



UNIVERSIDAD
DE GRANADA

Tema 2: Introducción a los sistemas operativos

Fundamentos del Software

1º Grado en Informática

Rosana Montes - rosana@ugr.es

Objetivos

- Conocer la evolución de los Sistemas Operativos (SO) y sus principales logros.
- Funciones de un SO.
- Conocer los elementos necesarios para implementar la multiprogramación en un SO.
- Entender el concepto de proceso y su
- Entender el concepto de hebra (hilo).
- Conocer el uso que realiza el SO de apoyo al hardware CPU y gestión de memoria.

2.1 Introducción: Funciones de los SO

- Una plataforma hardware es un conjunto de recursos de procesamiento (procesador, memoria, caché, registros, etc) y de dispositivos de E/S.
- El SO es una representación uniforme y abstracta de los recursos de un ordenador que son requeridos por las aplicaciones.
- El SO es una interfaz rica entre las aplicaciones y el hardware, que permite compartir recursos y la protección de éstos. Un SO es un programa que actúa de intermediario entre el usuario y el hardware. No siempre ha existido.
- Características deseables: comodidad, eficiencia y capacidad de evolución.
- Función: SO como máquina virtual y gestor de recursos.
- Abstracciones:
 1. Proceso: dedicación de la CPU a un programa
 2. Memoria virtual: capacidad de RAM infinita.
 3. Archivo: datos (con tipo) en memoria permanente.
 4. Canal: E/S asociada a un tipo de dato.
 5. Shell: potente interfaz de usuario programable.

Abstracción

El SO como interfaz Usuario/Computadora

Presenta al usuario una máquina abstracta más fácil de programar que el hardware subyacente:

- Oculta la complejidad del hardware.
- Da tratamiento homogéneo a diferentes objetos de bajo nivel (archivos, procesos, dispositivos, etc.).

Un programador de aplicaciones desarrolla más fácilmente una aplicación en un lenguaje de programación de alto nivel (que accede por sí misma a la API del SO) que en lenguaje máquina (lo que entiende el hardware).

APIs para programación

El SO como interfaz Usuario/Computadora

Un SO proporciona normalmente utilidades en las siguientes áreas:

- **Desarrollo de software:** editores de texto, compiladores, enlazadores, depuradores de memoria y de código.
- **Ejecución de programas:** cargador de programas y ejecución de éstos.
- **Acceso a dispositivos de E/S:** conjunto específico de instrucciones para cada dispositivo. Las llamadas al sistema resuelven estos accesos.

Seguridad

El SO como interfaz Usuario/Computadora

- **Acceso al sistema** En sistemas compartidos o públicos, el SO controla el acceso y uso de los recursos del sistema: Shell, permisos, propietario de archivo, grupos.
- **Detección y respuesta a errores** Tratamiento de errores a nivel software (*exceptions*) y hardware (*interruptions*), minimizando impacto (ej. pantallazo azul).
- **Contabilidad** Estadísticas de uso de los recursos y medida del rendimiento del sistema (*benchmark*)

Excepciones [Stal05] (pp.34-37)

Evento síncrono inesperado generado por alguna condición que ocurre durante la ejecución de una instrucción (ejemplo, desbordamiento aritmético, dirección inválida, instrucción privilegiada, etc.)

Administrador de Recursos

El SO requiere de un mecanismo de control:

- Las funciones del SO actúan de la misma forma que el resto del software, es decir, son programas ejecutados por el procesador.
- El SO cede el control y depende del procesador para volver a retomarlo.

Por lo tanto:

- El SO dirige el uso del procesador y el resto de los recursos.
- Controla la temporalización de la ejecución de los programas.
- Parte del código del SO se encuentra cargado en la Memoria Principal (*kernel* y, en ciertos momentos, otros servicios del SO que se estén usando). El resto de la memoria está ocupada por programas y datos de usuario.

Administración de Recursos

Por lo tanto: (cont.)

- La asignación de la memoria principal la realizan conjuntamente el SO y el hardware de gestión de memoria del procesador (MMU).
- El SO decide cuándo un programa en ejecución puede usar un dispositivo de E/S y también el acceso y uso de los ficheros.
- El procesador es también un recurso (uno de los más importantes).

2.1 Introducción: Evolución de los SO

- 40-50's - Procesamiento en serie de programas en lenguaje máquina.
STDin = tarjeta perforada. STDout=impresora.
- 50-60's - Sistemas por lotes (batch). Primer SO el monitor residente.
Poca interacción. Infrautilización de la CPU.
- 60-70's - Sistemas multiprogramados interactivos.
 - Aparece una facilidad hardware: la
 - Definir:
 - Los SO evolucionan a Tiempo Compartido.
 - Los SO evolucionan a
- 70-xx's Aparecen los sistemas multiusuario.

2.1 Introducción: Evolución de los SO

Clasificación:

- Según el modo de trabajo:
por lotes (batch) vs interactivo
- Según el aprovechamiento de la CPU:
uniprogramado vs multiprogramado
uniprocесador vs multiprocesador
- Según el usuario:
monousuario vs multiusuario

Proyecto en Prado2

Busca un SO, clasifícalo e incluye una captura de su interfaz gráfica (GUI).

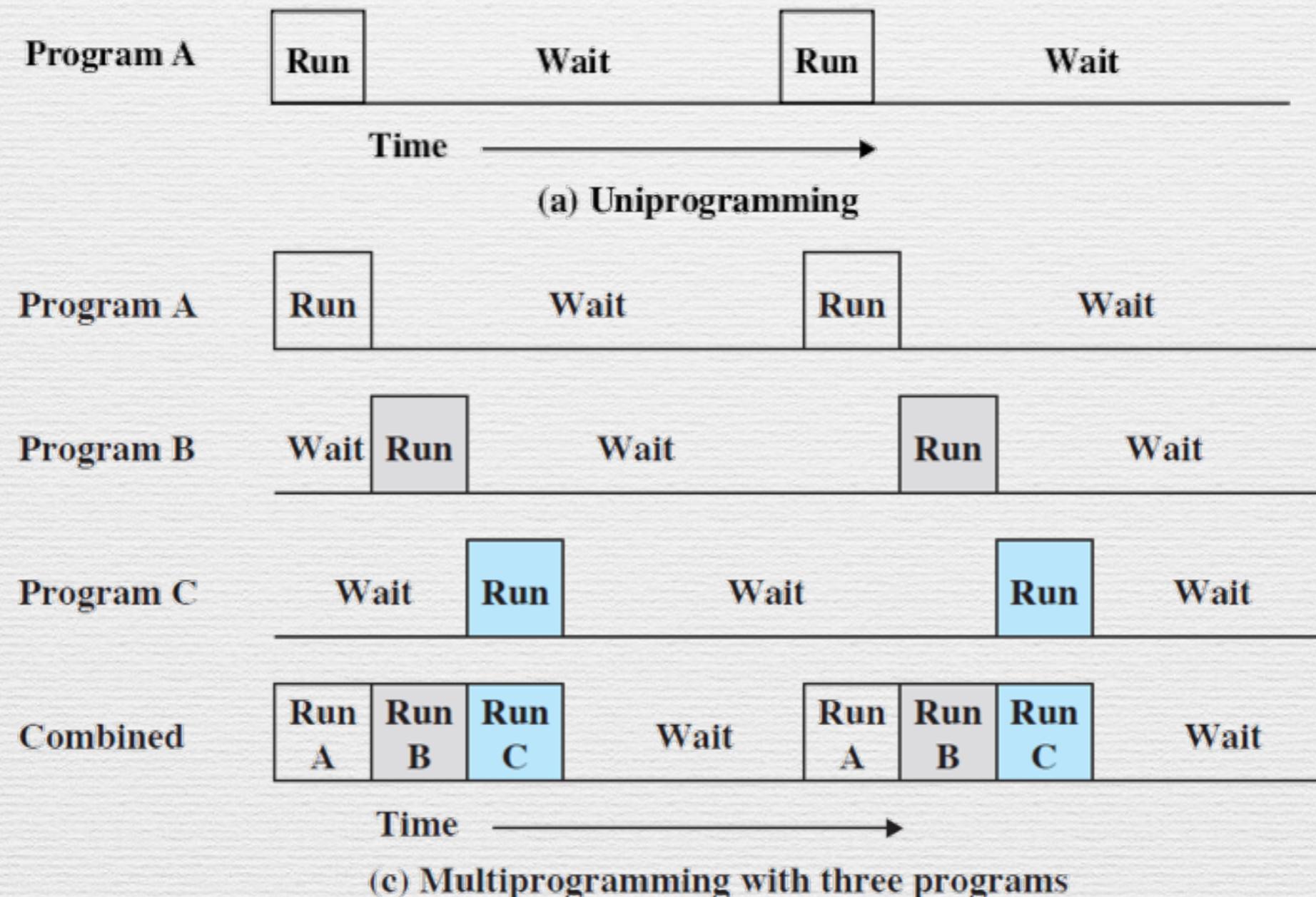
2.2 Componentes de un SO multiprogramado

Técnicas de E/S [Stal05] (pp.34-37) [Stal12] (pp.51-52)

- Un buen SO permite un uso eficiente del procesador y de los dispositivos E/S a través de diversas técnicas.
- Soporte hardware para los SO:
 - Modo dual de operación.
 - Interrupciones (evento).
 - Excepciones (evento)
 - Protección: E/S, Memoria, CPU.
- Los primeros SO eran simples sistemas MP dispone de MP puede cargarse más de una aplicación simultáneamente.
- Es necesario un algoritmo de

2.2 Componentes de un SO multiprogramado

Concepto de multiprogramación [Stall05] (pp. 58-67) [Stall12] (pp. 72-79)



Multiprogramación

	Job 1	Job 2	Job 3
<i>Tipo de trabajo</i>	<i>Computación pesada</i>	<i>Gran cantidad de E/S</i>	<i>Gran cantidad de E/S</i>
<i>Duración</i>	<i>5 minutos</i>	<i>15 minutos</i>	<i>10 minutos</i>
<i>Memoria requerida</i>	<i>50 MB</i>	<i>100 MB</i>	<i>75 MB</i>
<i>¿Necesita disco?</i>	<i>NO</i>	<i>NO</i>	<i>SI</i>
<i>¿Necesita terminal?</i>	<i>NO</i>	<i>SI</i>	<i>NO</i>
<i>¿Necesita impresora?</i>	<i>NO</i>	<i>NO</i>	<i>SI</i>

Job1 utiliza mucho la CPU,

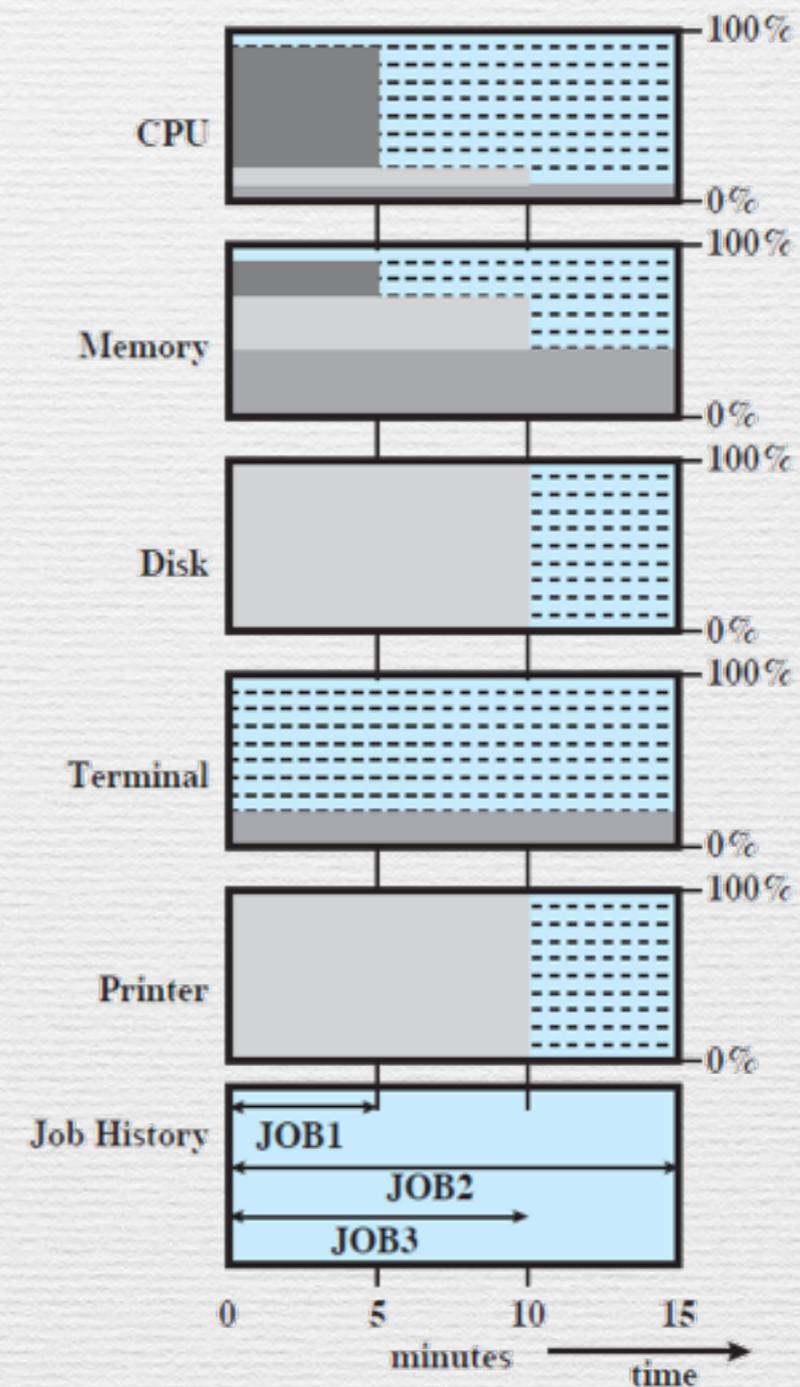
Job2 y Job3 utilizan mucho los periféricos de E/S.

Multiprogramación



(a) Uniprogramming

14



(b) Multiprogramming

Multiprogramación

Comparamos resultados con ambas técnicas

	Monoprogramación	Multiprogramación
<i>Uso del procesador</i>	20%	40%
<i>Uso de memoria</i>	33%	67%
<i>Uso de disco</i>	33%	67%
<i>Uso de impresora</i>	33%	67%
<i>Tiempo transcurrido</i>	30 minutos	15 minutos
<i>Productividad</i>	6 trabajos/hora	12 trabajos/hora
<i>Tiempo de respuesta medio</i>	18 minutos	10 minutos

Mejora: Si multaneamos varios trabajos en la CPU

Proceso

Concepto de proceso [Stal05] (pp. 68-71) [Stal12] (pp. 82-84,128-129)

- Un programa en ejecución.
- Una instancia de un programa ejecutándose en un ordenador
- La entidad que se puede asignar o ejecutar en un procesador.
- Una unidad de actividad caracterizada por un solo flujo de ejecución, un estado actual y un conjunto de recursos del sistema asociados.
- Un proceso está formado por:
 - ♦ Un programa ejecutable (también llamado *binario*).
 - ♦ Datos que necesita el SO para ejecutar el programa.

Proceso

PCB, Process Control Blocks **Figura 3.1**
[Stall05] (pp. 108-120) [Stal12] (pp. 82-84,128-130)

- El PCB es un conjunto de información, que incluye:
- **Identificador de proceso.** PID (*Process IDentifier*)
- **Estado.** En que situación se encuentra el proceso en cada momento (modelo de estados). Ej. en ejecución.
- **Prioridad.** Nivel propio frente a otros procesos.
- **Memoria.** Donde reside el programa y sus datos.
- **Contexto de ejecución:** Registros actuales del procesador.
- **Estado E/S.** Recursos del sistema que le han sido asignados.
- **Cuota.** Posibles limitaciones a los recursos.

Identificador
Estado
Prioridad
Contador de programa
Punteros de memoria
Datos de contexto
Información de estado de E/S
Información de auditoría
.

2.2 Componentes de un SO multiprogramado

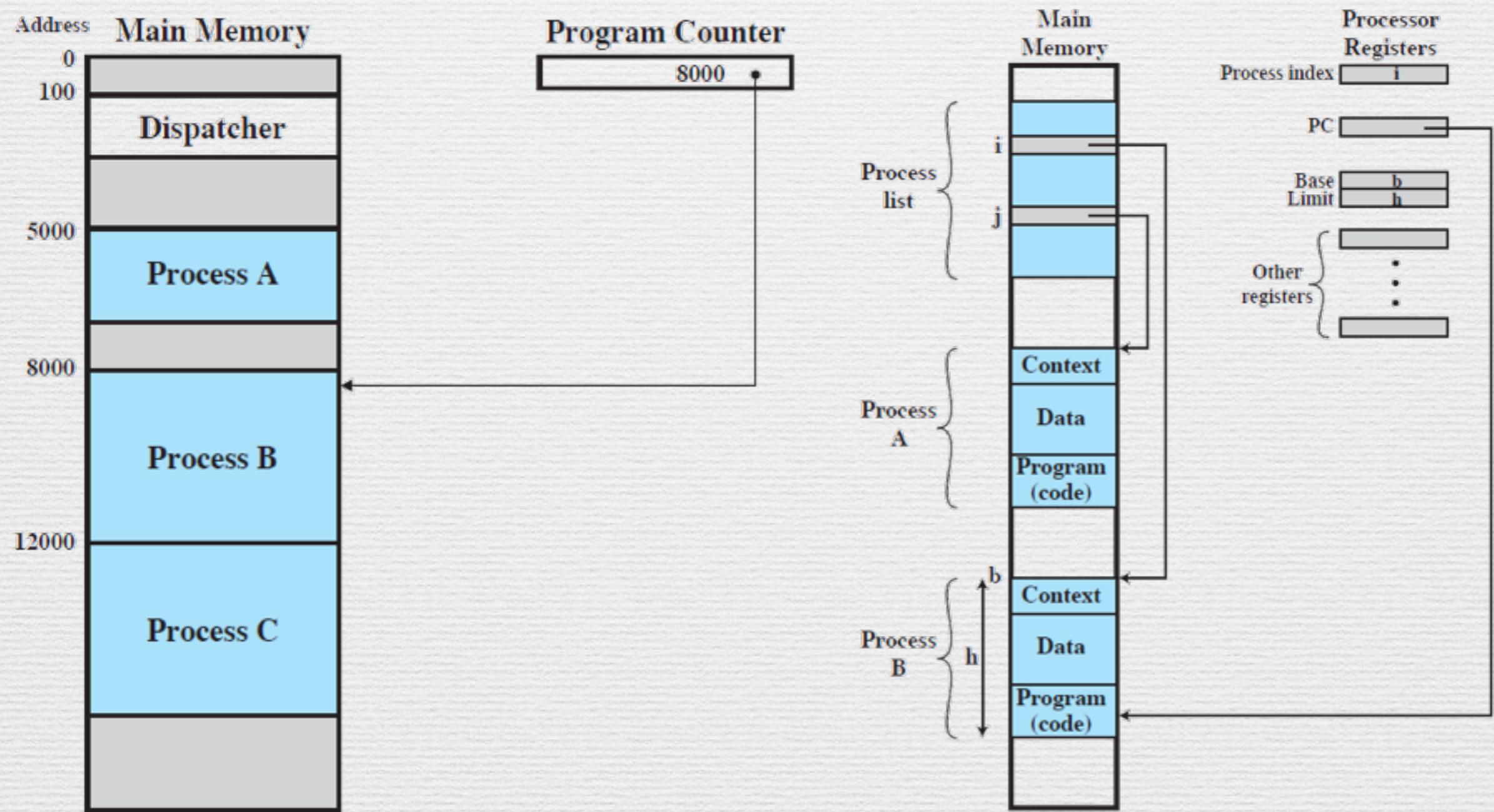
Implementación de Procesos Típica

Fig. 2.8. [Stal05] (pp. 70) [Stal12] (pp. 85)

- Esta implementación permite ver al **proceso** como una **estructura de datos**.
- El **estado** completo del proceso en un instante dado se almacena en su **contexto**.
- El **contexto** del procesador se refiere a los registros PC, PSW y SP.
- Esta estructura permite el desarrollo de técnicas potentes que aseguren la **coordinación** y la **cooperación** entre los **procesos**.
- Es necesario un algoritmo que resuelva la coordinación y cooperación de procesos (*scheduling algorithm / dispatcher*). Existen técnicas de **planificación** de la CPU.
- El cambio de proceso puede ocurrir en cualquier instante en que el SO obtiene el control sobre el proceso en ejecución.
- **Quantum:** tiempo máximo de uso del procesador.

2.2 Componentes de un SO multiprogramado

Disposición de memoria (*memory layout*)



Traza

Traza de ejecución [Stal12] (pp. 130-134)

- ♦ Una traza de ejecución es un listado de la secuencia de instrucciones de programa que realiza el procesador para un proceso.
- ♦ Desde el punto de vista del procesador se entremezclan las trazas de ejecución de los procesos y las trazas del código del sistema operativo.
- ♦ ¿Como consecuencia de qué situaciones se entremezclan las trazas de los procesos y las trazas del SO?
 - ♦ Llamadas al sistemas.
 - ♦ Multiprogramación.

Ejemplo de traza de ejecución

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000 = Starting address of program of Process A

8000 = Starting address of program of Process B

12000 = Starting address of program of Process C

100 = Starting address of dispatcher program

Shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

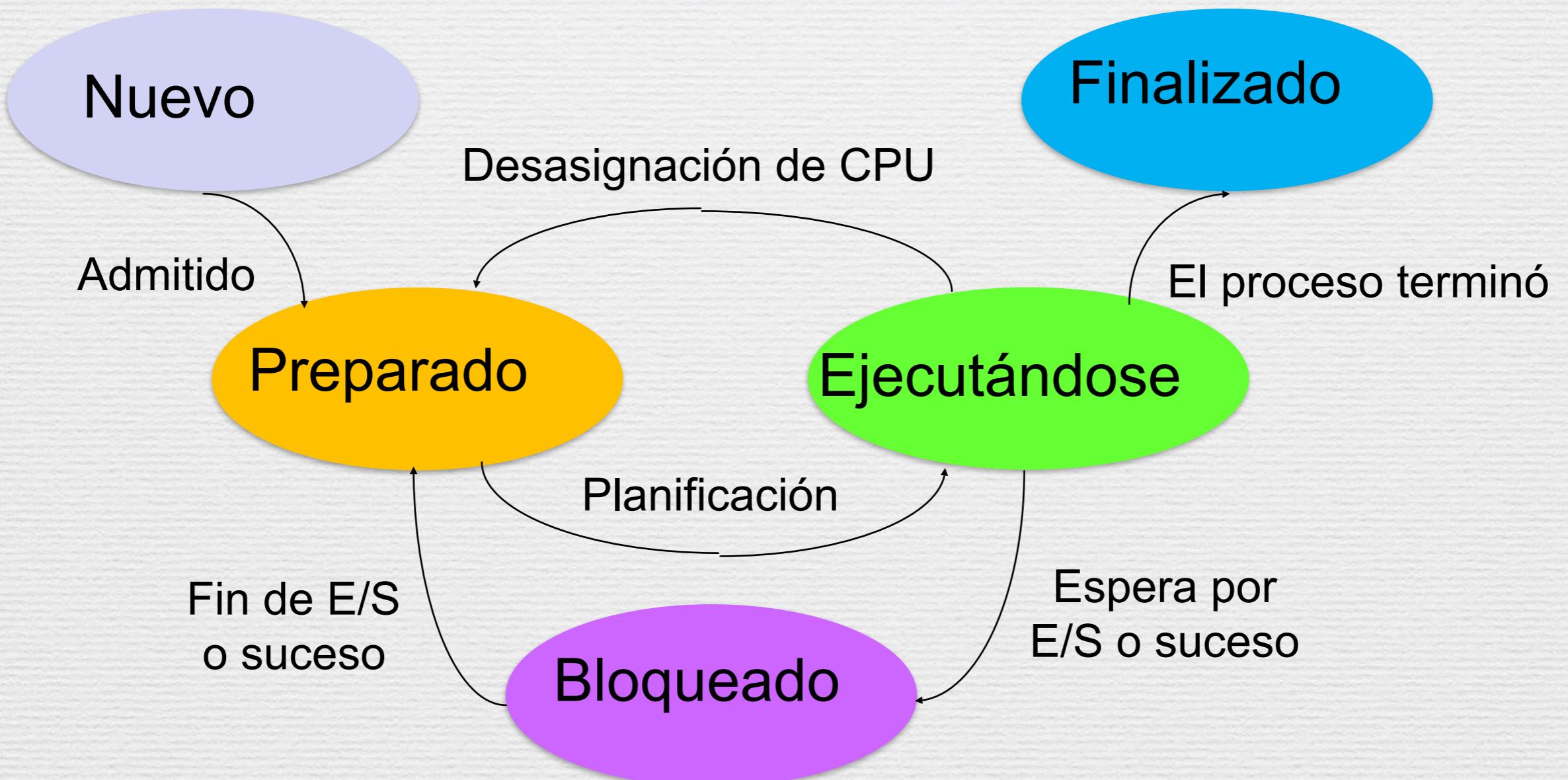
1	5000			27	12004
2	5001			28	12005
3	5002				----- Timeout
4	5003			29	100
5	5004			30	101
6	5005			31	102
7	100			32	103
8	101			33	104
9	102			34	105
10	103			35	5006
11	104			36	5007
12	105			37	5008
13	8000			38	5009
14	8001			39	5010
15	8002			40	5011
16	8003				----- Timeout
17	100			41	100
18	101			42	101
19	102			43	102
20	103			44	103
21	104			45	104
22	105			46	105
23	12000			47	12006
24	12001			48	12007
25	12002			49	12008
26	12003			50	12009
				51	12010
				52	12011
					----- Timeout

Objetivo: Simultanear varios trabajos en la CPU

Modelo 5 estados

- El modelo de cinco estados trata de representar las actividades que el SO lleva a cabo sobre los procesos:
 - Creación
 - Terminación
 - Multiprogramación
- Para ello hace uso de cinco estados:
 - ◆ Ejecutándose
 - ◆ Preparado (listo para ejecutarse)
 - ◆ Bloqueado o Esperando
 - ◆ Nuevo
 - ◆ Finalizado

Modelo 5 estados



El SO maneja varias
listas de ...

Transiciones

- **Nuevo → Preparado.** El PCB está creado y el programa está disponible en memoria.
- **Ejecutándose → Finalizado.** El proceso finaliza normalmente o es abortado por el SO a causa de un error no recuperable.
- **Preparado → Ejecutándose.** El SO (planificador CPU) selecciona un proceso para que se ejecute en el procesador.
- **Ejecutándose → Bloqueado.** El proceso solicita algo al SO por lo que debe esperar.
- **Ejecutándose → Preparado.** Un proceso ha alcanzado el máximo tiempo de ejecución ininterrumpida.
- **Bloqueado → Preparado.** Se produce el evento por el cual el SO bloqueó al proceso.
- **Preparado / Bloqueado → Finalizado.** Terminación de un proceso por parte de otro.

2.3 Descripción y control de procesos

Un proceso se describe mediante el **PCB**

- **Identificadores:** Del proceso, del padre del proceso, del usuario, ...
- **Contexto** de registros del procesador: **PC, PSW, SP, ...**
- **Información** para **control** del **proceso**:
 - **Estado** del proceso (según modelo de 5 estados).
 - Parámetros de **planificación**.
 - **Evento** que mantiene al proceso bloqueado.
 - Cómo acceder a la **memoria** que aloja el programa asociado al proceso: registros base y longitud.
 - **Recursos** utilizados por el proceso (dispositivos, puertos, etc.).

Creación

Creación e inicialización del PCB
[Stal05] (pp. 135-143) [Stal12] (pp. 154-159)

- Asignar **identificador** único al **proceso**
- Asignar un nuevo **PCB**
- Asignar **memoria** para el programa asociado
- **Iniciar PCB:**
 - **PC:** Dirección inicial de comienzo del programa
 - **SP:** Dirección de la pila de sistema
 - **Memoria** donde reside el programa
 - El resto de **campos** se inicializan a valores por omisión

PID
<i>Estado</i>
Prioridad
<i>Contador de programa</i>
Punteros de memoria
<i>Datos de contexto</i>
Información de estado de E/S
<i>Información de auditoría</i>

2.3 Descripción y control de procesos

Control de procesos: (1) cambio de proceso

- Un proceso en estado **Ejecutándose** cambia a otro estado y un proceso en estado **Preparado** pasa a estado **Ejecutándose**
- ¿Cuándo puede realizarse? Cuando el SO pueda ejecutarse y decida llevarlo a cabo. Luego solamente como resultado de:
 - Una interrupción: dispositivo ES listo; quantum *timeout*.
 - Una excepción: Evento síncrono inesperado(ej. desbordamiento aritmético, dirección inválida, instrucción privilegiada, etc.).
 - Una llamada al sistema (ej. modificación del reloj).

Cambio de contexto

- 1) Salvar los registros del procesador en el PCB del proceso que actualmente está en estado **Ejecutándose**.
- 2) Actualizar el campo estado del proceso al nuevo estado al que pasa e insertar el PCB en la cola de bloqueados o bloqueados por eventos.
- 3) Seleccionar un nuevo proceso del conjunto de los que se encuentran en estado **Preparado** (*Scheduler* o Planificador de CPU).
- 4) Actualizar el estado del proceso seleccionado a **Ejecutándose** y sacarlo de la lista de preparados.
- 5) Cargar los registros del procesador con la información de los registros almacenada en el PCB del proceso seleccionado.

2.3 Descripción y control de procesos

Control de procesos: (2) cambio de modo

- Se ejecuta una **rutina del SO** en el contexto del proceso que se encuentra en estado **Ejecutándose**.
- ¿Cuándo puede realizarse? Siempre que el SO pueda ejecutarse, luego solamente como resultado de:
 - Una interrupción.
 - Una excepción.
 - Una llamada al sistema.

Modos de ejecución

El control de procesos se efectua mediante:

- **Modo usuario.** El programa (de usuario) que se ejecuta en este modo sólo se tiene acceso a:
 - Un subconjunto de los registros del procesador.
 - Un subconjunto del repertorio de instrucciones máquina.
 - Un área de la memoria.
- **Modo núcleo (kernel).** El programa (SO) que se ejecuta en este modo tiene acceso a todos los recursos de la máquina.

Modos de ejecución

¿Cómo utiliza el SO el modo de ejecución?

- El modo de ejecución (bit en el PSW) cambia a modo kernel, automáticamente por hardware, cuando se produce:
 - Una interrupción.
 - Una excepción.
 - Una llamada al sistema.
- Seguidamente se ejecuta la rutina del SO correspondiente al evento producido.
- Finalmente, cuando termina la rutina, el hardware restaura automáticamente el modo de ejecución a modo usuario.

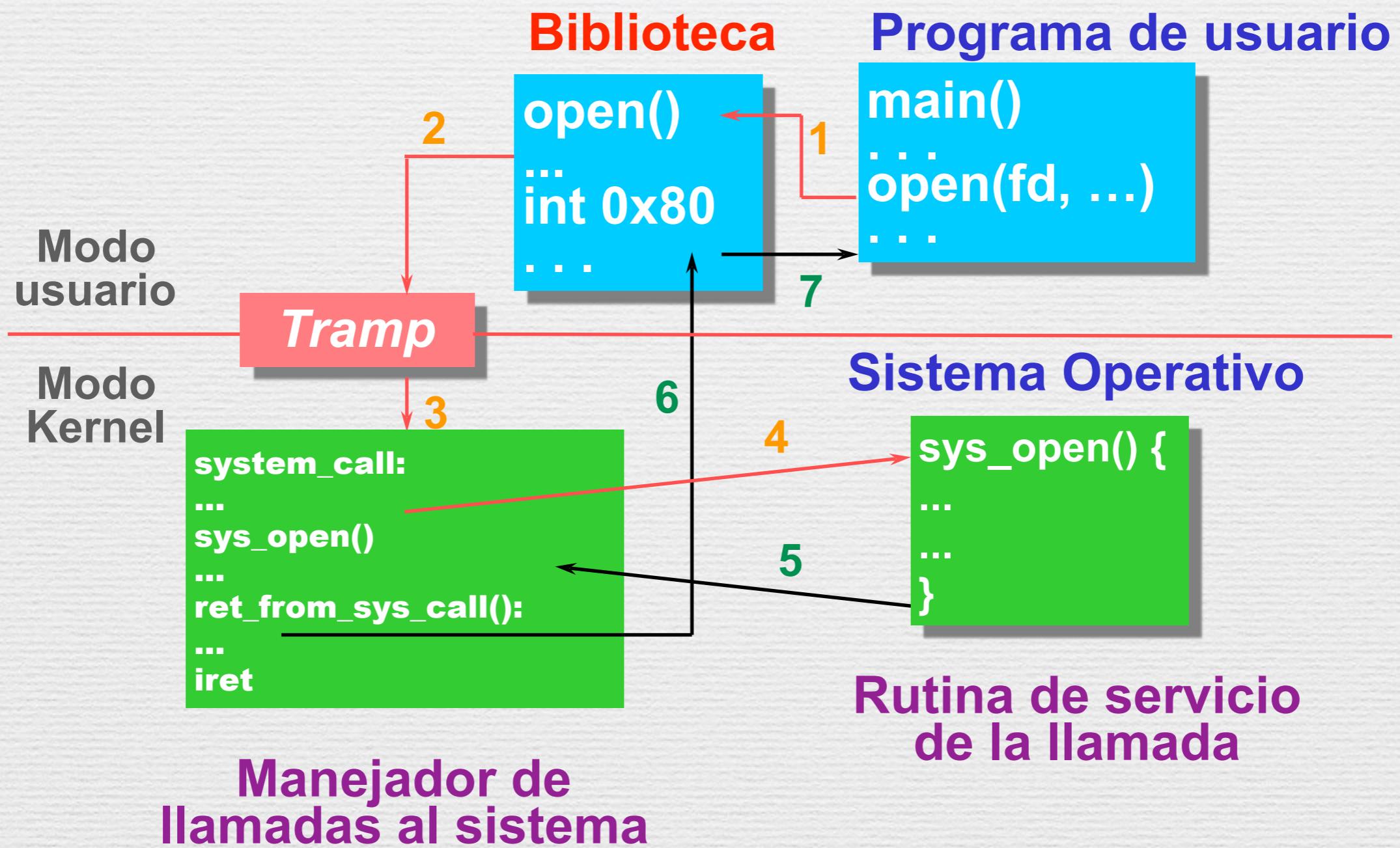
Cambio de modo

- 1) El hardware automáticamente salva como mínimo el PC + PSW y cambia a modo kernel.
- 2) Determinar automáticamente la rutina del SO que debe ejecutarse y cargar el PC con su dirección de comienzo.
- 3) Ejecutar la rutina. Posiblemente la rutina comience salvando el resto de registros del procesador y termine restaurando en el procesador la información de registros previamente salvada.
- 4) Volver de la rutina del SO. El hardware automáticamente restaura en el procesador la información del PC y PSW previamente salvada.

Llamadas al Sistema

- Es la forma en la que se comunican los programas de usuario con el SO en tiempo de ejecución.
- Son solicitudes al SO de petición de servicio. Modelo cliente/servidor
- Ejemplos de llamadas al sistema:
 - **Solicitudes de E/S**: manipulación de archivos y disco.
 - **Gestión de procesos**: control de procesos y comunicaciones.
 - **Gestión de memoria**: mantenimiento de la información
- Se implementan a través de una *trampa* (instrucción trap 12 en ensamblador) o **interrupción software**.

Llamadas al Sistema



Hebras

- Un proceso permite al SO:
 - Controlar la asignación de los recursos necesarios para la ejecución de programas (incluida la memoria).
 - Intercalar la ejecución del programa asociado al proceso con otros programas.
- Una hebra es una ejecución independiente de un proceso. Puede haber varias hebras en ejecución.
- La hebra permite separar los recursos del proceso con su ejecución:
 - La tarea se encarga de soportar todos los recursos necesarios.
 - Cada una de las hebras permite la ejecución del programa de forma *independiente* del resto de hebras.

Hebras

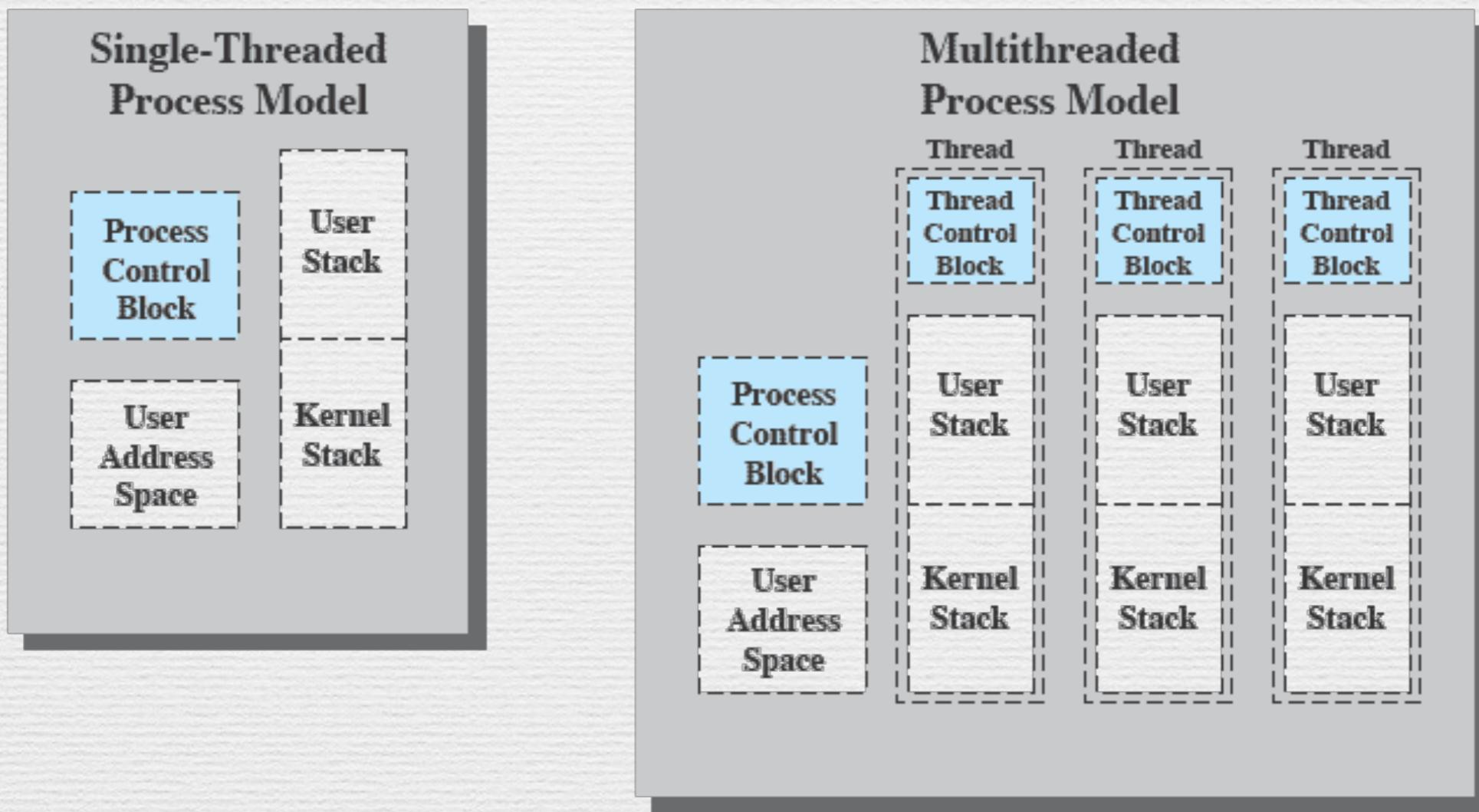
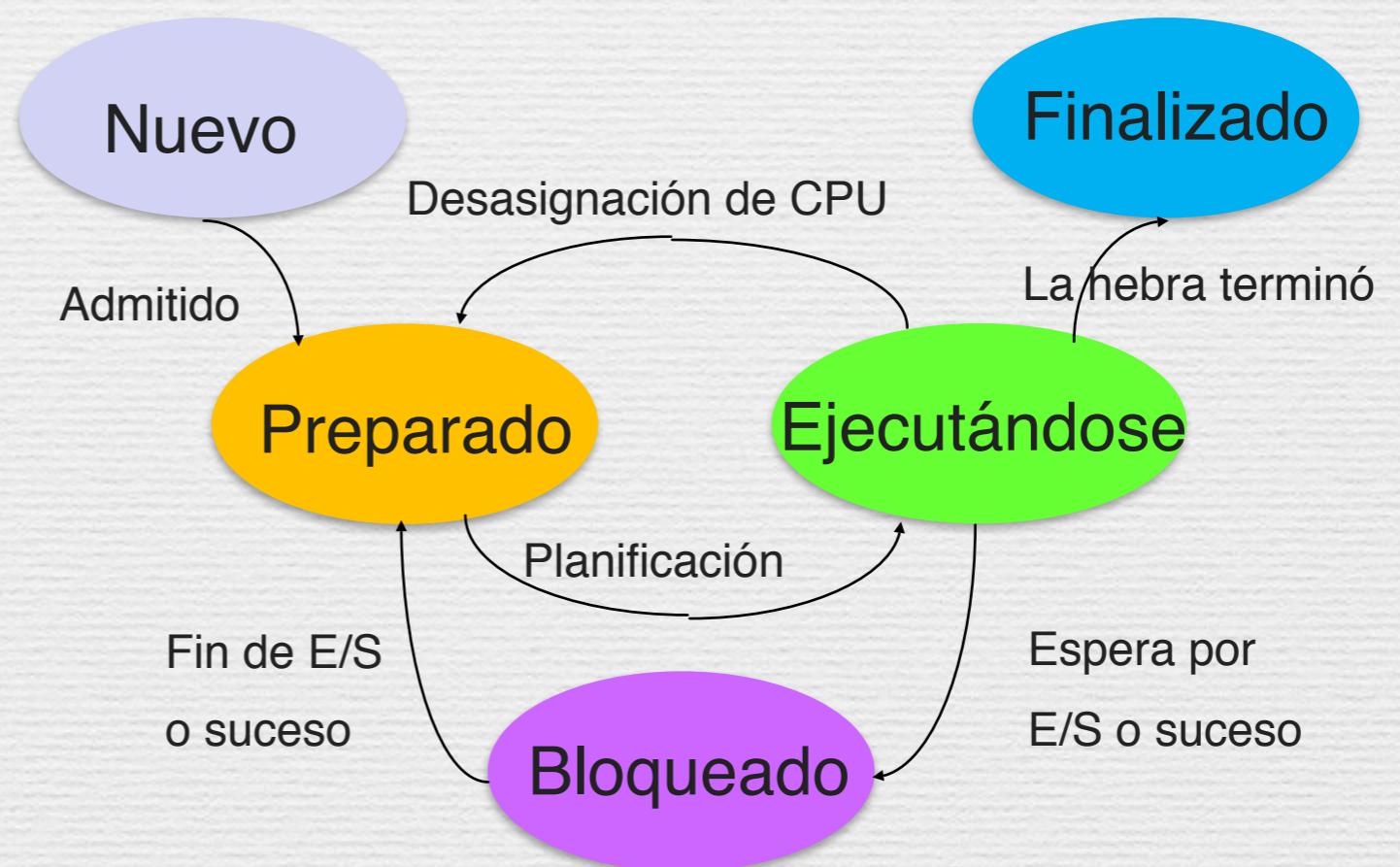


Figure 4.2 Single Threaded and Multithreaded Process Models

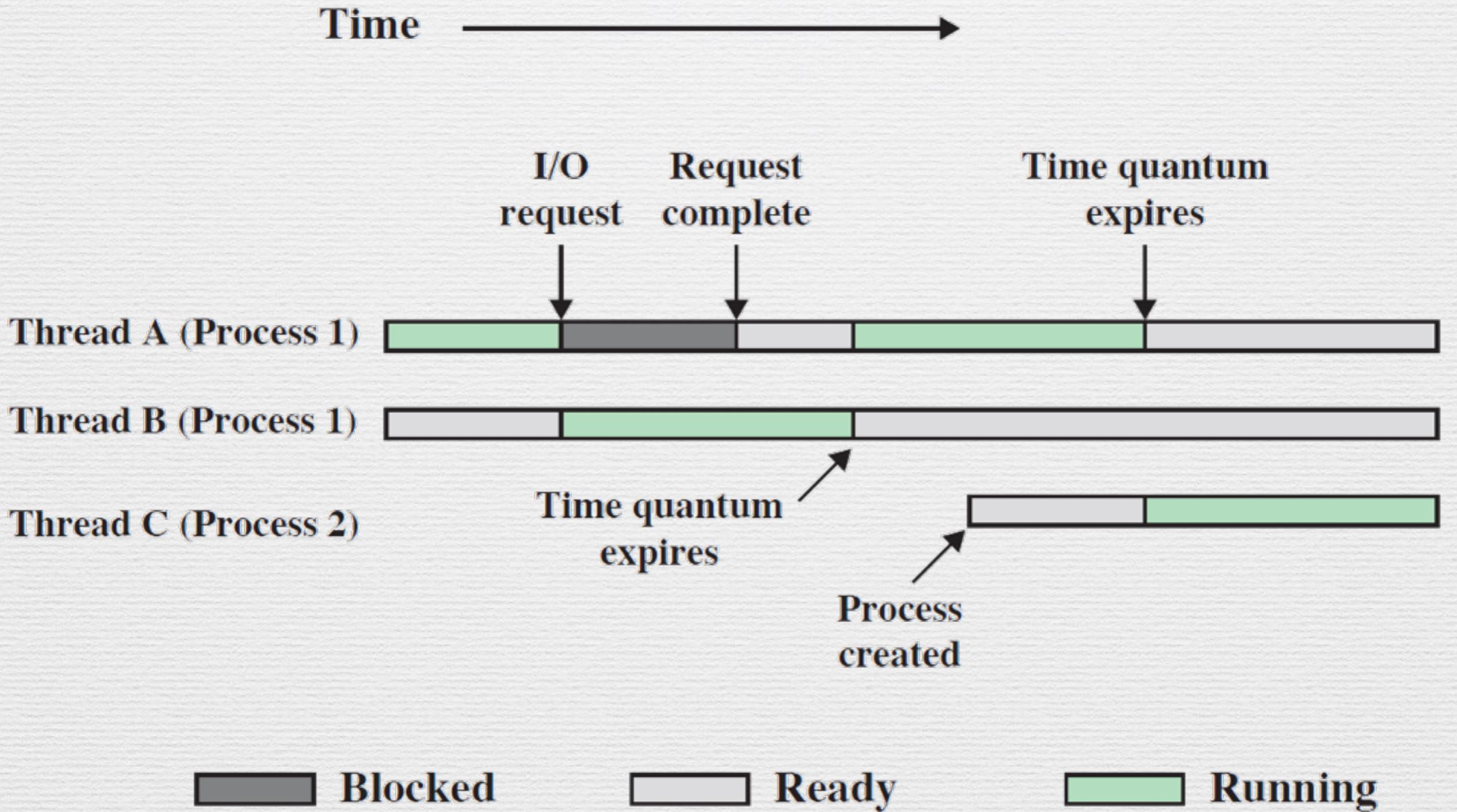
Modelo 5 estados

- Las hebras debido a su característica de ejecución de programas presentan cinco estados análogos al modelo de estados para procesos:

- Ejecutándose**
- Preparado**
- Bloqueado**
- Nuevo**
- Finalizado**



Ejemplo con hebras



Hebras

Ventajas de las hebras

- Menor tiempo de creación de una hebra en un proceso ya creado que la creación de un nuevo proceso.
- Menor tiempo de finalización de una hebra que de un proceso.
- Menor tiempo de cambio de contexto (hebra) entre hebras pertenecientes al mismo proceso.
- Facilitan la comunicación entre hebras pertenecientes al mismo proceso.
- Permiten aprovechar las técnicas de programación concurrente y el multiprocesamiento simétrico.

2.4 Gestión básica de memoria

Tarea realizada por el SO que consiste en gestionar la jerarquía de memoria para cargar y descargar procesos hacia o desde la memoria principal.

El dispositivo de hardware Memory Management Unit-MMU transforma direcciones lógicas en direcciones físicas (tipos de direcciones).

Objetivos de la Gestión de Memoria:

- ♦ Ofrecer a cada proceso un espacio de memoria lógica propio.
- ♦ Proporcionar protección entre los procesos.
- ♦ Permitir que los procesos compartan memoria.
- ♦ Maximizar el rendimiento del sistema.

2.4 Gestión básica de memoria

Requisitos de la Gestión de Memoria

[Stall05] (pp. 308-316) [Stall12] (pp. 327-335)

- **Reubicación:** Como la memoria no es infinita existe la necesidad de cargar en memoria solo parte de los procesos y no de forma permanente (swap o intercambio). Las direcciones de memoria que ocupa un proceso varían de acuerdo a la disponibilidad de espacios libres.
- **Protección:** No se puede permitir que los procesos accedan a las direcciones del SO ni a la de los otros procesos.
- **Compartición:** Bajo la supervisión y control del sistema operativo, puede ser provechoso que los procesos compartan memoria.
- **Organización Lógica:** Correspondencia entre el SO y el hardware al tratar la concepción lógica que tienen los procesos y sus datos, con la organización física de la memoria.

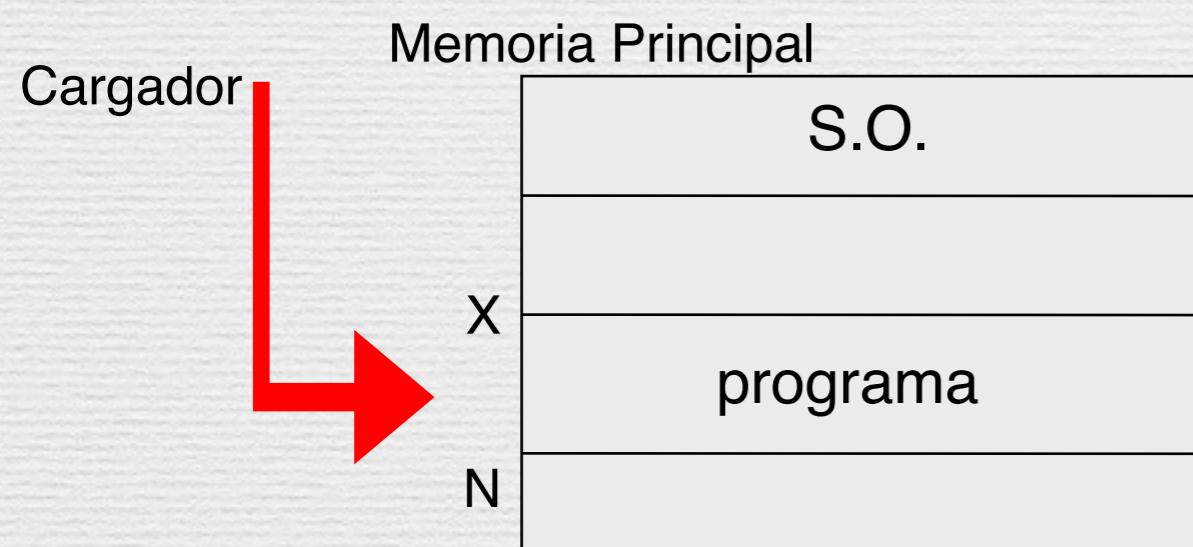
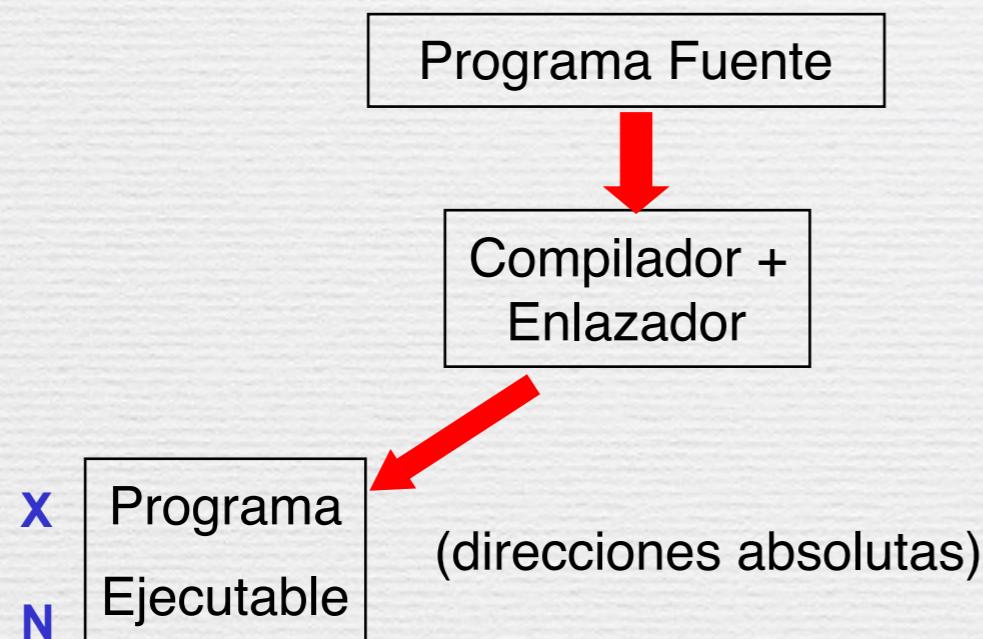
2.4 Gestión básica de memoria

Carga absoluta y reubicación

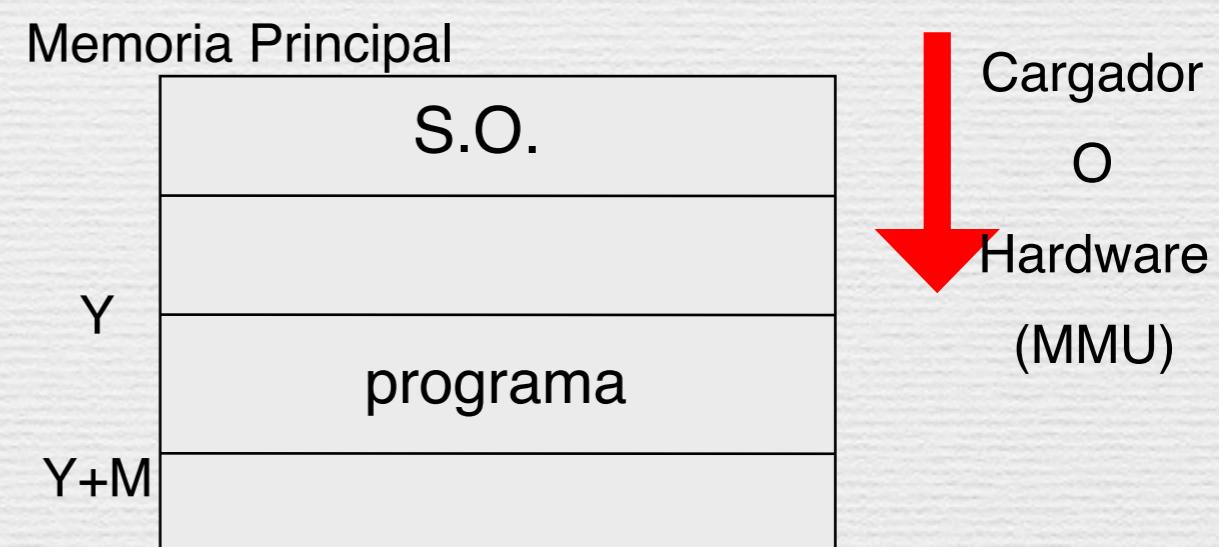
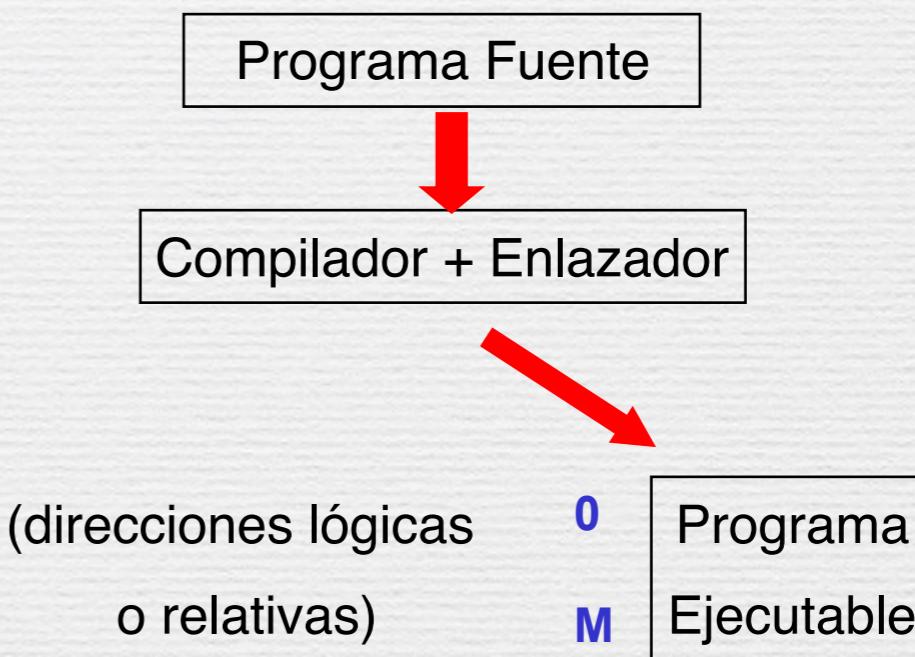
- **Tipos de direcciones.**
 - Físicas (directas a memoria) o absolutas.
 - Relativas al origen del programa.
 - Lógicas, o referencia a una localización no asignada.
- **Carga absoluta.**
 - Asignar direcciones físicas al programa en tiempo de compilación.
 - El programa no es reubicable.
- **Reubicación.**
 - Capacidad de cargar y ejecutar un programa en un lugar arbitrario de la memoria.

2.4 Gestión básica de memoria

Carga absoluta

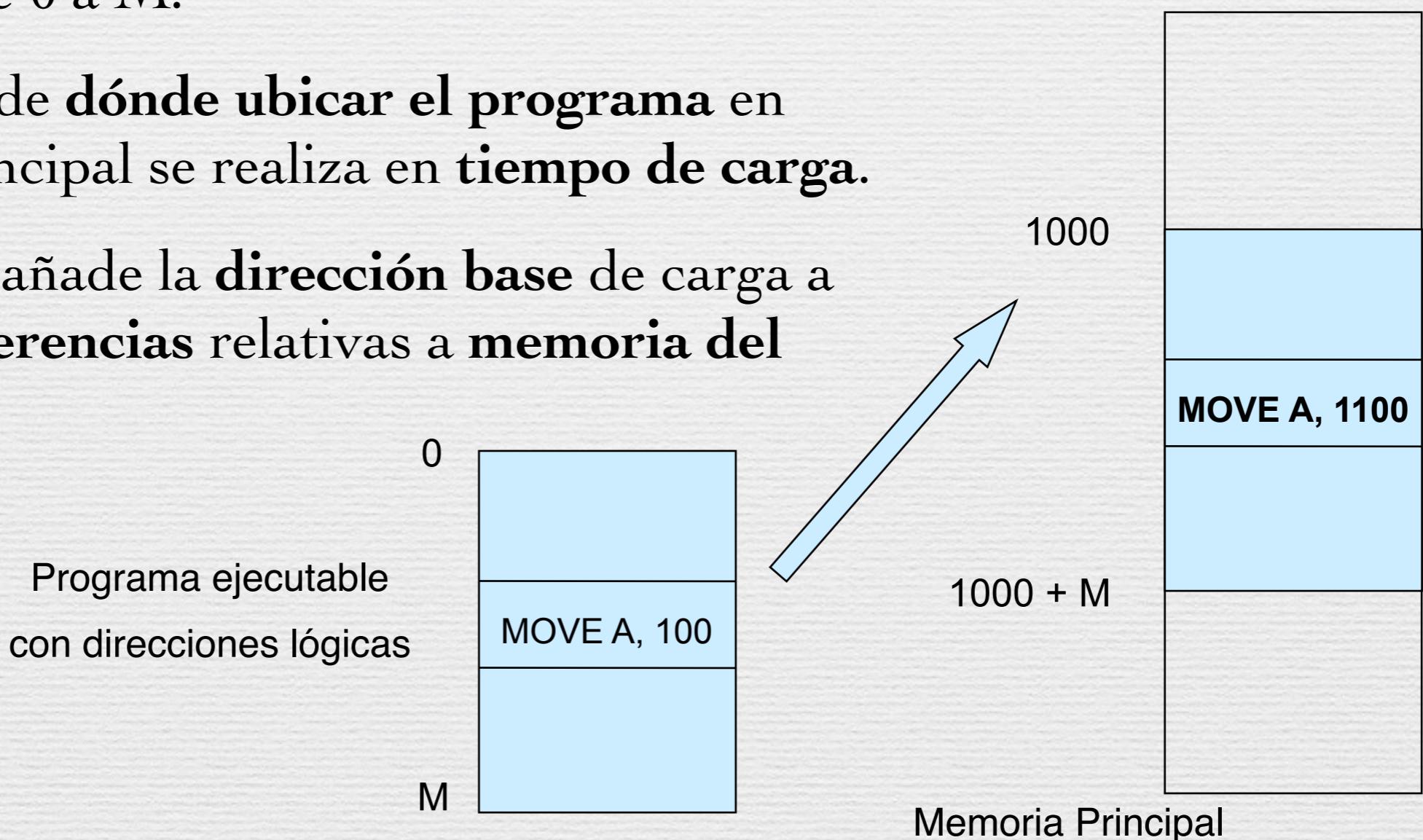


Reubicación



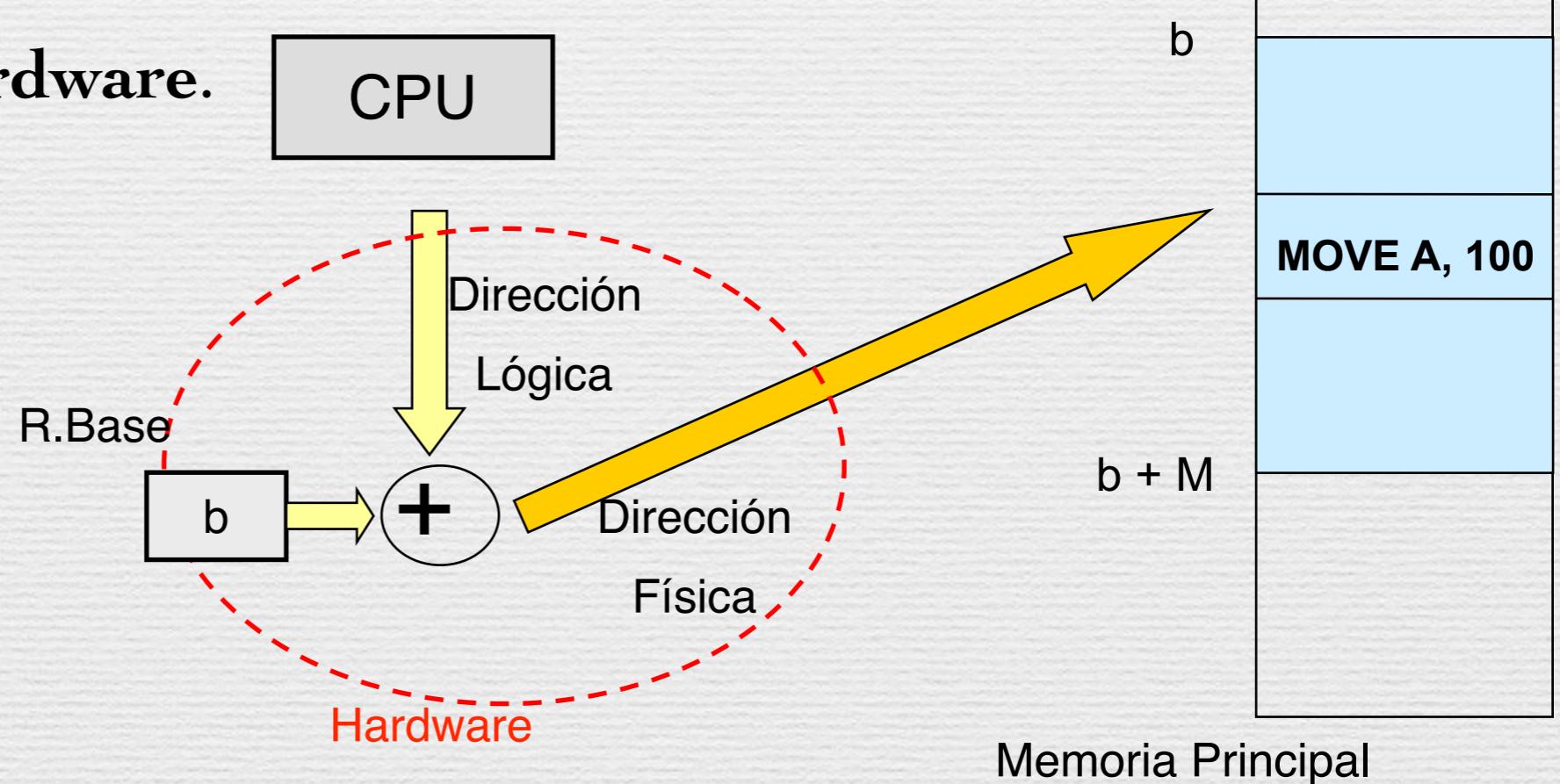
Reubicación estática

- El **compilador** genera **direcciones lógicas** (relativas) de 0 a M.
- La **decisión** de **dónde ubicar el programa** en memoria principal se realiza en **tiempo de carga**.
- El **cargador** añade la **dirección base** de carga a todas las **referencias** relativas a **memoria del programa**.



Reubicación dinámica

- El **compilador** genera **direcciones lógicas** (relativas) de 0 a M.
- La traducción de **direcciones lógicas** a **físicas** se realiza en **tiempo de ejecución** luego el programa está cargado con referencias relativas.
- Requiere apoyo **hardware**.



2.4 Gestión básica de memoria

Definiciones asociadas al manejo de memoria

- ♦ **Espacio de direcciones lógico.** Conjunto de direcciones lógicas (*o relativas*) que utiliza un programa ejecutable.
- ♦ **Espacio de direcciones físico.** Conjunto de direcciones físicas (memoria principal) correspondientes a las direcciones lógicas del programa en un instante dado.
- ♦ **Mapa de memoria de un ordenador.** Todo el espacio de memoria direccionable por el ordenador. Normalmente depende del tamaño del bus de direcciones.
- ♦ **Mapa de memoria de un proceso.** Estructura de datos (que reside en memoria) que indica el tamaño total del espacio de direcciones lógico y la correspondencia entre las direcciones lógicas y las físicas.

2.4 Gestión básica de memoria

Técnicas de Asignación Contigua de Memoria

Debido a la multiprogramación, varios procesos residen simultáneamente en MP. En cada cambio de contexto, su intercambio con HD es costoso.

- **Particiones fijas.**

- La MP se divide en cierto nº de partes de tamaño constante.
- Cada nuevo proceso es ubicado en una partición.
- La liberación del proceso libera la partición.
- El espacio no ocupado es inutilizable.

- **Particiones variables.**

- El tamaño de la partición en MP es exactamente el que el proceso solicita.

2.4 Gestión básica de memoria



Particiones de Igual Tamaño



Particiones de Distinto Tamaño

- **Particiones fijas.**
 - Genera fragmentación interna.
- **Particiones variables.**
 - Genera fragmentación externa. La compactación es costosa.
 - Es necesario un mecanismo de relocalización

2.4 Gestión básica de memoria

Problema de la fragmentación de memoria



2.4 Gestión básica de memoria

Técnicas de Asignación No Contigua de Memoria

- ♦ Trocear el espacio lógico en unidades más pequeñas: **páginas** (elementos de longitud fija), o **segmentos** (elementos de longitud variable).
- ♦ Los trozos no tienen por qué ubicarse consecutivamente en el espacio físico de memoria.
- ♦ Los esquemas de organización del espacio lógico de direcciones y de traducción de una dirección del espacio lógico al espacio físico que comentaremos son:
 - Paginación
 - Segmentación

Paginación

- ♦ El espacio de direcciones físicas de un proceso puede ser no contiguo.
- ♦ La memoria física se divide en bloques de tamaño fijo, denominados marcos de página. El tamaño es potencia de dos, de 512 B a 8 KB.
- ♦ El espacio lógico de un proceso se divide conceptualmente en bloques del mismo tamaño, denominados páginas.
- ♦ Los marcos de página contendrán páginas de los procesos
- ♦ Calcular: n° máximo de páginas, n° máximo de marcos y tamaño de la página.

Paginación

- ♦ Las direcciones lógicas, que son las que genera la CPU se convierten a los valores número de página (p) y desplazamiento dentro de la página (d)



- ♦ Las direcciones físicas contienen el número de marco (m, dirección base del marco donde está almacenada la página) y desplazamiento (d)



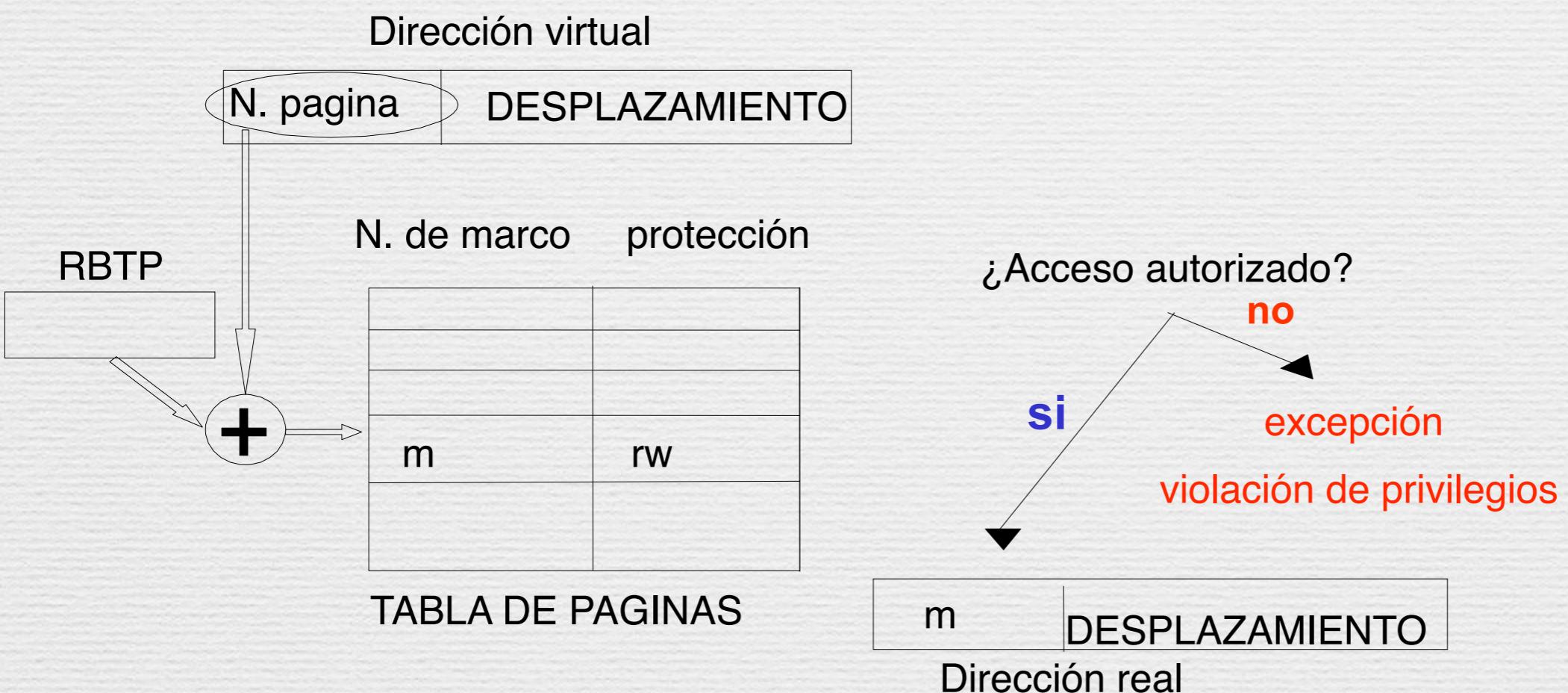
Paginación

- ♦ Cuando la CPU genere una dirección lógica será necesario traducirla a la dirección física correspondiente.
- ♦ La tabla de páginas mantiene información necesaria para realizar dicha traducción.
- ♦ Existe una tabla de páginas por proceso.
- ♦ Tabla de marcos de página, usada por el S.O. y contiene información sobre cada marco de página

Tabla de páginas

- ♦ Una entrada por cada página del proceso:
 - ♦ Número de marco (dirección base del marco) en el que está almacenada la página si está en MP.
 - ♦ Modo de acceso autorizado a la página (bits de protección)
 - ♦ Adicionalmente otros bits de presencia, modificación, etc.

Esquema de traducción



2.4 Gestión básica de memoria

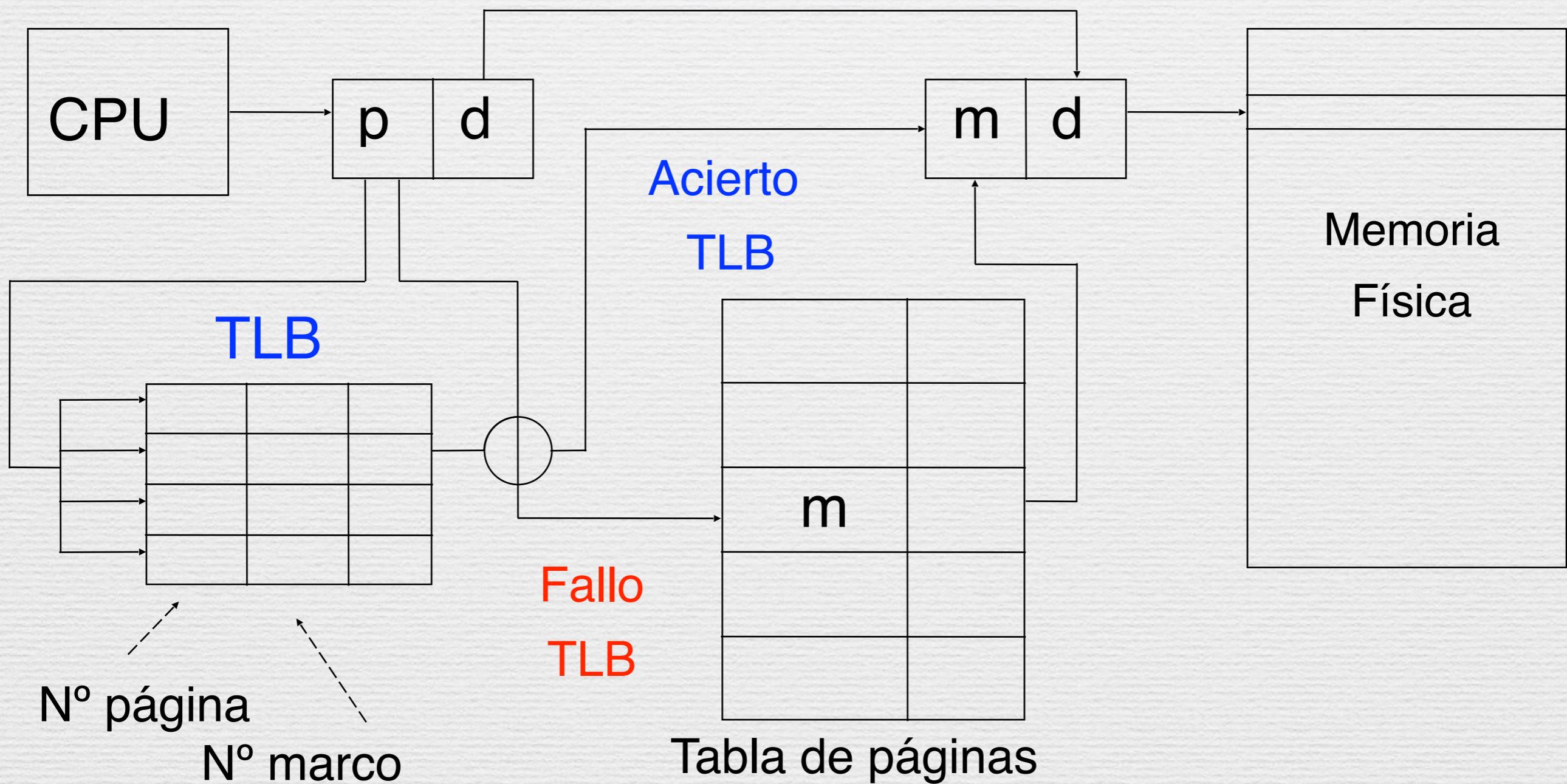
Implementación de la paginación:

- La tabla de páginas y la lista de marcos libres se mantienen en MP.
- El registro base de la tabla de páginas (RBTP) apunta a la tabla de páginas. Suele almacenarse en el PCB del proceso.
- En este esquema cada acceso a una instrucción o dato requiere dos accesos a memoria, uno a la tabla de páginas y otro a memoria.

TLB - Buffer de Traducción Adelantada [Stall05] (pp. 349-351)

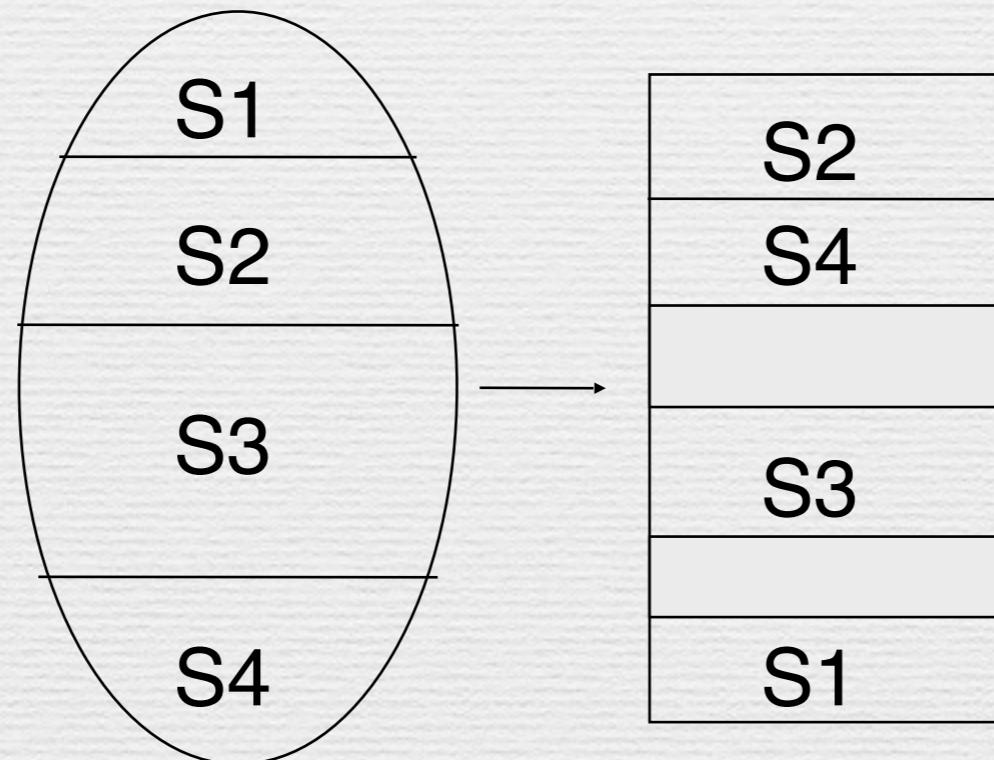
- Caché hardware de consulta rápida TLB (Translation Look-aside Buffer)
- Conjunto de registros asociativos que permiten una búsqueda en paralelo.
- Para traducir una dirección:
 - Si existe ya en el registro asociativo, obtenemos el marco.
 - Si no, la buscamos en la tabla de páginas y se actualiza el TLB con esta nueva entrada

Traducción con TLB



Segmentación

- Esquema de organización de memoria que soporta mejor la visión de memoria del usuario
- Un programa es una colección de unidades lógicas segmentos. Ej. procedimientos, funciones, pila, tabla de símbolos, matrices, etc.



Mapa memoria proceso

Memoria física

Tabla de Segmentos

- ♦ Una dirección lógica es una tupla:
 $\langle \text{número_de_segmento}, \text{desplazamiento} \rangle$
- ♦ La Tabla de Segmentos aplica direcciones bidimensionales definidas por el usuario en direcciones físicas de una dimensión.
- ♦ Cada entrada de la tabla tiene los siguientes elementos:
 - ♦ registro base:
memoria.
 - ♦ registro límite: tamaño o longitud del segmento
 - ♦ información de protección, presencia, modificación...

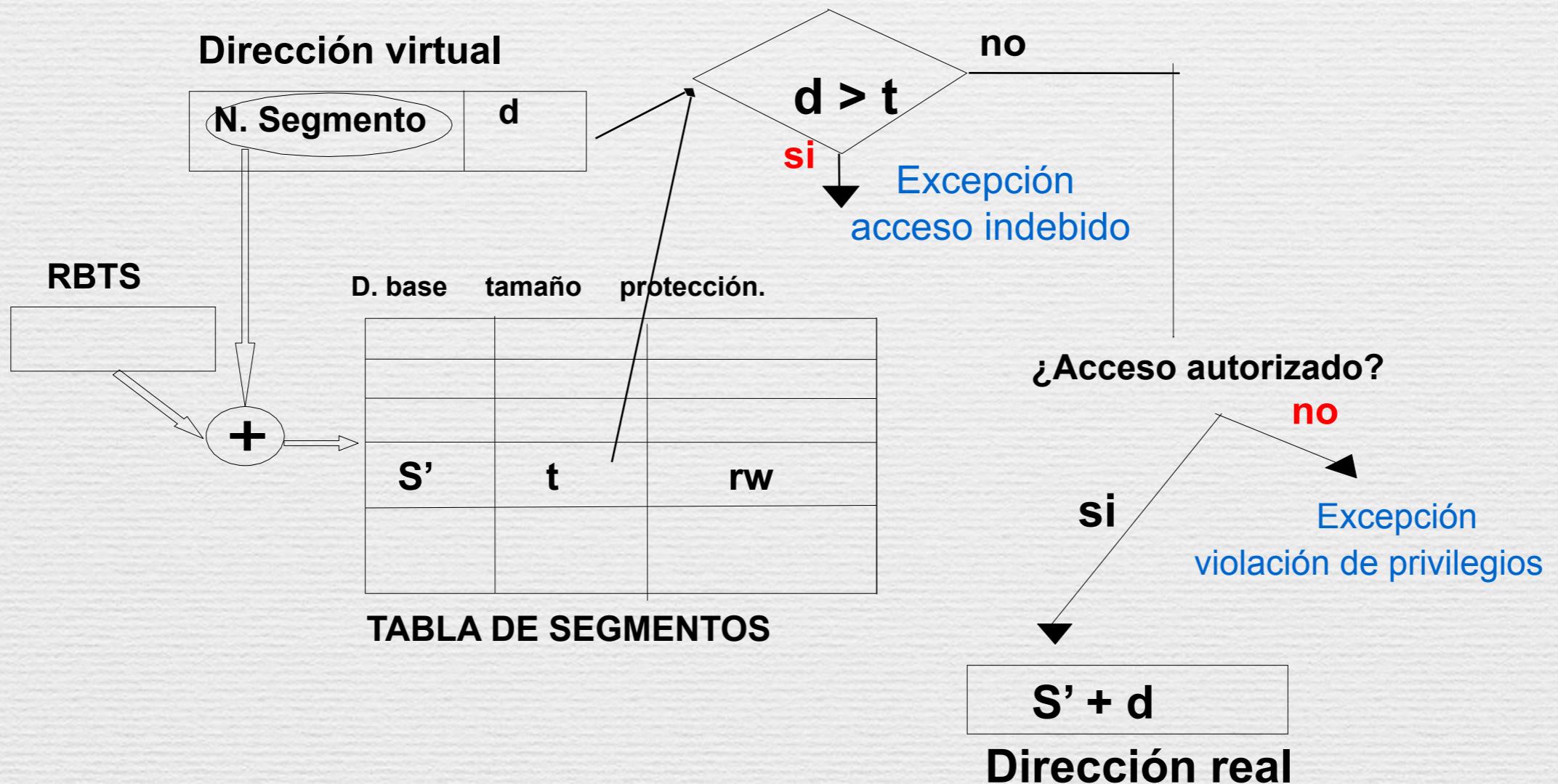
2.4 Gestión básica de memoria

Implementación de la Tabla de Segmentos

- ♦ La Tabla de Segmentos, y la lista de segmentos libres se mantienen
- ♦ El PCB del proceso almacena:
 - ♦ el Registro Base de la Tabla de Segmentos (RBTS): apunta a la dirección de inicio de la tabla de segmentos
 - ♦ el Registro Longitud de la Tabla de Segmentos (RLTS) indica el número de segmentos del proceso.
- ♦ Una dirección lógica $\langle s, d \rangle$ es legal si:

$$s < R$$

Esquema de traducción



Autoevaluación

Entre teoría y prácticas yo puedo:

- ♦ Comprender el rol del sistema operativo como capa de abstracción, gestor de recursos y catalizador de medidas de seguridad.
- ♦ Clasificar los SO existentes desde la perspectiva de la interfaz con el usuario y aspectos de multiprocesamiento.
- ♦ Sacar beneficio a las llamadas al SO, mediante la creación de scripts desde la consola del sistema (
- ♦ Saber usar los comandos de control de procesos e información del sistema.
- ♦ Comprender conceptos de programación concurrente; coordinación de procesos y hebras.
- ♦ Comprender las distintas técnicas de gestión de memoria.

Y ademas ...