Topic 5

Operating Systems in Industrial Applications

5 Operating Systems

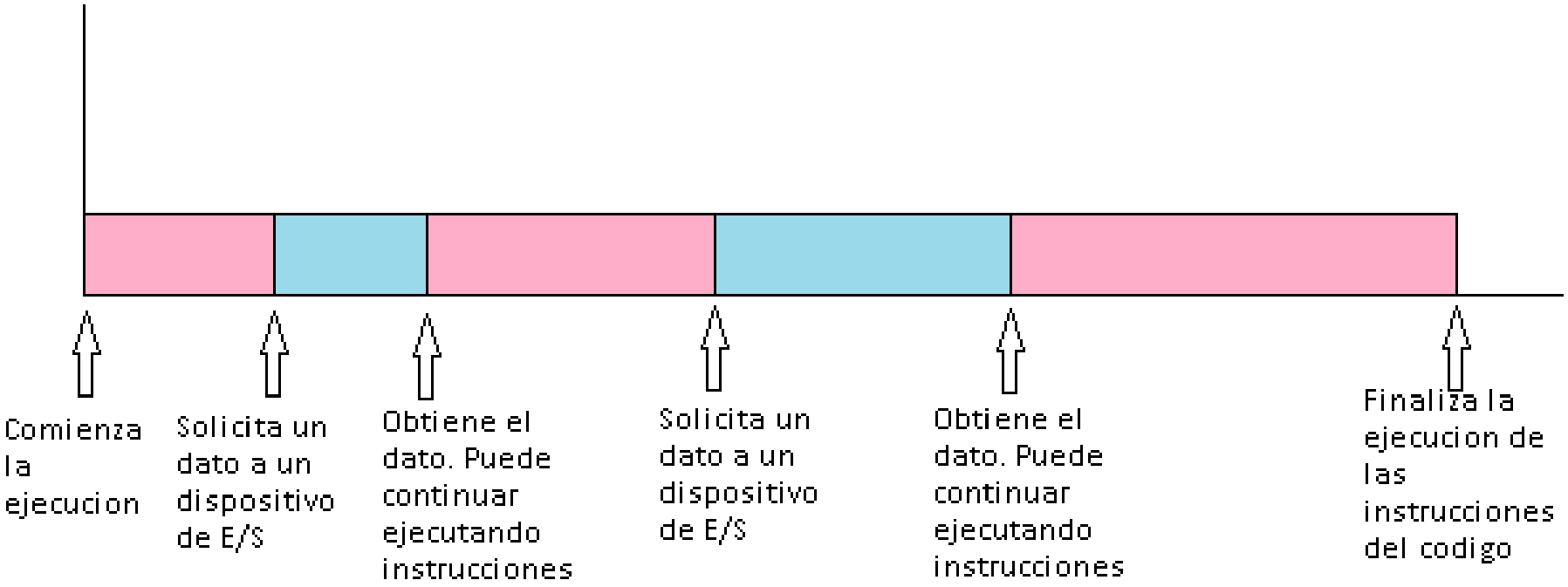
- La informatica industrial a menudo trata de la programacion de pequeños sistemas informáticos sin muchos recursos.
- Si aumentamos la complejidad del hardware o de los algoritmos a emplear, necesitaremos de otra aplicación que nos ofrezca las funciones del sistema
  - Sistema Operativo
    - Aplicación industrial
      - se ejecuta en modo usuario
    - Sistema Operativo
      - en modo supervisor

5 Operating Systems

- **Tarea o Proceso → Programa en ejecucion**
- **Tarea es cualquier programa que se encuentre cargado en memoria desde la que es procesado por la CPU.**
- **Caracteristicas de las tareas**
  - **Deben ser lo mas independientes posibles del resto de tareas. No deben compartir datos con otras tareas**
  - **Tendrá sistemas para** intercambiar informacion con es exterior (E/S) y con el resto de tareas (sincronizacion)
  - Dispondrá de areas de memoria propias

5 Operating Systems

- **Tarea o Proceso → Programa en ejecucion**



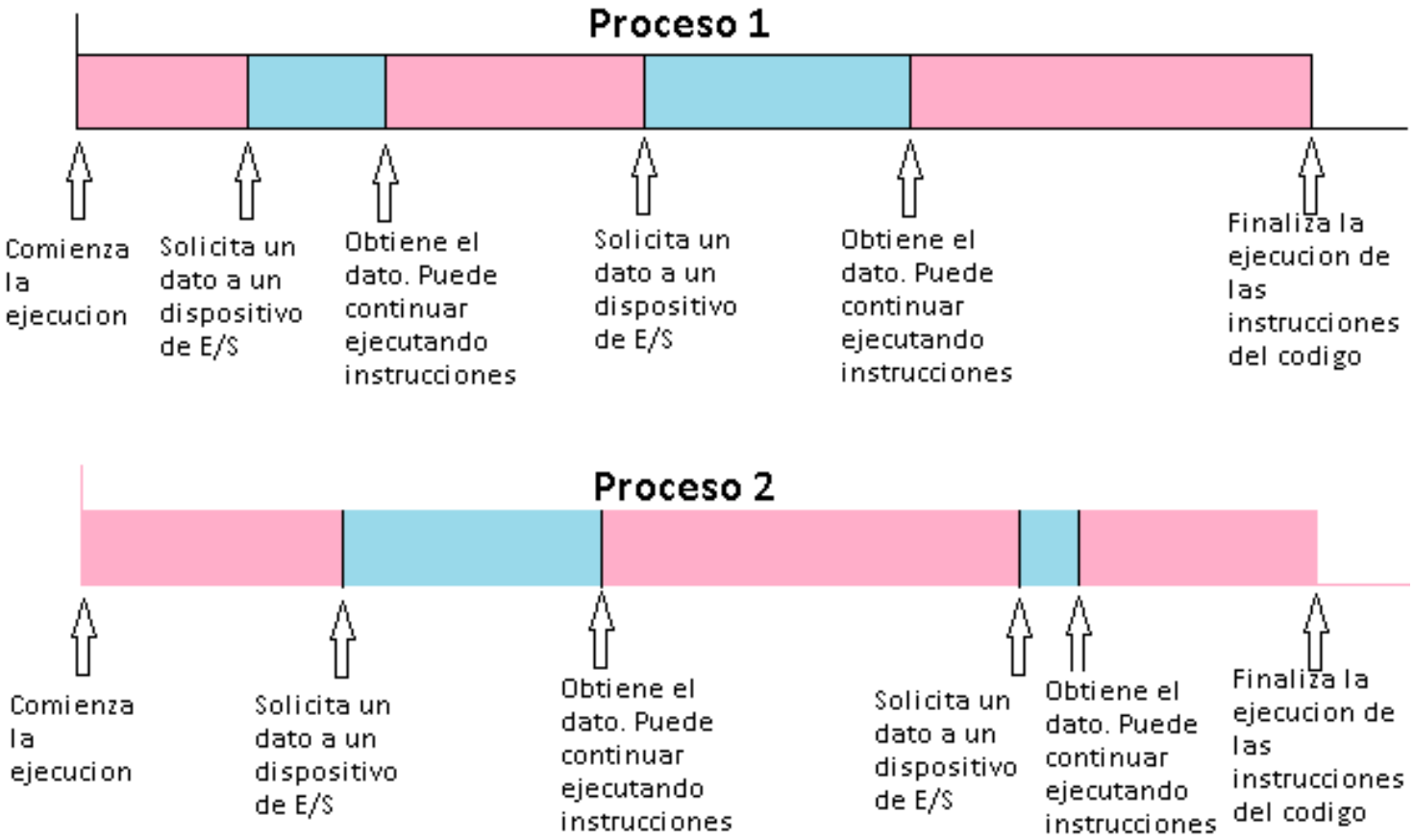
5 Operating Systems

- **Multitarea**
  - **Monotarea.**
    - **Sistema informatico que solo puede mantener simultaneamente un programa de aplicación cargado en memoria**

5 Operating Systems

Multitarea

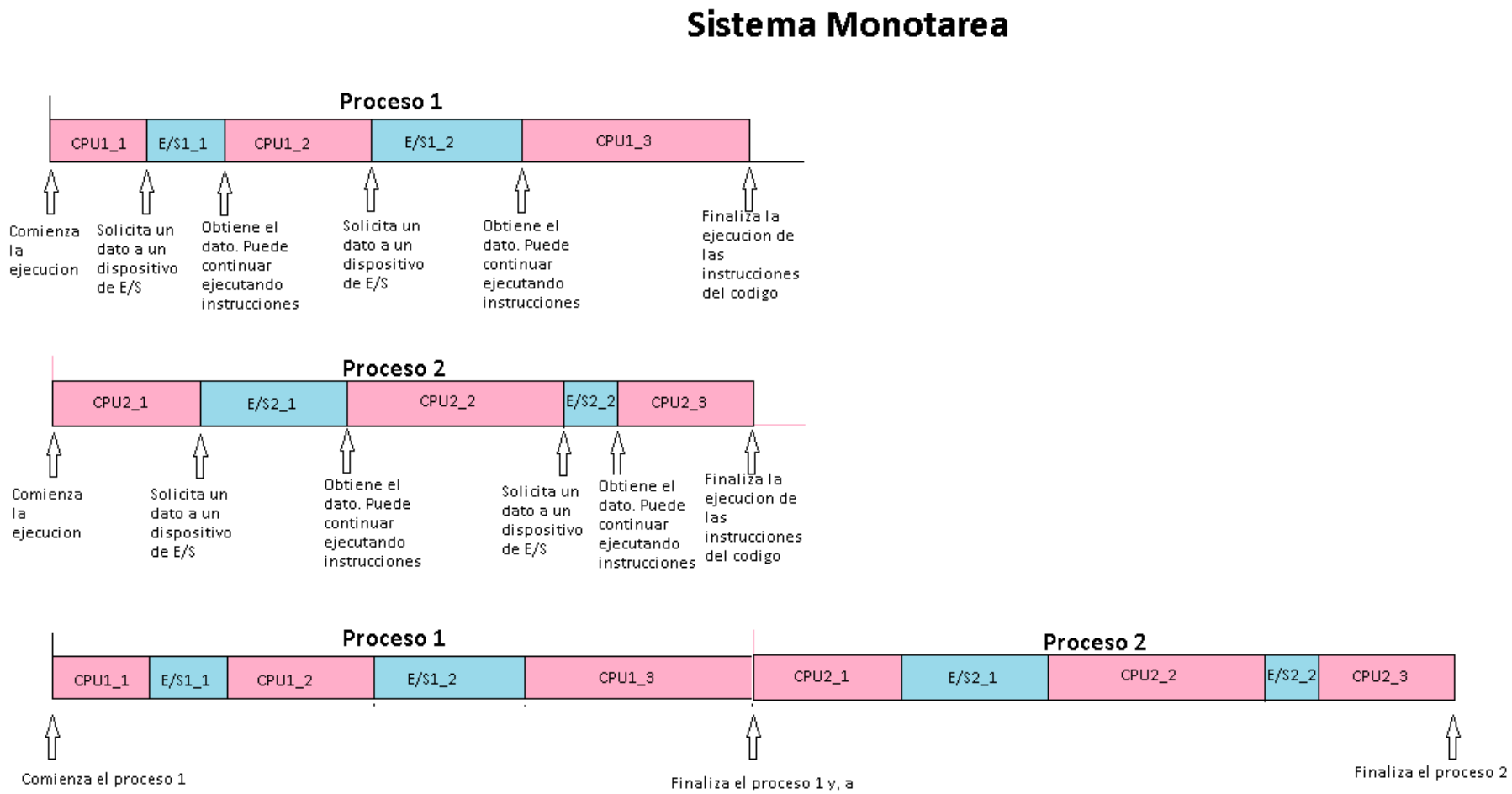
- **Monotarea.**



5 Operating Systems

Multitarea

Monotarea.



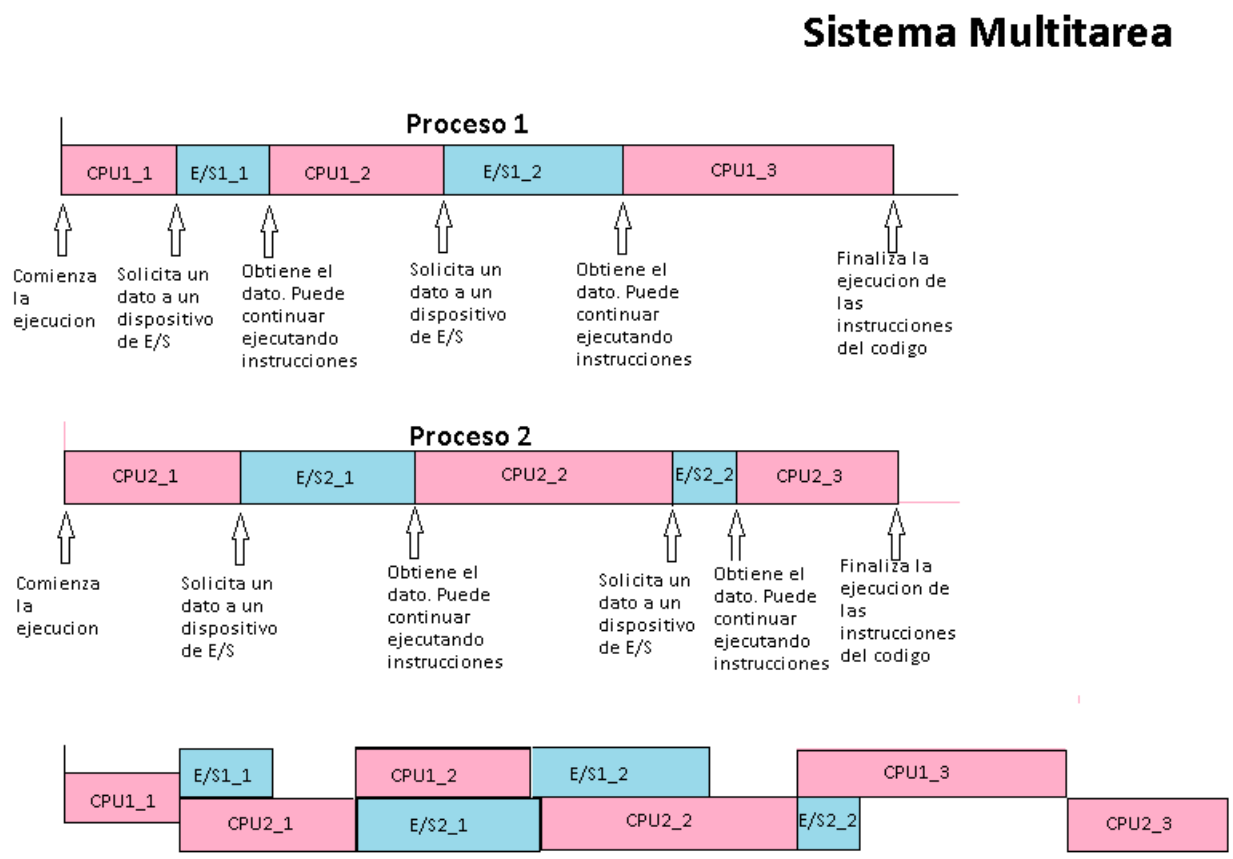
5 Operating Systems

- **Multitarea**
  - **Monotarea.**
    - **Sistema informatico que solo puede mantener simultaneamente un programa de aplicación cargado en memoria**
  - **Multitarea.**
    - **Permite a varios programas simultaneamente**
    - **Paralelismo de grano gordo. ->Dan la sensacion de que varios trabajos independientes se ejecutan en paralelo**
    -
    - **Se aprovecha los tiempos de espera de una tarea para que ejecuten instrucciones otras tareas**

5 Operating Systems

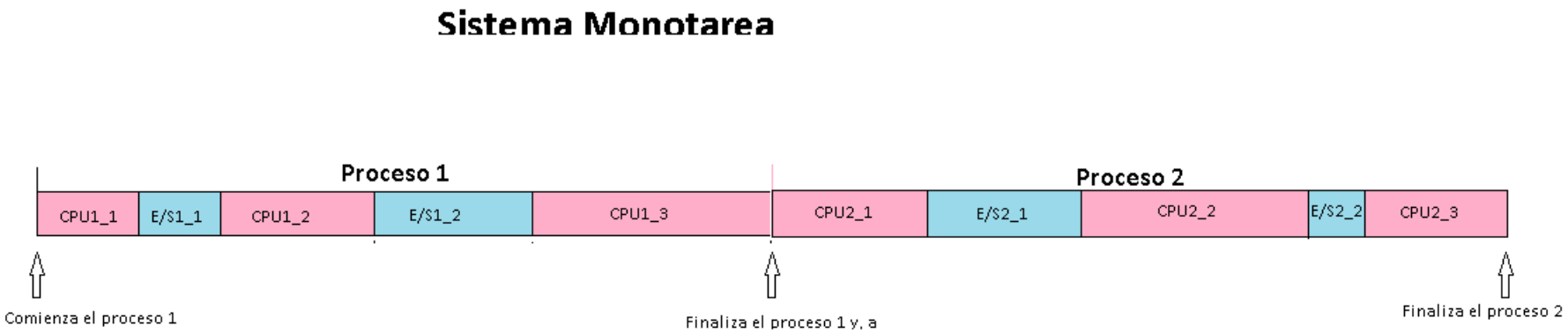
Multitarea

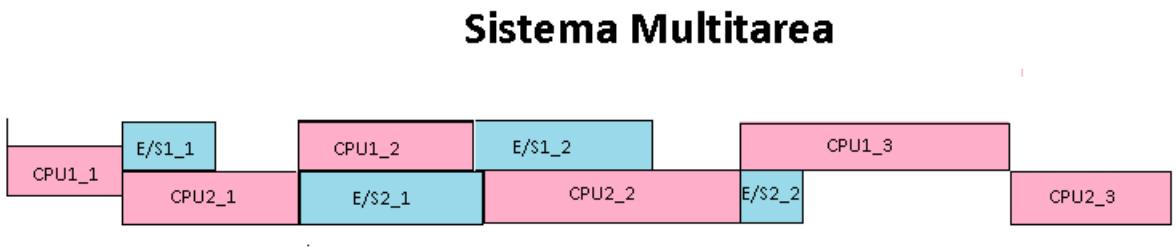
Multitarea.



5 Operating Systems

Multitarea





5 Operating Systems

- **Procesos**
  - El Sistema Operativo se encarga de administrar el tiempo, sincronizar y comunicar entre si a los procesos
  - Permite a tareas trabajar mientras otras esperan otros eventos
  - En un mismo sistema pueden existir tareas de tiempo real con tareas sin requerimientos temporales estrictos
  - El Sistema Operativo es el encargado de asegurar que se cumplen los tiempos de las tareas de TR

5 Operating Systems

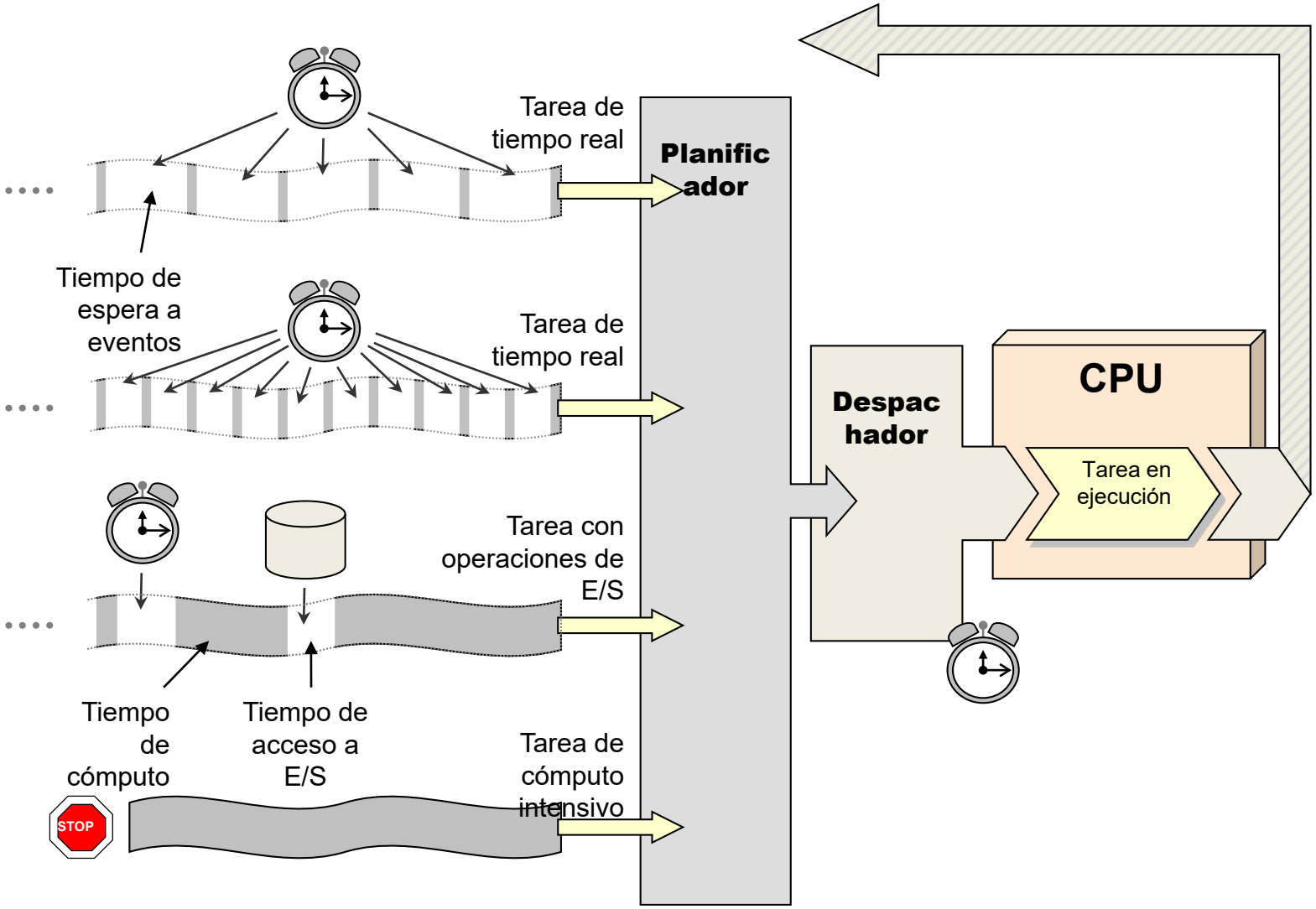


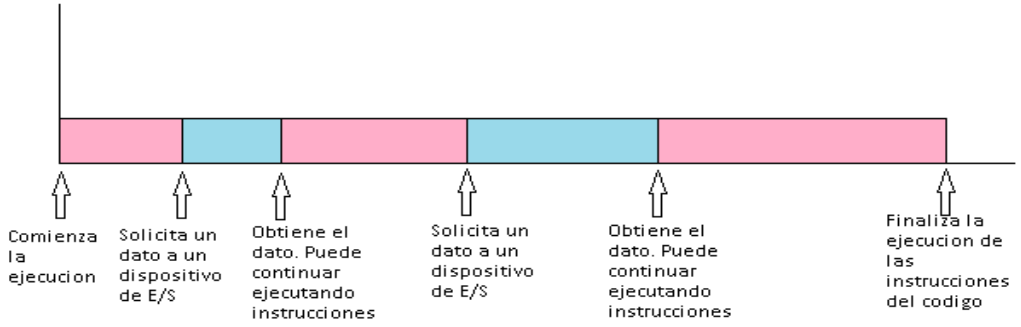
Figura -. Colaboración entre planificador y despachador.

5 Operating Systems

- **Procesos**
  - **Proceso→ Se pueden ver como contenedores de recursos**
    - **Tareas que ejecutan codigo**
    - **Memoria**
    - **Descriptores de archivos**
    - **Objetos que indican elementos software o hardware que se tienen en uso**

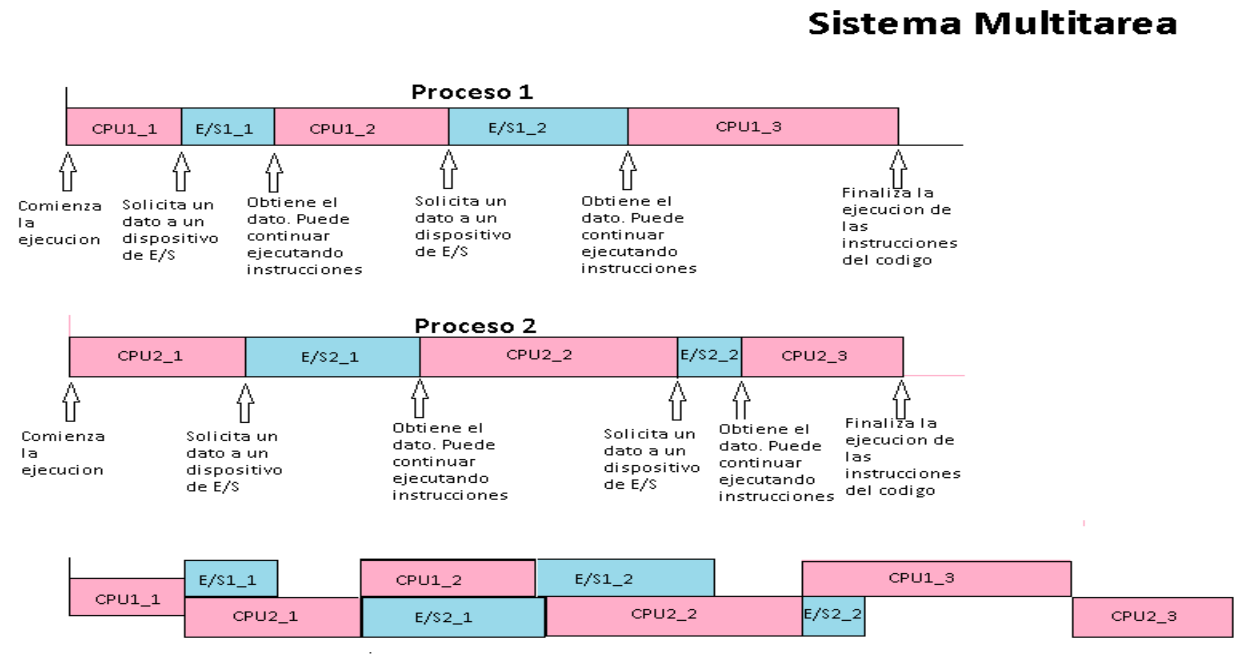
5 Operating Systems

- **Si una aplicación necesita realizar más de un trabajo,¿Cómo lo estructuramos?**
  - **Solucion 1**
    - Realizar una aplicación que vaya ejecutando en orden todos los trabajos que hay que realizar.
  - **Ventajas**
    - Sencillo
  - **Inconveniente**
    - No es eficiente. Si en uno de los trabajos se accede a hardware, toda la aplicación debe esperar a que el hardware responda



5 Operating Systems

- **Si una aplicación necesita realizar más de un trabajo,¿Cómo lo estructuramos?**
  - **Solucion 2**
    - Dividir toda la aplicación en trabajos, y asignar cada uno de ellos a un procesos.
    - Si necesito comunicar datos entre ellos, utilizar metodos de comunicación entre procesos (pipes, sockets, buffers de memoria,…)

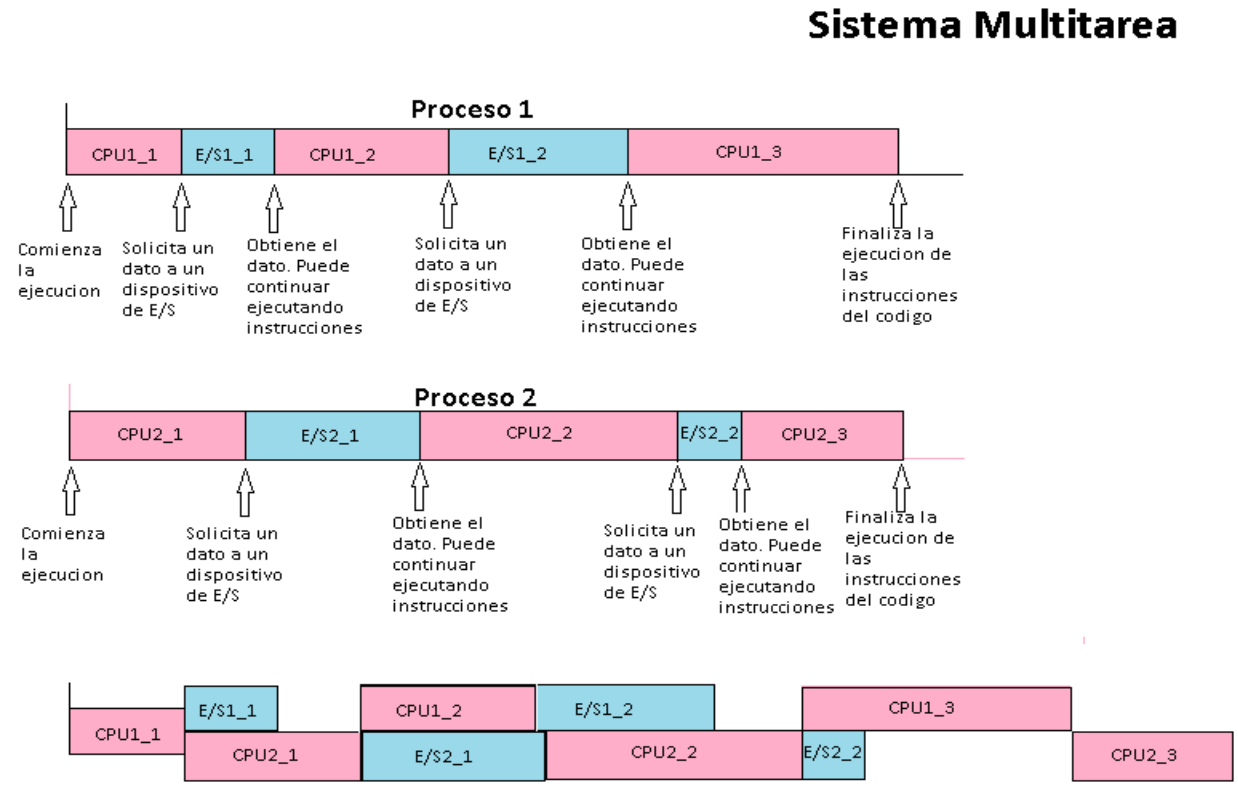


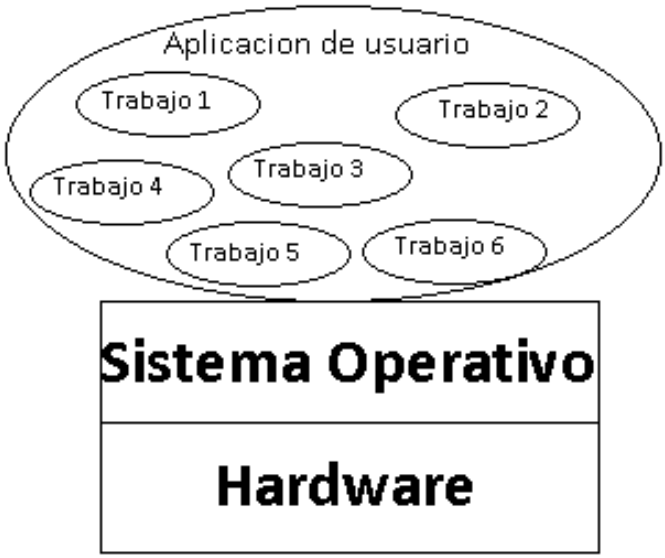
- **Si una aplicación necesita realizar más de un trabajo,¿Cómo lo estructuramos?**
  - **Solucion 2**
    - Dividir toda la aplicación en trabajos, y asignar cada uno de ellos a un procesos.
    - Si necesito comunicar datos entre ellos, utilizar metodos de comunicación entre procesos (pipes, sockets, buffers de memoria,…)
  - **Ventajas**
    - Puede incrementar el rendimiento de la aplicación
    - Disminuye el tiempo de respuesta por la ejec. paralelo
  - **Inconveniente**
    - Se duplican los recursos
    - La comunicación entre procesos puede llegar a ser lenta

- **Si una aplicación necesita realizar más de un trabajo,¿Cómo lo estructuramos?**
  - **Solucion 3**
    - Permitir una ejecucion paralela de los diferentes trabajos **en el mismo proceso**
    - No necesito comunicar datos entre los flujos, ya que estos pueden acceder a todos los recursos
  - **Surgen para poder obtener esta solucion los subprocesos (o procesos ligeros o hilos)**

5 Operating Systems

- Si una aplicación necesita realizar más de un trabajo,¿Cómo lo estructuramos?
  - Solucion 3





5 Operating Systems

- El proceso es un contenedor de los recursos que utilice la aplicación
  - Memoria
  - Descriptores de archivos
  - Objetos que indican elementos software o hardware que se tienen en uso
  - Y..... Los hilos de los que haga uso

Hilos

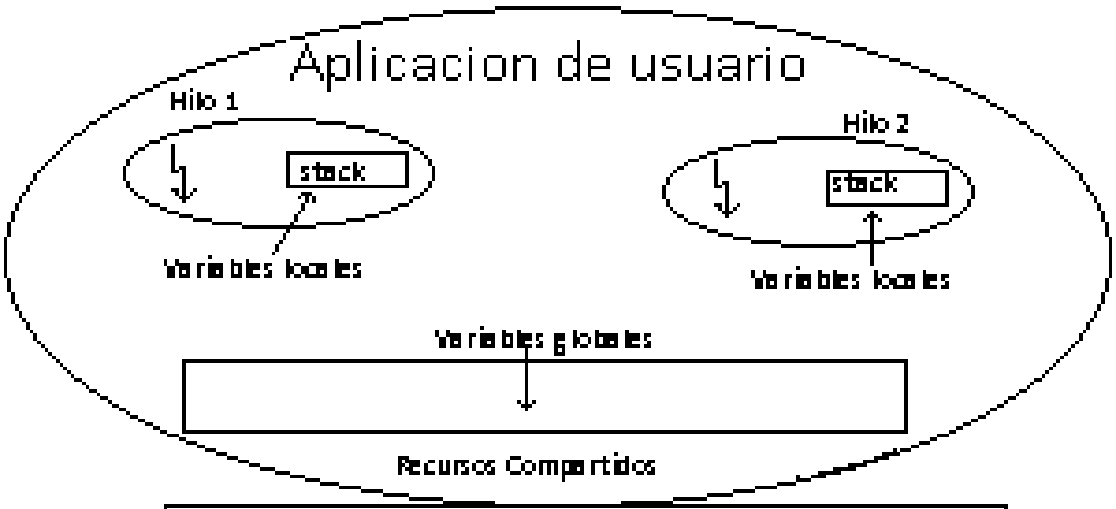
- Cada proceso tendrá por lo menos un hilo
- Los hilos son entes de ejecución de código, tareas que comparten recursos
- Estado de un hilo lo define
  - Su propia pila
  - Los registros del procesador a nivel de usuario
  - Una estructura interna del núcleo con informacion adicional del estado del hilo
- En la pila es donde se almacena la parte fundamental de su estado, que incluye fundamentalmente su estado

- Hilos


Aplicacion de usuario
Hilo 1
stack
Hilo 2
stack
Recursos Compartidos
Sistema Operativo
Hardware

5 Operating Systems

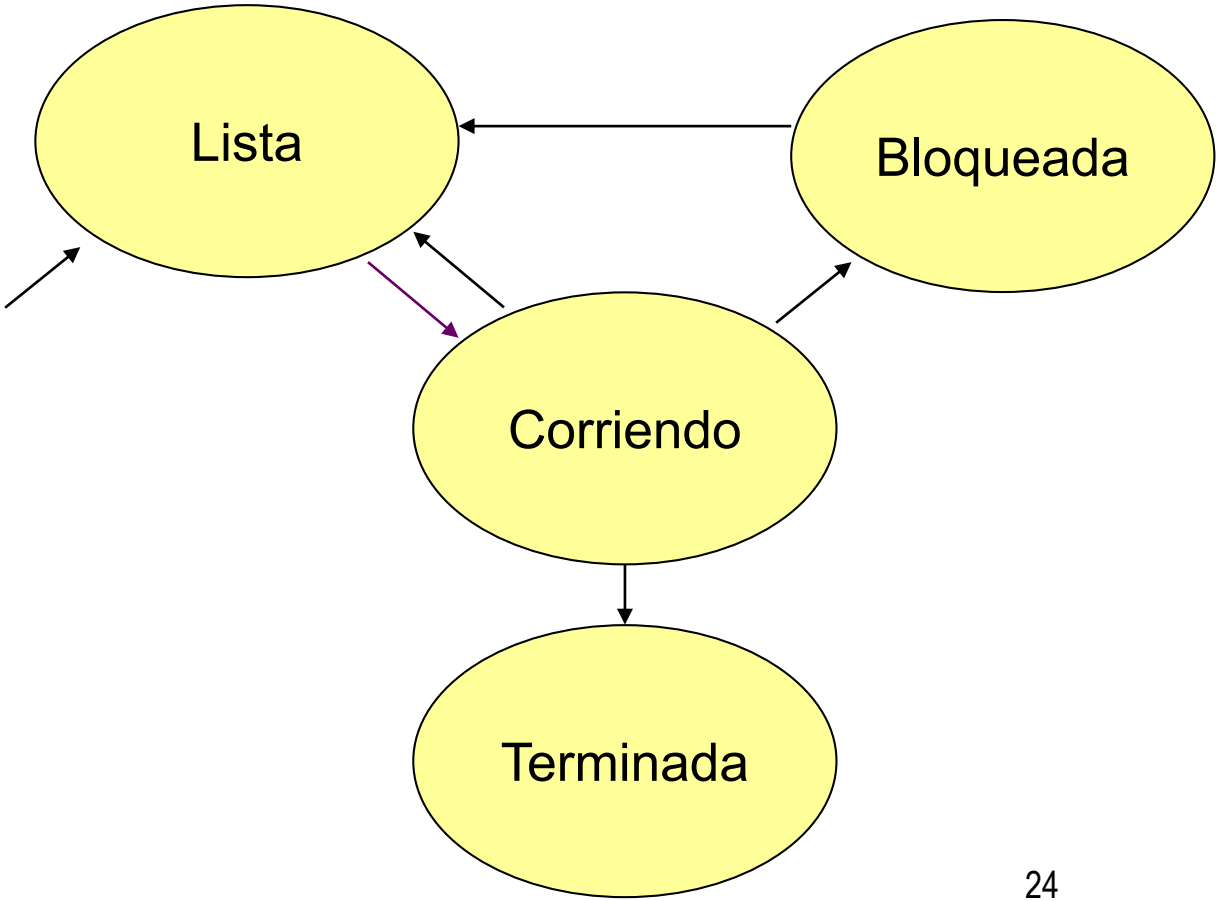
- **Hilos**



- **Hilos**
  - **Todos los recursos son compartidos por todos los hilos del proceso**
  - **Las variables globales y la memoria dinámica (como vimos está en la zona de memoria asignada a la parte de código) será compartida por todos los procesos→ Problema de concurrencia**

Hilos

Estados de un hilo



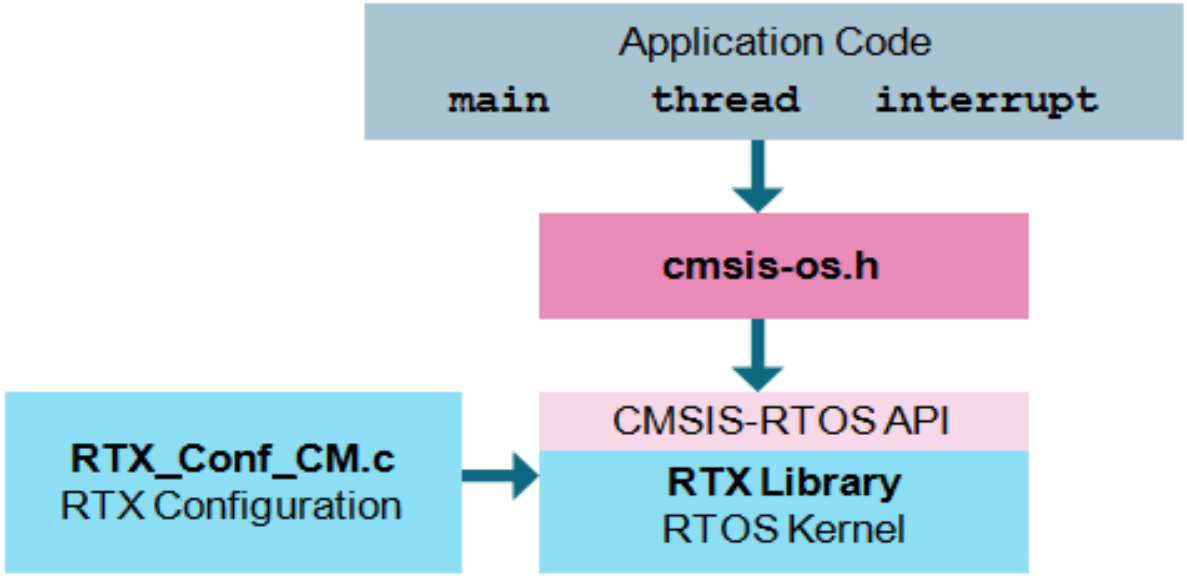
5 Operating Systems

- **Creacion de Hilos.**
  - Los sistemas operativos multihilados ofrecen a los usuarios la forma de crear hilos.
  - Nosotros en esta asignatura nos vamos a basar en un Sistema Operativo soportado por Cortex-M3: RTX.
  - RTX es un sistema operativo de tiempo real creado para dispositivos basados en el procesador Cortex-M. El Kernel de RTX se puede utilizar para crear aplicaciones que realizan varias tareas al mismo tiempo. Permite la programacion de aplicaciones de usuario usando el estándar de C y C++ y compilados con ARMCC, GCC o IAR compilator.

5 Operating Systems

Creacion de Hilos.

- El CMSIS-RTOS es una API (Interfaz de programación de aplicaciones) común para RTOS. Proporciona una interfaz de programación estándar que es portátil para muchos RTOS y por lo tanto permite que las plantillas de Software, middleware, bibliotecas y otros componentes soportados por RTOS
- El RTX CMSIS-RTOS gestiona los recursos del micro implementa el concepto de hilos paralelos que se ejecutan simultáneamente.



Creacion de Hilos.

- **osThreadId osThreadCreate (const osThreadDef_t *thread_def, void *argument)**
  - **Crea un hilo y lo añadie a la lista de Hilos Activos en el estado de READY**
  - **Parameters:**
    - [entrada] thread_def Puntero a una definicion de hilo realizada mediante osThreadDef y referenciada con osThread.
    - [entrada] argument Puntero a una variable que se pasa a la funcion del hilo como un argumento de entrada. t.
  - **returns:** ID del hilo para referenciar por otras funciones o NULL en caso de error.

- **Inicia hilo asignado a una funcion y lo añade a la lista hilos activos, estableciendo su estado como READY. La función del hilo recibe el *argumento* como argumento de la función cuando se ésta se inicia. Cuando la prioridad de la función del hilo creado es superior al hilo RUNNING, el hilo creado comienza inmediatamente y se convierte en el nuevo hilo RUNNING .**

5 Operating Systems

Creacion de Hilos.

- La definicion de un hilo se realiza en dos fases:
  - Primero se define la funcion que va a ejecutar
    - void FuncionHilo(void const *arg);
  - A continuacion se define el tipo de hilo
    - osThreadDef (funtion_name,priority,instancias,stacksize)
- La prioridad del hilo tendra los valores
  - osPriorityIdle = -3,
    osPriorityLow = -2,
    osPriorityBelowNormal = -1,
    osPriorityNormal = 0,
    osPriorityAboveNormal = +1,
    osPriorityHigh = +2,
    osPriorityRealtime = +3,
    osPriorityError = 0x84
- Instancias: numero maximo de hilos con esta definicion que se crearan.
- StackSize. Tamaño de la pila del hilo (en bytes). 0->Tamaño standar.

Creacion de Hilos.

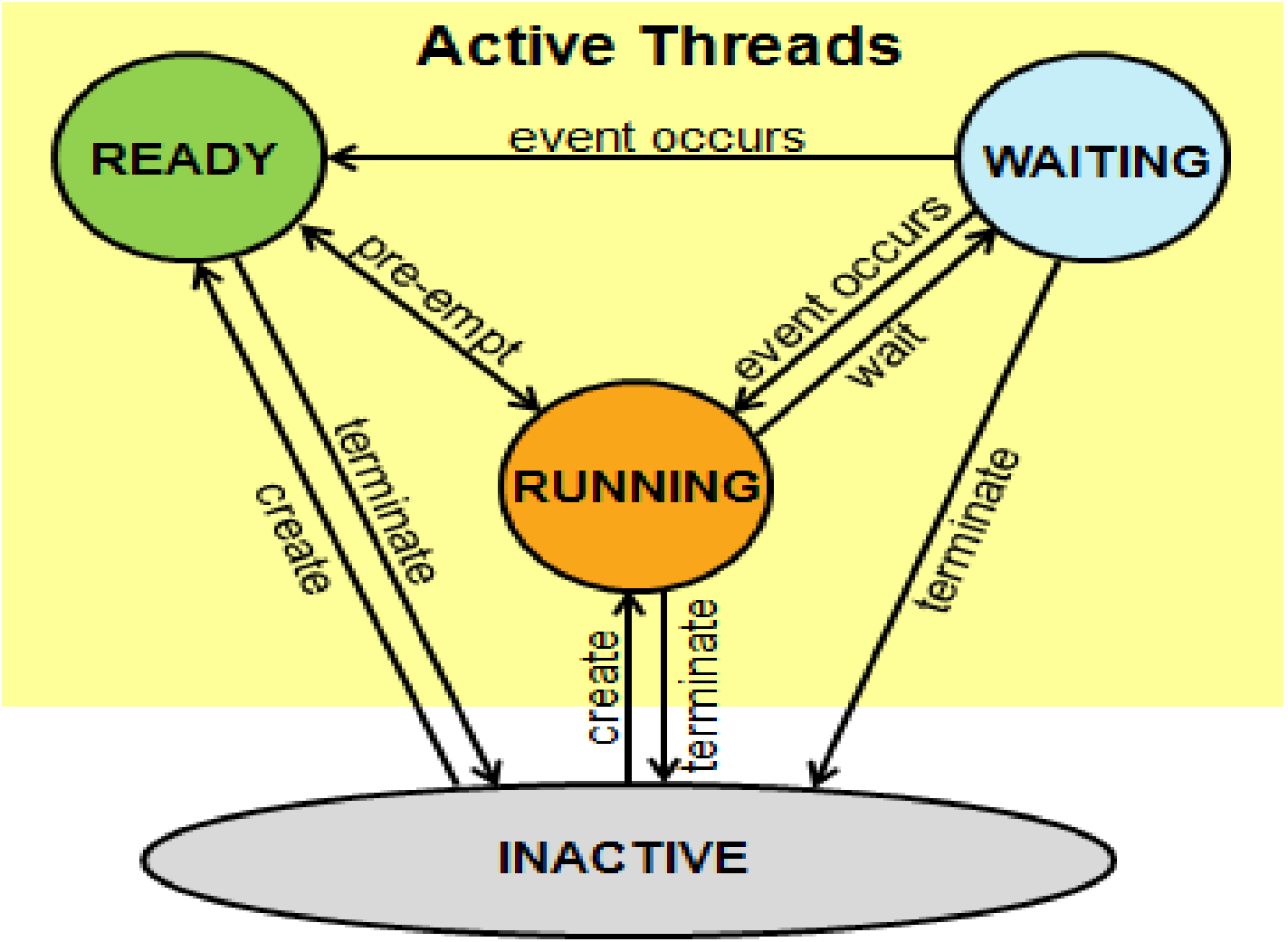
- **Example**

```
#include "cmsis_os.h“

void Thread_1 (void const *arg); // function prototype for Thread_1
osThreadDef (Thread_1, osPriorityNormal, 1, 0); // define Thread_1

void ThreadCreate_example (void) {
osThreadId id;
id = osThreadCreate (osThread (Thread_1), NULL); // create the thread
if (id == NULL)
                    { // handle thread creation
                   // Failed to create a thread
                   }
osThreadTerminate (id); // stop the thread
}
```

5 Operating Systems



Creacion de Hilos.

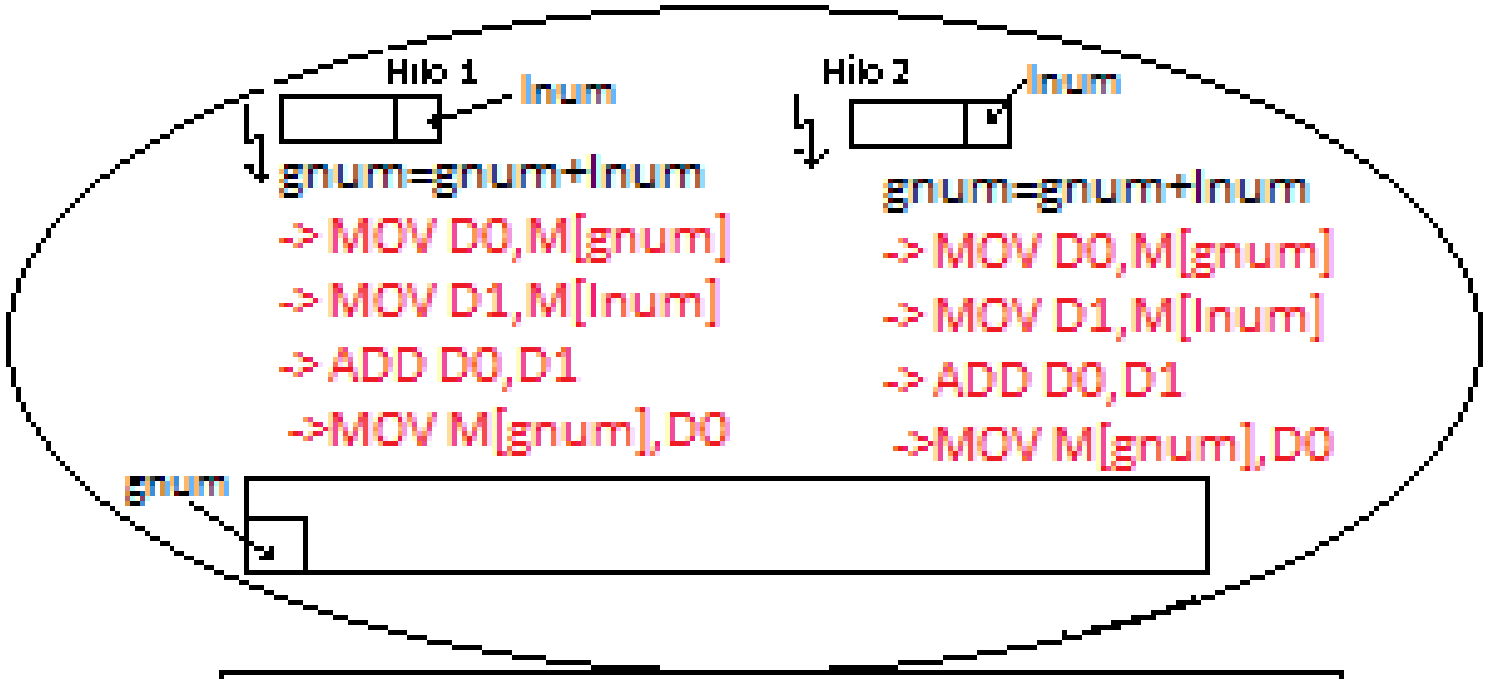
- osThreadId osThreadGetId (void)
  - Retorna el identificador osThreadId del hilo que la llama
- osStatus osThreadTerminate (osThreadId thread_id)
  - Finaliza la ejecucion del hilo que la llama y lo borra de la lista de Hilos Activos.
- osStatus osThreadSetPriority (osThreadId thread_id, osPriority priority)
  - Cambia la prioridad del hilo thread_id a la prioridad priority.

Concurrencia

- Si dos sistemas pretenden usar simultaneamente un mismo recurso se dice que lo usan concurrentemente
- Eso puede provocar errores
  - de codigo→ modificaciones de la memoria compartida.
  - Dispositivo hardware→corrupcion de datos

- g_num=gnum+num
  - MOV D0,Mem(g_num)
  - MOV D1,Mem(num)
  - ADD D0,D1
  - MOV Mem(g_num),D0

5 Operating Systems



Sistema Operativo

Hardware

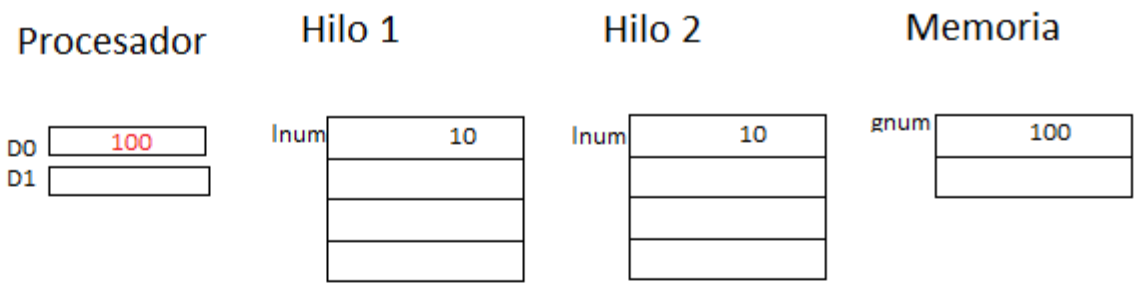
5 Operating Systems

5 Operating Systems

SSOO entrega CPU Hilo 1
->MOV D0,M[gnum]

5 Operating Systems

SSOO entrega CPU Hilo 1
->MOV D0,M[gnum]

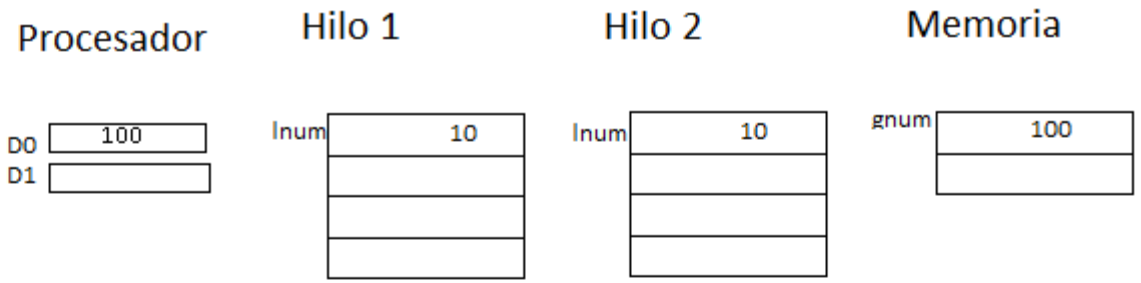


5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

->MOV D1,M[lnum]

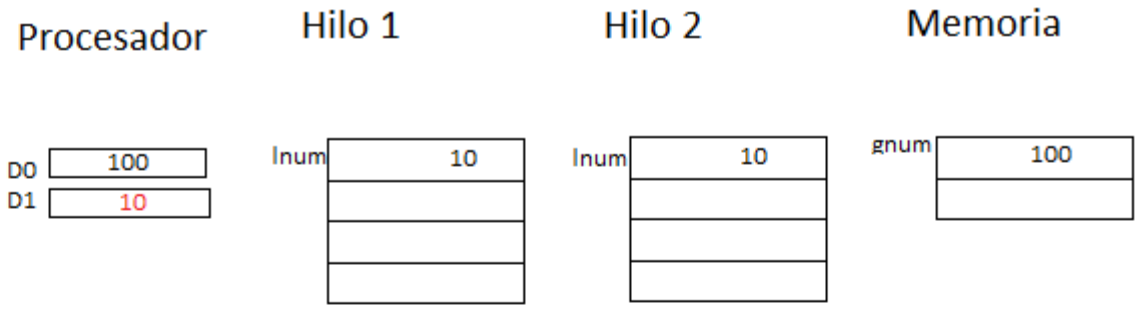


5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

->MOV D1,M[lnum]



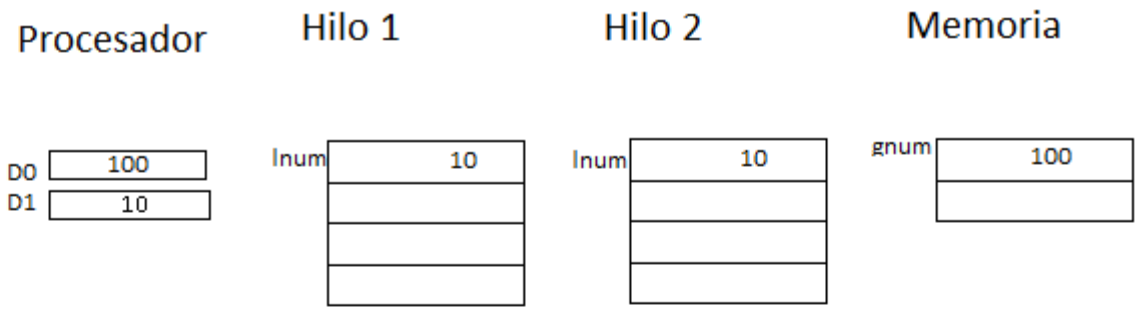
5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

-> ADD D0,D1



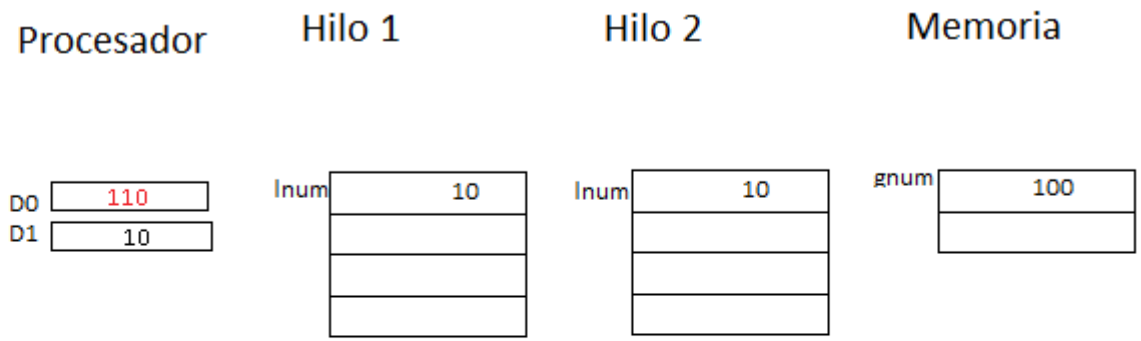
5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

-> ADD D0,D1



5 Operating Systems

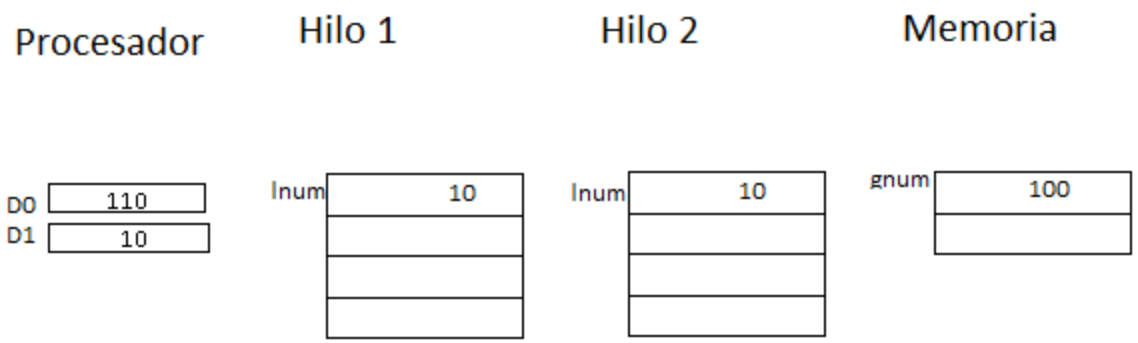
SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

->MOV M[gnum],D0



5 Operating Systems

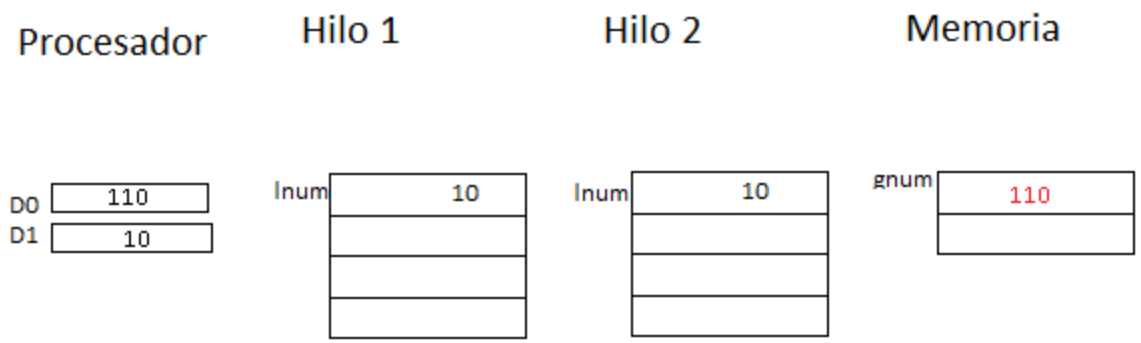
SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

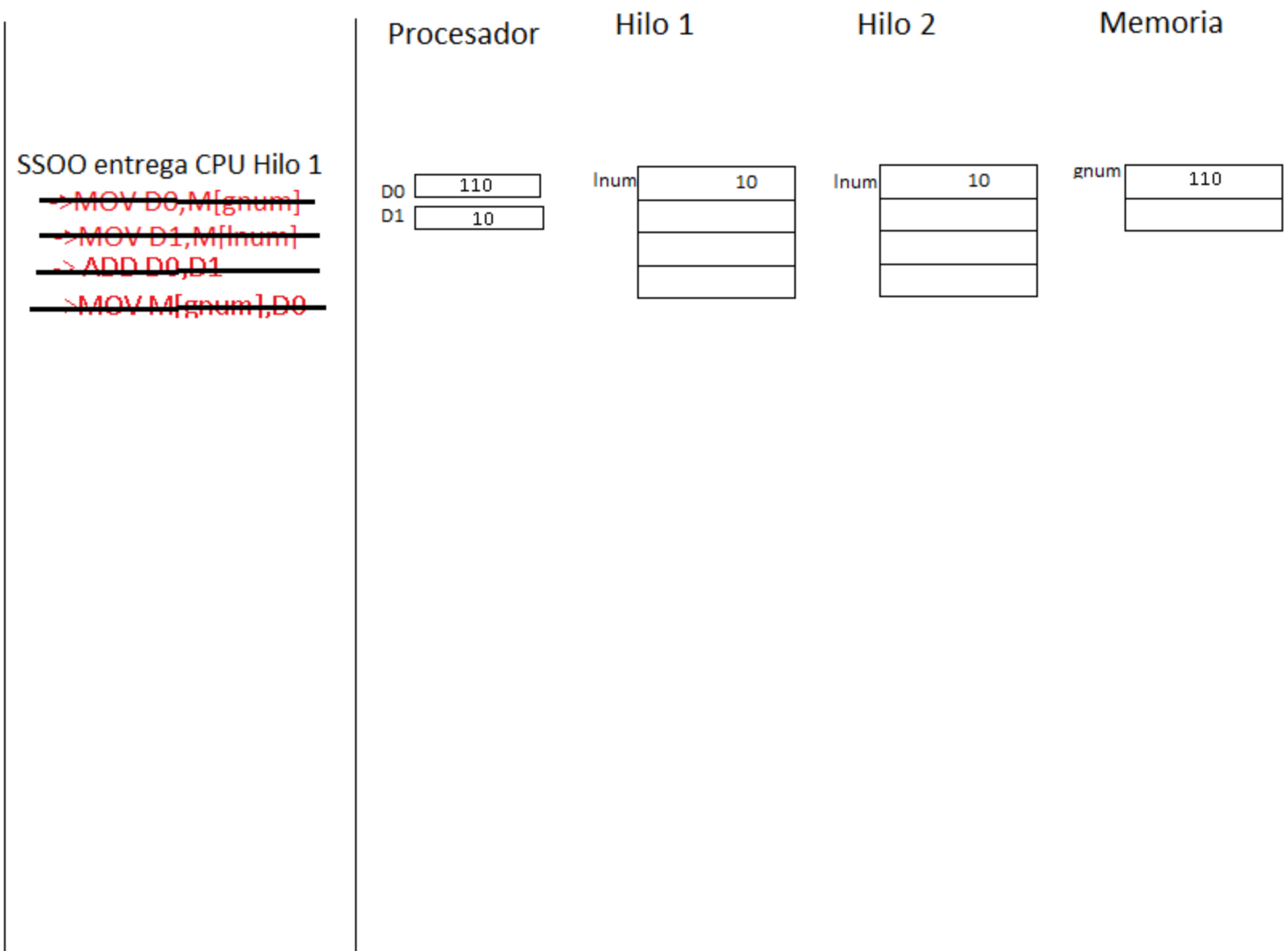
~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

->MOV M[gnum],D0



5 Operating Systems



5 Operating Systems

SSOO entrega CPU Hilo 1

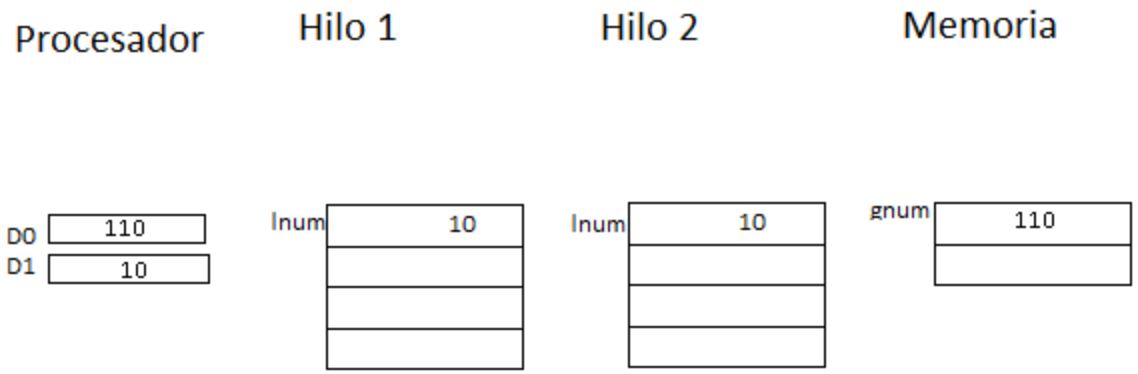
~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila



5 Operating Systems

SSOO entrega CPU Hilo 1

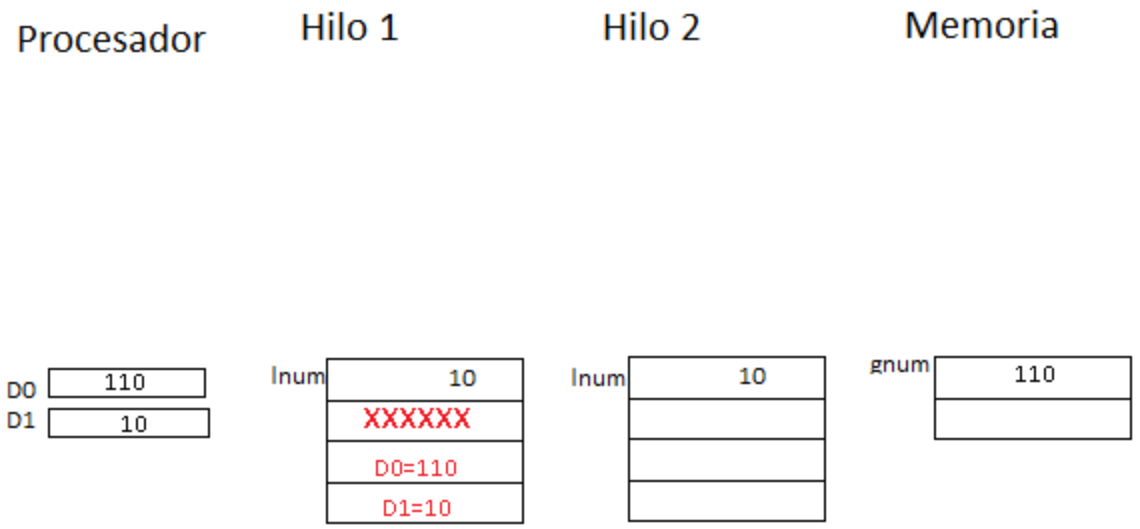
~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

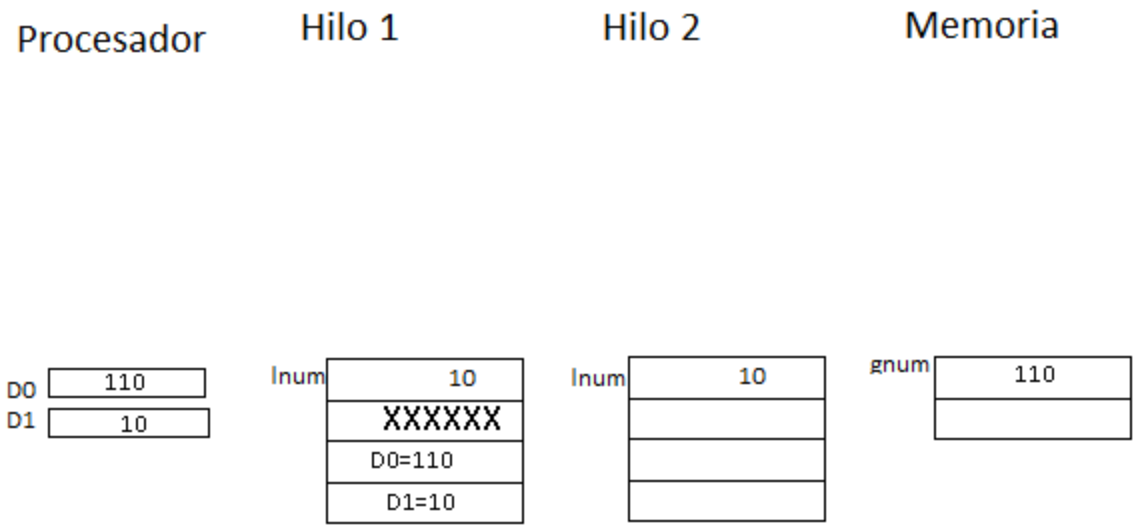
~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

-> MOV D0,M[gnum]



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

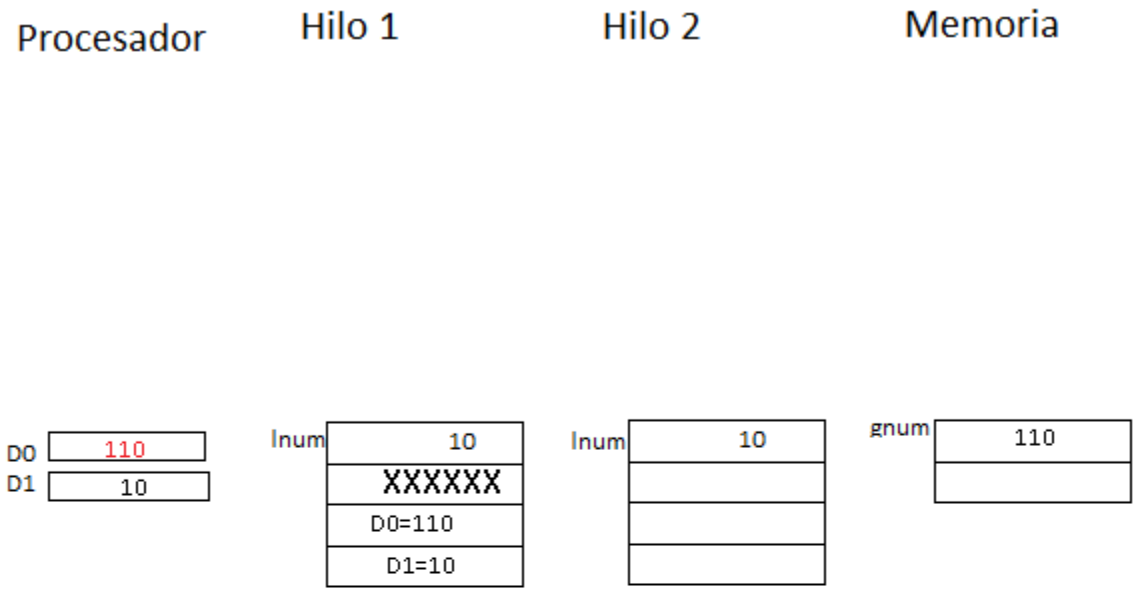
~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

-> MOV D0,M[gnum]



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

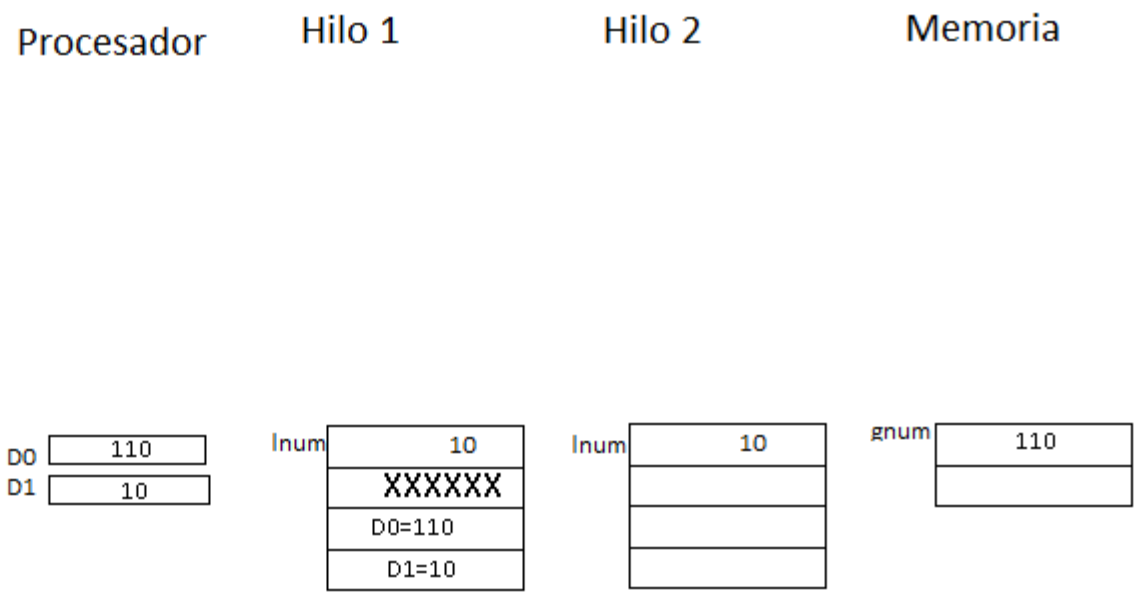
~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

-> MOV D1,M[lnum]



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

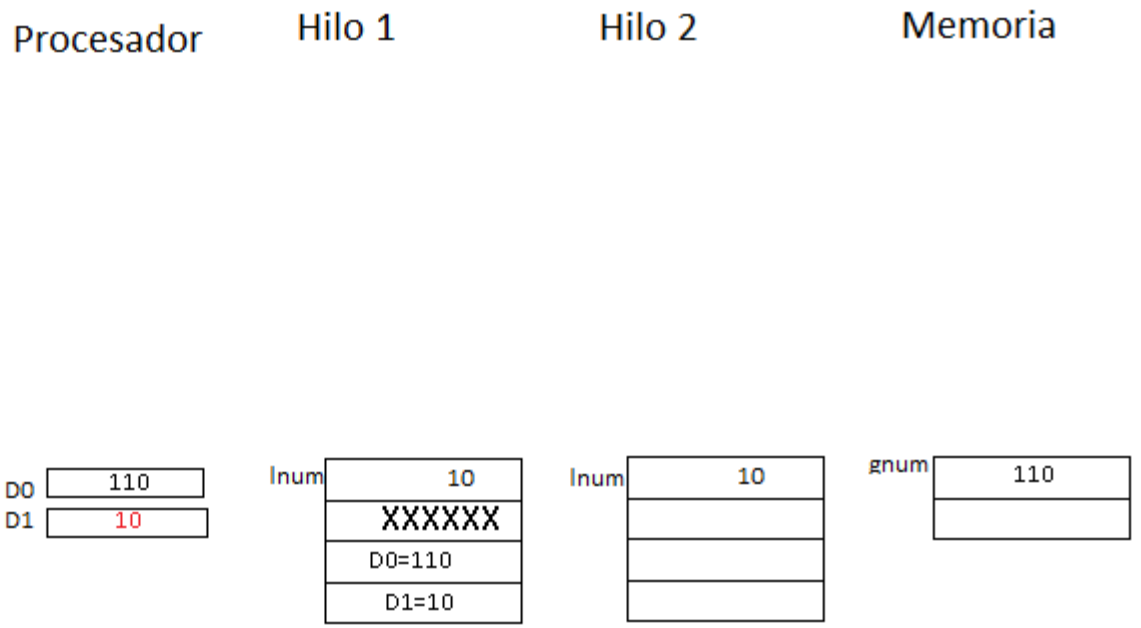
~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

-> MOV D1,M[lnum]



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

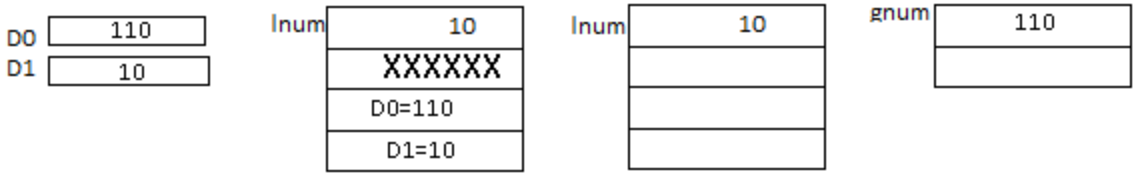
~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

-> ADD D0,D1



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

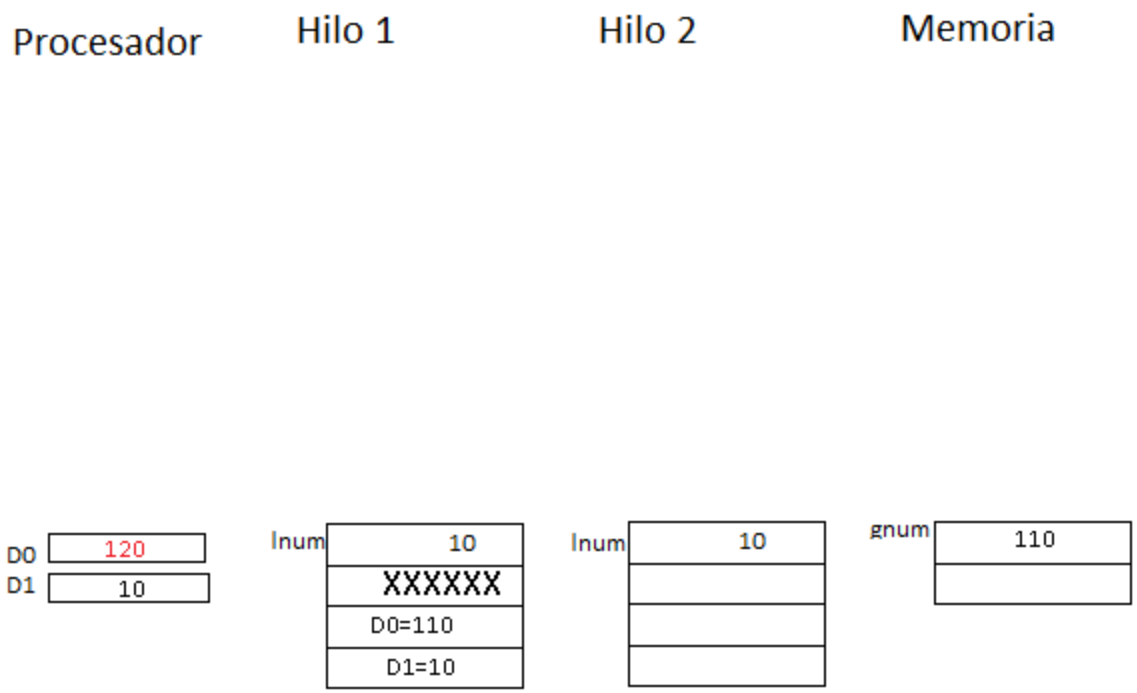
~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

-> ADD D0,D1



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

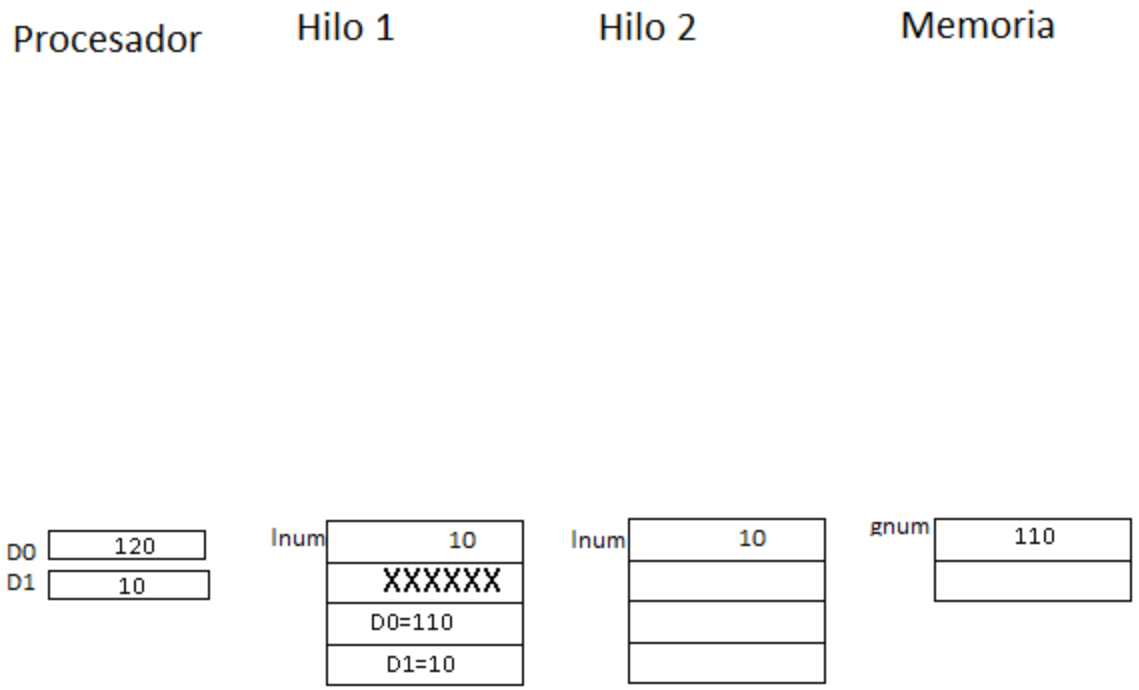
SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

-> MOV M[gnum],D0



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

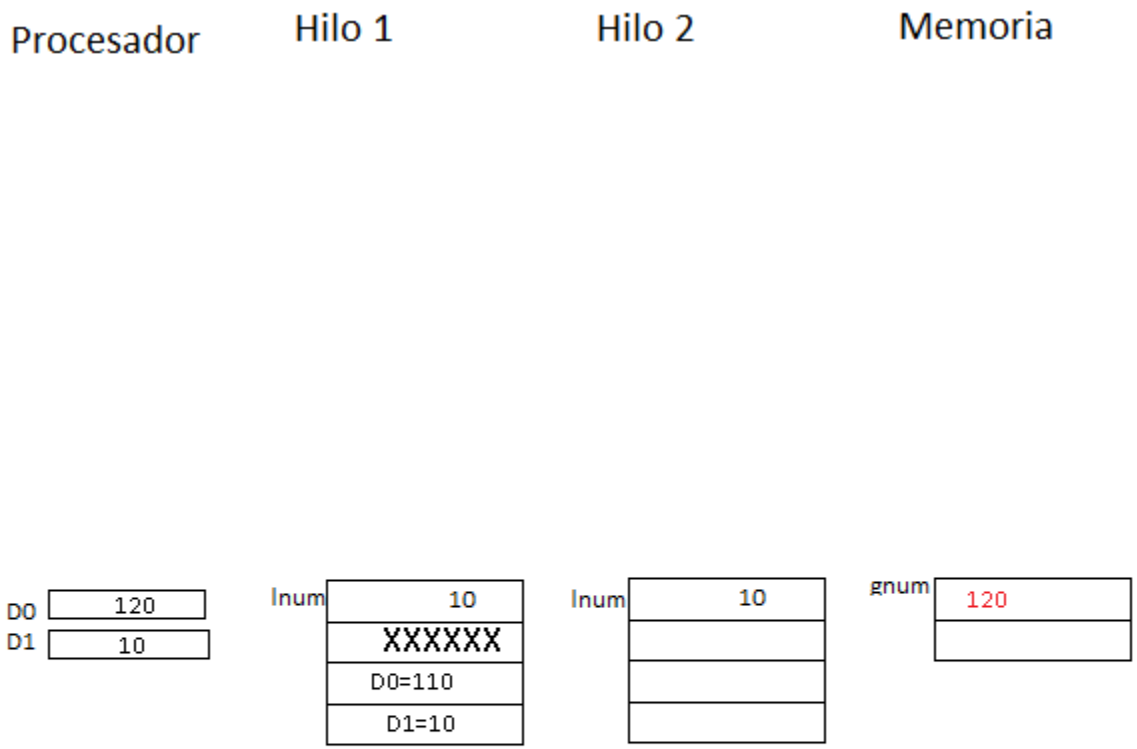
SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

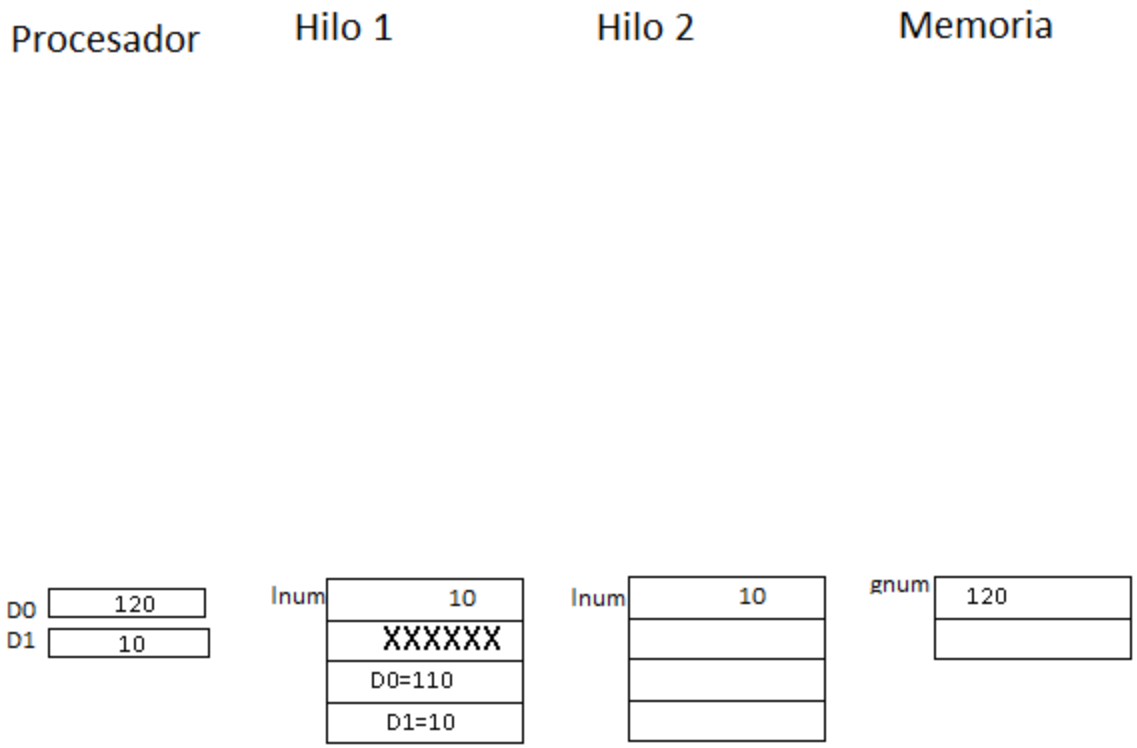
SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

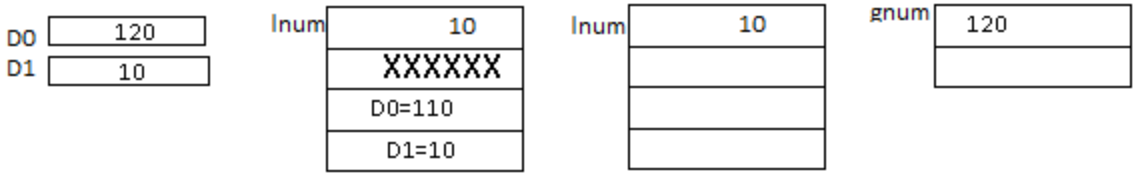
~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

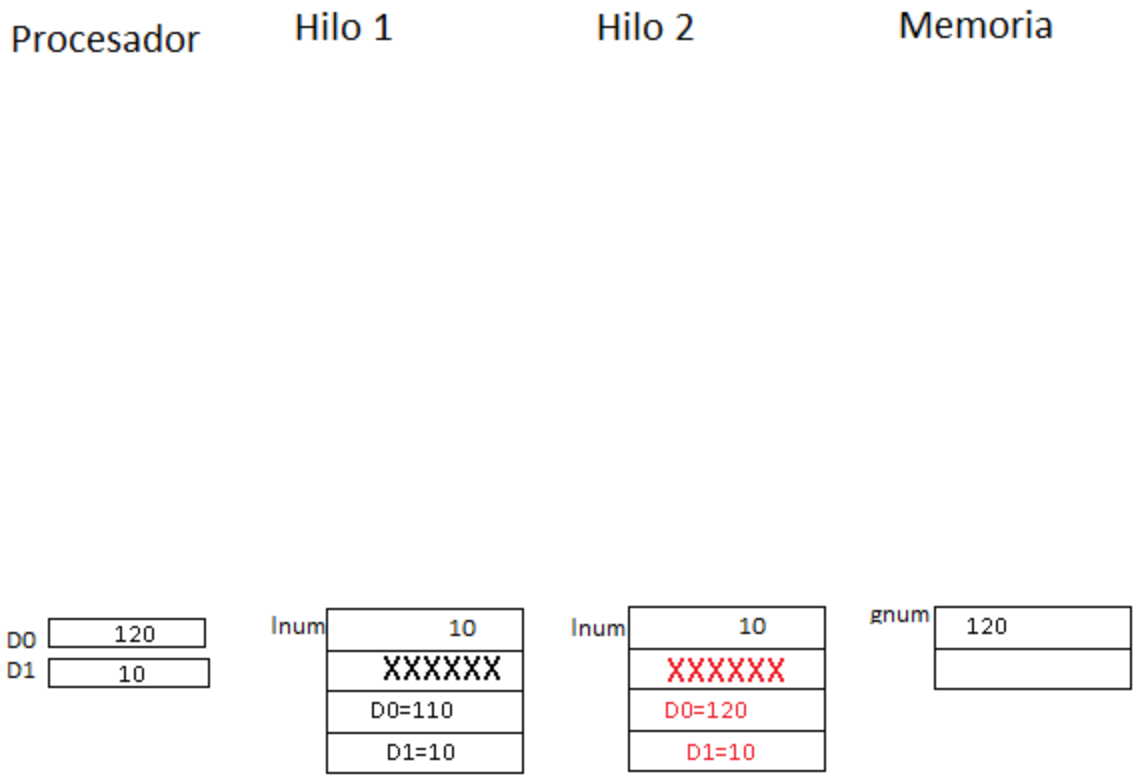
~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

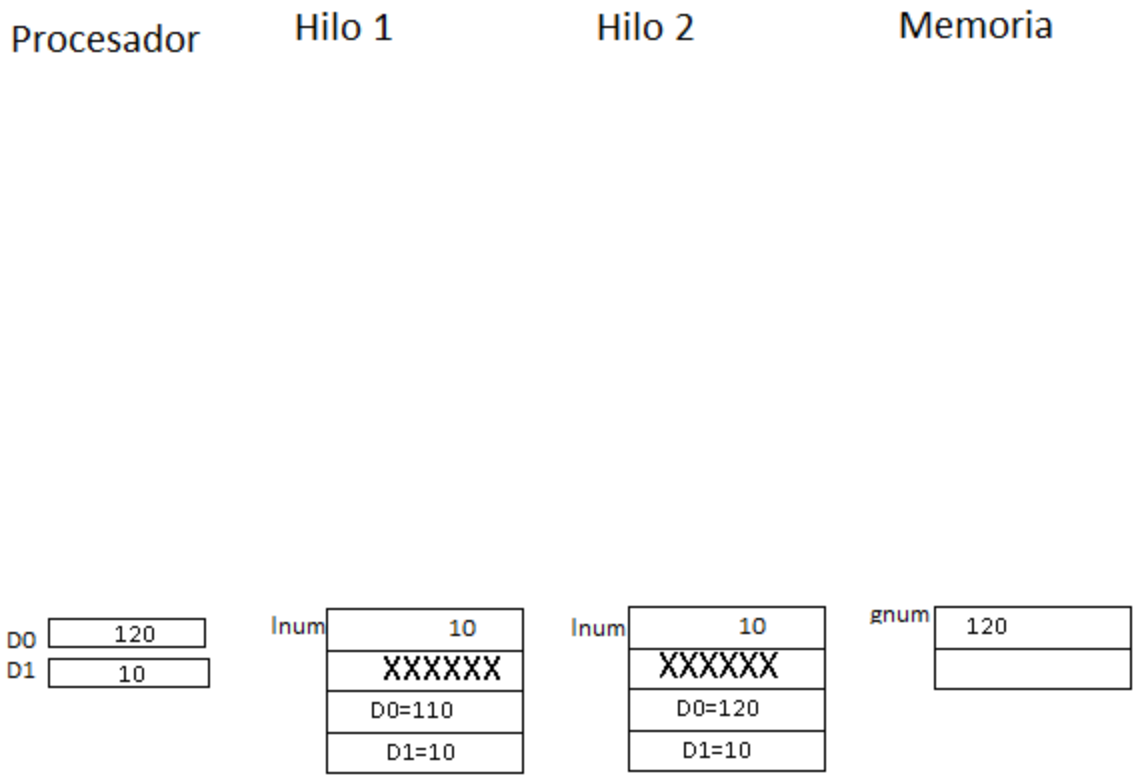
~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila Hilo 1 restaura su estado desde la pila



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~->MOV M[gnum],D0~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

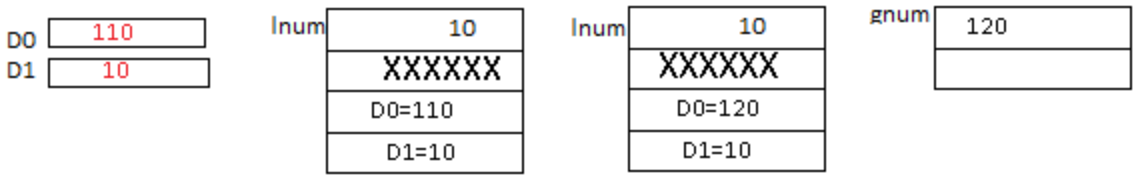
~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila
Hilo 1 restaura su estado desde la pila

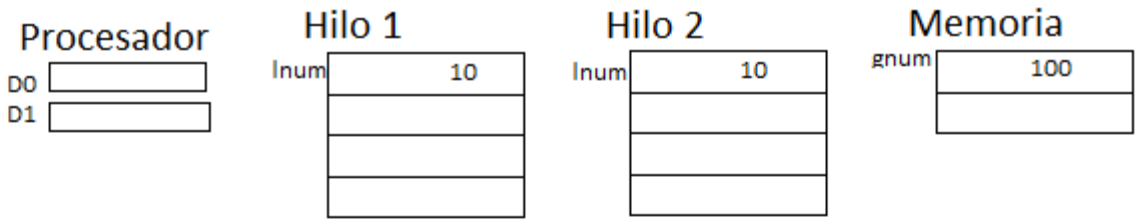


5 Operating Systems

5 Operating Systems

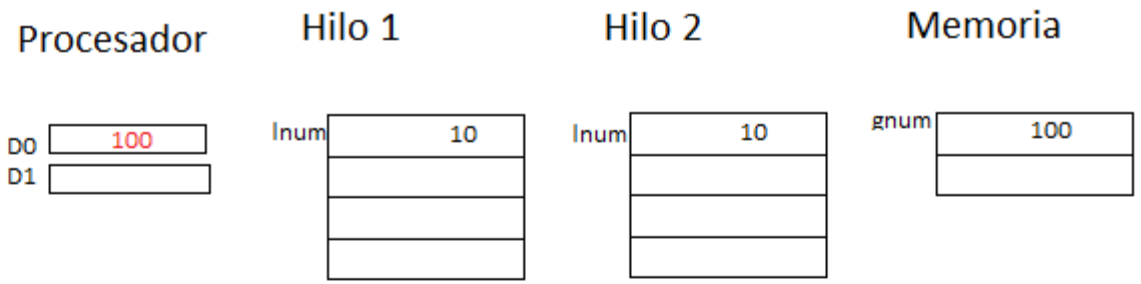
5 Operating Systems

SSOO entrega CPU Hilo 1
->MOV D0,M[gnum]



5 Operating Systems

SSOO entrega CPU Hilo 1
->MOV D0,M[gnum]

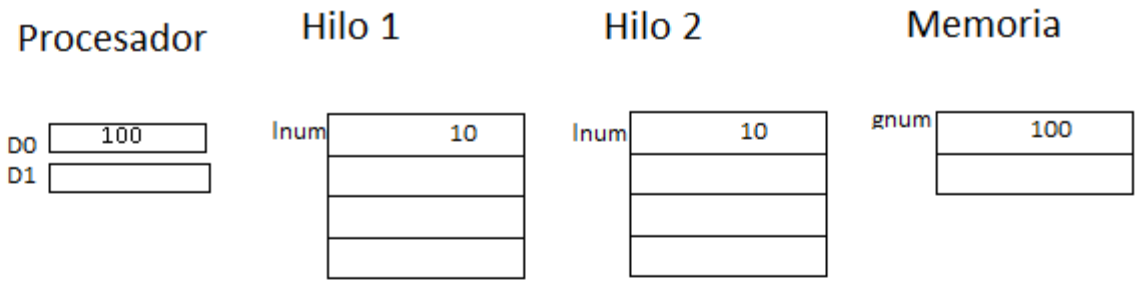


5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

->MOV D1,M[lnum]

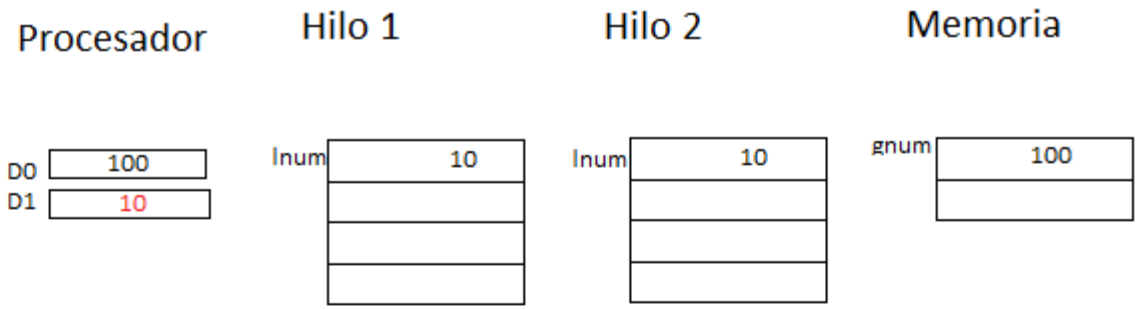


5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

->MOV D1,M[lnum]



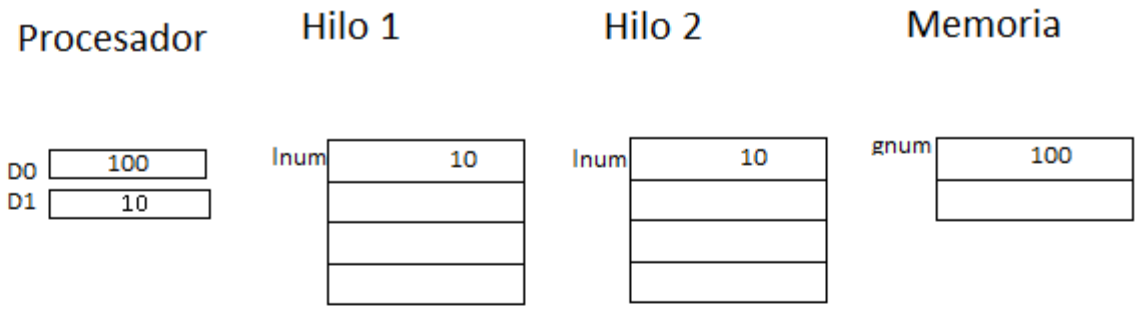
5 Operating Systems

SSOO entrega CPU Hilo 1

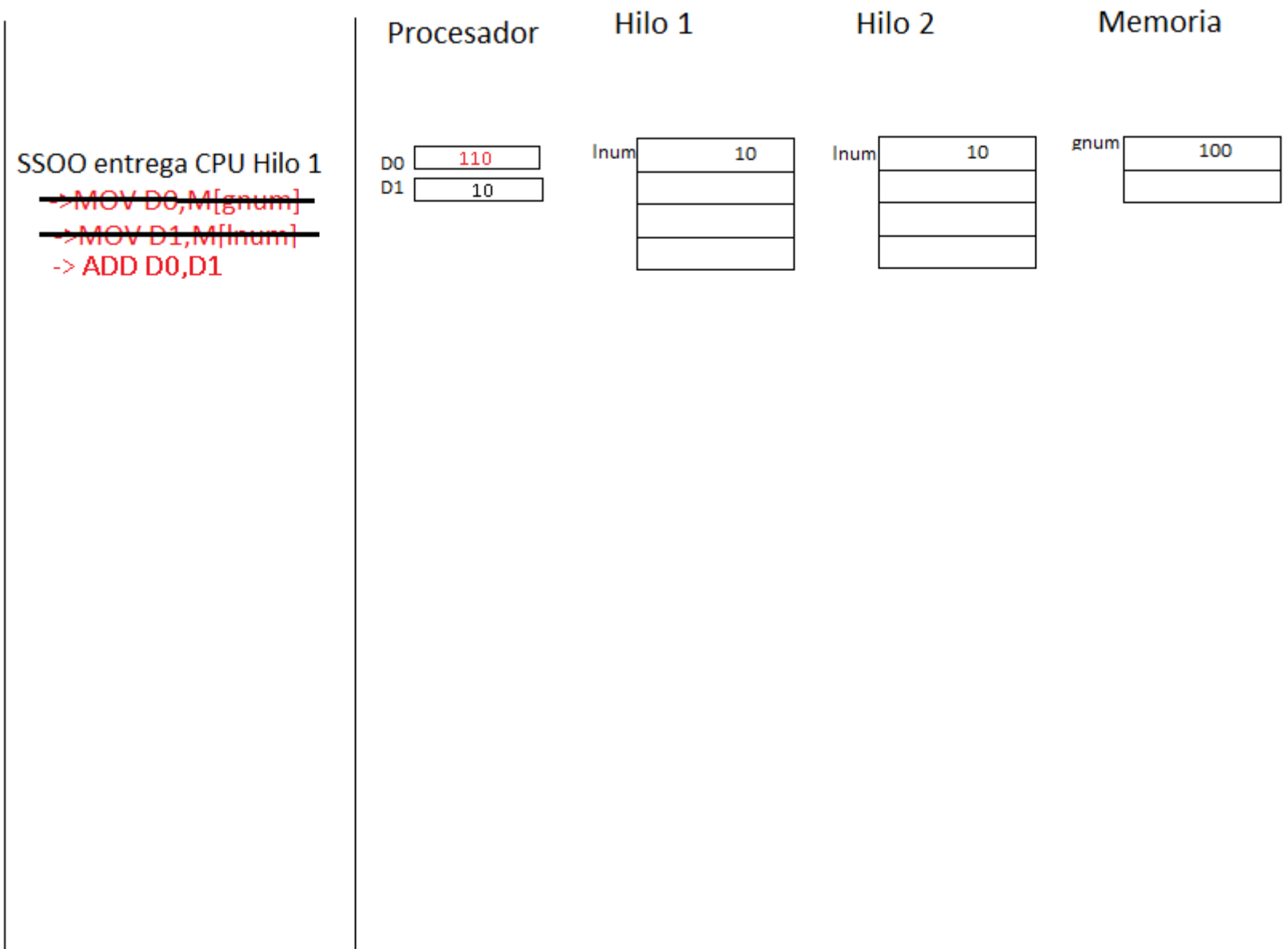
~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

-> ADD D0,D1



5 Operating Systems



5 Operating Systems

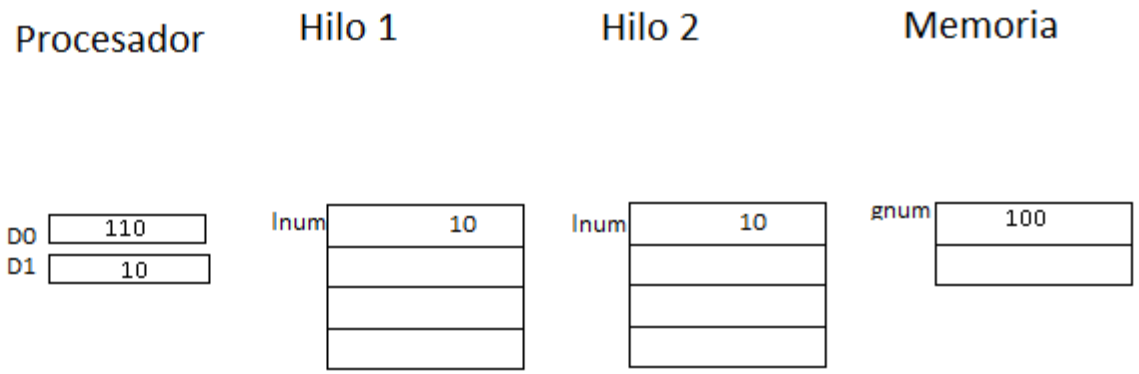
SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila



5 Operating Systems

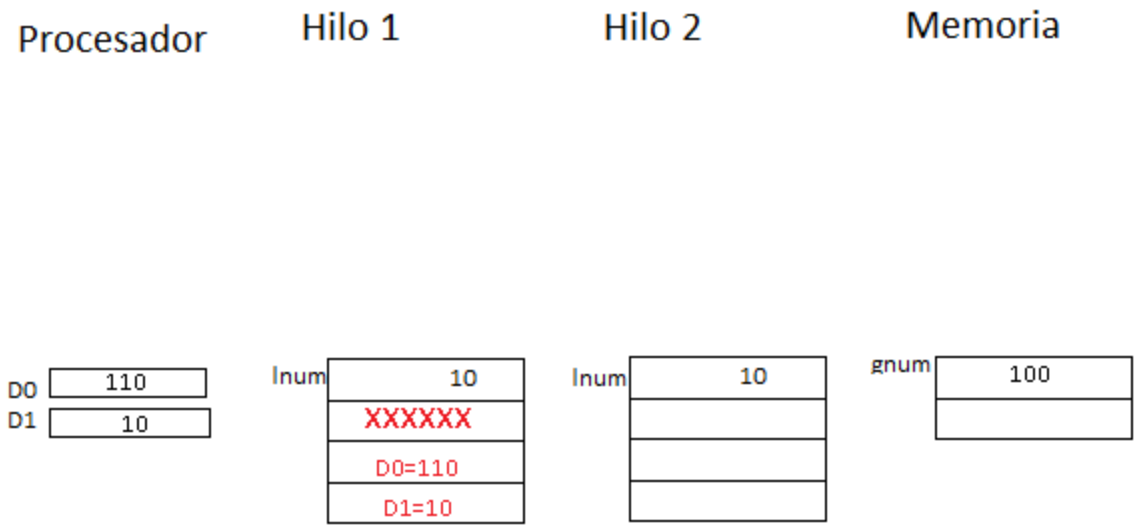
SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila



5 Operating Systems

SSOO entrega CPU Hilo 1

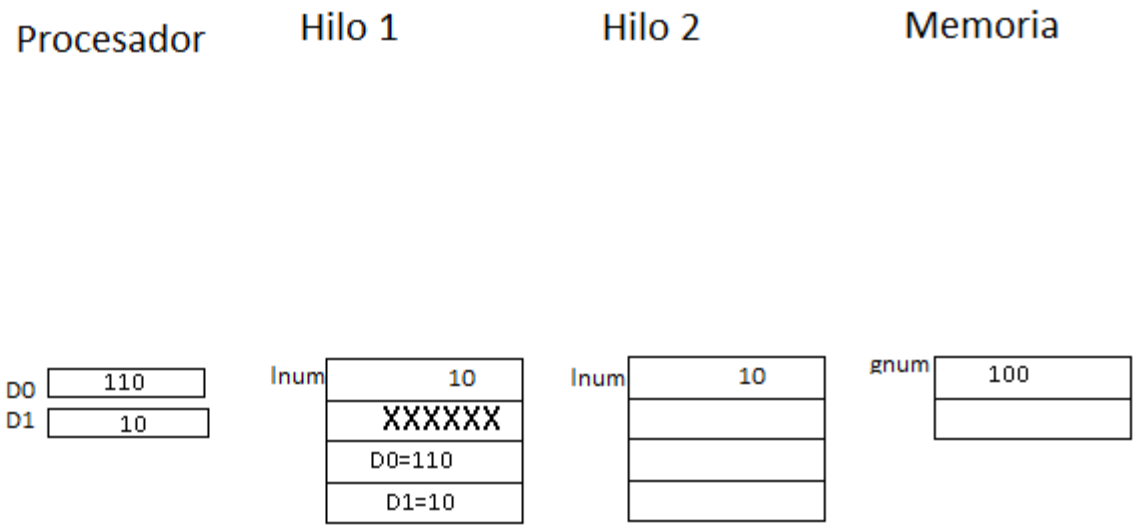
~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

-> MOV D0,M[gnum]



5 Operating Systems

SSOO entrega CPU Hilo 1

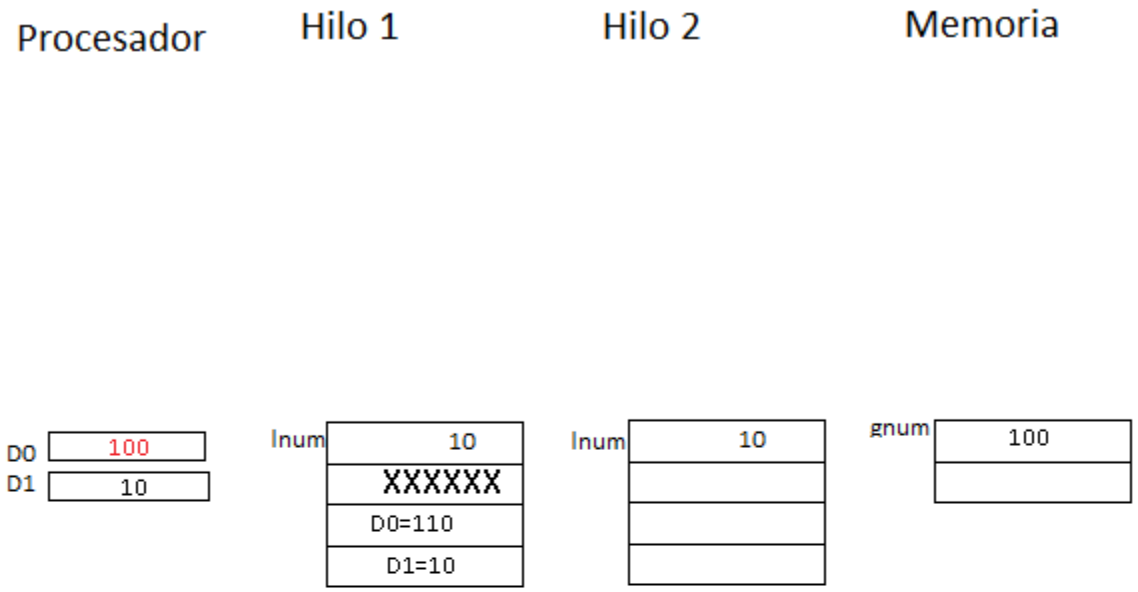
~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

-> MOV D0,M[gnum]



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

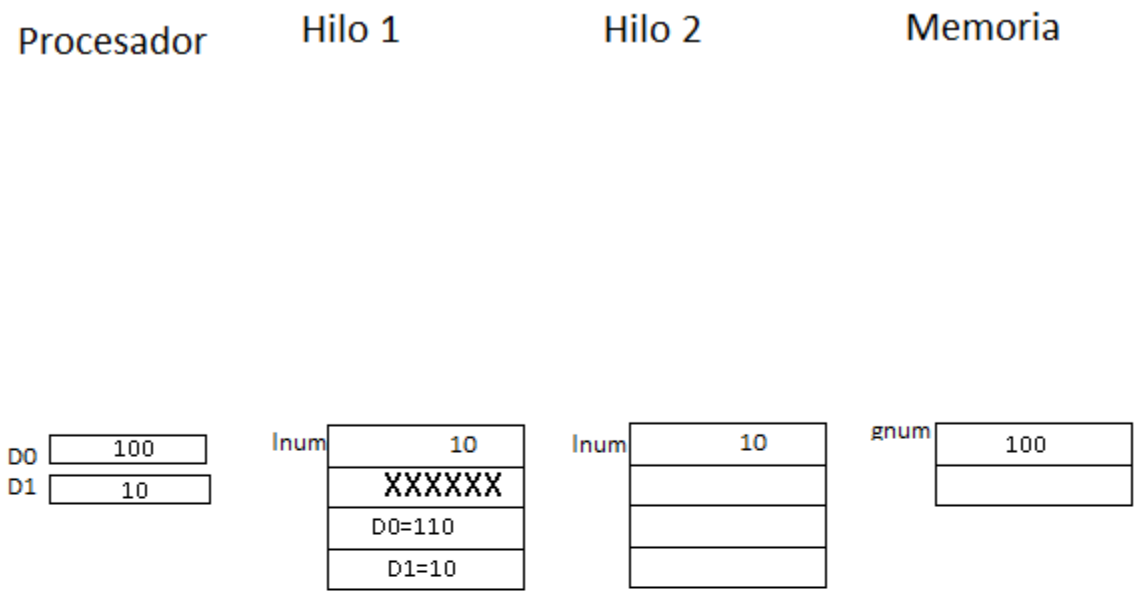
~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

-> MOV D1,M[lnum]



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

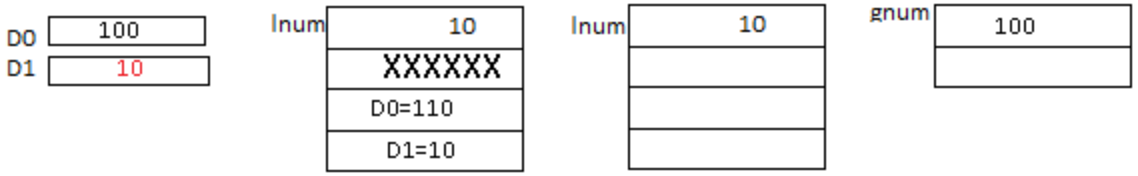
~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

-> MOV D1,M[lnum]



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

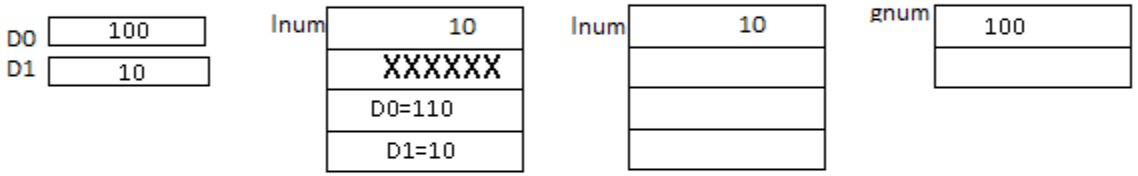
~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

-> ADD D0,D1



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

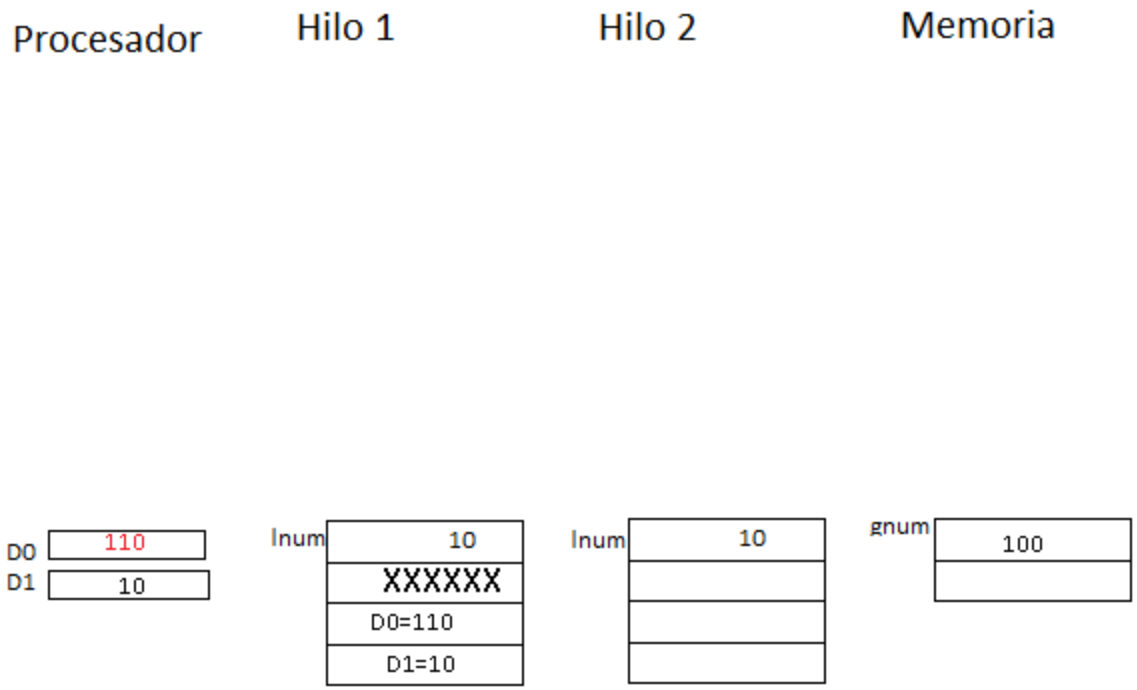
~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

-> ADD D0,D1



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

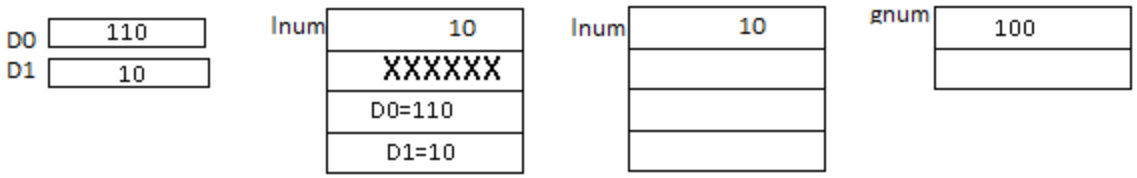
SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

-> MOV M[gnum],D0



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

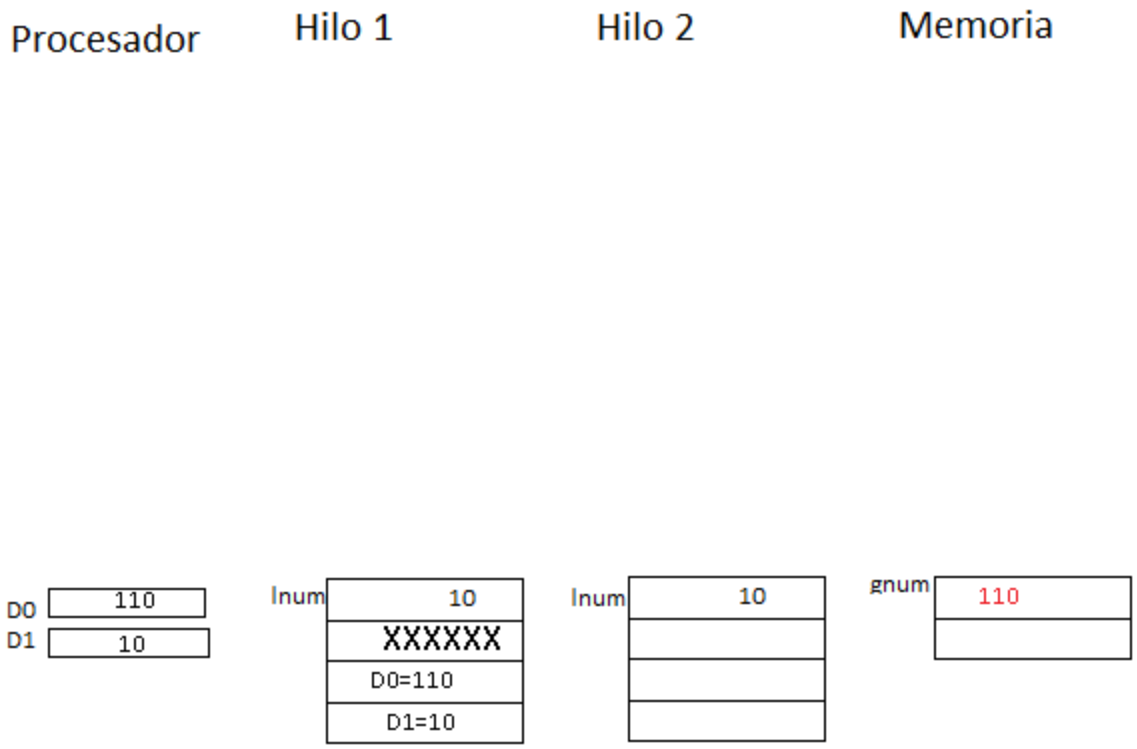
SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

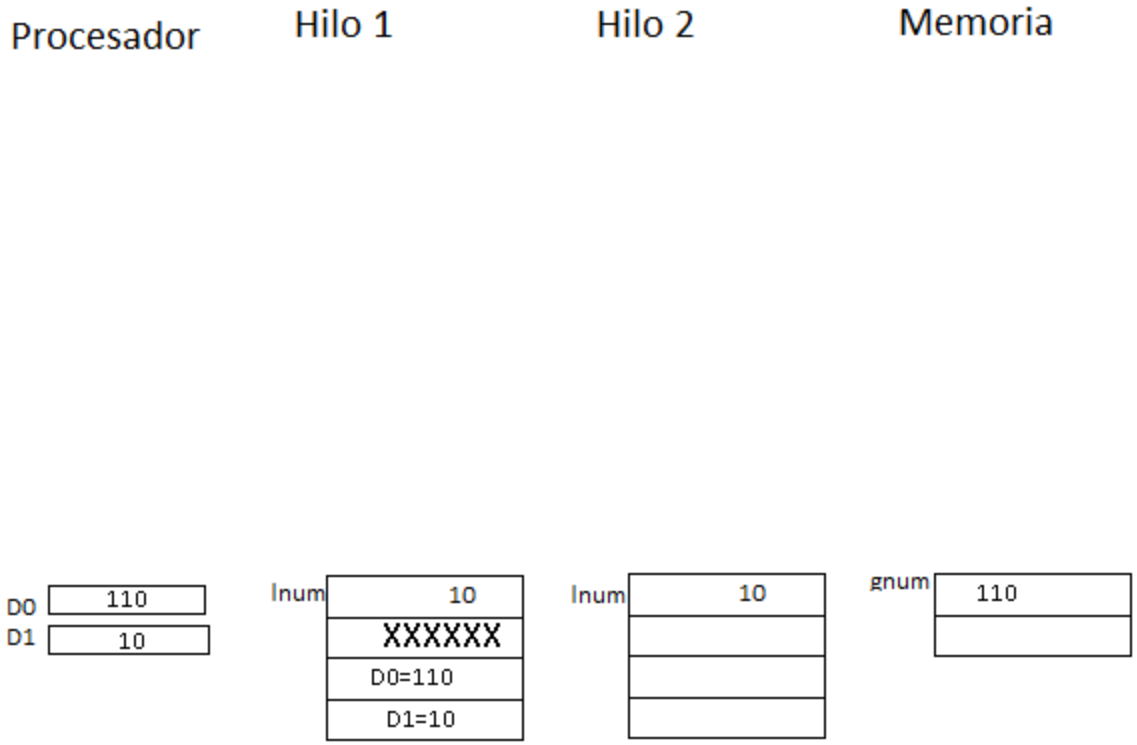
SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

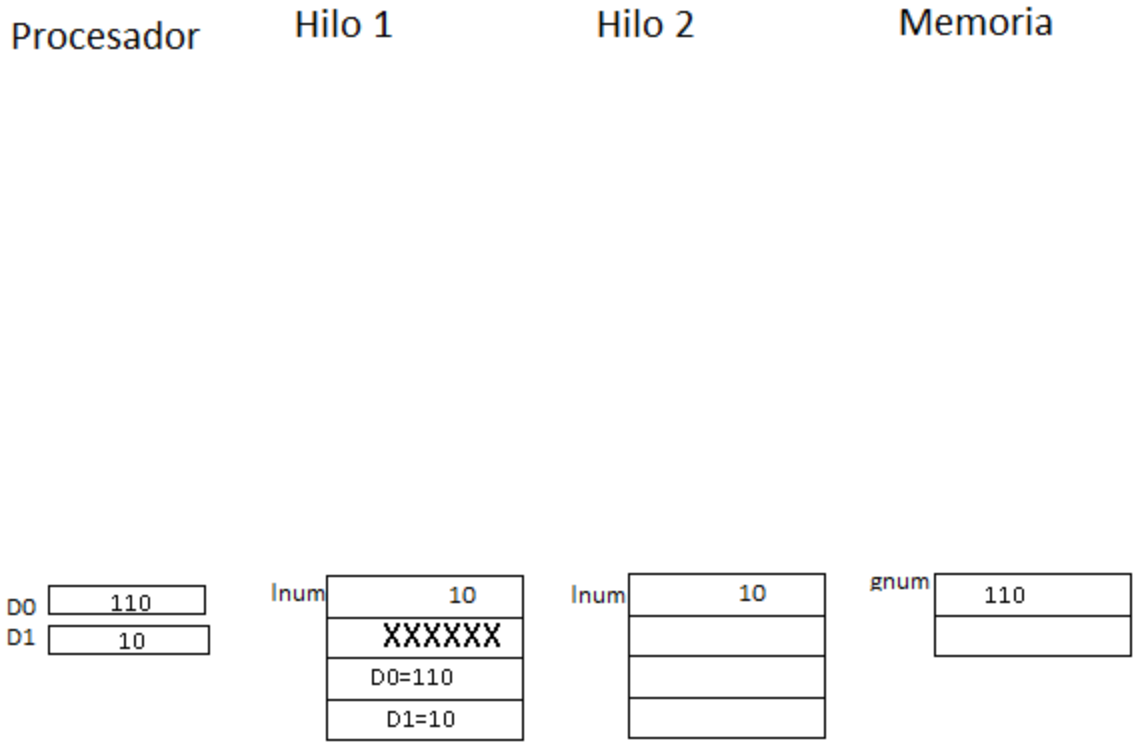
~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

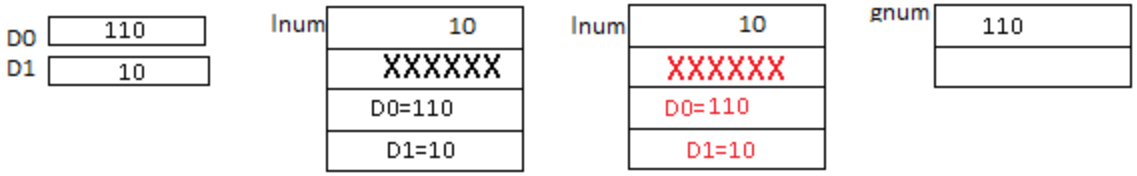
~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila



5 Operating Systems

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

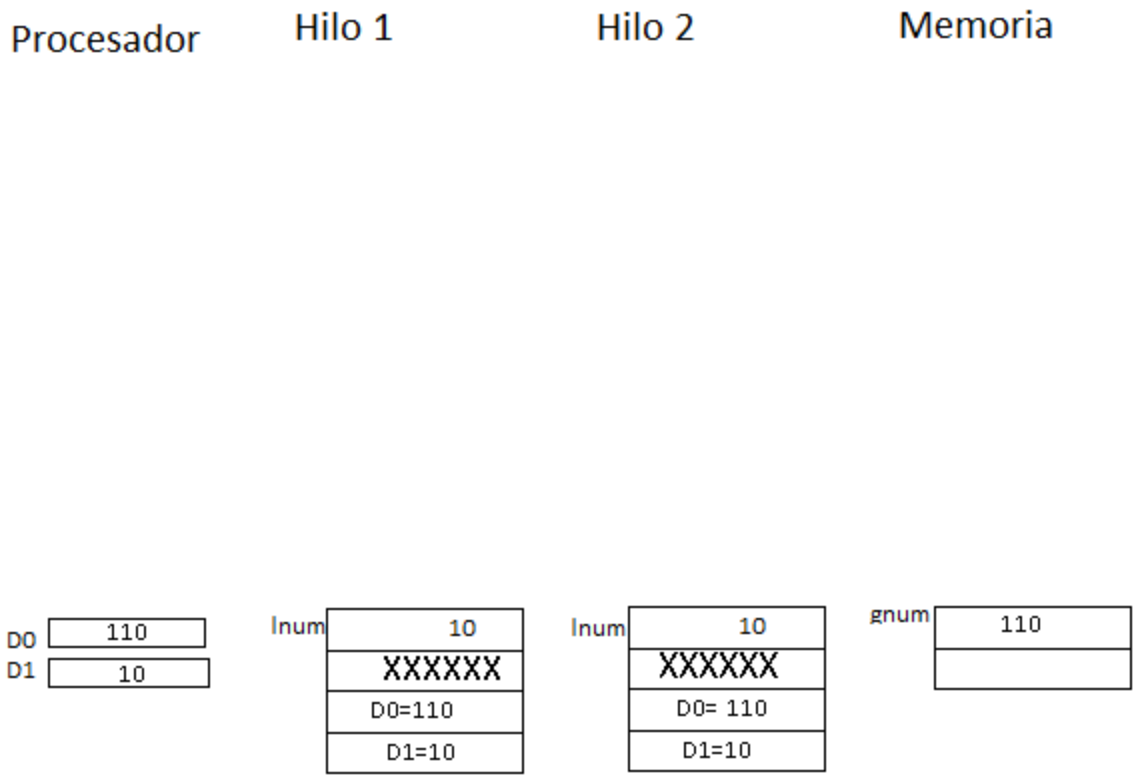
~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila Hilo 1 restaura su estado desde la pila



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

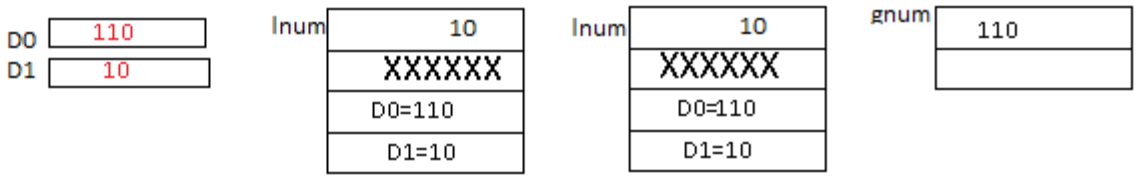
~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila Hilo 1 restaura su estado desde la pila



Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

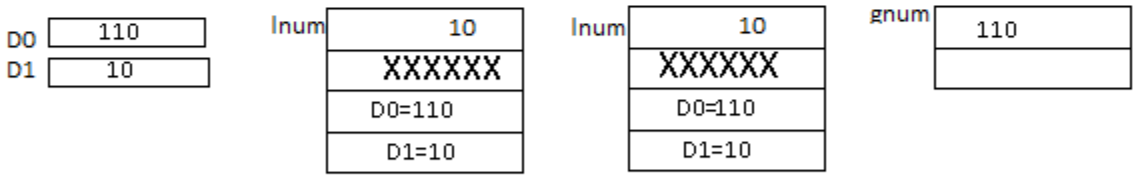
~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila

Hilo 1 restaura su estado desde la pila

-> MOV D0,M[gnum]



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

~~-> MOV D1,M[lnum]~~

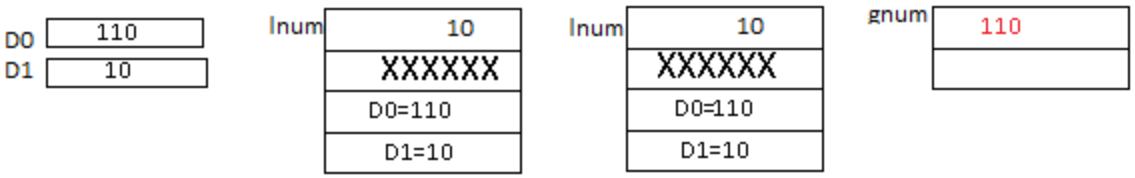
~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila

Hilo 1 restaura su estado desde la pila

-> MOV D0,M[gnum]



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

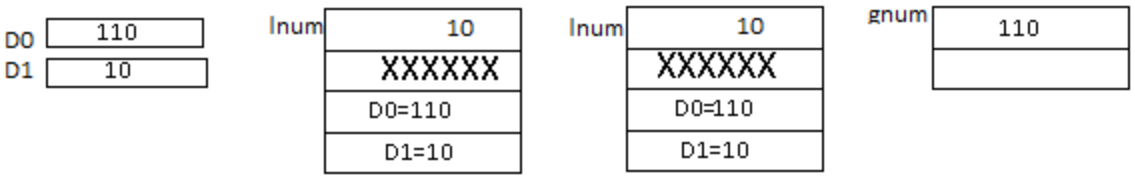
~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila
Hilo 1 restaura su estado desde la pila

~~-> MOV D0,M[gnum]~~



5 Operating Systems

Procesador | Hilo 1 | Hilo 2 | Memoria

SSOO entrega CPU Hilo 1

~~->MOV D0,M[gnum]~~

~~->MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

SSOO entrega CPU Hilo2. Hilo 1 guarda su estado en la pila

~~-> MOV D0,M[gnum]~~

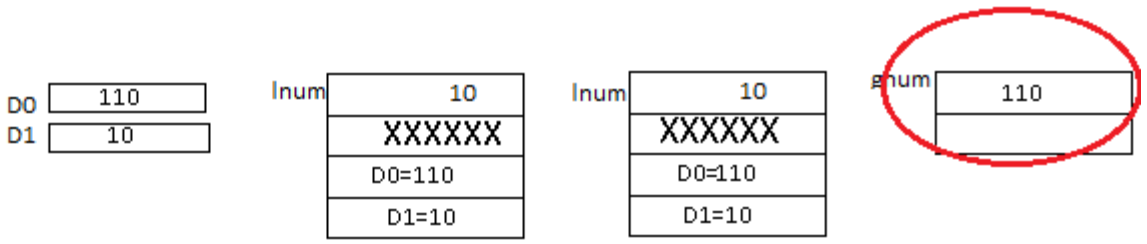
~~-> MOV D1,M[lnum]~~

~~-> ADD D0,D1~~

~~-> MOV M[gnum],D0~~

SSOO entrega CPU Hilo 1. Hilo 2 guarda su estado en la pila
Hilo 1 restaura su estado desde la pila

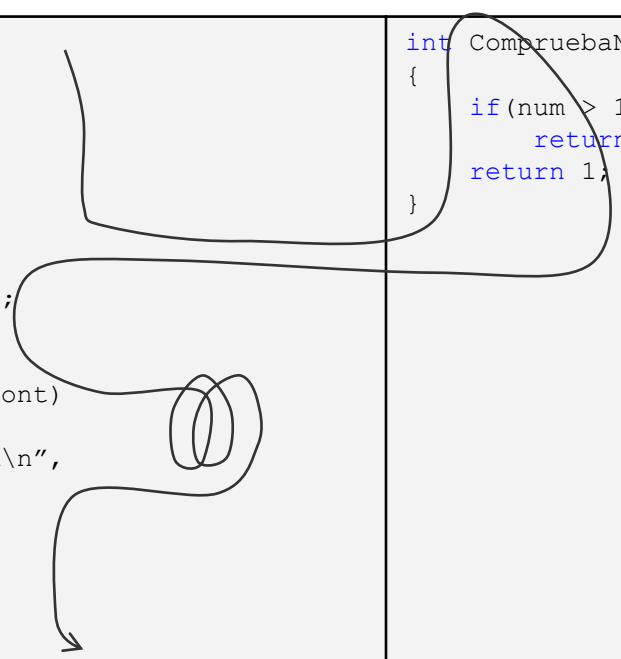
~~-> MOV D0,M[gnum]~~



Código en lenguaje C

```
int main(int argc, char** argv)
{
    int iNum, iCubo, iCont;
    printf("Introduzca un n°:");
    scanf("%d", &iNum);
    iCubo = iNum;
    if(!CompruebaNumero(iNum))
    {
        printf("Número no válido");
        return -1;
    }
    for(iCont = 0; iCont < 2; ++iCont)
        iCubo *= iNum;
    printf( "\nEl cubo de %d es %d\n",
            iNum, iCubo);
    return 0;
}
```

```
int CompruebaNumero(int num)
{
    if(num > 1000)
        return 0;
    return 1;
}
```

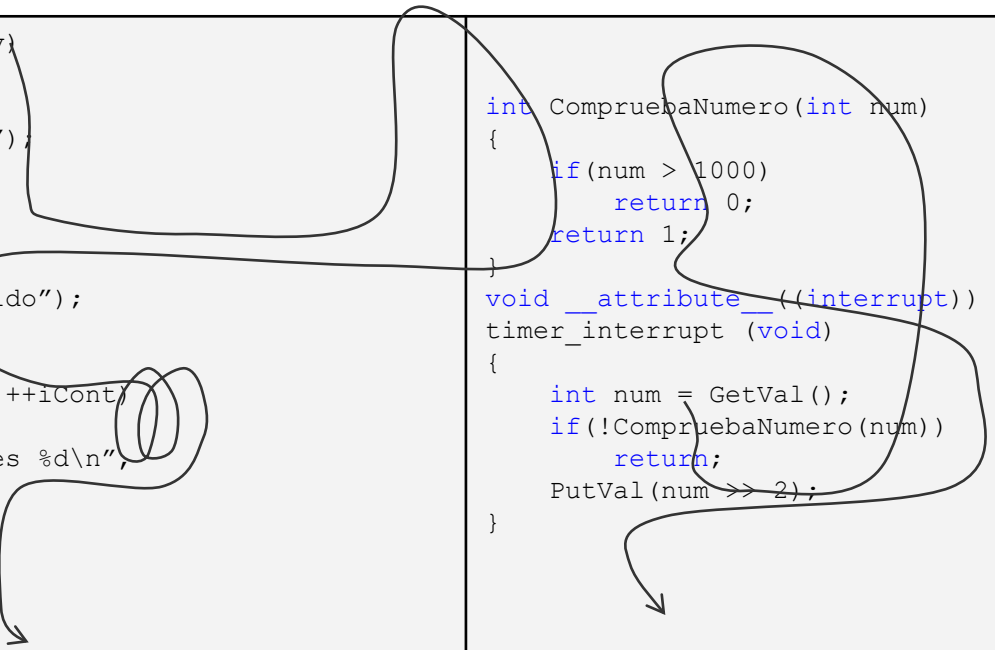


5 Operating Systems

Código en lenguaje C

```c
int main(int argc, char** argv)
{
    int iNum, iCubo, iCont;
    printf("Introduzca un n°:");
    scanf("%d", &iNum);
    iCubo = iNum;
    if(!CompruebaNumero(iNum))
    {
        printf("Número no válido");
        return -1;
    }
    for(iCont = 0; iCont < 2; ++iCont)
        iCubo *= iNum;
    printf( "\nEl cubo de %d es %d\n",
            iNum, iCubo);
    return 0;
}
```

```c
int CompruebaNumero(int num)
{
    if(num > 1000)
        return 0;
    return 1;
}
void __attribute__ ((interrupt))
timer_interrupt (void)
{
    int num = GetVal();
    if(!CompruebaNumero(num))
        return;
    PutVal(num >> 2);
}
```



- **Concurrencia**
  - En las zonas de código que usan variables globales o recursos compartidos, para evitar problemas se puede hacer el acceso indivisible o atómico→ deshabilitar interrupciones (no permitido en modo usuario)

5 Operating Systems

Código en lenguaje C

```c
#include <stdio.h>
volatile int g_num;
int CompruebaNumero(int num);
int main()
{
    int iNum, iCubo, iCont;
    printf("Introduzca un n°:");
    scanf("%d", &iNum);
    iCubo = iNum;
    if(!CompruebaNumero(iNum))
    {
        printf("Número no válido");
        return -1;
    }
    for(iCont = 0; iCont < 2; ++iCont)
        iCubo *= iNum;
    printf( "\nEl cubo de %d es %d\n",
            iNum, iCubo);
    return g_num;
}
```

```c
int CompruebaNumero(int num)
{
    if(num > 10)
    {
    DeshabilitarInt();
   g_num = g_num + num;
    HabilitarInt();
        return 0;
    }
    return 1;
}
void __attribute__((interrupt))
timer_interrupt (void)
{
    int num = GetVal();
    if(!CompruebaNumero(num))
        return;
    PutVal(num >> 2);
}
```

- **Hilos**
  - **Las variables locales de un hilo no se comparten con otros hilos, aunque son físicamente accesibles desde otro hilo dentro del proceso.→El núcleo no las protege porque pertenecen al mismo proceso**

- **Hilos**
  - **Ventajas**
    - Comparticion de recursos dentro del proceso
    - Rápidez de creación y destrucción de los hilos→ Se reserva espacio para una pila
    - Agil conmutacion entre los hilos que están dentro del mismo procesos.
    - Si un hilo dentro de un proceso se bloquea se puede pasar al siguiente hilo del mismo proceso y así no se bloquea toda la aplicación.
    - Se puede asignar prioridades a hilos del mismo proceso
    - Se puede dedicar hilos a cálculos de larga duración sin tener que paralizar a todo el proceso

- **Hilos**
  - **Inconveniente**
    - **Al compartir recursos, si no se programa con cuidado, la actuación de un hilo sobre un recurso puede corromper el uso del recurso de otro hilo del mismo proceso.**
    - **Ejemplo, Si dos hilos del mismo proceso esperan un dato cada uno de un lugar comun (buffer), si no se hace esta espera con cuidado, puede ocurrir que cada hilo se lleve el dato que espera el otro.**
    - **Ejemplo2, Problema visto de concurrencia en variables globales**