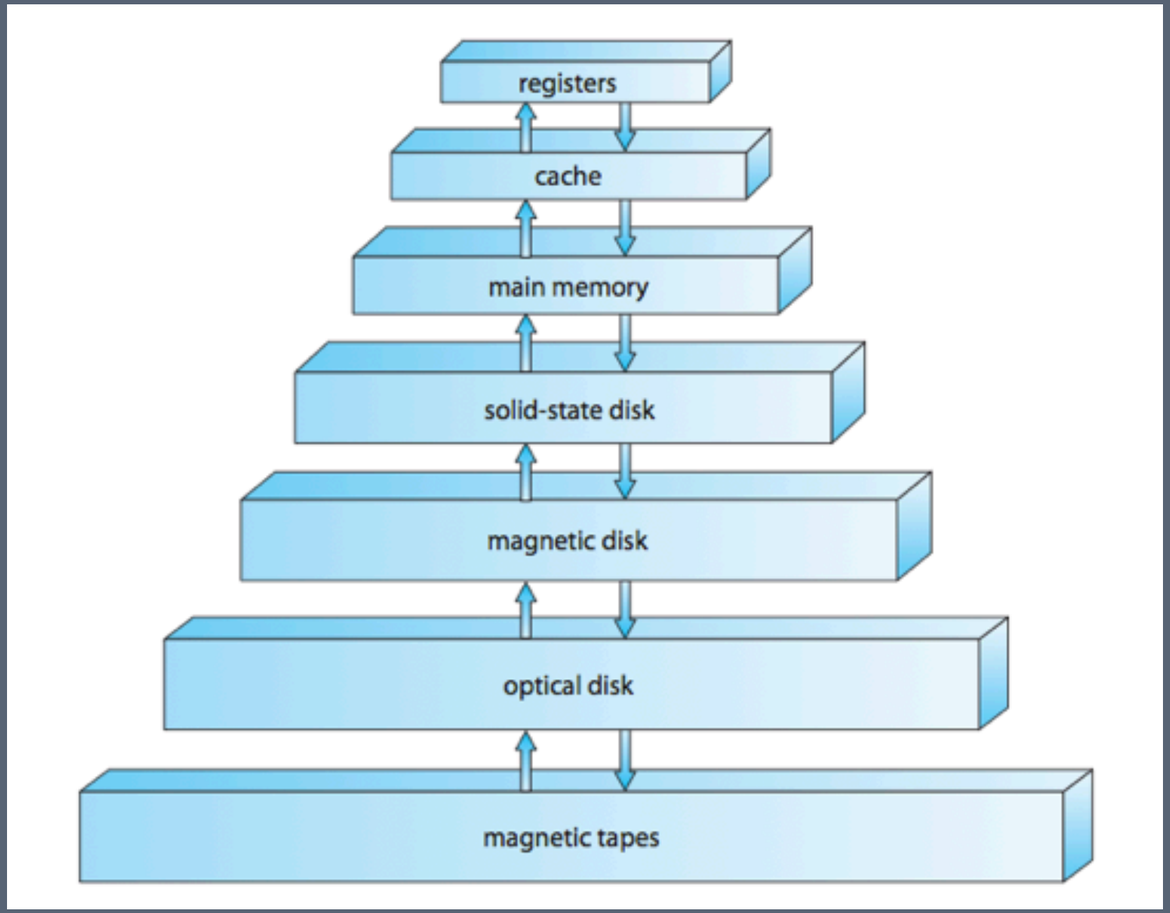
- TOMAR INSTRUCCION DE MEMORIA. DECODIFICARLA Y EJECUTAR
  - CONJUNTO DE INSTRUCCIONES ESPECÍFICO
- REGISTROS DE PROPÓSITO GENERAL. CONTADOR DE PROGRAMA. PUNTERO A LA PILA. PSW

# TIPOS DE CPU




- (A) PIPELINE 3 ETAPAS
- (B) SUPERESCALAR

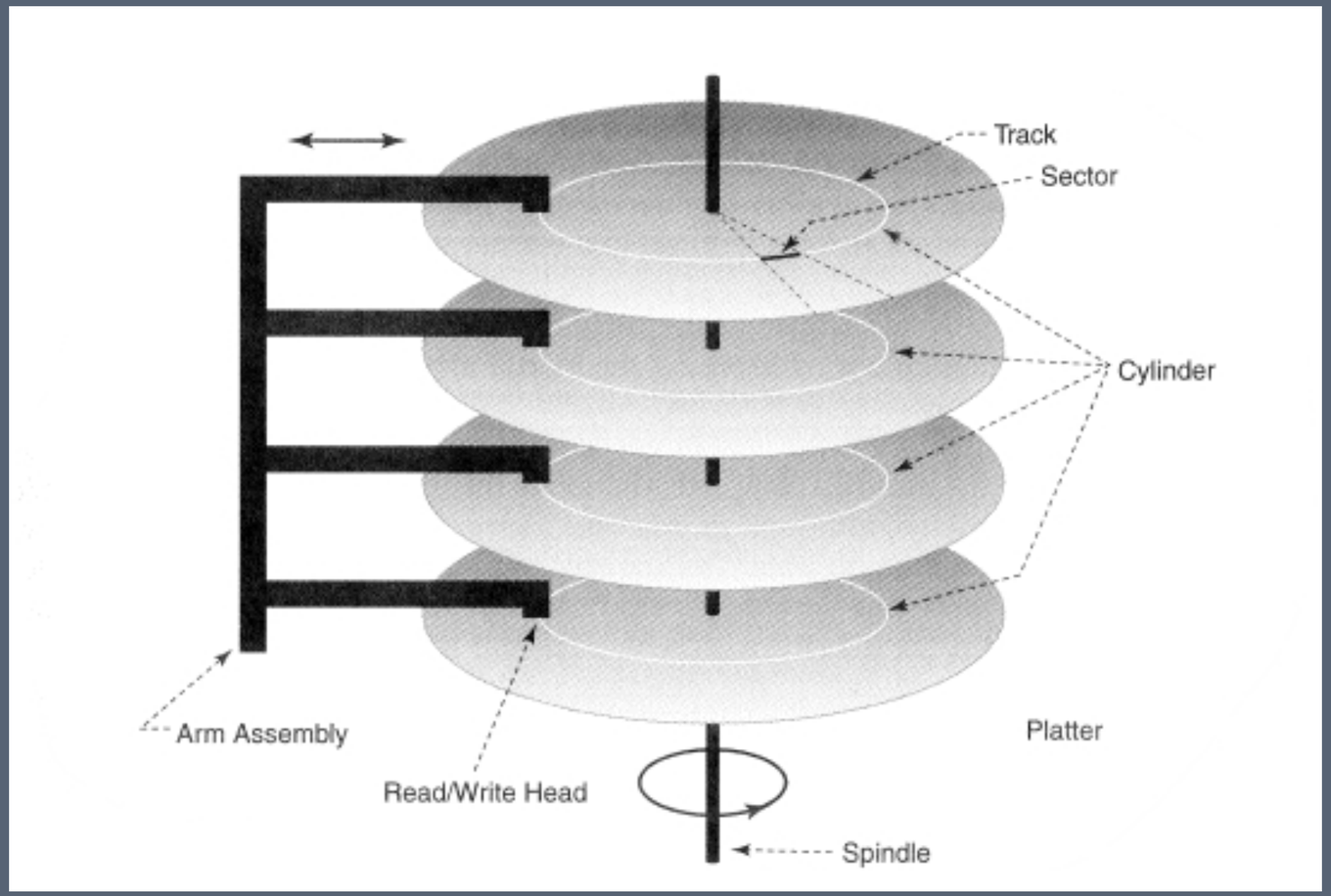
# ALMACENAMIENTO



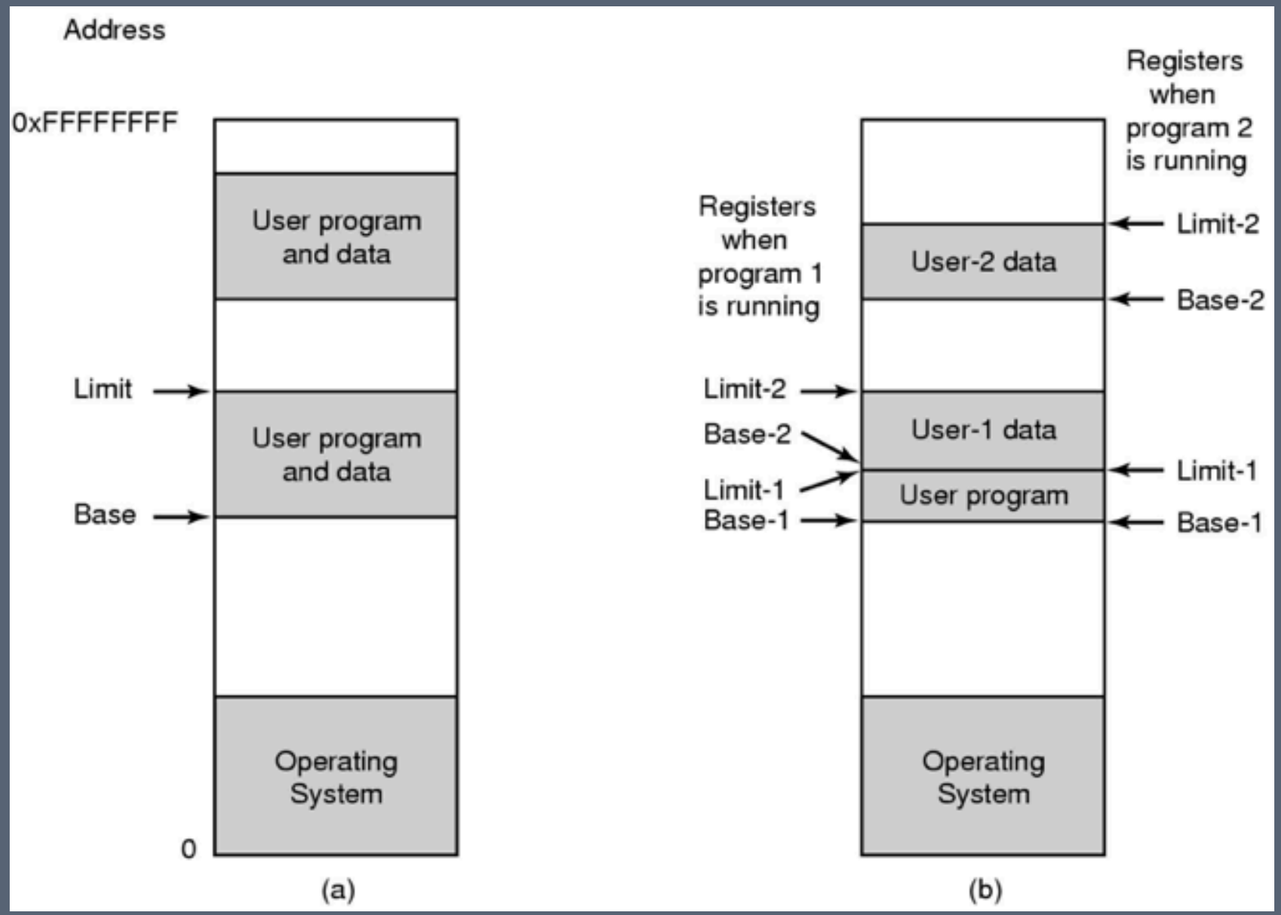
# ALMACENAMIENTO

| NIVEL | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| TIPO | REG | CACHE | MEM PPAL | DISCO |
| TAMAÑO | < 1KB | > 16MB | 16GB | 128GB |
| TECNOLOGÍA | CMOS | CMOS/SRAM | CMOS/DRAM | SSD |
| T ACCESO (NS) | 0.25 | 0.5 | 100 | 35000 |
| BANDWITH (MB/S) | 100000 | 10000 | 5000 | 250 |
| MANEJADO | COMPILADOR | HARDWARE | SO | SO |

# DISCOS MAGNÉTICOS



# MEMORIA PRINCIPAL

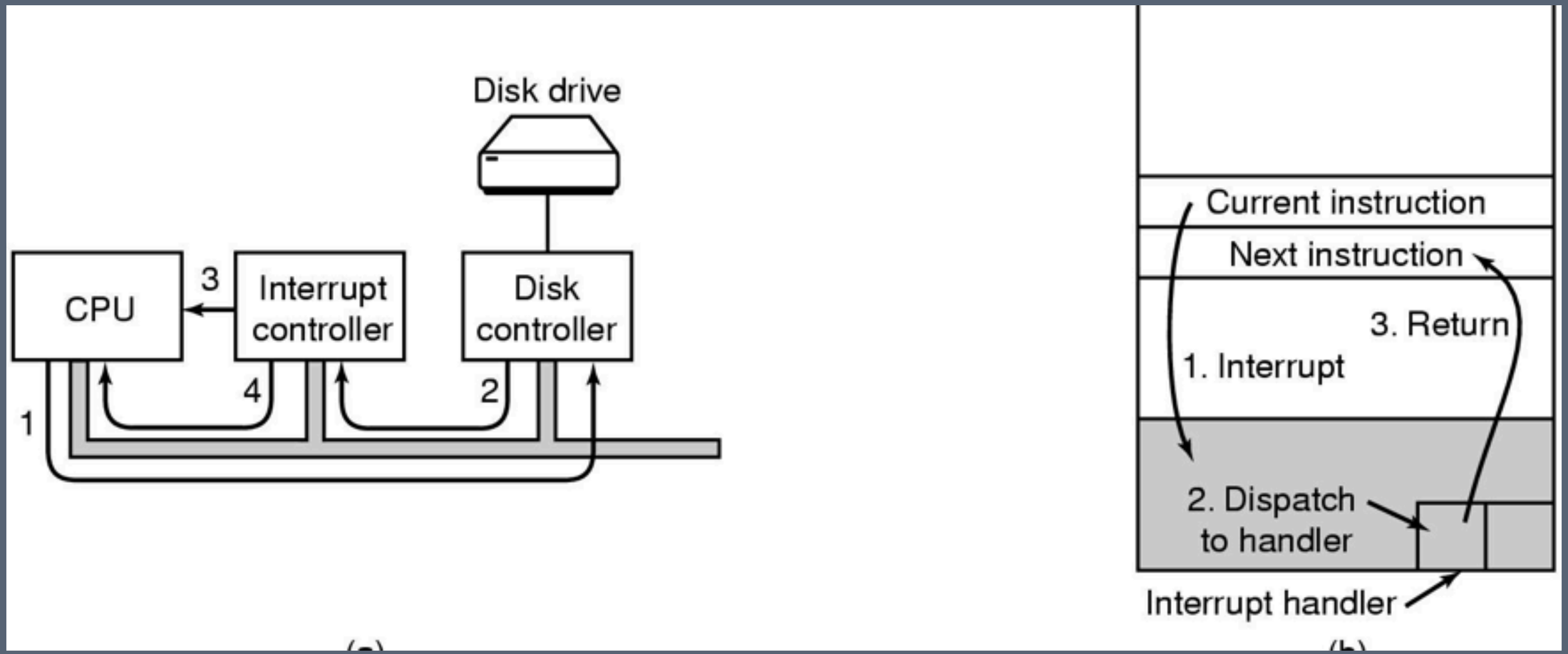


# ENTRADA/ SALIDA

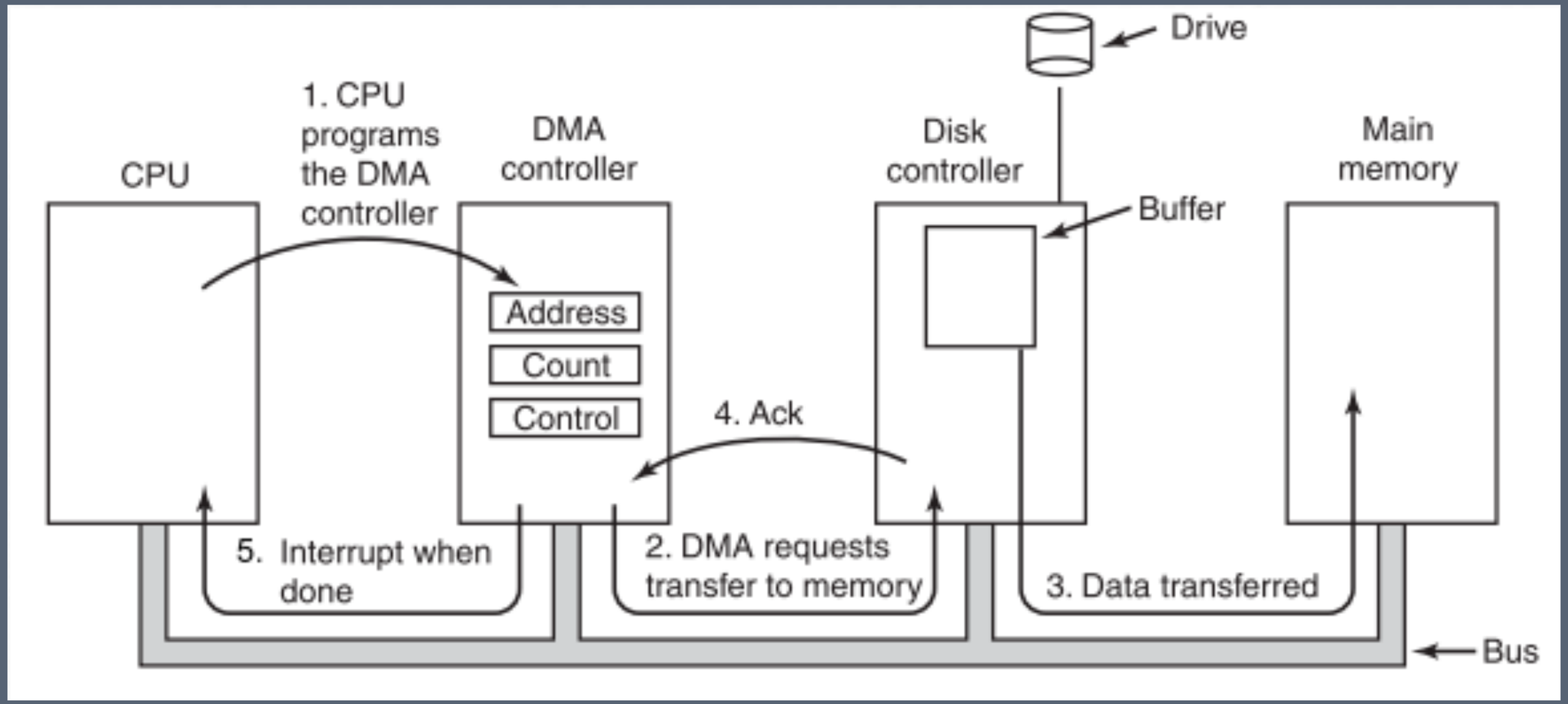
# SOLICITUDES DE E/S

- BUSY WAITING
- INTERRUPCIONES
- DMA

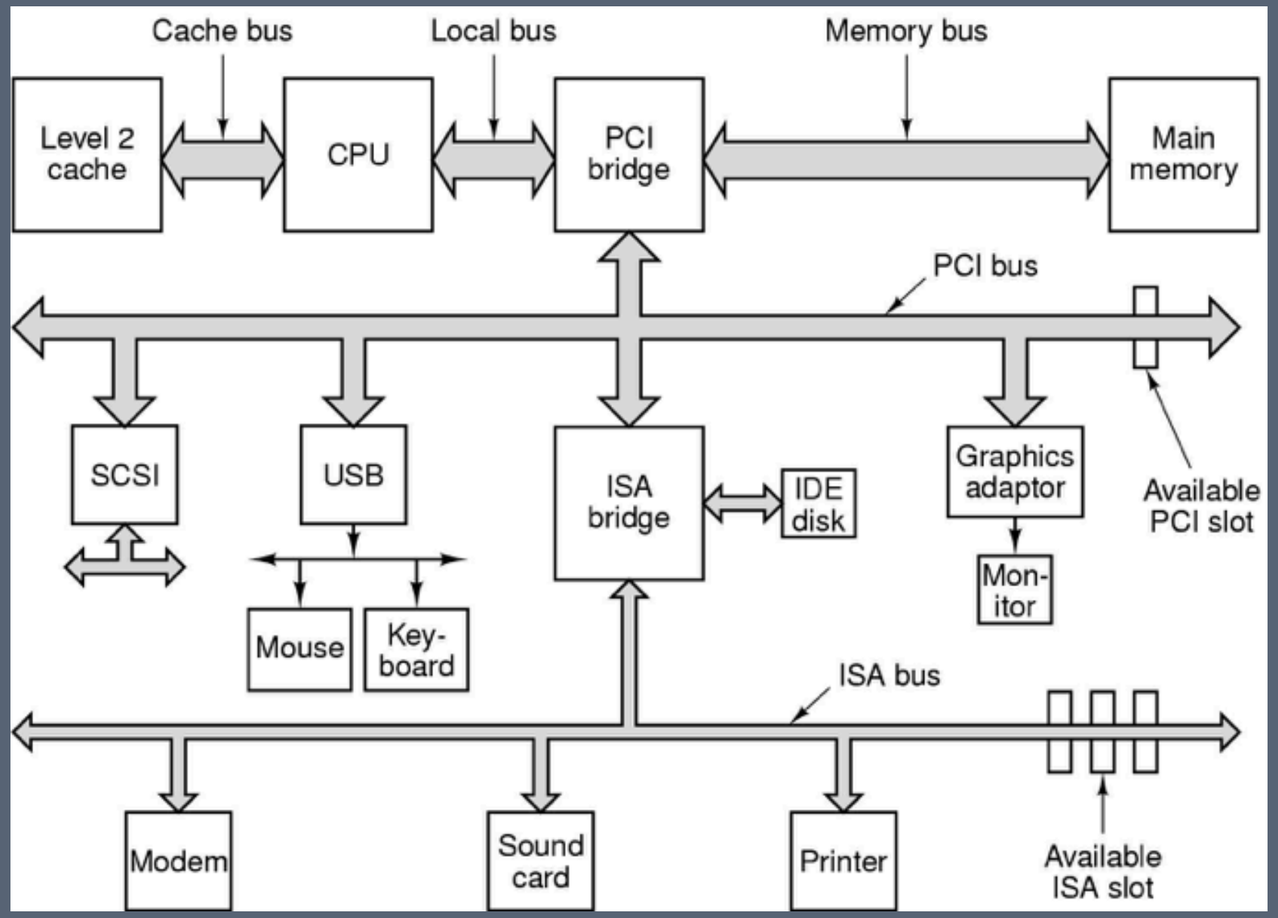
# INTERRUPCIONES



# DMA

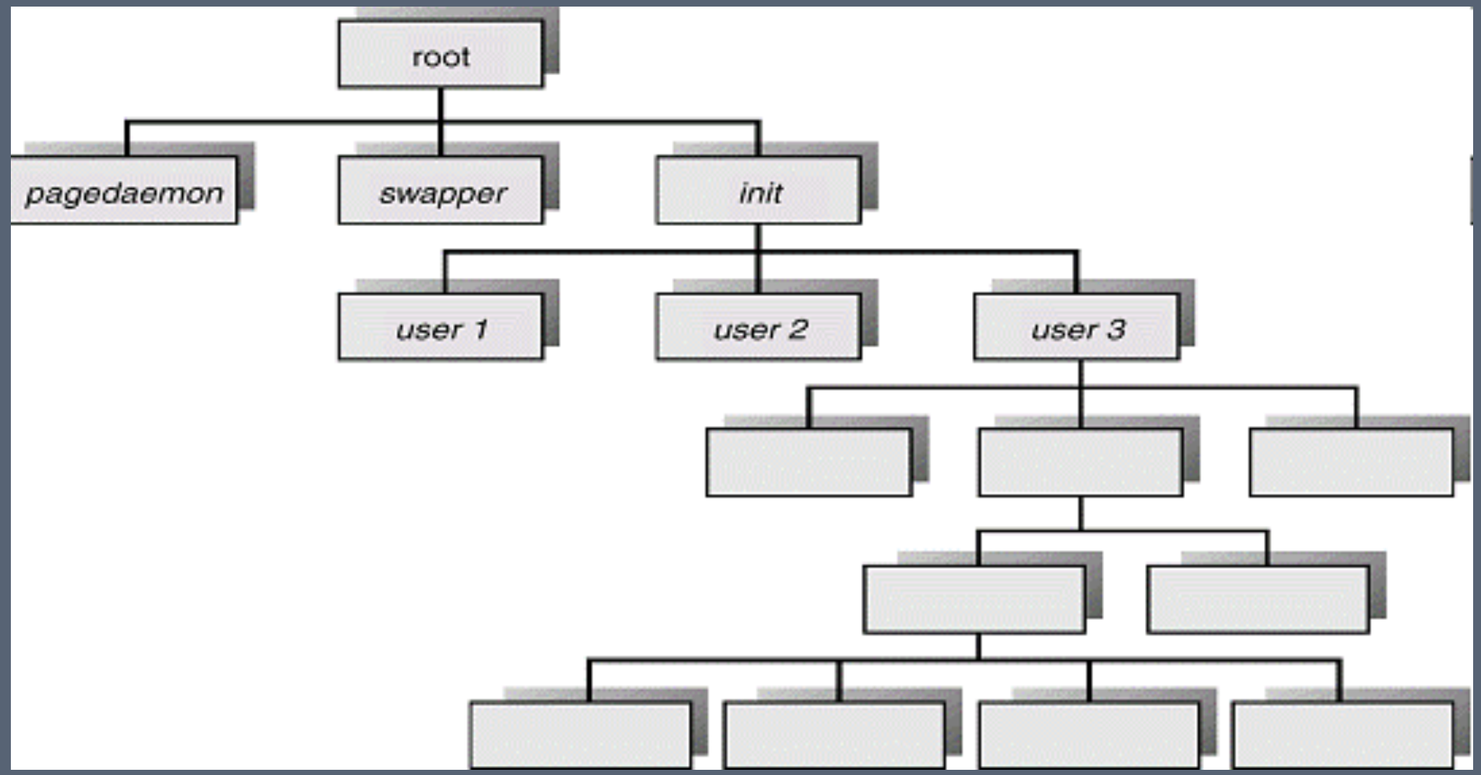

Drive
1. CPU programs the DMA controller
DMA controller
Disk controller
Main memory
CPU
Buffer
Address
Count
Control
4. Ack
5. Interrupt when done
2. DMA requests transfer to memory
3. Data transferred
Bus

# COMPLEJIDAD DEL HARDWARE



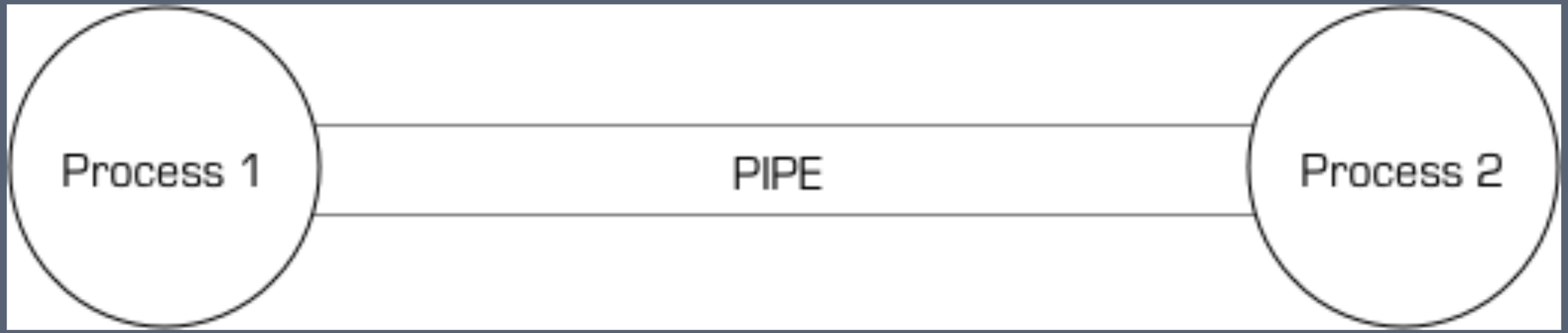
# CONCEPTOS

# PROCESOS

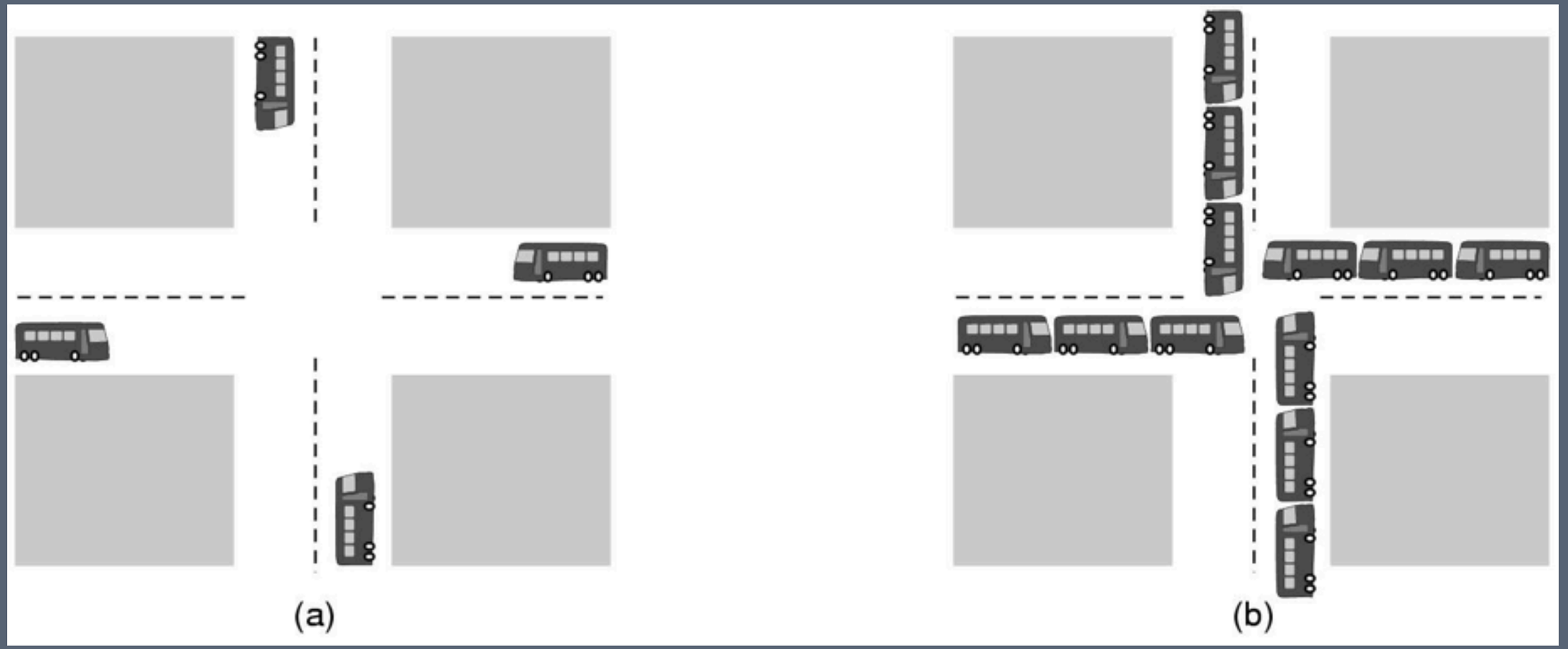



- ARBOL DE PROCESOS

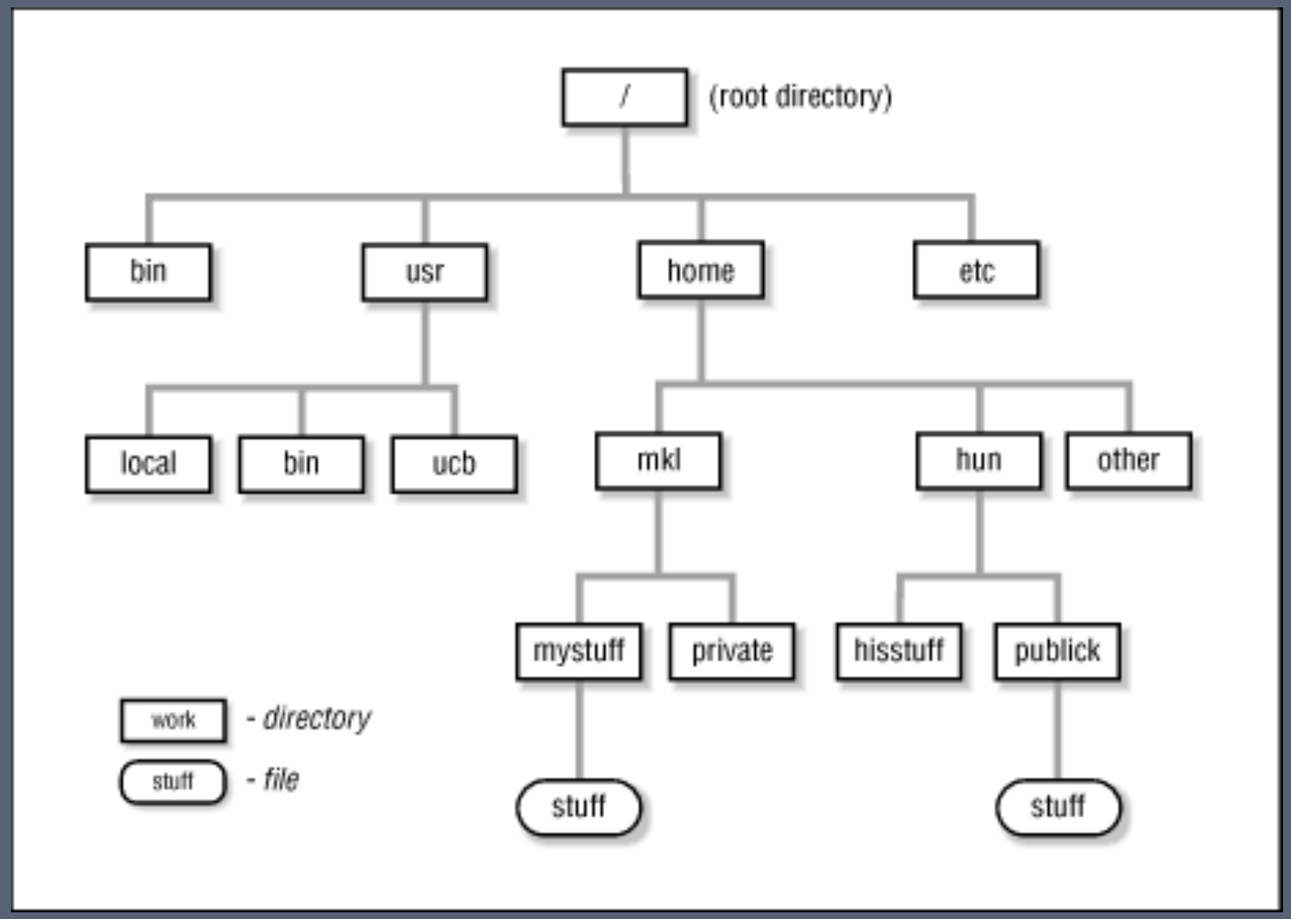
# COMUNICACIÓN ENTRE PROCESOS



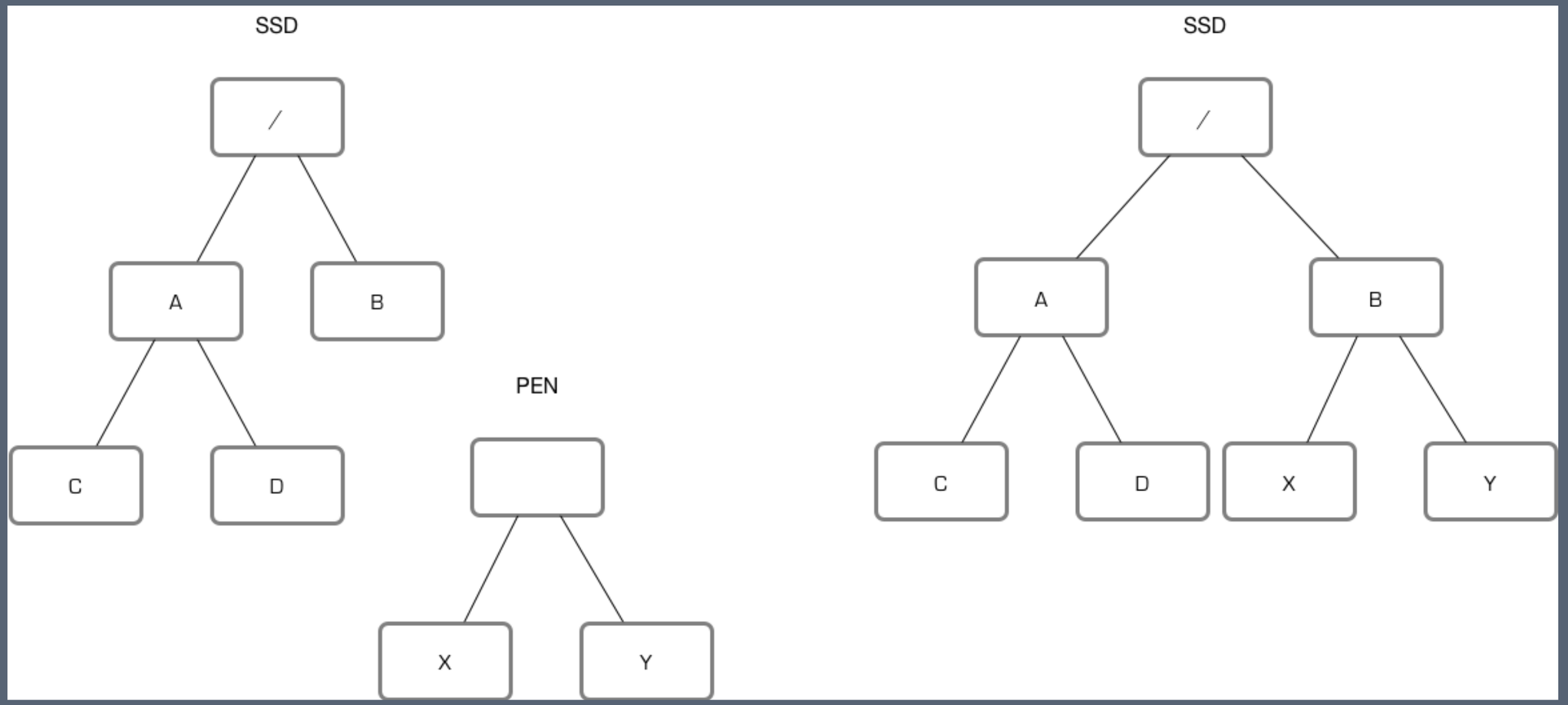
# DEADLOCKS



# SISTEMAS DE ARCHIVOS



# VOLUMENES Y PUNTOS DE MONTADO

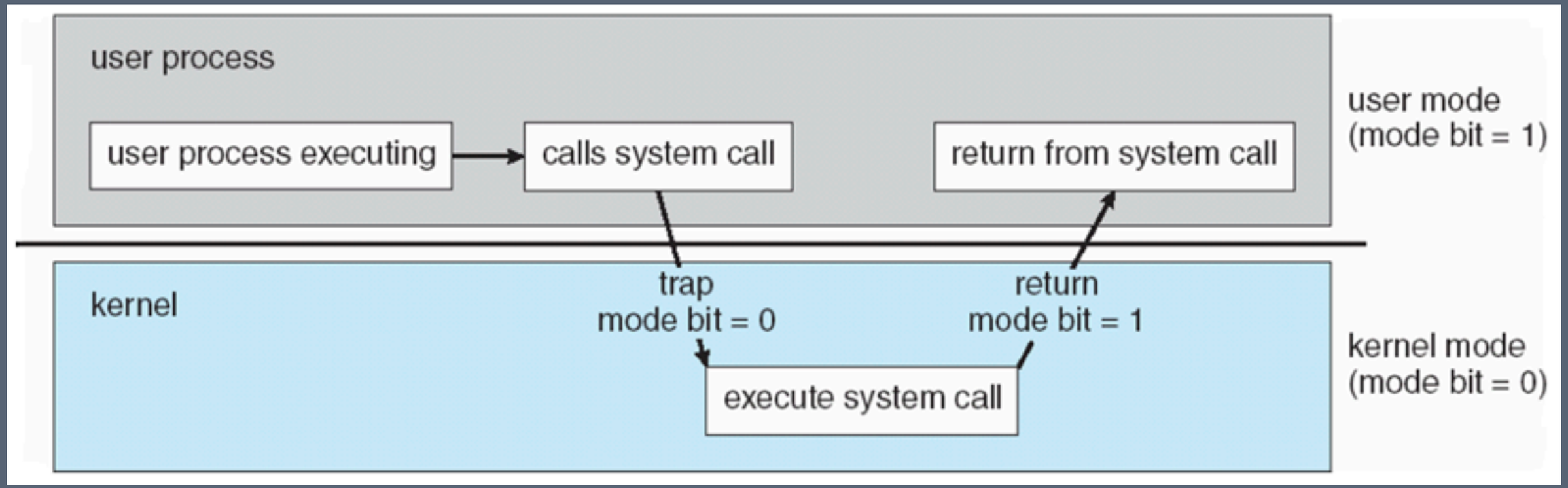


# MODO PROTEGIDO

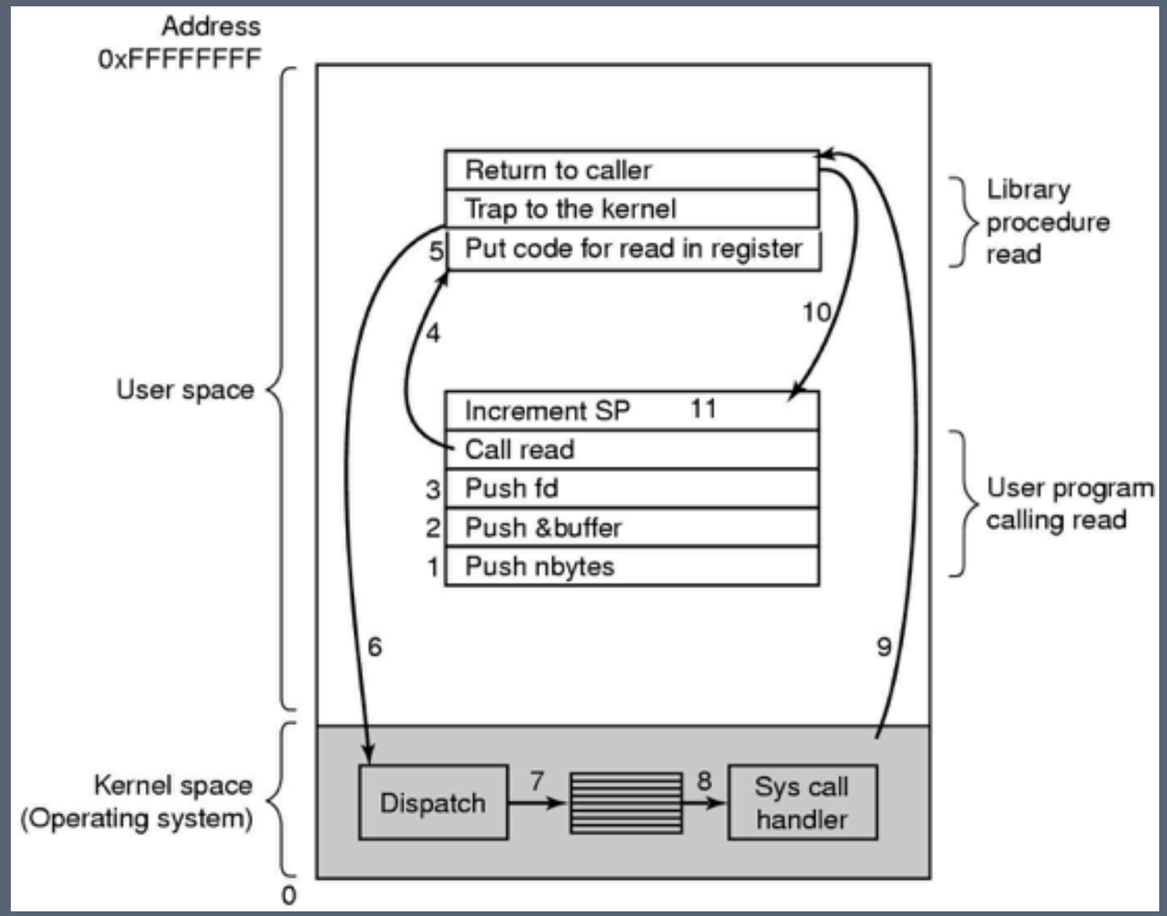
EL MODO DUAL DE EJECUCION PERMITE AL SO PROTEJERSE ASI MISMO Y OTROS COMPONENTES DEL SISTEMA

- 2 MODOS: USUARIO Y SUPERVISOR (KERNEL)
- EL BIT DE MODO PERMITE:
- DISTINGUIR CUANDO EL SISTEMA ESTÁ EJECUTANDO CÓDIGO DEL SISTEMA O USUARIO
- ALGUNAS INSTRUCCIONES DESIGNADAS COMO PRIVILEGIADAS, SOLO EJECUTAR EN MODO KERNEL
- LAS LLAMADAS AL SISTEMA CAMBIA A MODO KERNEL, CUANDO RETORNAN VUELVEN AL MODO USUARIO

# EJECUCIÓN EN MODO PROTEGIDO



# LLAMADAS AL SISTEMA



# AMINISTRACIÓN DE PROCESOS

| LLAMADA | DESCRIPCIÓN |
| --- | --- |
| `pid = fork()` | CREA UN NUEVO PROCESO IDÉNTICO AL HIJO |
| `pid = waitpid(pid, ...)` | ESPERA QUE UN PROCESO TERMINE |
| `s = execv(name, argv, env)` | REEMPLAZA LA IMAGEN DEL PROCESO |
| `exit(status)` | TERMINA LA EJECUCIÓN DE UN PROCESO |

# MANEJO DE ARCHIVOS

| LLAMADA | DESCRIPCIÓN |
| --- | --- |
| `fd = open(file, ...)` | ABRE UN ARCHIVO |
| `s = close(fd, ...)` | CIERRA UN ARCHIVO ABIERTO |
| `n = read(fd, buffer, nbytes)` | LEE DE UN ARCHIVO A UN BUFFER |
| `n = write(fd, buffer, nbytes)` | ESCRIBE DE UN BUFFER A UN ARCHIVO |
| `pos = lseek(fd, offset, whence)` | MUEVE EL PUNTERO |
| `s = stat(fd, &buf)` | RETORNA INFORMACIÓN DE ESTADO DE UN ARCHIVO |

# MANEJO DE ARCHIVOS

| LLAMADA | DESCRIPCIÓN |
| --- | --- |
| `s = mkdir(name, mode)` | CREA UN NUEVO DIRECTORIO |
| `s = rmdir(name)` | BORRA UN DIRECTORIO VACÍO |
| `s = link(name1, name2)` | CREA UNA NUEVA ENTRADA NAME2 APUNTANDO A NAME1 |
| `s = unlink(name)` | REMUEVE UNA ENTRADA |
| `s = mount(special, name, flag)` | MONTA UN SISTEMA DE ARCHIVOS |
| `s = umount(special)` | DESMOMTA UN SISTEMA DE ARCHIVOS |

# UN SHELL SIMPLIFICADO

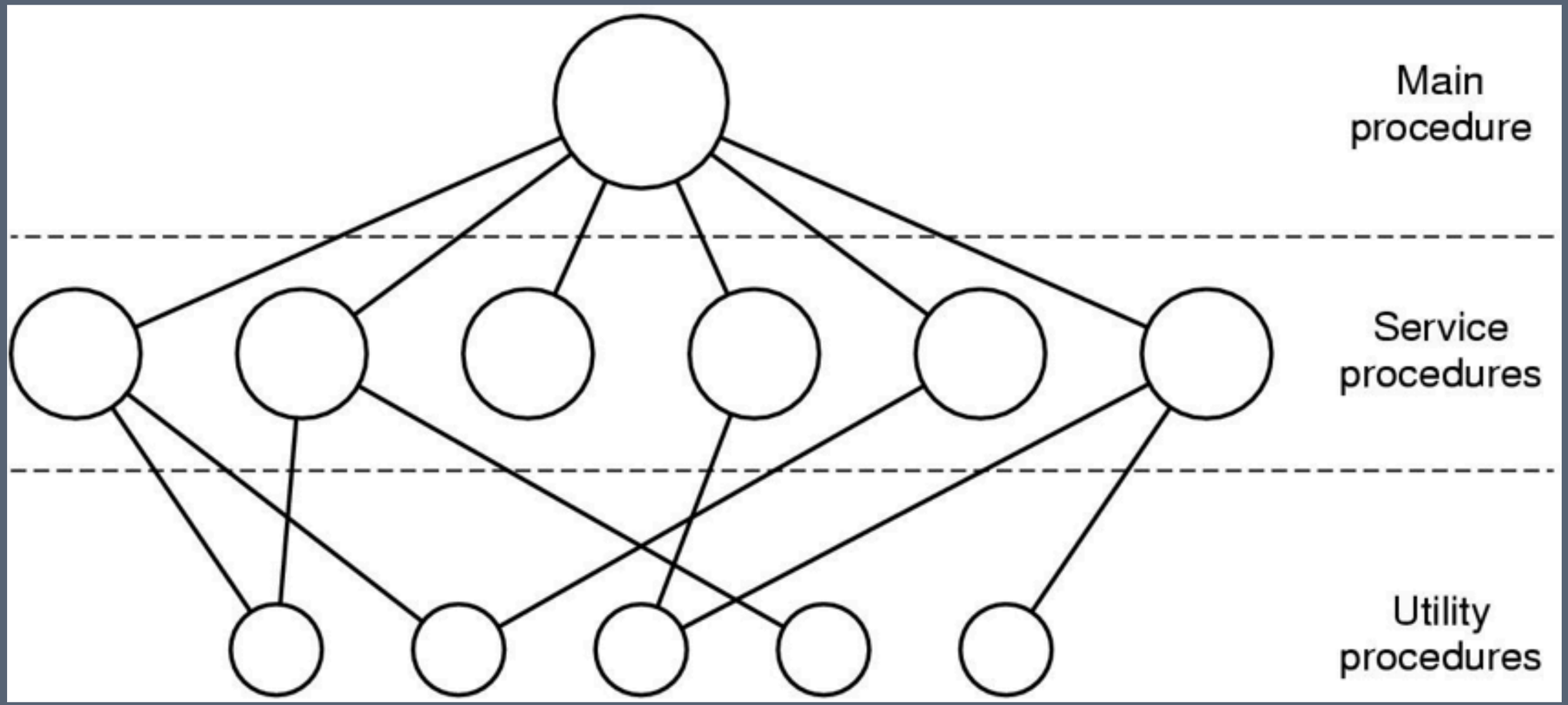
```
while (TRUE) {                                  /* repite por siempre */
    type_prompt( );                             /* mostrar el prompt */
    read_command(command, parameters)           /* ingreso desde la terminal */
    if (fork() != 0) {                          /* fork de un proceso */
        /* Parent code */
        waitpid( -1, &status, 0);               /* esperar finalización del hijo */
    } else {
      /* Child code */
      execve(command, parameters, 0);           /* ejecutar comando */
   }
}
```

# LLAMADAS AL SISTEMA WIN32

| UNIX | WIN32 | DESCRIPCIÓN |
|---|---|---|
| fork() | CreateProcess | CREA UN NUEVO PROCESO IDÉNTICO AL HIJO |
| waitpid | WaitForsingleObject | ESPERA QUE UN PROCESO TERMINE |
| execv | - | REEMPLAZA LA IMAGEN DEL PROCESO |
| exit | ExitProcess | TERMINA LA EJECUCIÓN DE UN PROCESO |
| open | CreateFile | ABRE UN ARCHIVO |
| close | CloseHandle | CIERRA UN ARCHIVO ABIERTO |
| read | ReadFile | LEE DATOS DE UN ARCHIVO |
| write | WriteFile | ESCRIBE DATOS A UN ARCHIVO |
| lseek | SetFilePointer | MUEVE EL PUNTERO |
| ... | ... | ... |

# ESTRUCTURA DE LOS SISTEMAS OPERATIVOS

# SISTEMA MONOLÍTICO



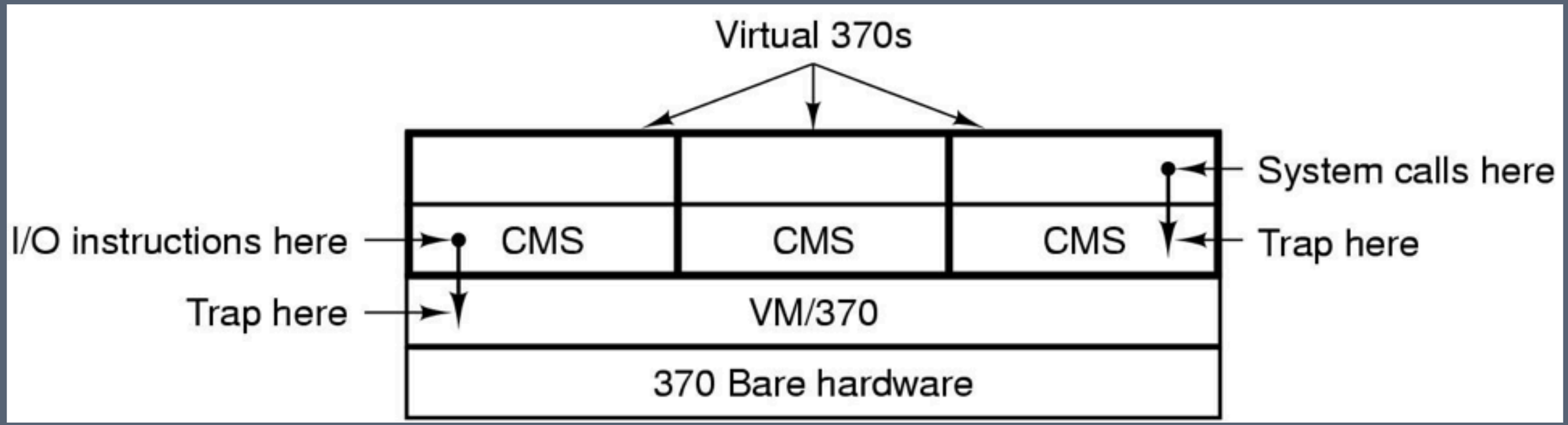
- LOS PROGRAMAS DE APLICACION INVOCAN LOS SERVICIOS DEL SISTEMA
- TIENEN UN CONJUNTO DE SERVICIOS DEL SISTEMA IMPLEMENTAN LAS LLAMADAS AL SISTEMA
- UN CONJUNTO DE PROCEDIMIENTOS DE UTILIDAD QUE AYUDAN A LOS SERVICIOS DEL SISTEMA

# ESTRUCTURA EN CAPAS

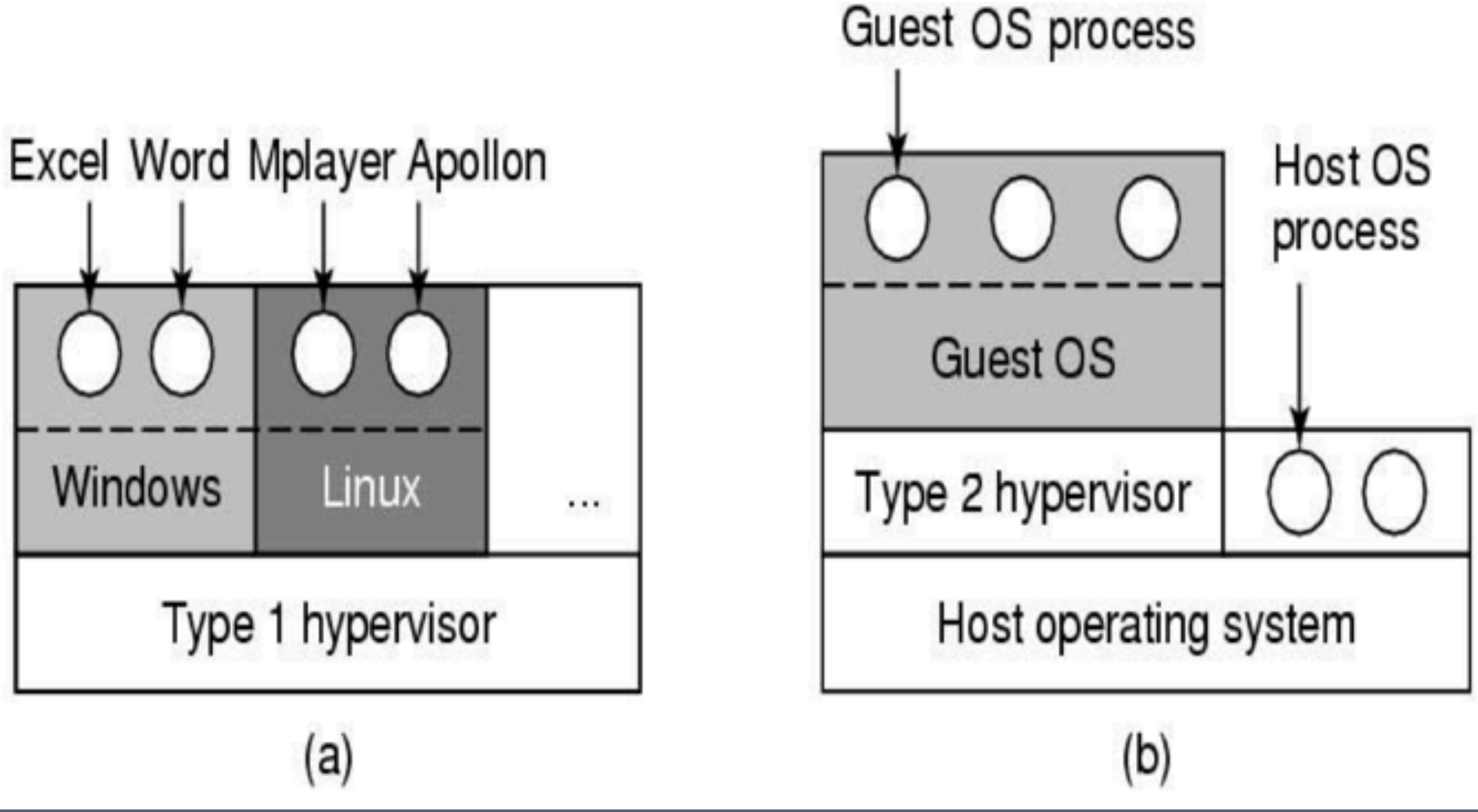
| LAYER | FUNCIÓN |
| --- | --- |
| 5 | EL OPERADOR |
| 4 | PROGRAMAS DEL USUARIO |
| 3 | MANEJO DE ENTRADA/SALIDA |
| 2 | COMUNICACIÓN ENTRE PROCESOS |
| 1 | MANEJO DE MEMORIA |
| 0 | ASIGNACIÓN DE PROCESADOR Y MULTIPROGRAMACIÓN |

- EL SISTEMA OPERATIVO ESTA DIVIDIDO EN UN NUMERO DE CAPAS (NIVELES), CADA UNA CONSTRUIDO SOBRE LAS CAPAS – INFERIORES
- LA CAPA INFERIOR (CAPA 0) ES EL HARDWARE; LA CAPA MÁS ALTA (CAPA N) ES LA INTERFAZ DEL USUARIO
- EN UN SISTEMA MODULAR, LAS CAPAS SON ELEGIDAS DE FORMA TAL QUE CADA CAPA USA SOLO LAS FUNCIONES Y SERVICIOS DE LAS CAPAS INFERIORES

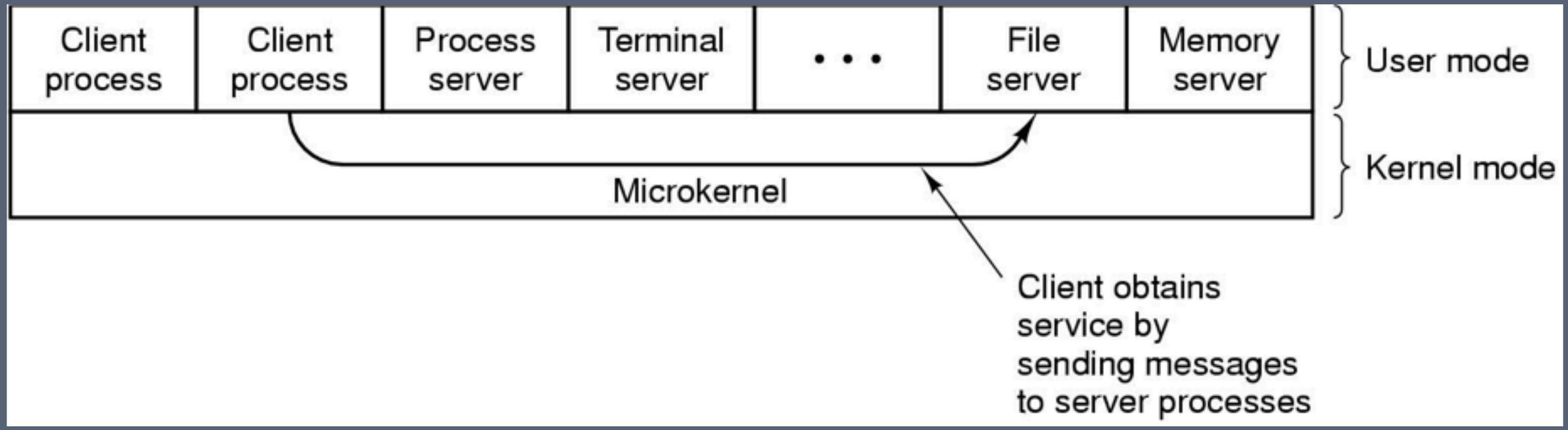
# MÁQUINAS VIRTUALES



- UNA MAQUINA VIRTUAL TIENE UNA APROXIMACION DE UN SISTEMA EN CAPAS RESPECTO DE SU SIGUIENTE CAPA LÓGICA. TRATA AL SISTEMA OPERATIVO Y AL HARDWARE COMO SI FUESEN HARDWARE
- UNA MÁQUINA VIRTUAL BRINDA UNA INTERFAZ IDÉNTICA AL HARDWARE SUBYACENTE
- EL SISTEMA OPERATIVO HOST CREA LA ILUSIÓN QUE CADA PROCESO TIENE SU PROPIO PROCESADOR Y MEMORIA
- A CADA HUÉSPED SE LE BRINDA UNA COPIA VIRTUAL DE LA COMPUTADORA SUBYACENTE


Excel
Word
Mplayer
Apollon
Windows
Linux
...
Type 1 hypervisor
(a)
Guest OS process
Host OS process
Guest OS
Type 2 hypervisor
Host operating system
(b)

# MICROKERNEL

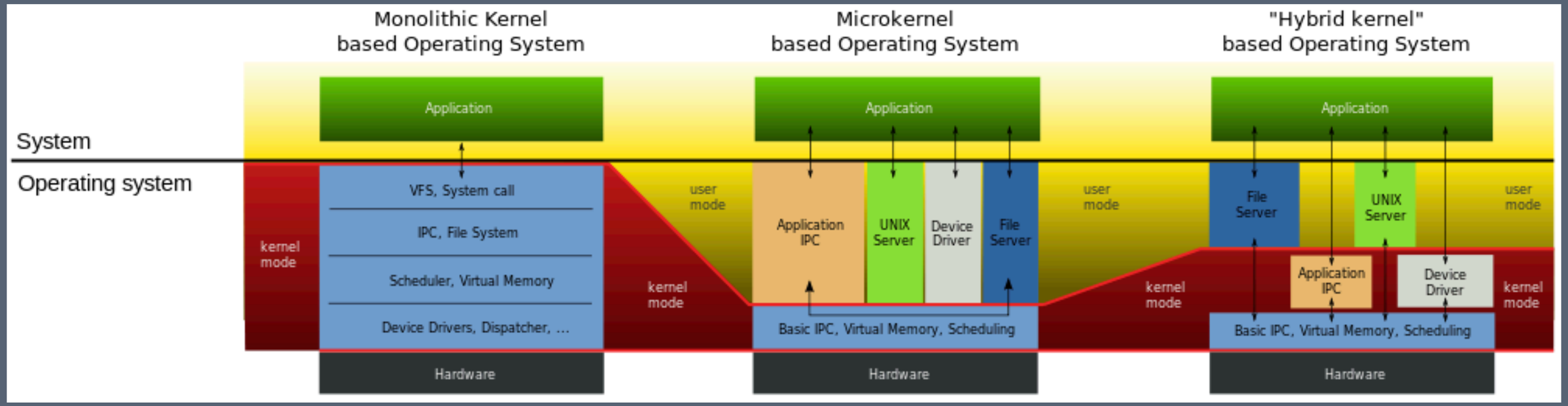


- MUEVE LA MAYOR CANTIDAD DE FUNCIONALIDAD POSIBLE DEL KERNEL AL ESPACIO DE "USUARIO"
- SOLO ALGUNAS FUNCIONES ESENCIALES QUEDAN EN EL KERNEL: MANEJO PRIMITIVO DE MEMORIA (ESPACIO DE DIRECCIONES), E/S Y MANEJO DE INTERRUPCIONES, COMUNICACIÓN ENTRE PROCESOS (IPC), PLANIFICACIÓN BÁSICA
- OTROS SERVICIOS DEL SO SON PROVISTOS POR PROCESOS QUE EJECUTAN EN ESPACIO DEL USUARIO:
DRIVERS DE DISPOSITIVOS, SISTEMA DE ARCHIVOS, MEMORIA VIRTUAL

# LA COMUNICACION ENTRE LOS MODULOS DEL USUARIO SE HACE MEDIANTE PASAJE DE MENSAJES

- BENEFICIOS:
- MÁS FÁCIL DE EXTENDER
- MÁS FÁCIL DE PORTAR
- MÁS SÓLIDO
- MÁS SEGURO
- DESVENTAJAS:
- HAY UN OVERHEAD DE PEFORMANCE POR LA COMUNICACIÓN ENTRE EL ESPACIO DE USUARIO Y ESPACIO DE KERNEL

# COMPARACIÓN DE ESTRUCTURAS



# ARBOL GENEALOGICO