Administrador de Procesos

Sistemas Operativos

M. en C. Violeta del Rocío Becerra Velázquez

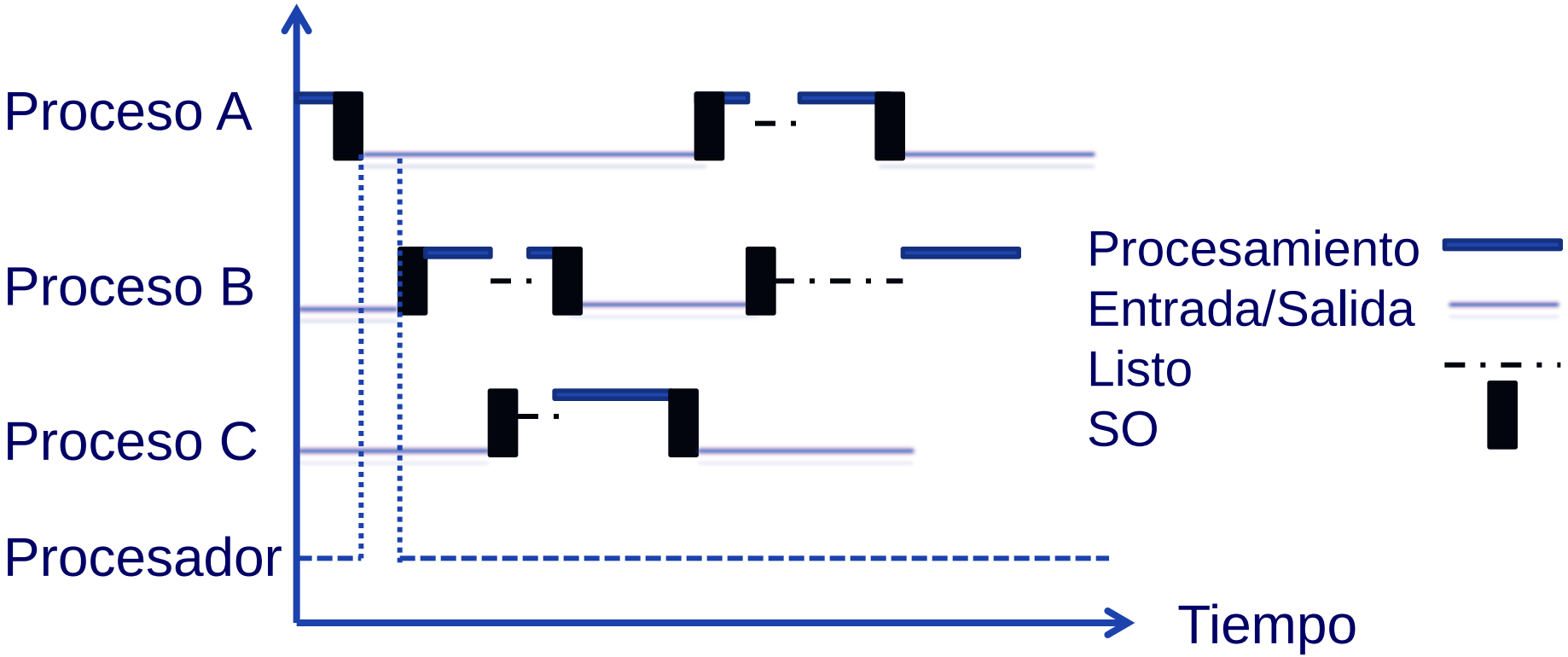
Marzo de 2014.

Multitarea

La multitarea se basa en:

- Paralelismo real entre E/S y procesador.
- Alternancia en los procesos de fases de E/S y de procesamiento.
- Memoria principal capaz de almacenar varios procesos.

Ejemplo de Ejecución en un Sistema Multitarea



Cambio de Contexto

- Cuando se produce una interrupción se cambia de contexto.
- Cambio de contexto es el conjunto de dos operaciones:
  - Se salva el estado del procesador en el correspondiente BCP
  - Se pasa a ejecutar la rutina de tratamiento de interrupción del SO
- Planificador: Módulo del SO que decide el siguiente proceso a ejecutar.
- Activador: Módulo del SO que pone a ejecutar un proceso.
  - Copia el estado del BCP a los registros
  - Termina con una instrucción RETI (retorno de interrupción)
    - Restituye el registro de estado (bit de nivel de ejecución)
    - Restituye el contador de programa (para el nuevo proceso).

Información del Proceso

- **Estado del procesador: (conformado por todos sus registros)**
  - Registros generales.
  - Contador de programa.
  - Puntero de pila.
  - Registro o registros de estado.
  - Registros especiales

Información del Proceso

Imagen de Memoria del Proceso

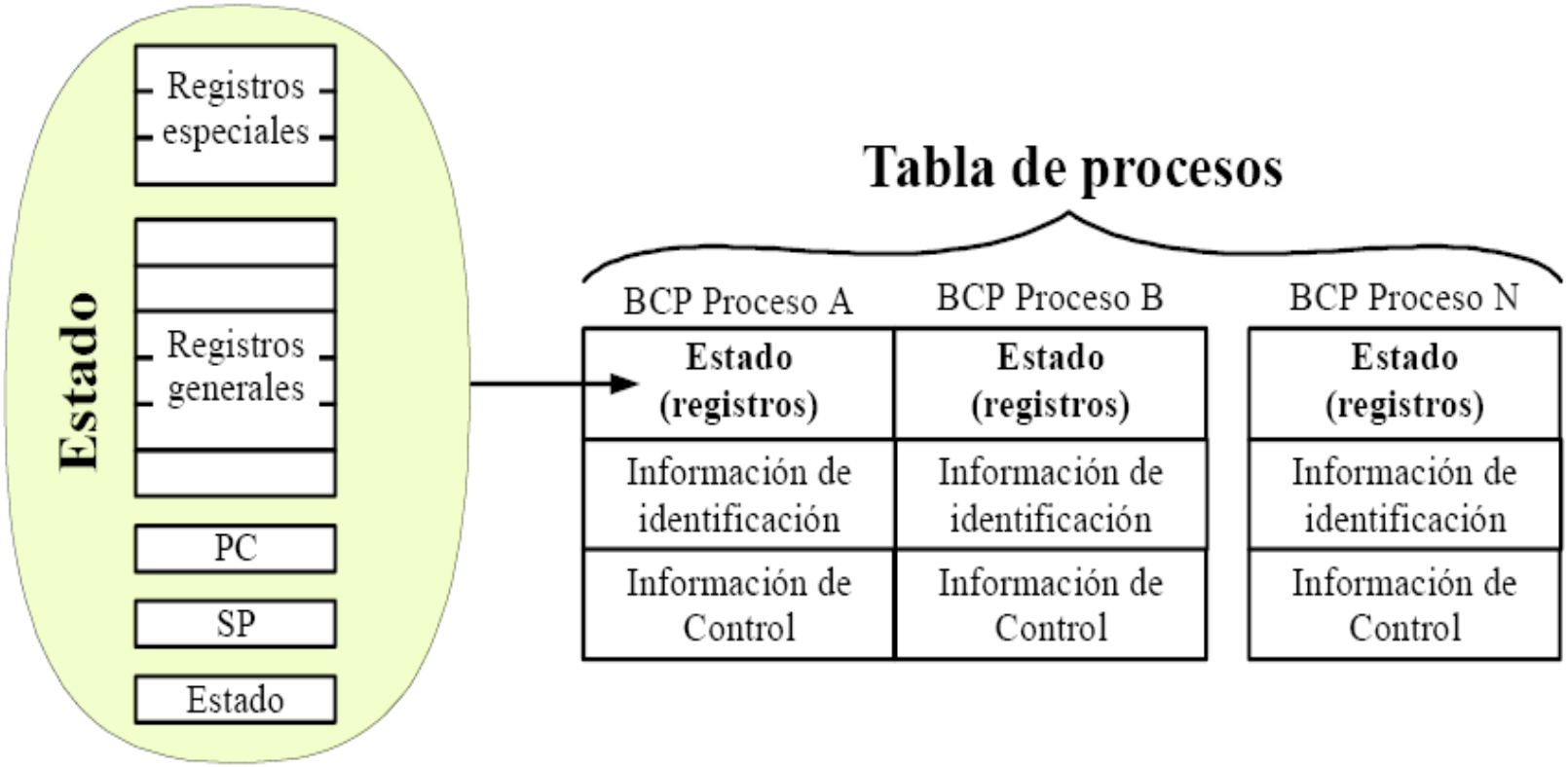
- Esta formada por los espacios de memoria que está autorizado a utilizar.

Información del Proceso

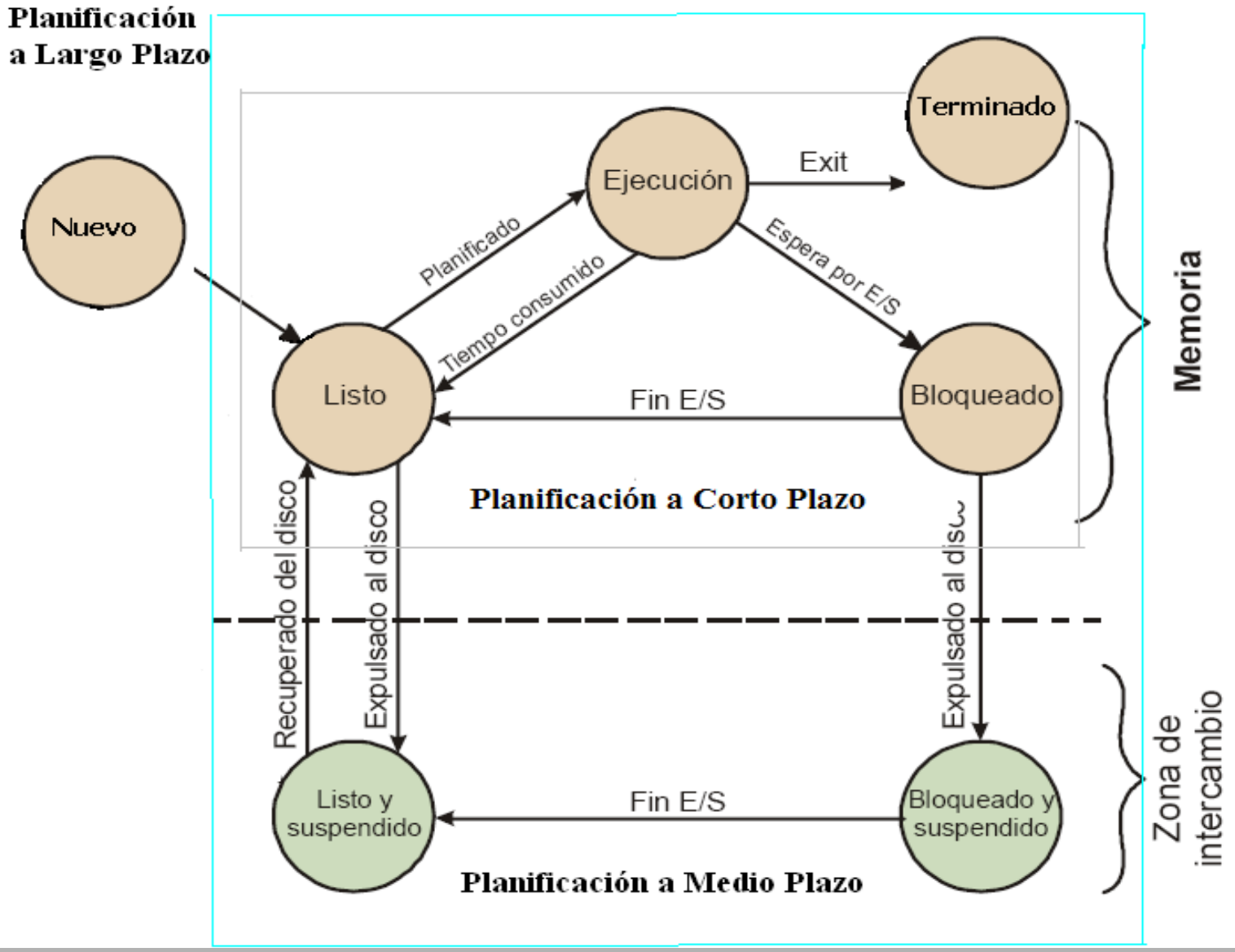
Información del BCP

- Información de Identificación: esta identifica al usuario y al proceso.
- Estado del procesador: contiene los valores iniciales del estado del procesador.
- Información de control del proceso.
  - Información de planificación y estado.
  - Descripción de segmentos de memoria asignados.
  - Recursos asignados.
  - Punteros.
  - Comunicación entre procesos.

Tabla de Procesos



Tipos de Planificación



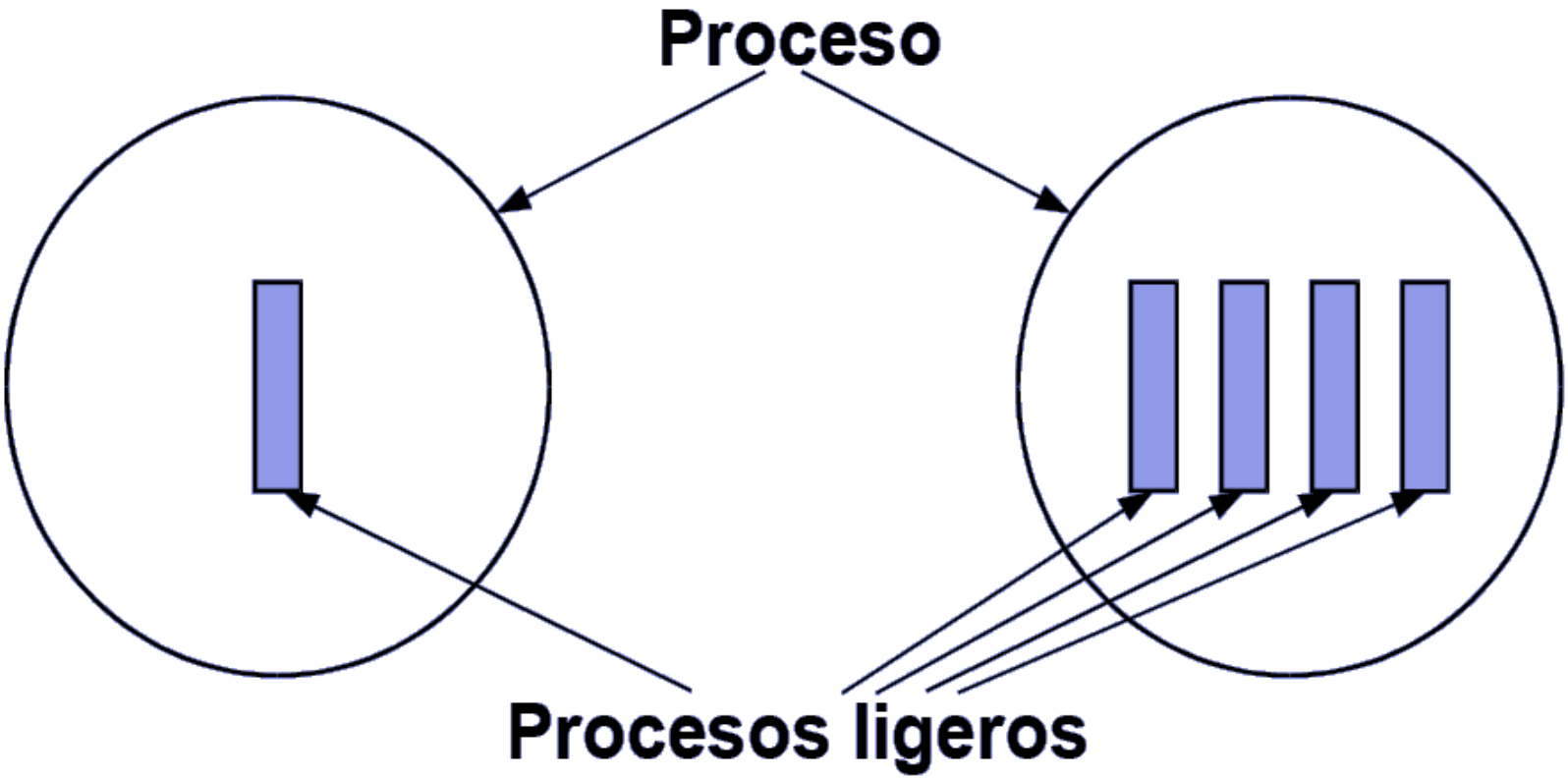
Objetivos de Planificación

- Reparto de UCP equitativo
- Eficiencia (optimizar UCP)
- Mejor tiempo de respuesta en uso interactivo
- Mejor tiempo de espera en lotes (batch)
- Mayor número de trabajos por unidad de tiempo

Procesos Ligeros

- **Un hilo es una secuencia de instrucciones relacionadas que se ejecuta de forma independiente a otras secuencias.**
- **Todos los programas tienen al menos un hilo principal**
  - Inicializa el programa y comienza la ejecución de las instrucciones iniciales.
  - Puede crear otros hilos que ejecuten varias tareas o puede hacer todo el trabajo el solo

Procesos Ligeros o Threads



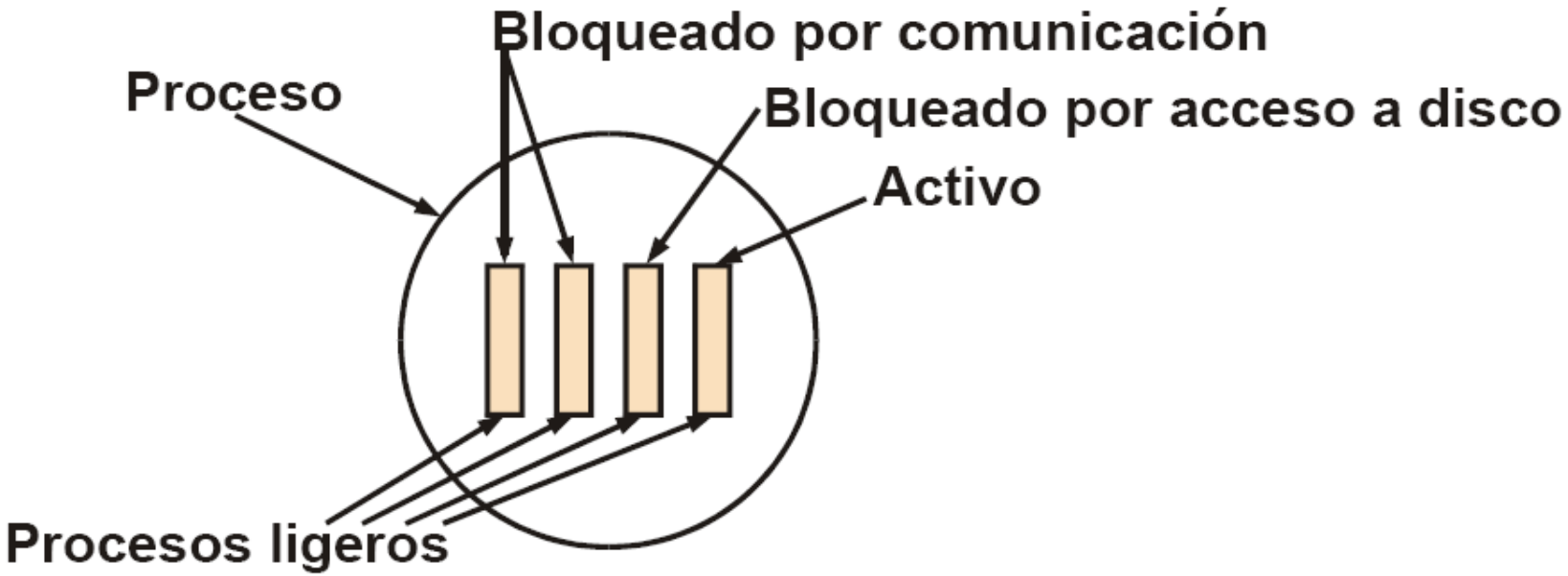
Procesos Ligeros o Threads

- Un hilo es una unidad básica de utilización del CPU; que consiste de:
  - Contador de programa
  - Conjunto de registros
  - Espacio de stack

Procesos Ligeros o Threads

- Un hilo comparte con sus hilos pares:
  - Sección de código
  - Sección de datos
  - Recursos del sistema operativo
  - En conjunto se les conoce como tarea
- Un proceso tradicional o pesado, es igual a una tarea con un hilo.

Estados del Proceso Ligero



Estados del Proceso Ligero

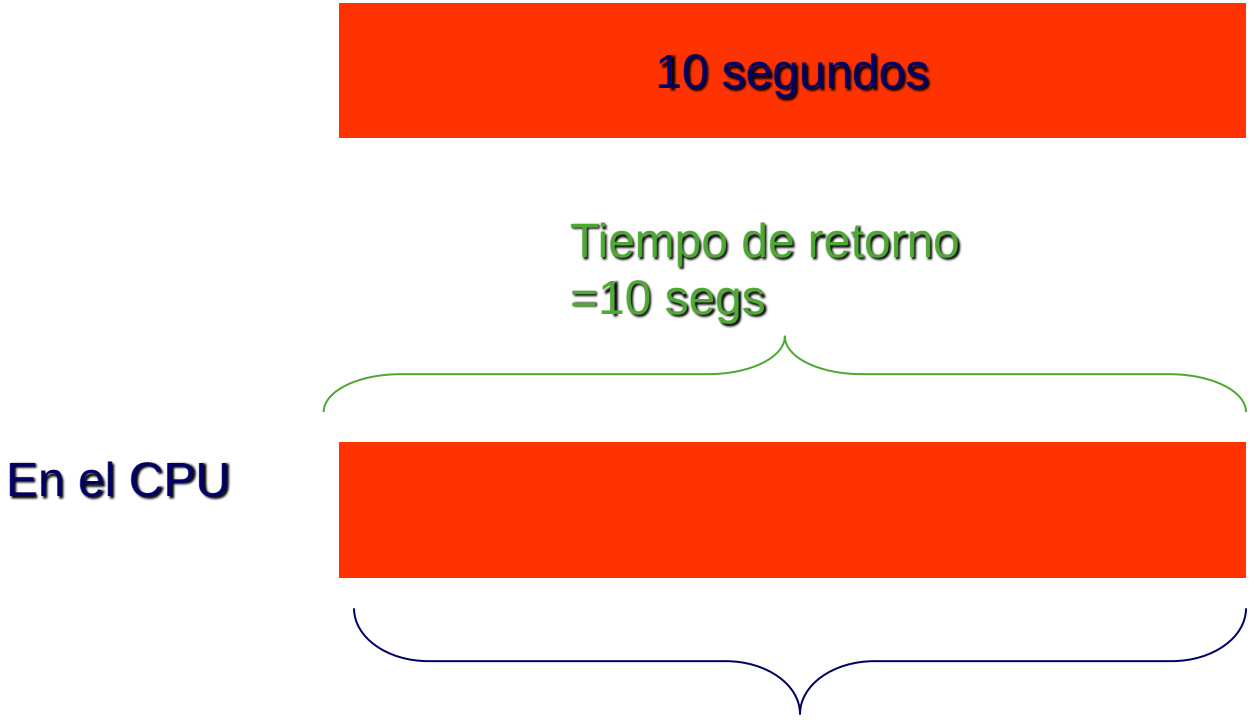
- **En una tarea donde hay múltiples hilos, mientras un hilo servidor está bloqueado y esperando, otro hilo en la misma tarea puede ejecutarse.**
  - Cooperación de múltiples hilos en la misma tarea aumenta la tasa de trabajos por unidad tiempo y mejora el rendimiento.
  - Aplicaciones que requieren compartir un buffer común (productor-consumidor) sacan provecho de la utilización de hilos.

Dividiendo una aplicación

- **Un proceso puede dividirse en múltiples hilos**
  - Mejor uso de los recursos de hardware
- **Es necesario entender**
  - Diseño y estructura de la aplicación.
  - La interface de programación con hilos.
  - El compilador y el ambiente de ejecución de la aplicación.
  - Las plataformas destino donde la aplicación se va a ejecutar.

Uso intensivo de CPU

Un trabajo en un núcleo



1 trabajo en 10 segundos

Tasa = 1 / 10 = 0.1

Uso intensivo de CPU

Dos trabajos en un núcleo

Tiempo de retorno P0 =20 segs

Tiempo de retorno P1 =20 segs

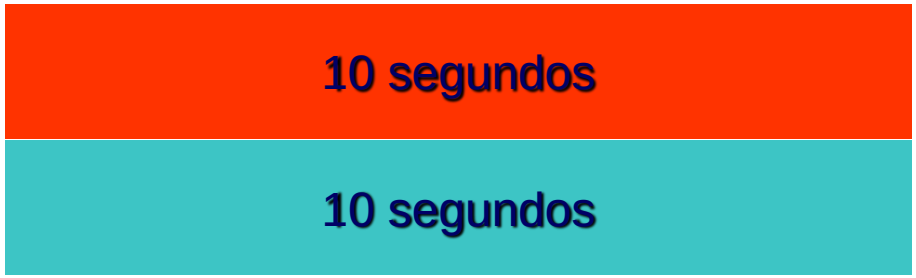
En el CPU

2 trabajos en 20 segundos

Tasa = 2 / 20 = 0.1

Uso intensivo de CPU

Dos trabajos en un núcleo

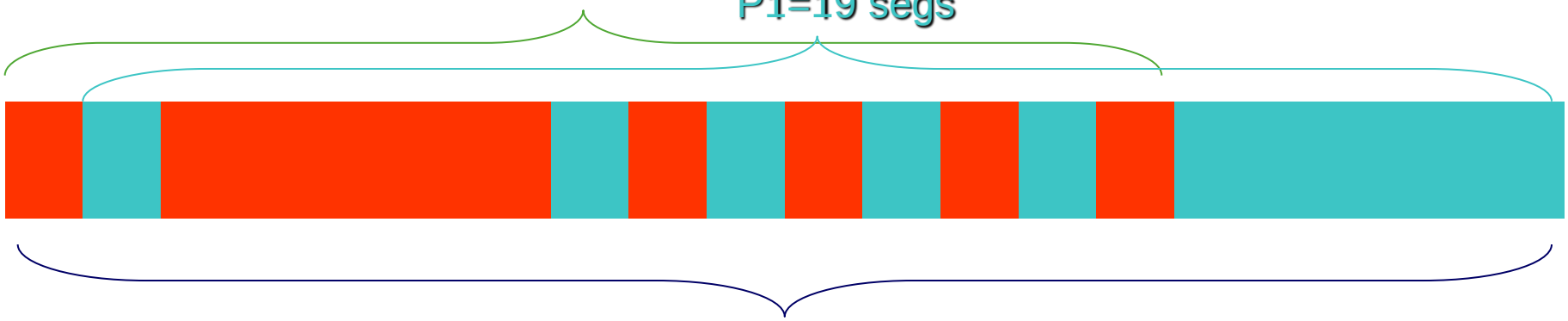


El tiempo de retorno de los procesos puede variar
- Prioridades
- Políticas de planificación

En el CPU

Tiempo de retorno P0 =15 segs

Tiempo de retorno P1=19 segs



2 trabajos en 20 segundos

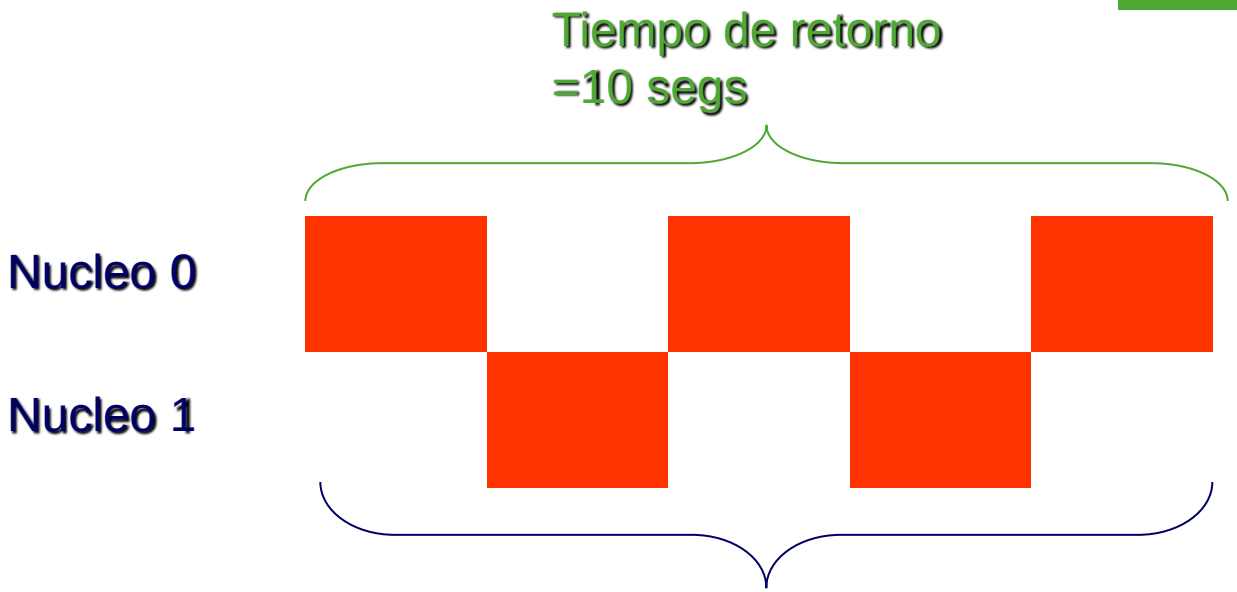
Tasa = 2 / 20 = 0.1

La tasa de trabajos se mantiene igual

Uso intensivo de CPU

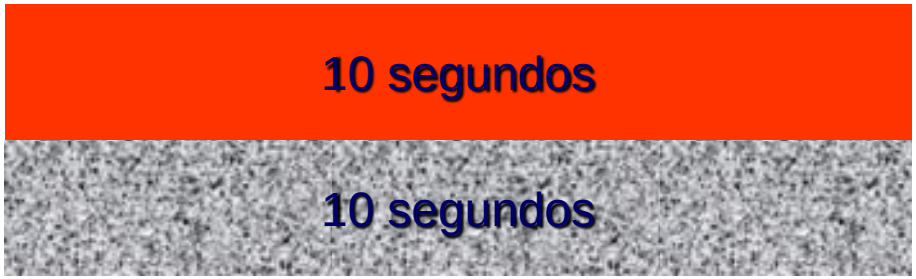
Un trabajo en dos núcleos

No hay mejora en el tiempo de retorno ni la tasa de trabajos



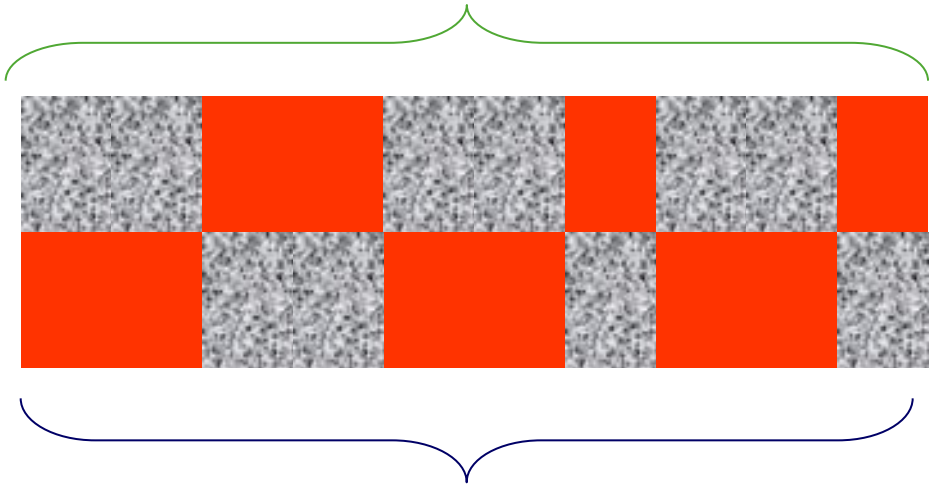
Uso intensivo de CPU

Dos trabajos en dos núcleos



El tiempo de retorno se mantiene igual que cuando ejecutamos un trabajo

Tiempo de retorno
=10 segs



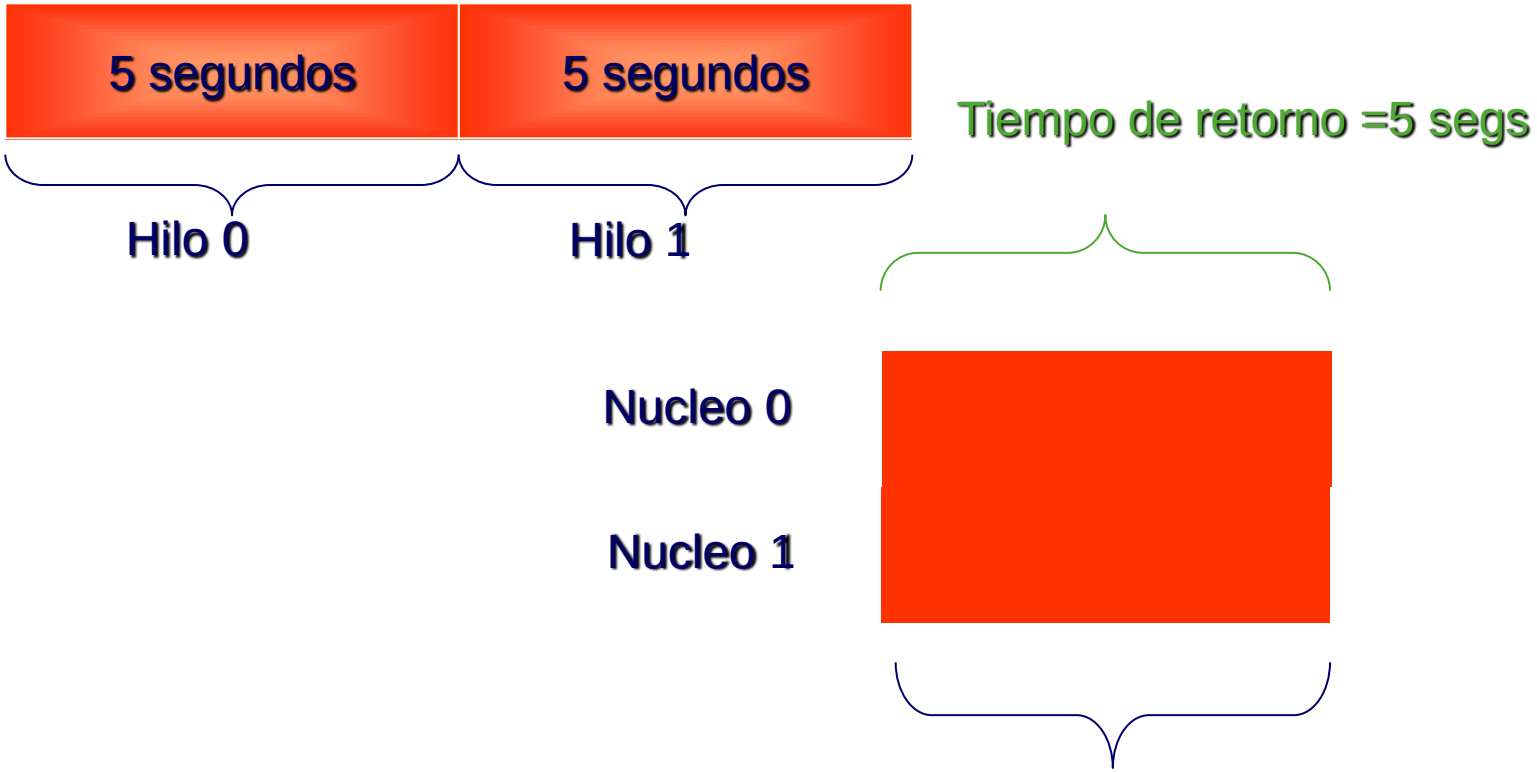
2 trabajos en 10 segundos

La tasa de trabajos aumenta al doble

Tasa = 2 / 10 = 0.2

Uso intensivo de CPU

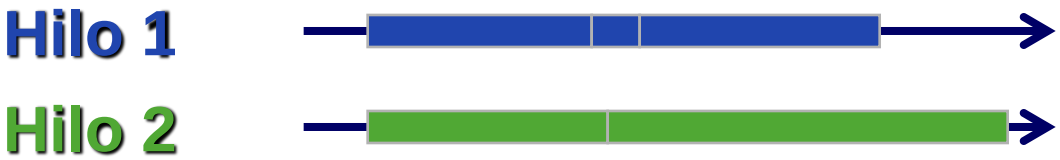
Un trabajo optimizado en dos núcleos



Concurrencia vs. Paralelismo

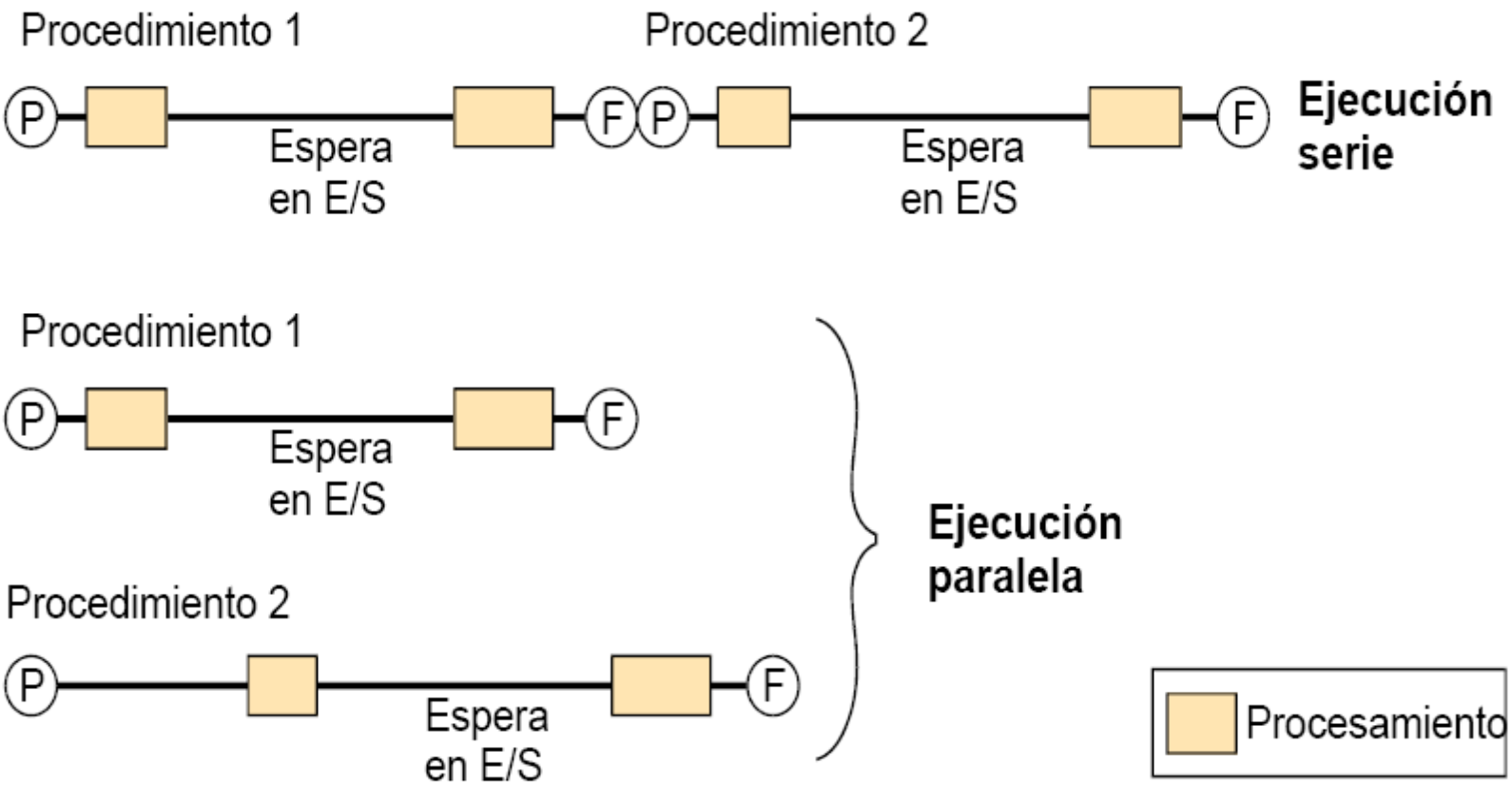
- Concurrencia: dos o más hilos están en progreso al mismo tiempo

- Paralelismo: dos o más hilos están ejecutandose al mismo tiempo



- Se requieren varios núcleos

Paralelismo con hilos



Creando hilos por funcionalidad

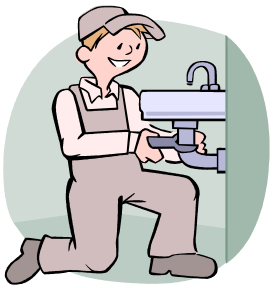
- **Asignar hilos a diferentes funciones de la aplicación**

  El método más simple ya que no hay como sobreponer

- **Ejemplo: *Construyendo una casa***

  Albañil, carpintero, pintor, plomero,…




Hilos para el rendimiento

- **Mejora el rendimiento de computadoras**
- **Paralelizar para mejorar el tiempo de retorno o la tasa de trabajos**
- **Ejemplo**
  - Línea de ensamblado de automóviles
    - Cada trabajador hace una función asignada.

Tiempo de retorno

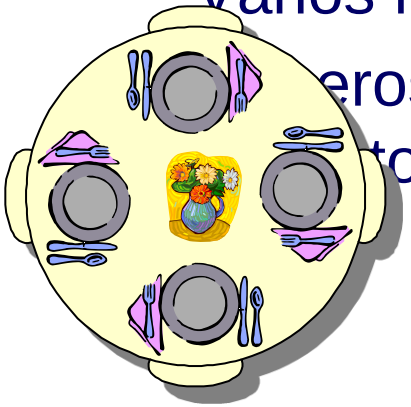
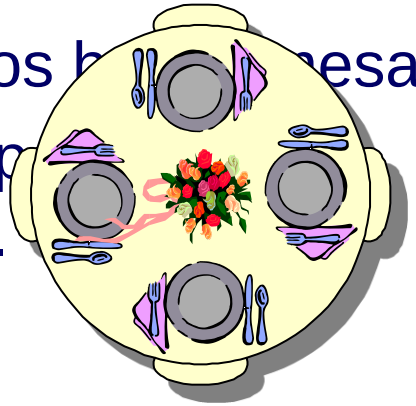
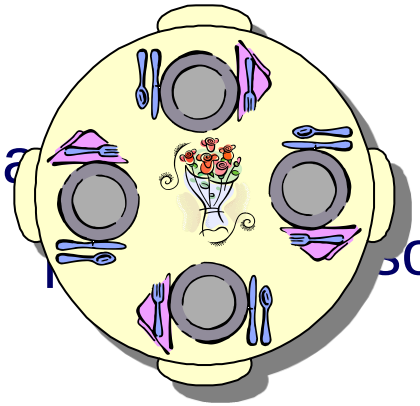
- **Complete una sola tarea en la menor cantidad de tiempo**
- **Ejemplo: *Poner la mesa para la cena***

  - Uno pone los platos
  - Uno que dobla y pone las servilletas
  - Uno que ponga los cubiertos
    - Cucharas, cuchillos, tenedores
  - Uno que ponga los vasos

Tasa de trabajos

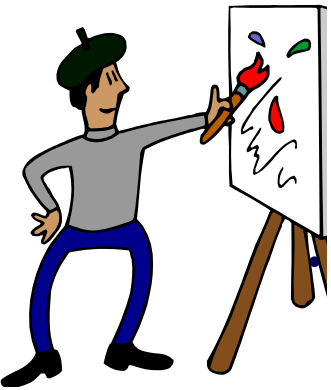
- **Completar la mayor cantidad de trabajos en un tiempo fijo**

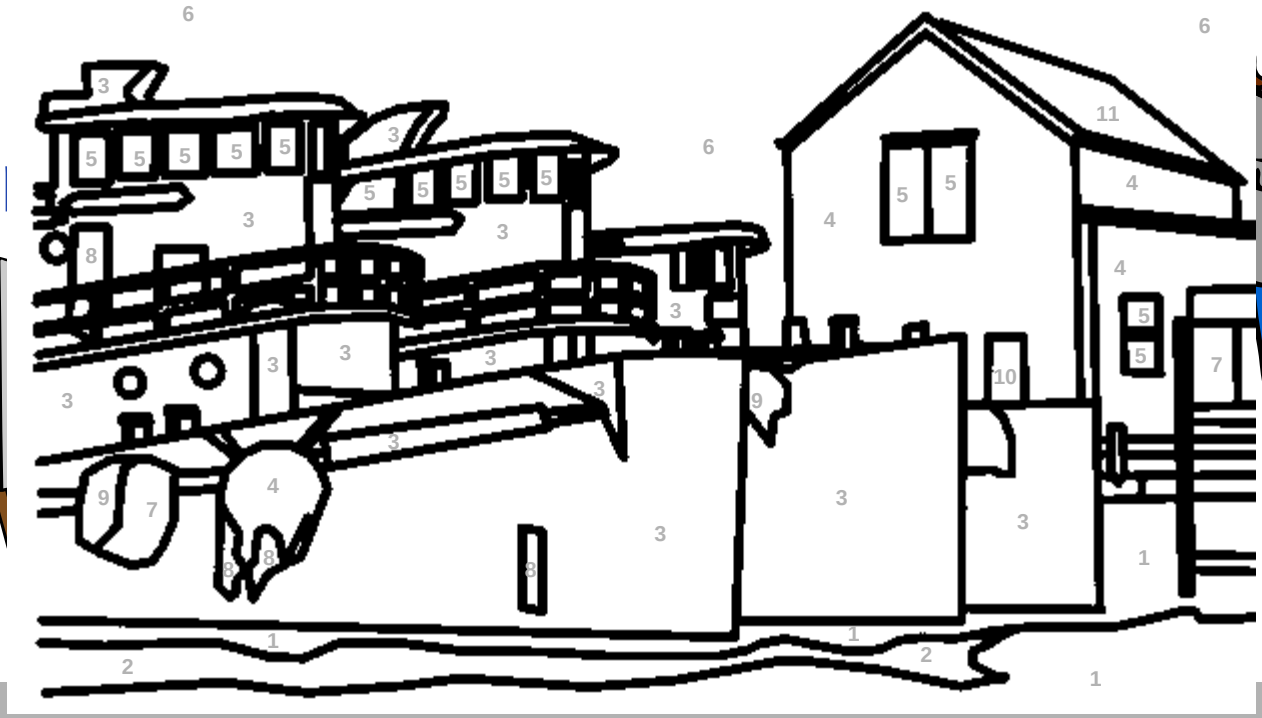
- **Ejemplo: *Poner las mesas de un banquete***
  - Varios meseros h esas sepa
    eros esp para os,
    os, etc.

Descomposición de una tarea

- **Divide el tiempo de computación basado en un conjunto natural de tareas independientes**
  - Asignar datos para cada tarea conforme se van
- **Eje**

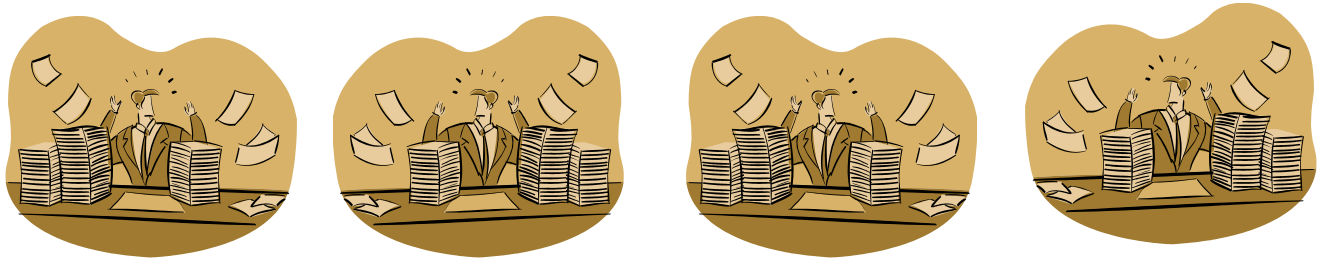




Descomposición de los datos

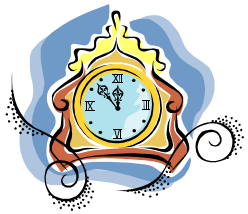
- **Conjuntos de datos grandes cuyos elementos pueden computarse de manera independiente**
  - Divide datos y asocia la computación entre los hilos

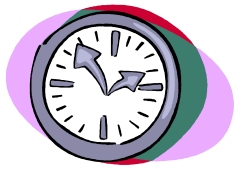
- ***Ejemplo: Calificar exámenes***
  - Varios profesores con la misma llave



Aceleración y Eficiencia

- **Medir que tanto se acelera la ejecución de cómputo vs. el mejor código serial**
  - Tiempo serial dividido por tiempo paralelo
- **Ejemplo: *Pintar una barda de tablitas***
  - 30 minutos se preparación (serial)
  - Un minuto para pintar una tabla
  - 30 minutos para limpiar (serial)





- **Por lo tanto, 300 tablas toman 360 minutos (tiempo serial)**

Computando Aceleración

Número de pintores	Tiempo	Velocidad
1	30 + 300 + 30 = 360	1.0X
2	30 + 150 + 30 = 210	1.7X
10	30 + 30 + 30 = 90	4.0X
100	30 + 3 + 30 = 63	5.7X
Infinito	30 + 0 + 30 = 60	6.0X

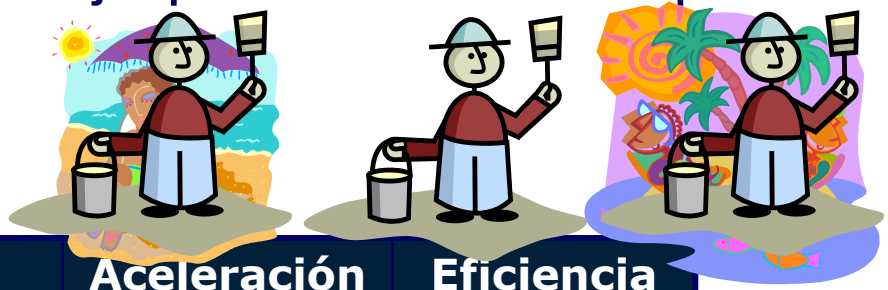
Illustra la ley de Amdahl

La aceleración potencial está restringida por la porción serial

- ¿Que pasa si el dueño de la barda usa un spray para pintar 300 tablas en una hora ?
  - Mejor algoritmo serial
  - Si no hay sprays disponibles para varios pintores, ¿cuál es la máxima paralelización?

Eficiencia

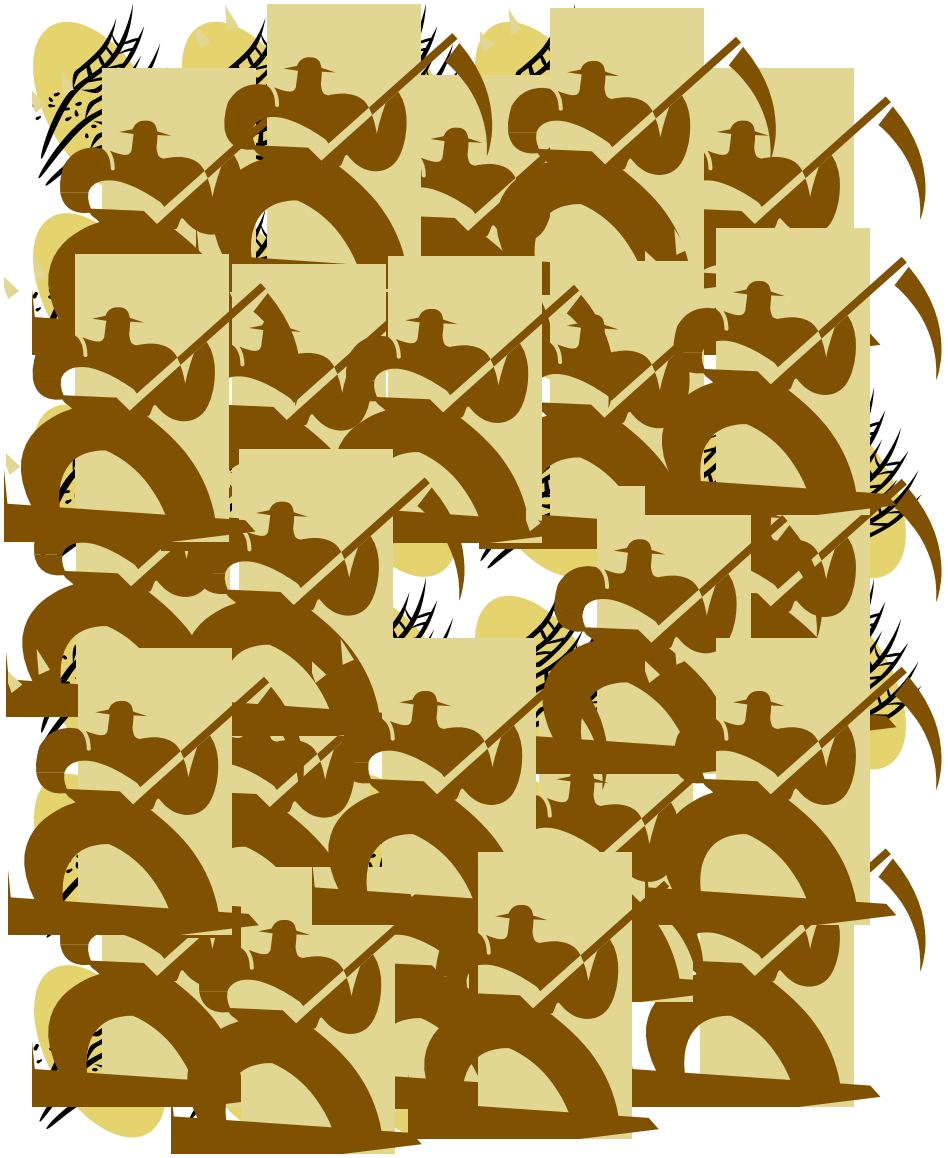
- **Medir que tan efectivamente los recursos de cómputo están ocupados**
  - Aceleración dividida entre el número de hilos
  - Expresada como porcentaje promedio de tiempo no ocioso



Número de pintores	Tiempo	Aceleración	Eficiencia
1	360	1.0X	100%
2	30 + 150 + 30 = 210	1.7X	85%
10	30 + 30 + 30 = 90	4.0X	40%
100	30 + 3 + 30 = 63	5.7X	5.7%
Infinito	30 + 0 + 30 = 60	6.0X	Muy baja

Granularidad

- Pobremente definido como tasa de computación a sincronización
- Asegurarse que hay suficiente trabajo que merezca cómputo en paralelo
- *Ejemplo: Dos granjeros dividen un campo. ¿Cuántos granjeros más se pueden agregar?*



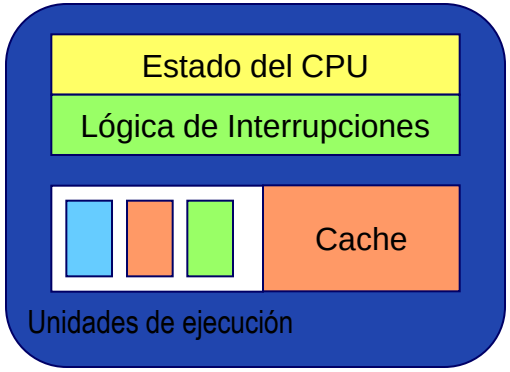
Balanceo de carga

- **La distribución más efectiva es tener la misma cantidad de trabajo por hilo**
  - Los primeros hilos que terminan se convierten en ociosos
  - Los hilos deben terminar casi al mismo tiempo
- **Ejemplo: *Poner las mesas de un banquete***
  - Lo mejor es asignar la misma cantidad de mesas a cada cuadrilla

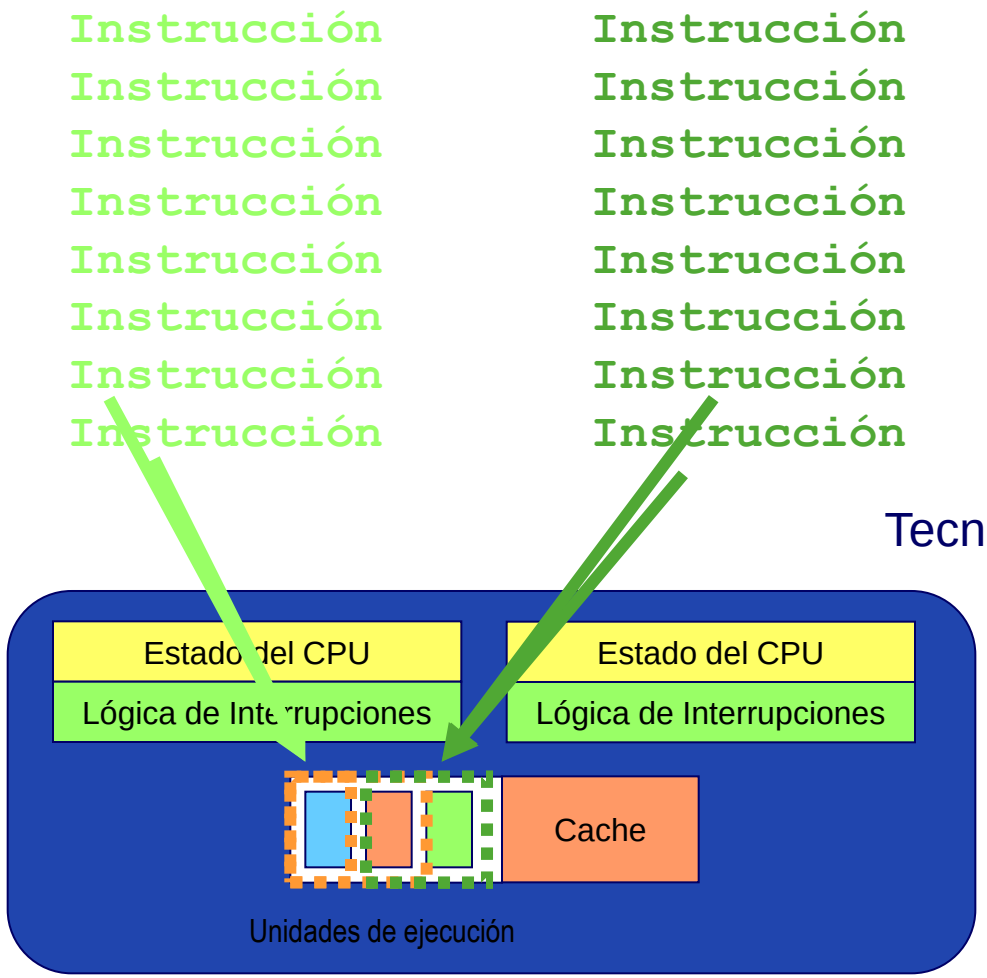
Hilos en plataforma single core

Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción

Single Core



Hilos en plataforma Hyper Threading



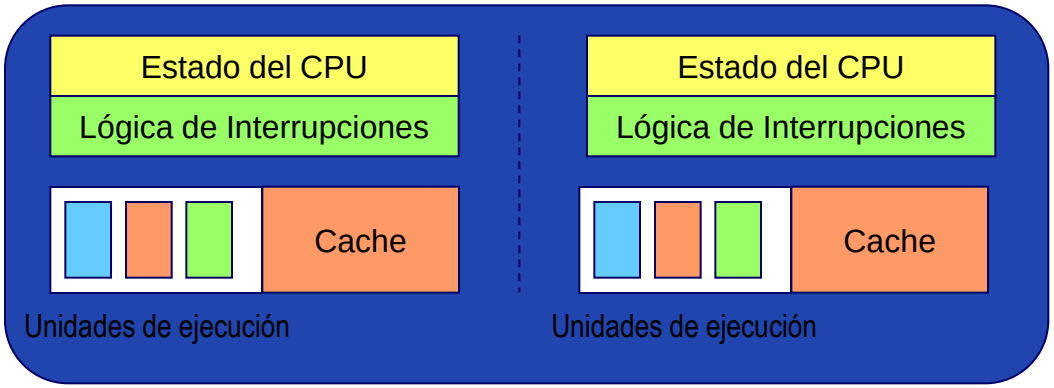
Tecnología Hyper Threading

Ambas instrucciones requieren diferentes unidades funcionales

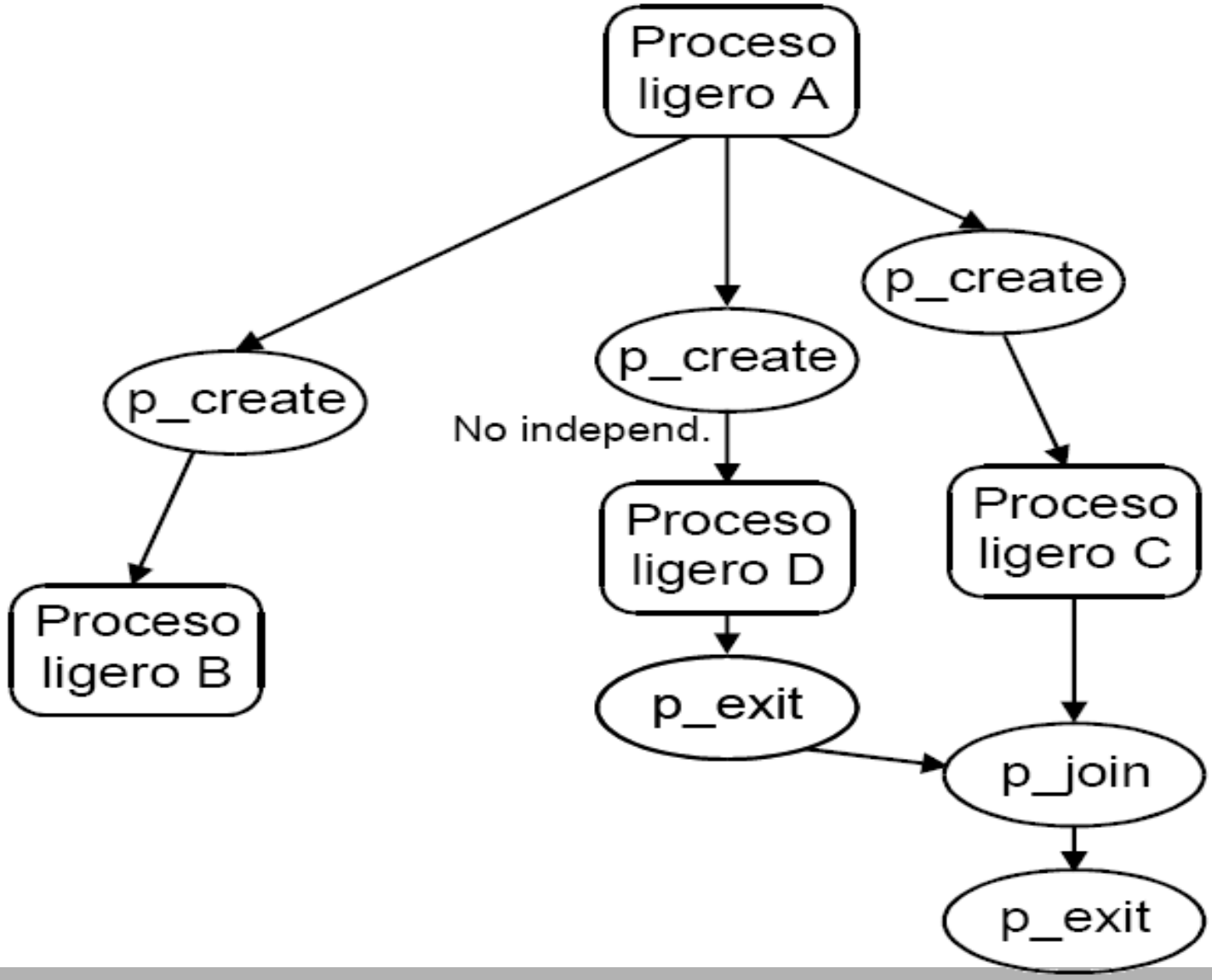
Multi-hilos en plataforma multi core

Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción
Instrucción	Instrucción

Multi Core



Jerarquía de Procesos Ligeros



Interrupciones

- Todas las computadoras proporcionan un mecanismo mediante el cual otros módulos (E/S, memoria) pueden interrumpir la ejecución normal del procesador.
- El objetivo de las interrupciones es ser una vía para mejorar la eficiencia del procesamiento.

Tipos de Interrupciones

- De programa.
- De reloj.
- De E/S.
- Por fallo de hardware.

Tipos de Interrupciones

- **De programa:**

  Generadas por alguna condición que se produce como resultado de la ejecución de una instrucción, como el desbordamiento aritmético, la división por cero, el intento de ejecutar una instrucción ilegal de la máquina o una referencia a una zona de memoria fuera del espacio permitido al usuario.

Tipos de Interrupciones

- **De reloj:**

  Generadas por un reloj interno del procesador. Esto permite al sistema operativo llevar a cabo ciertas funciones con determinada regularidad.

Tipos de Interrupciones

- **De Entrada/Salida:**

  Generadas por un controlador de Entrada/Salida, para indicar que una operación ha terminado normalmente o para indicar diversas condiciones de error.

Tipos de Interrupciones

- **Por fallo de hardware:**

  Generadas por fallos tales como un corte de energía o un error de paridad de la memoria.

Tratamiento de Interrupciones

1) **El dispositivo emite una señal de interrupción al procesador**
2) **El procesador finaliza la ejecución de la instrucción en curso, antes de responder a la interrupción.**
3) **El procesador pregunta por la interrupción, comprueba que hay una y envía una señal de reconocimiento al dispositivo que generó la interrupción.**

Tratamiento de Interrupciones

4) El procesador necesita ahora prepararse para transferir el control a la rutina de interrupción.
5) El procesador carga ahora el contador de programa, con la ubicación de entrada del programa de tratamiento de la interrupción, que responderá a esta interrupción.
6) En este punto, el contador de programa y la PSW relativa al programa interrumpido se han salvado en la pila de sistema. (cambio de contexto)

Tratamiento de Interrupciones

7) La rutina de tratamiento de la interrupción puede ahora proceder a procesar la interrupción.
8) Cuando se completa el tratamiento de la interrupción, se recuperan de la pila los registros de los valores que se salvaron y se restauran sobre los registros del procesador.
9) La operación final es restaurar los valores de la PSW y del contador de programa a partir de la pila. Como resultado, la próxima instrucción ejecutada será la del programa interrumpido previamente.