Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas

Sistemas Operativos

Unidad II:

Aplicabilidad y seguridad de los sistemas operativos

Cómputo paralelo y distribuido

M. en C. Hermes Francisco Montes Casiano
hermes.upiicsa@gmail.com

Índice

Hilos y procesos ligeros

- La concepción de una aplicación multiproceso no suele ser sencilla, pero para algunos problemas suele ser la solución más eficiente y elegante.
- Es deseable que el núcleo del sistema operativo encapsule los aspectos de
  - creación,
  - planificación,
  - distribución y
  - comunicación

  de los distintos flujos de ejecución secuencial de una aplicación.

Hilos y procesos ligeros

- Es deseable que el núcleo del sistema operativo encapsule los aspectos de
  - creación,
  - planificación,
  - distribución y
  - comunicación

  de los distintos flujos de ejecución secuencial de una aplicación.

UNIX

Las variantes modernas de UNIX abordan esta necesidad y ofrecen el mecanismo de *hilos* como la herramienta para crear aplicaciones concurrentes de una forma más sencilla y ligera, que el mecanismo clásico de procesos.

Hilos y procesos ligeros

- Los hilos no surgen para sustituir a los procesos sino para enriquecerlos
- Un proceso es una entidad compuesta por recursos y por hilos.
  - Los recursos están relacionados con los datos del proceso.
  - Los *hilos* son objetos dinámicos cada uno de los cuales ejecuta una secuencia de instrucciones y comparten entre sí los recursos del proceso.

UNIX

En los sitemas UNIX tradicionales cada proceso se compone de un solo hilo mientras que los sistemas multihilo permiten más de un hilo por proceso.
Cada hilo dispone de su propio contador de programa, su pila y su contexto de registros.

Hilos y procesos ligeros

- Los hilos no surgen para sustituir a los procesos sino para enriquecerlos
- Un proceso es una entidad compuesta por recursos y por hilos.
  - Los recursos están relacionados con los datos del proceso.
  - Los *hilos* son objetos dinámicos cada uno de los cuales ejecuta una secuencia de instrucciones y comparten entre sí los recursos del proceso.

Comunicación entre hilos

La comunicación entre hilos es a través de memoria compartida, ya que los datos del proceso son comunes para todos ellos.

Paralelismo y concurrencia

Concurrencia

La *concurrencia* de una aplicación es el número máximo de flujos de ejecución secuenciales que podría estar ejecutando simultáneamente si dispusiera de un número ilimitado de procesadores y depende de cómo esté escrita la aplicación.

Paralelismo

El *paralelismo* es el número real de hilos que se están ejecutando simultáneamente en un momento determinado y por lo tanto está limitado por el número real de procesadores y por la carga actual del sistema [Black, 1990].

Paralelismo y concurrencia

- Idealmente, una aplicación con concurrencia n podría alcanzar igual paralelismo si se ejecuta en una arquitectura con n o más procesadores, reduciendo su tiempo de ejecución en un factor de $1/n$.

En la práctica

No se alcanza nunca ese factor debido a la sobrecarga que supone:

1. crear,
2. distribuir y
3. sincronizar los hilos

Tipos de hilos

A la hora de implementar aplicaciones concurrentes se pueden considerar tres tipos de hilos:

1. **Hilos del núcleo:** no están asociados con ningun proceso de usuario; son creados y destruidos según las necesidades internas del núcleo y comparten el código y los datos de éste.
2. **Procesos ligeros:** es un hilo del usuario sostenido por el núcleo. Un proceso se puede componer de varios procesos ligeros, cada uno de los cuales se apoya en un hilo del núcleo.

   Como son visibles para el núcleo, éste se encarga de despacharlos y alojarlos en los procesadores que se encuentren disponibles, traduciendo así la concurrencia de una aplicación en paralelismo real.

Tipos de hilos

A la hora de implementar aplicaciones concurrentes se pueden considerar tres tipos de hilos:

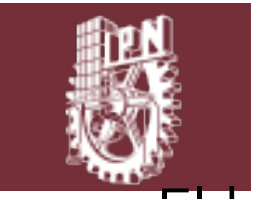
3. **Hilos de usuario:** son gestionados por la aplicación, sin la intervención del núcleo. Esto se consigue a través de bibliotecas como *C-threads* de Match o *pthreads* de POSIX.

   La implementación de los hilos de usuario es posible gracias a que la pila y el contexto de registros pueden ser guardados y restaurados sin la intervención del núcleo.

Tipos de hilos

Ventaja

Una ventaja, en una arquitectura monoprocesador, es que las operaciones bloqueantes bloquean hilos individuales y no al proceso en su totalidad, como ocurre con los procesos de usuario.



Hilos de usuario

El hecho de que los hilos de usuario no sean conocidos por el núcleo plantea dos inconvenientes:

- **No es posible traducir la concurrencia de la aplicación en paralelismo real**, ya que es el núcleo el encargado de alojar los hilos en los procesadores y esto sólo lo puede hacer con los hilos del núcleo y los procesos ligeros.
- **LLamadas al sistema,** al ser ejecutadas en modo supervisor y por lo tanto por el núcleo, bloquea el proceso al que pertenece el hilo y por lo tanto bloquea todos los hilos de ese proceso.

  La ventaja principal que presentan los hilos de usuario es la mejora en el rendimiento de la aplicación para conmutar de hilo, ya que no se consumen recursos del núcleo a menos que esté asociado a un proceso ligero.

Creación de hilos

- Entre las interfaces para trabajar con hilos, la conocida como *pthreads* de POSIX es la adoptada mayoritariamente.
- POSIX, no define un sistema operativo y tampoco indica cómo deben escribirse las aplicaciones, solo define la interfaz entre ambos.
- La biblioteca de hilos POSIX tiene un enfoque orientado a objetos donde cada hilo es un objeto que representa un flujo de control secuencial dentro de un proceso.
- A cada hilo se le asocia un identificador de tipo *pthread_t* y unos atributos de tipo *pthread_attr_t*, ambos definidos en *pthread.h*

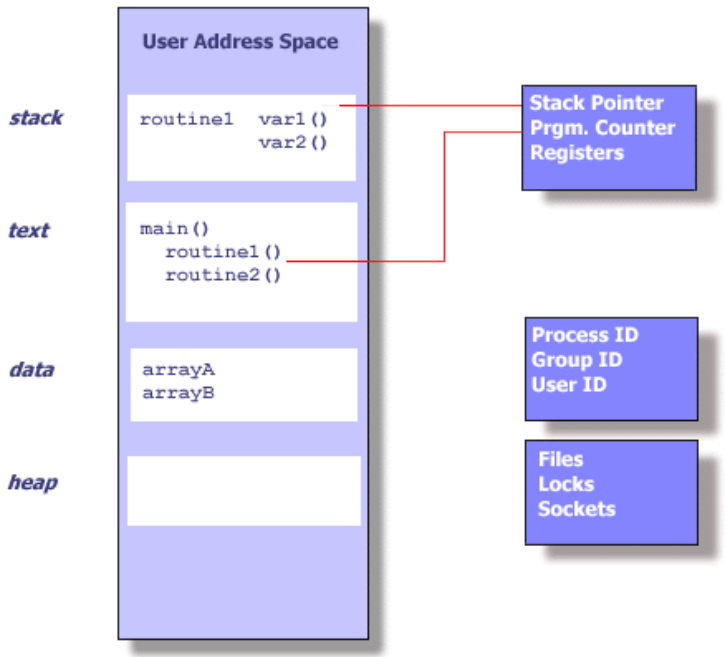
Creación de hilos

Función	Descripción
pthread_create	Crear un hilo
pthread_self	Obtener su identificador
pthread_detach	Reclamar los recursos que utiliza
pthread_exit	Ordenar su terminación
pthread_join	Esperar su terminación
pthread_equal	Comparar dos hilos

Índice

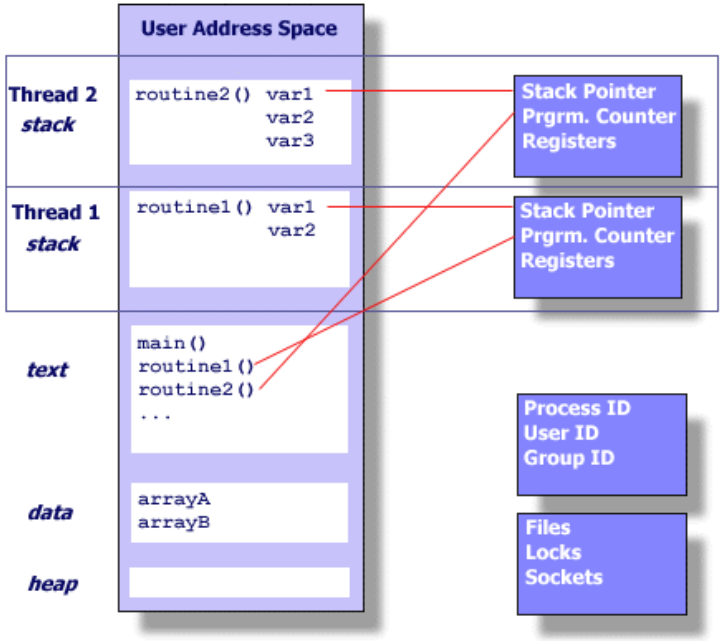
Introducción

- Technically, a thread is defined as an independent stream of instructions that can be scheduled to run as such by the operating system.



Introducción

- Technically, a thread is defined as an independent stream of instructions that can be scheduled to run as such by the operating system.



Introducción

- Threads use and exist within these process resources, yet are able to be scheduled by the operating system and run as independent entities largely because they duplicate only the bare essential resources that enable them to exist as executable code.
  - Stack pointer
  - Registers
  - Scheduling properties (such as policy or priority)
  - Set of pending and blocked signals
  - Thread specific data.

Introducción

- Because threads within the same process share resources:
  - Changes made by one thread to shared system resources (such as closing a file) will be seen by all other threads.
  - Two pointers having the same value point to the same data.
  - Reading and writing to the same memory locations is possible, and therefore requires explicit synchronization by the programmer.

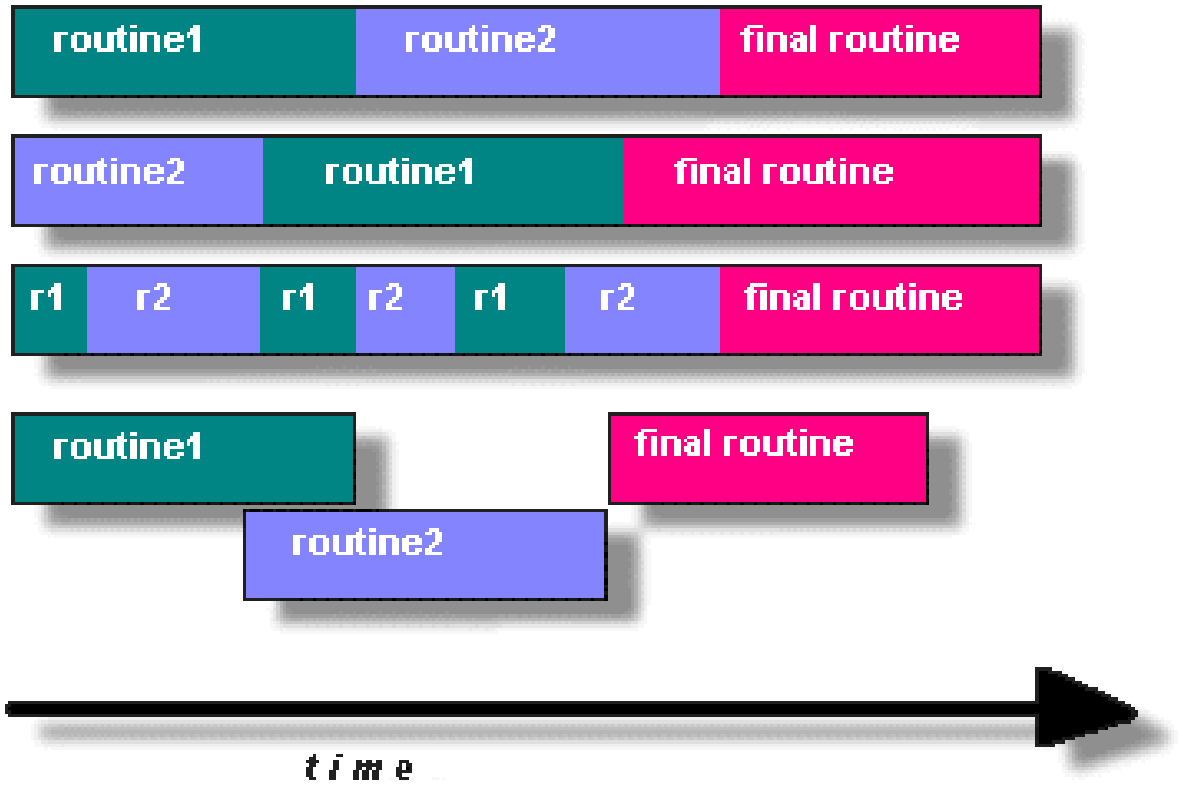
¿Por qué Pthreads?

- In the world of high performance computing, the primary motivation for using Pthreads is to realize potential program performance gains.
- When compared to the cost of creating and managing a process, a thread can be created with much less operating system overhead. Managing threads requires fewer system resources than managing processes.

Programación paralela 1

- On modern, multi-cpu machines, pthreads are ideally suited for parallel programming, and whatever applies to parallel programming in general, applies to parallel pthreads programs.
- There are many considerations for designing parallel programs, such as:
  - What type of parallel programming model to use?
  - Problem partitioning
  - Load balancing
  - Communications
  - Data dependencies
  - Synchronization and race conditions
  - Memory issues
  - I/O issues
  - Program complexity
  - Programmer effort/costs/time

Programación paralela



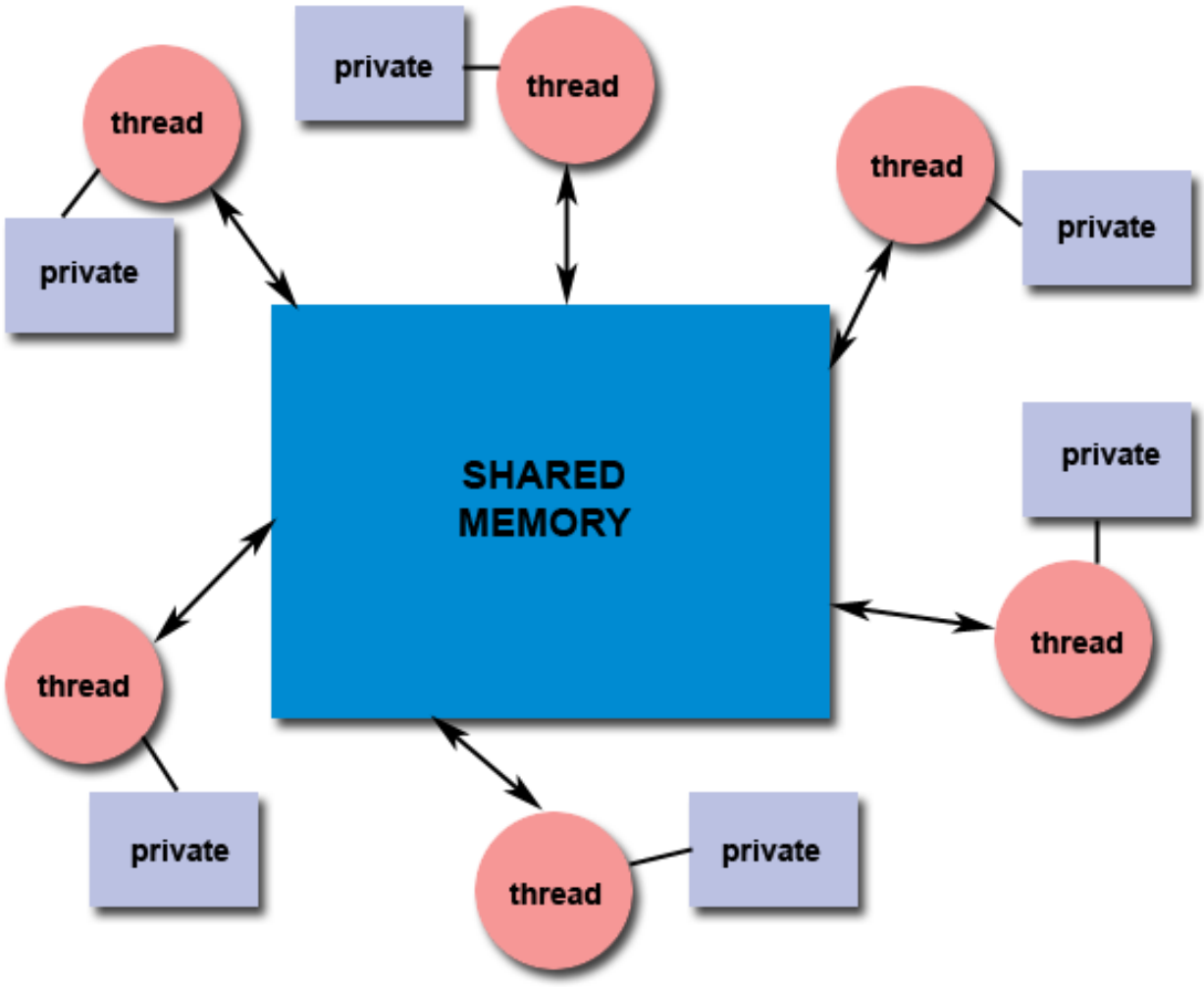
Modelos de programas multihilo

Manager/worker: a single thread, the manager assigns work to other threads, the workers. Typically, the manager handles all input and parcels out work to the other tasks. At least two forms of the manager/worker model are common: static worker pool and dynamic worker pool.

Pipeline: a task is broken into a series of suboperations, each of which is handled in series, but concurrently, by a different thread. An automobile assembly line best describes this model.

Peer: similar to the manager/worker model, but after the main thread creates other threads, it participates in the work.

Modelo de memoria compartida

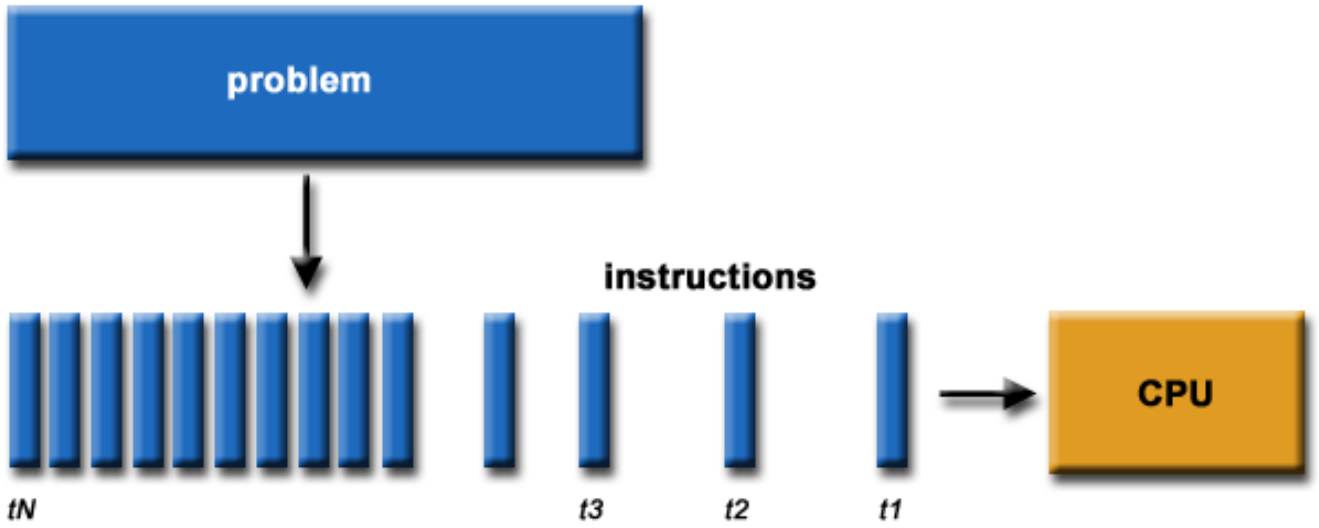


Limitaciones de los hilos

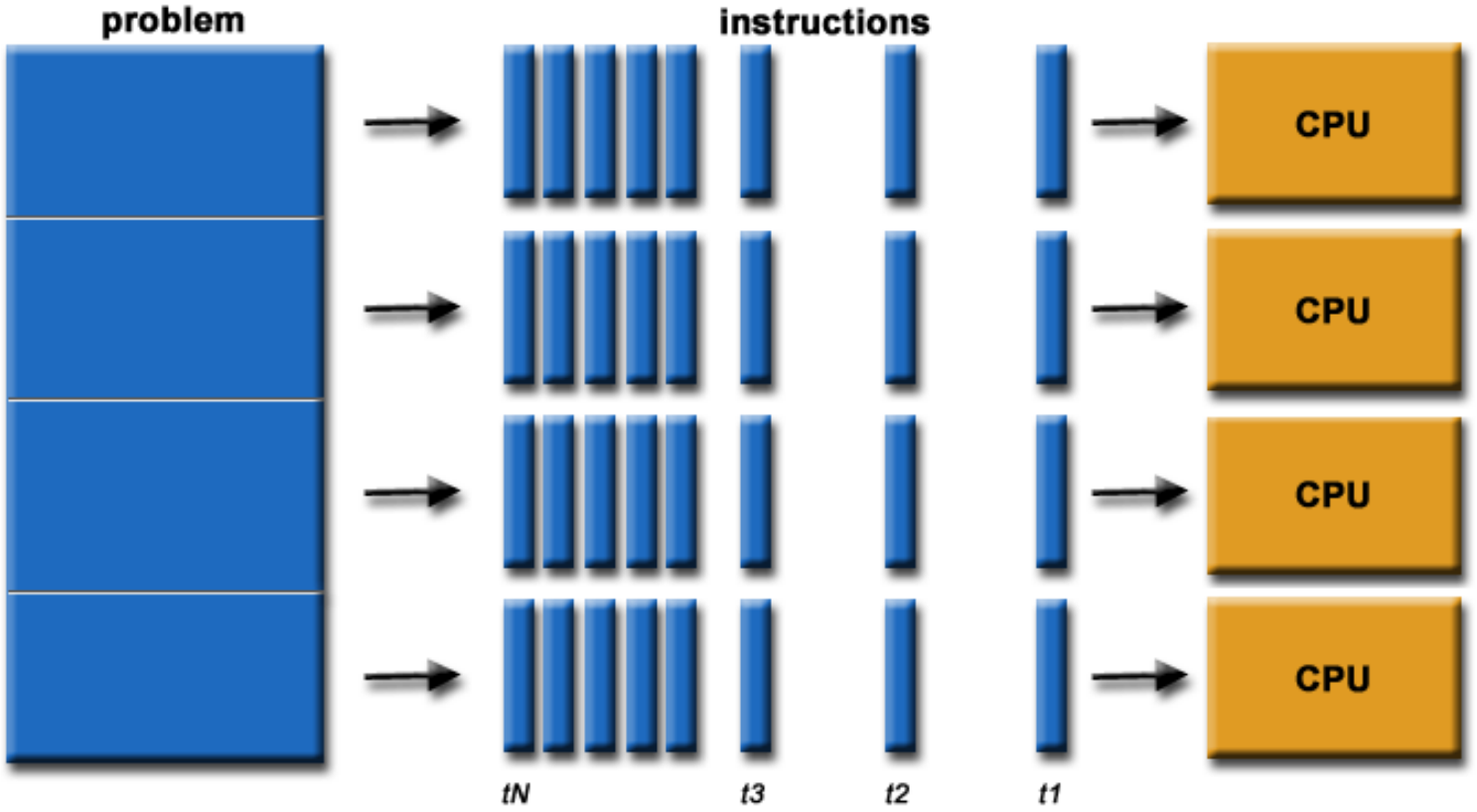
- Although the Pthreads API is an ANSI/IEEE standard, implementations can, and usually do, vary in ways not specified by the standard.
- Because of this, a program that runs fine on one platform, may fail or produce wrong results on another platform.
- The maximum number of threads permitted, and the default thread stack size are two important limits to consider when designing your program.

Índice

Cómputo serial



Cómputo paralelo



¿Por qué usar cómputo paralelo?

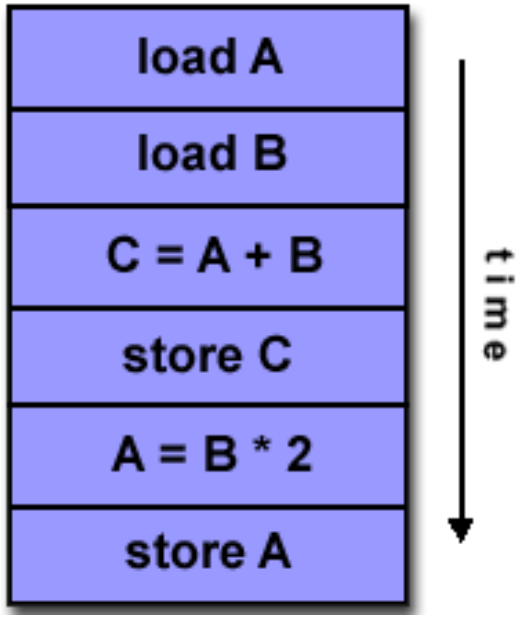
1. Ahorrar tiempo y dinero
2. Resolver problemas complejos
3. Proveer concurrencia
4. Utilizar recursos remotos

Clasificación de Flynn

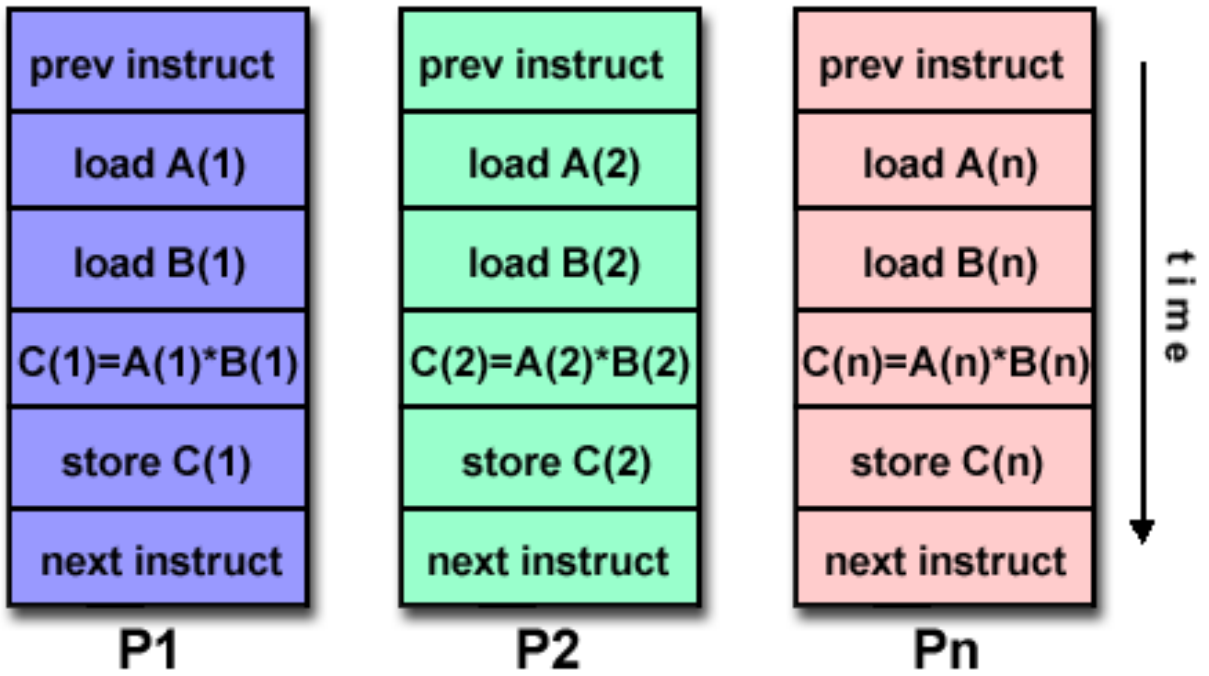
Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of Instruction and Data. Each of these dimensions can have only one of two possible states: Single or Multiple.

SISD **Single Instruction, Single Data**	**SIMD** **Single Instruction, Multiple Data**
MISD **Multiple Instruction, Single Data**	**MIMD** **Multiple Instruction, Multiple Data**

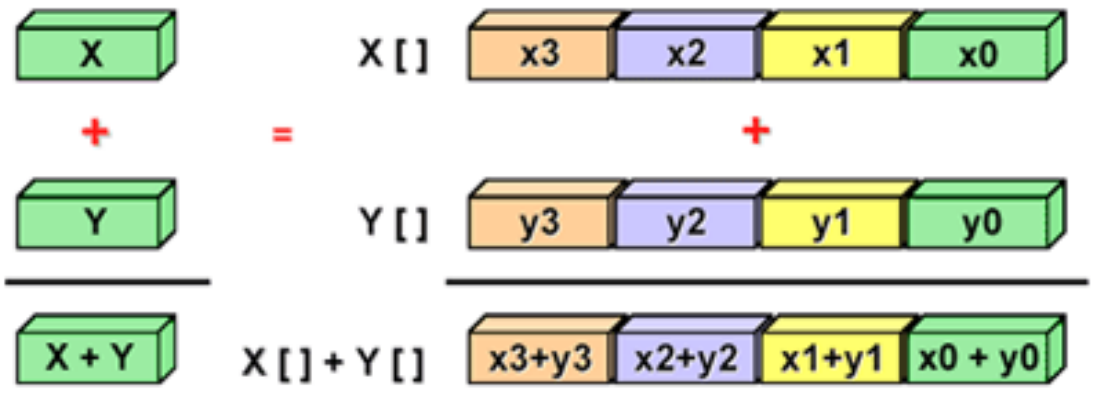
Clasificación de Flynn



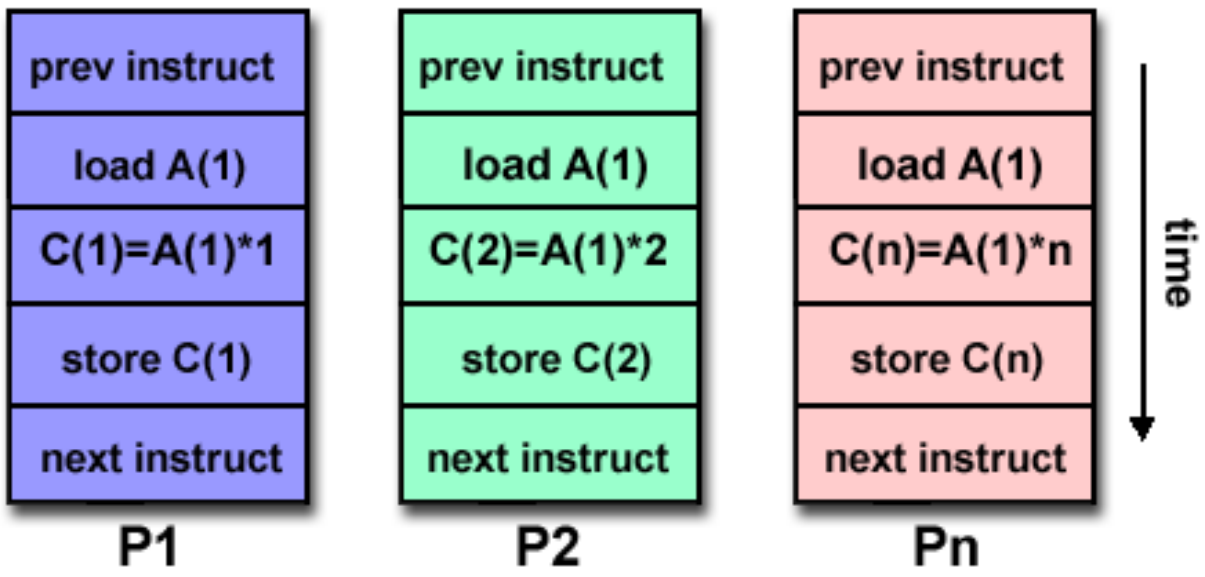
Clasificación de Flynn



Clasificación de Flynn



Clasificación de Flynn



Clasificación de Flynn