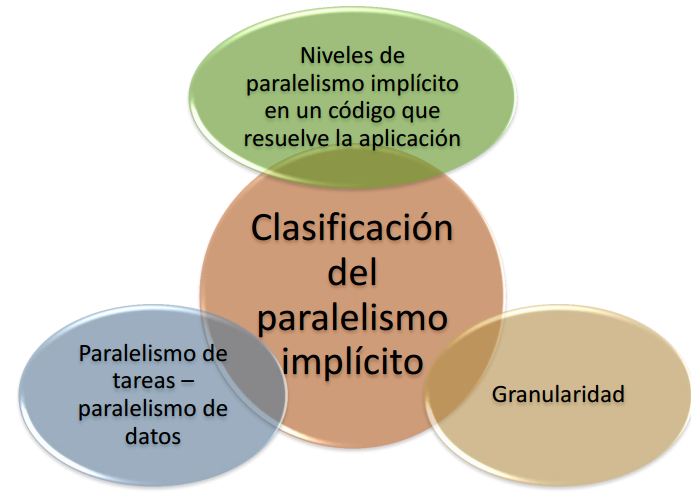
**Lección 1. Clasificación del paralelismo implícito en una aplicación**

**|Criterios de clasificaciones del paralelismo implícito en una aplicación|**



**|Niveles de paralelismo implícito en una aplicación|**

En una aplicación podemos aplicar la clasificación por niveles

* Los programas serian Grano Grueso
* Funciones Grano Medio
* Bucles Grano Medio-Fino
* Operaciones Basicas Grano Fino

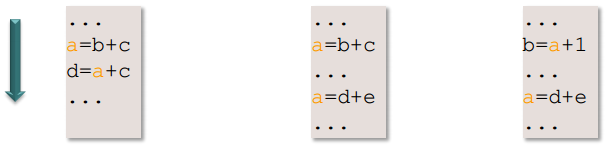
**|Dependencias de datos |**

Condiciones que se deben cumplir para que el bloque de código B2 presente dependencia de datos con respecto a B1 :

* Deben hacer referencia a una misma posición de memoria M (variable).
* B1 aparece en la secuencia de código antes que B2

Tipos de dependencias de datos (de B2 respecto a B1 ):

* RAW (Read After Write) o dependencia verdadera
* WAW (Write After Write) o dependencia de salida
* WAR (Write After Read) o antidependencia



**|Paralelismo implícito en una aplicación |**

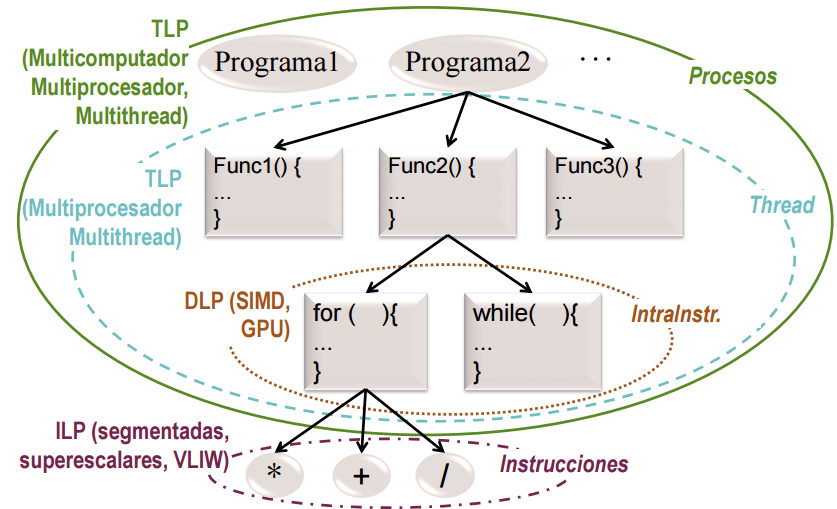
Paralelismo de tareas (task parallelism o TLP -Task Level Par.)

* Se encuentra extrayendo la estructura lógica de funciones de una aplicación.
* Está relacionado con el paralelismo a nivel de función

Paralelismo de datos (data parallelism o DLP-Data Level Par.)

* Se encuentra implícito en las operaciones con estructuras de datos (vectores y matrices)
  + Por ejemplo, la operación vectorial V1=V2+V3 engloba múltiples sumasν de escalares.
* Se puede extraer de la representación matemática de la aplicación.
* Está relacionado principalmente con el paralelismo a nivel de bucle

**|Paralelismo implícito (nivel de detección), explícito y arquitecturas paralelas |**



**|Nivel de paralelismo explícito. Unidades en ejecución en un computador|**

Instrucciones

* La unidad de control de un core o procesador gestiona la ejecución de instrucciones por la unidad de procesamiento

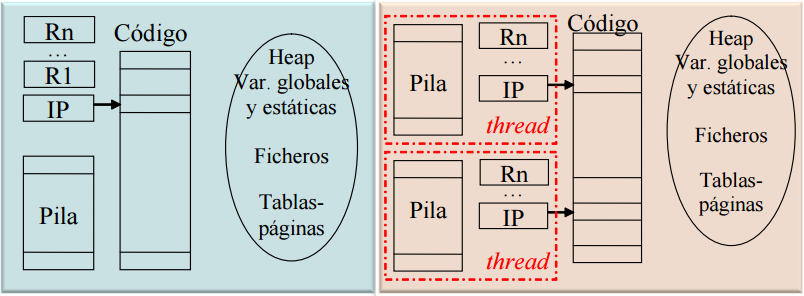
Thread o light process (concepto del SO)

* Es la menor unidad de ejecución que gestiona el SO
* Menor secuencia de instrucciones que se pueden ejecutar en paralelo o concurrentemente

Proceso o process (concepto del SO)

* Mayor unidad de ejecución que gestiona el SO Un proceso consta de uno o varios thread

**|Nivel de paralelismo explícito. Threads versus procesos |**



Proceso: comprende el código del programa y todo lo que hace falta para su ejecución:

* Datos en pila, segmentos (variables globales y estáticas) y en heap
* Contenido de los registros
* Tabla de páginas
* Tabla de ficheros abiertos

Para comunicar procesos hay que usar llamadas al SO

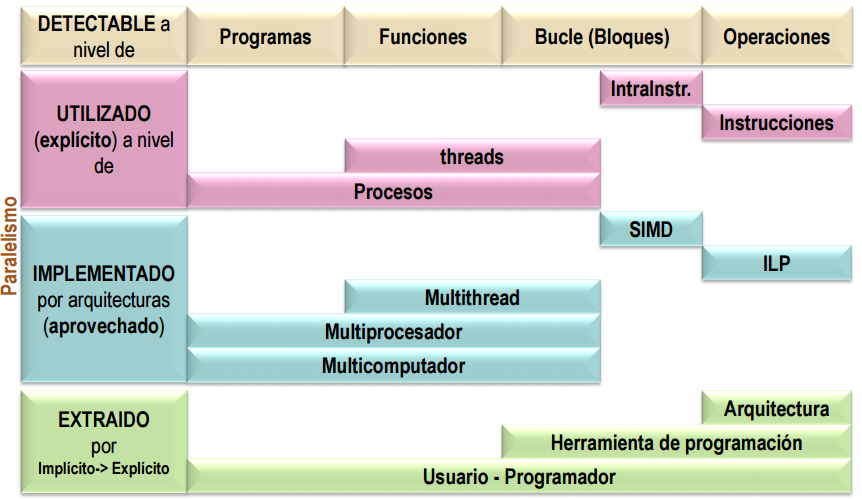
Un proceso puede constar de múltiples flujos de control, llamados threads o procesos ligeros. Cada thread tiene:

* Su propia pila
* Contenido de los registros, en particular el contador de programa o instruction pointer y el puntero de pila o stack pointer

Para comunicar threads de un proceso se usa la memoria que comparten



**|Detección, utilización, implementación y extracción del paralelismo|**



**Lección 2. Clasificación de arquitecturas paralelas**

**|** **Computación paralela – Computación distribuida|**

Computación paralela (se estudia en AC)

Estudia los aspectos hardware y software relacionados con el desarrollo y ejecución de aplicaciones en un sistema de cómputo compuesto por múltiples cores/procesadores/computadores que es visto externamente como una unidad autónoma (multicores, multiprocesadores, multicomputadores, cluster)

Computación distribuida

Estudia los aspectos hardware y software relacionados con el desarrollo y ejecución de aplicaciones en un sistema distribuido; es decir, en una colección de recursos autónomos (PC, servidores -de datos, aplicaciones, …-, supercomputadores …) situados en distintas localizaciones físicas

**|** **Computación distribuida a gran escala: Computación grid |**

Computación distribuida baja escala

Estudia los aspectos relacionados con el desarrollo y ejecución de aplicaciones en una colección de recursos autónomos de un dominio administrativo situados en distintas localizaciones físicas conectados a través de infraestructura de red local

Computación grid

Estudia los aspectos relacionados con el desarrollo y ejecución de aplicaciones en una colección de recursos autónomos de múltiples dominios administrativos geográficamente distribuidos conectados con infraestructura de telecomunicaciones

**|** **Computación distribuida a gran escala: Computación nube o cloud |**

Computación cloud

Comprende los aspectos relacionados con el desarrollo y ejecución de aplicaciones en un sistema cloud

Sistema cloud

Ofrece servicios de infraestructura, plataforma y/o software, por los que se paga cuando se necesitan (pay-per-use) y a los que se accede típicamente a través de una interfaz (web) de auto-servicio

Consta de recursos virtuales que

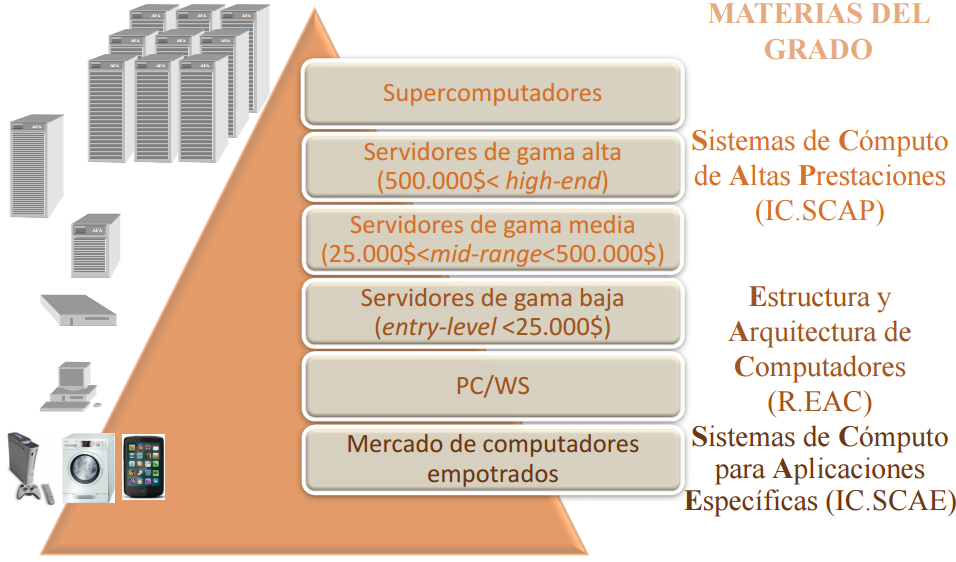
* son una abstracción de los recursos físicos
* parecen ilimitados en número y capacidad y son reclutados/liberados de forma inmediata sin interacción con el proveedor
* soportan el acceso de múltiples clientes (multitenant)
* están conectados con métodos estándar
* independientes de la plataforma de acceso.

**|** **Criterios de clasificación de computadores |**

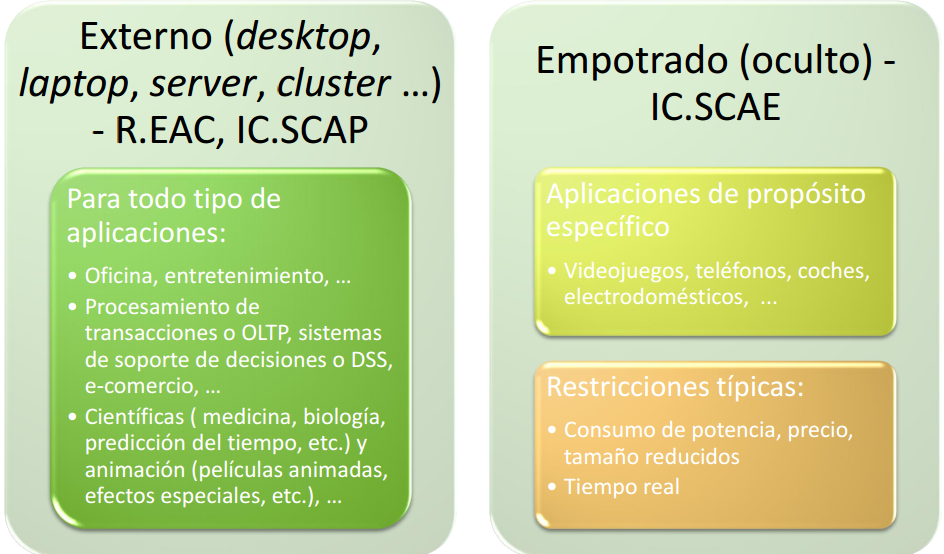
**Comercial**

Segmento del mercado

* embebidos, servidores gama baja



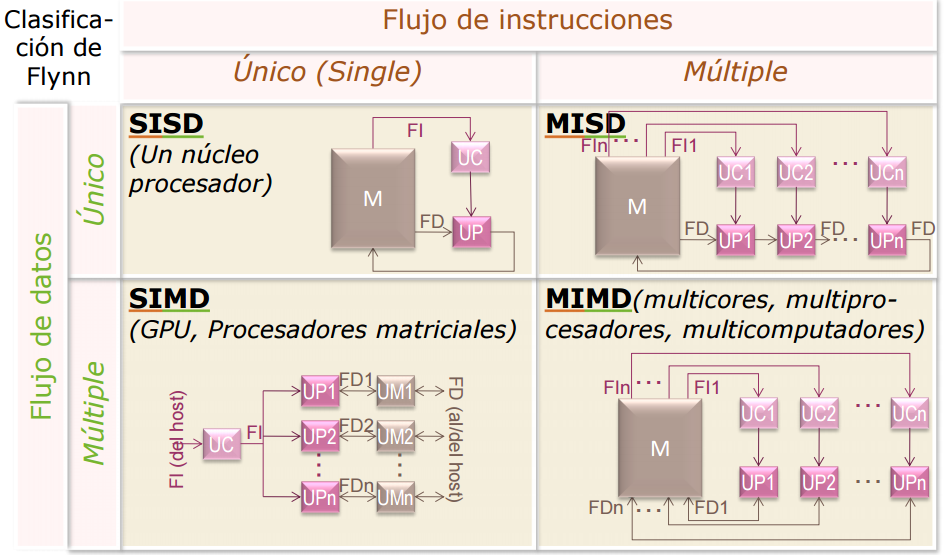
Clasificación de Computadores según segmento



**Educación, investigación (también usados por fabricantes y vendedores)**

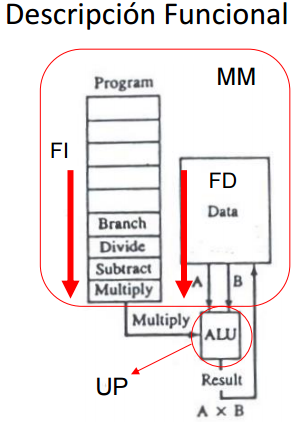
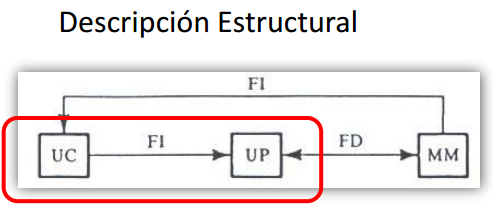
* Flujos de control y flujos de datos: clasificación de Flynn (1972)
* Sistema de memoria
* Flujos de control (propuesta de clasificación de arquitecturas con múltiples flujos de control)
* Nivel del paralelismo aprovechado (propuesta de clasif.)

Clasificación de Flynn de arquitecturas, 1972 (Flujo instr./flujo de datos)



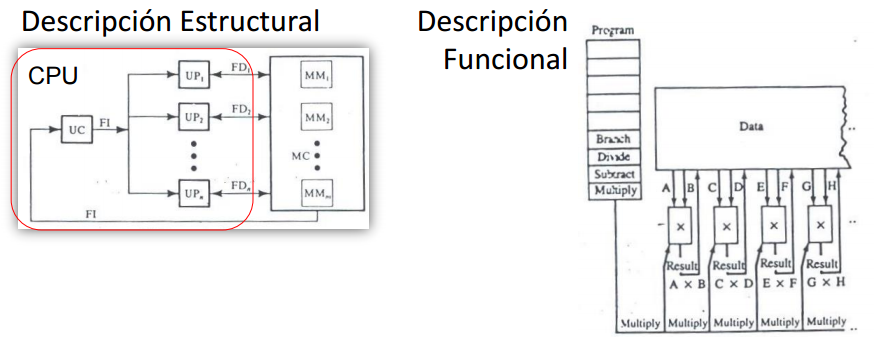
Arquitecturas SISD

Corresponde a los computadores uni-procesador (un núcleo o core procesador)



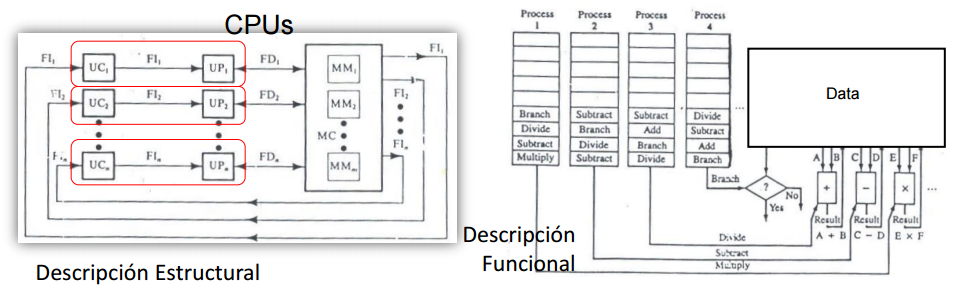
Arquitecturas SIMD

Aprovechan paralelismo de datos



Arquitecturas MIMD

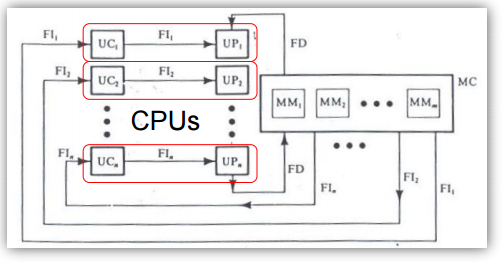
Corresponde con Multinúcleos, Multiprocesadores y Multicomputadores: Puede aprovechar, además, paralelismo funcional



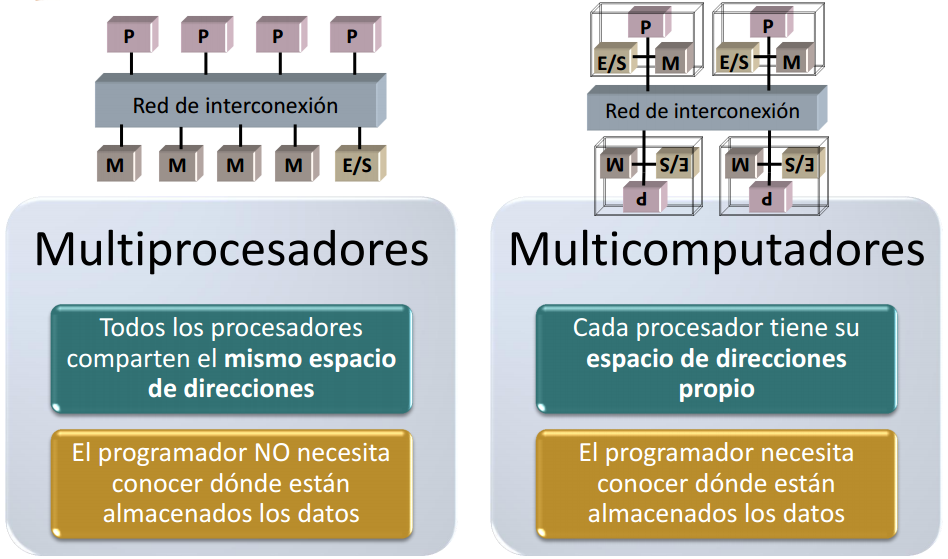
Arquitecturas MISD

No existen computadores que funcionen según este modelo

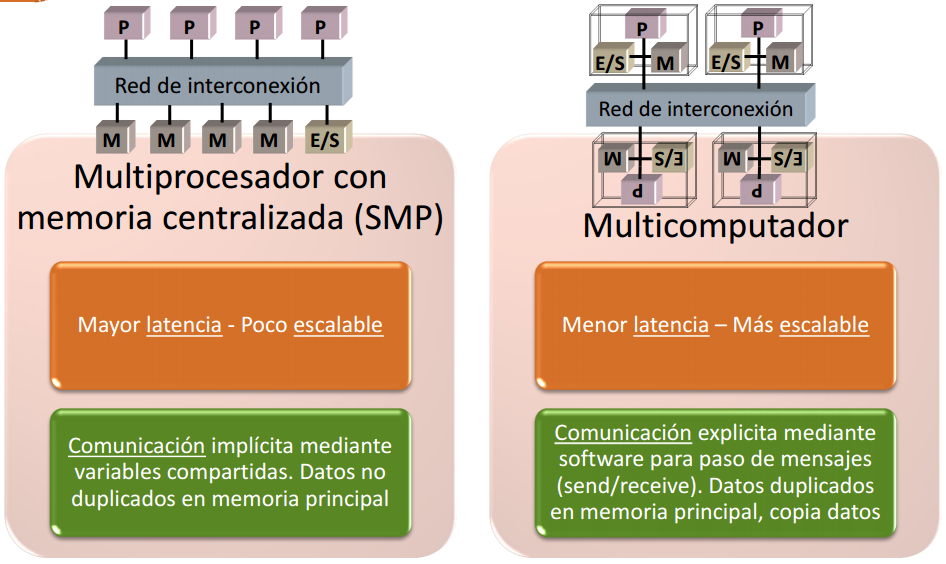
Se puede simular en un código este modelo para aplicaciones que procesan una secuencia o flujo de datos

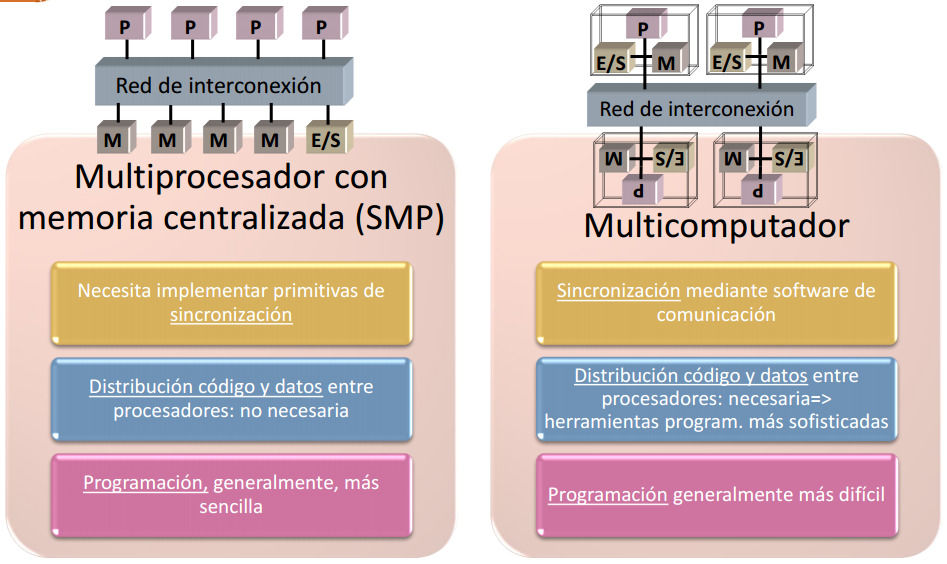


Clasificación de Computadores Paralelos MIMD según el sistema de memoria

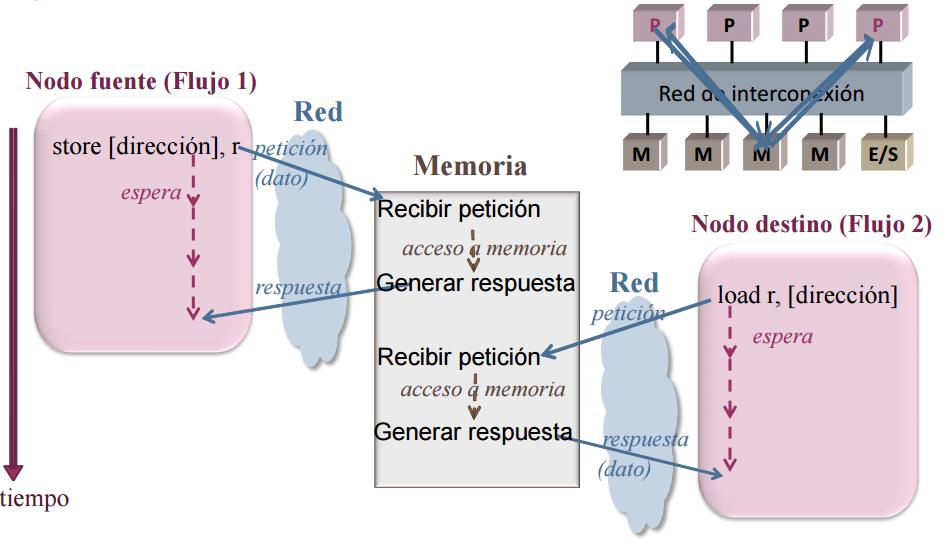


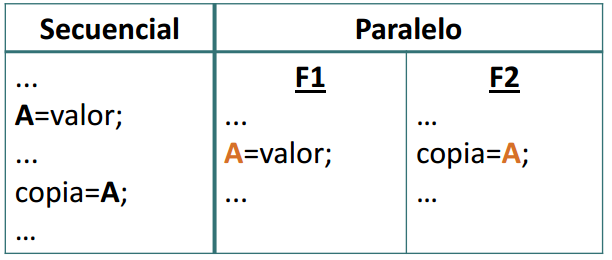
Comparativa SMP (Symmetric MultiProcessor) y multicomputadores





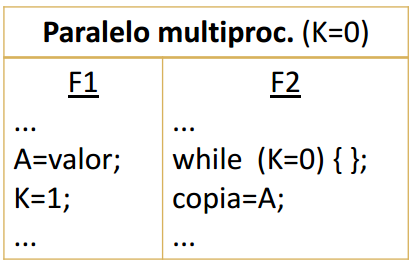
Comunicación uno-a-uno en un multiprocesador





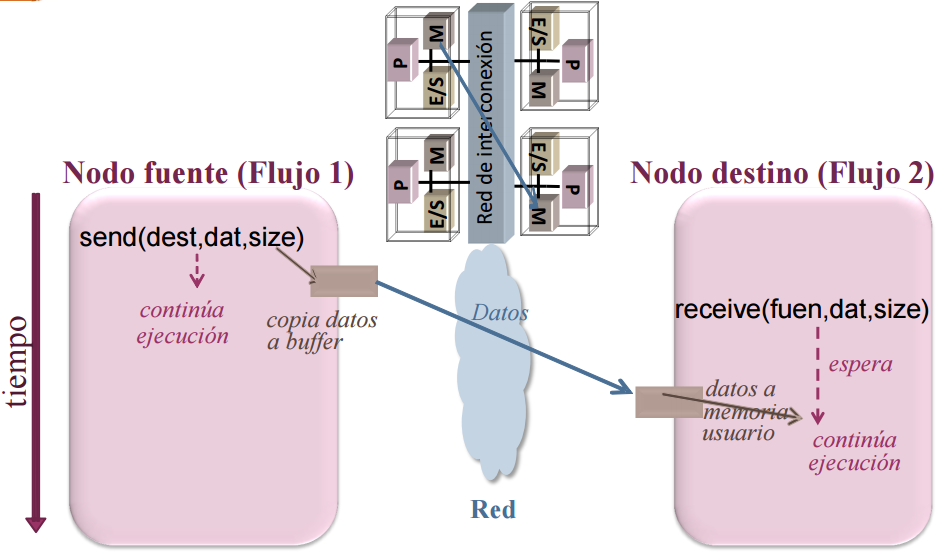
F1 es el flujo de control productor del dato (envía el dato)

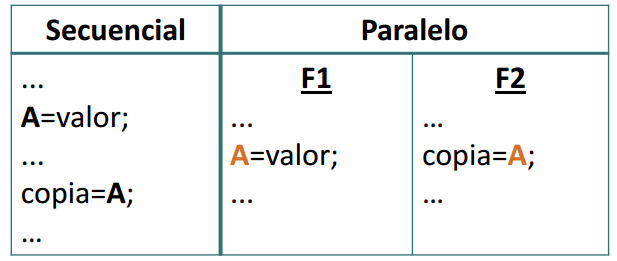
F2 es el flujo de control consumidor del dato (recibe el dato)



Se debe garantizar que el flujo de control consumidor del dato lea la variable compartida (A) cuando el productor haya escrito en la variable el dato.

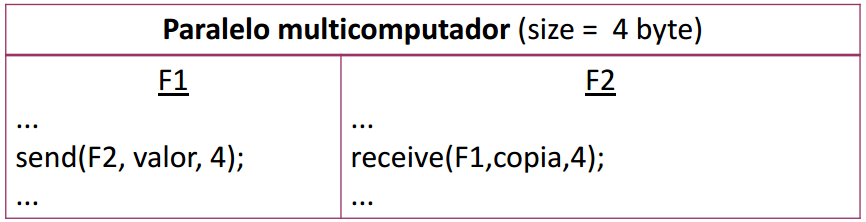
Comunicación uno-a-uno en multicomputador (receive bloqueante)





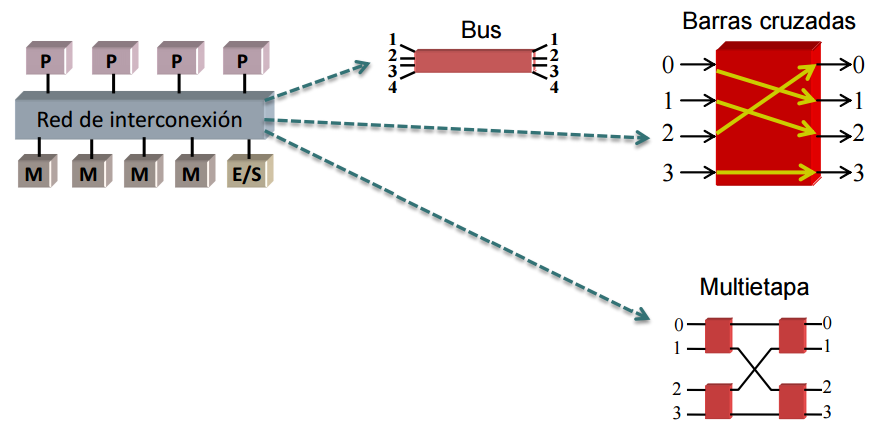
F1 es el flujo de control productor del dato (envía el dato)

F2 es el flujo de control consumidor del dato (recibe el dato)

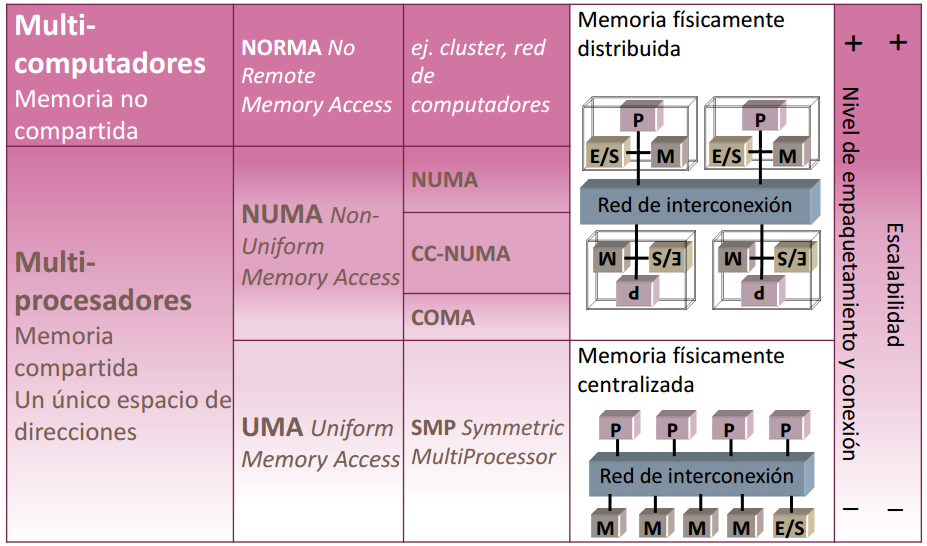


Incremento de escalabilidad en multiprocesadores y red de interconexión

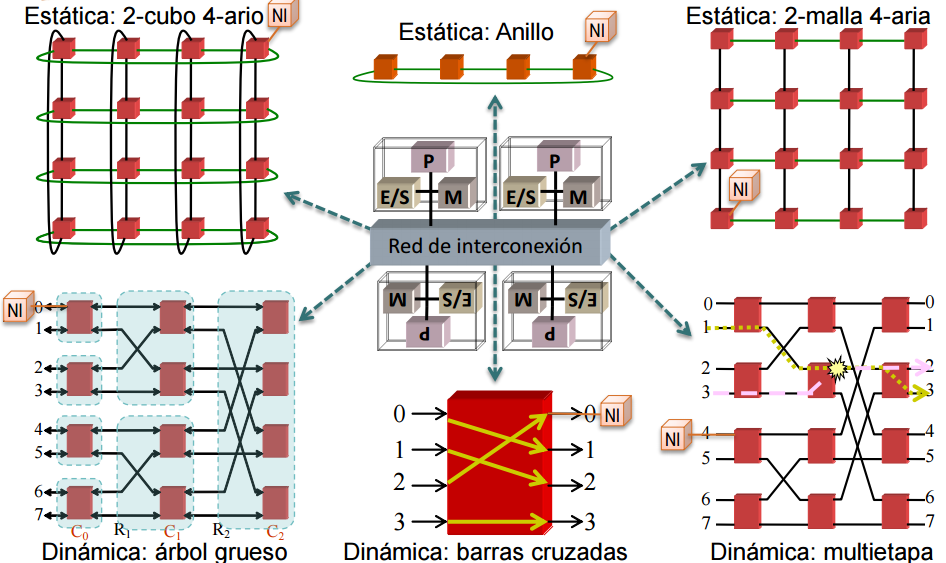
Incremento escalabilidad multiprocesadores:

* Aumentar cache del procesador
* Usar redes de menor latencia y mayor ancho de banda que un bus (jerarquía de buses, barras cruzadas, multietapa)
* Distribuir físicamente los módulos de memoria entre los procesadores (pero se sigue compartiendo espacio de direcciones)

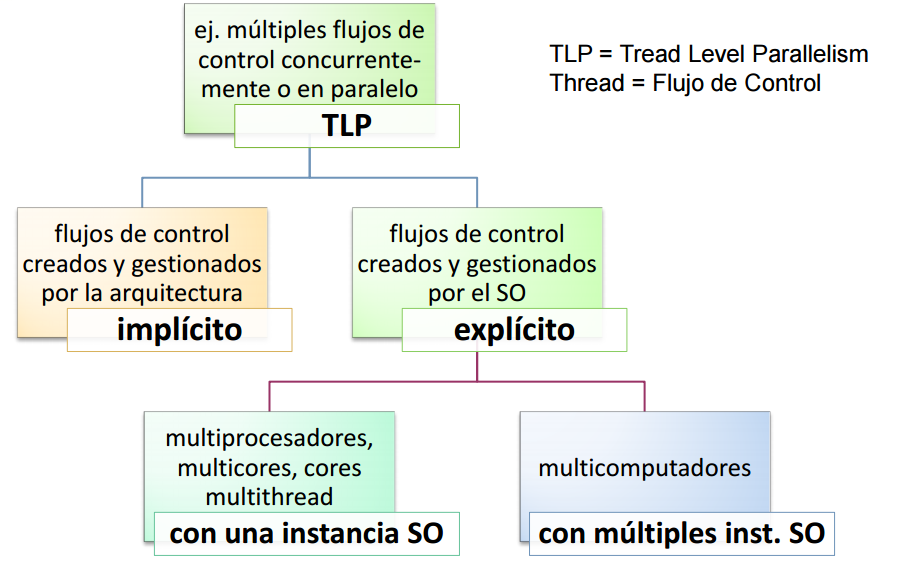
Clasificación completa de computadores según el sistema de memoria



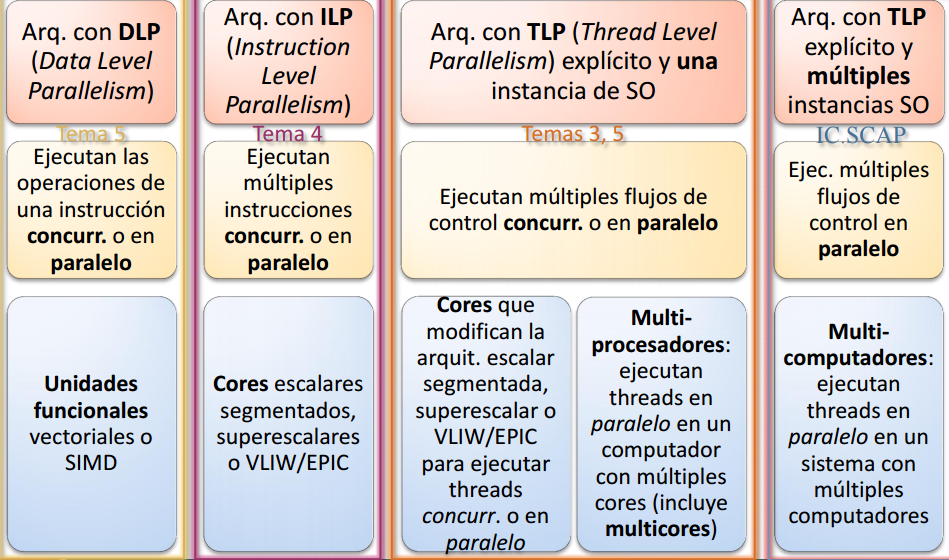
*Red en sistemas con memoria físicamente distribuida (NI: Network Interface)*

**

Propuesta clasificación arquitecturas con múltiples flujos de control o threads



Arquitecturas con DLP, ILP y TLP (thread=flujo de control)



**|Nota histórica|**

**Tema5**

DLP (Data Level Parallelism)

Unidades funcionales (o de ejecución) SIMD (o multimedia)

1989 (Intel i860). 1991 (motorola M88110). 1993 (repertorio MAX en HP PA :ν PA7100LC). 1995 (repertorio VIS en Sun Sparc: Ultra I). 1997 (repertorio MMX en Intel x86: Pentium MMX). 1999 (repertorio SSE en Intel x86: Pentium III; repertorio Altivec en IBM Power: PowerPC 8000)

**Tema4**

ILP (Instruction Level Parallelism) :

Procesadores/cores segmentados

1961 (IBM 7030). 1982 (chip Intel i286, Motorola 68020). 1986 (chip MIPS R2000).ν 1987 (chip AMD Am29000). 1988 (chip Sun Sparc)…

Procesadores con múltiples unidades funcionales

1967 (IBM 360/91) …

Procesadores/cores superescalares

1989 (chip Intel 960CA (3)). 1990: (chip IBM Power1 (4)). 1992: (chips DEC α21064ν (2/4), HP PA 7100 (2/2), Sun SuperSparc (3/5)) …

Procesadores/cores VLIW

1990 (chip DSP Intel i860 (2)).1997 (chip DSP TMS320C6x (8)). 2001 (chip Intelν Itanium)…

**Tema3**

TLP explícito con una instancia de SO:

Multithread grano fino (FGMT)

1975 (Denelcor HEP). 2005 (chip Sun UltraSPARC T1) …

Multithread grano grueso (CGMT)

1990 (MIT Alewife). 2000 (chip IBM PowerPC RS64 IV (2)). 2006 (chip Intel Itaniumν Montecito (2)) …

Multithread simultánea (SMT)

2002 (chip Intel Pentium 4/Xeon Hyper-Threading). 2004 (chip IBM Power5) …

Multiprocesadores en un chip (CMP) o multicores

2001 (chip IBM Power4). 2004 (chip Sun UltraSPARC IV). 2006 (chip Intel Core Duo.ν 2008 (chip Intel Celeron Dual-core) …

Multiprocesadores

1962 (Burroughs D825 - red barras cruzadas). 1966 (UNIVAC 1108 - red bus). 1985ν (IBM RP3 - red multietapa). 1996 (SGI Origin 2000 -CC-NUMA, red estática+multietapa) .2006 (SGI Altix 4000) …

**IC.SCAP**

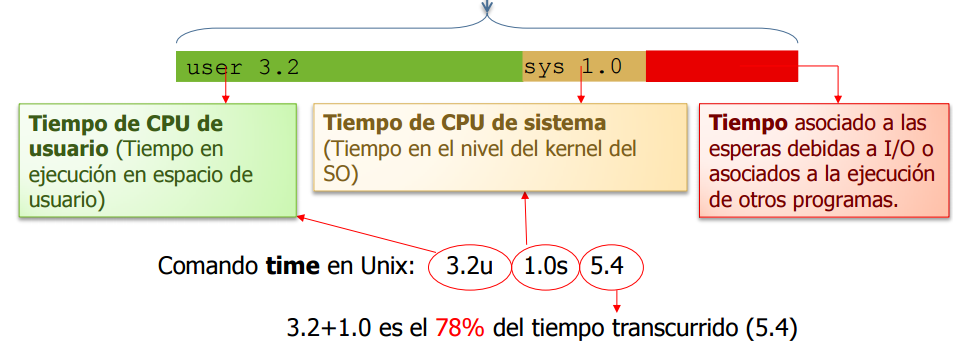
TLP explícito con múltiples instancias del SO (multicomputadores)

1985 (Intel iPSC1 - i286+red estática con Ethernet) … cualquier clusterν

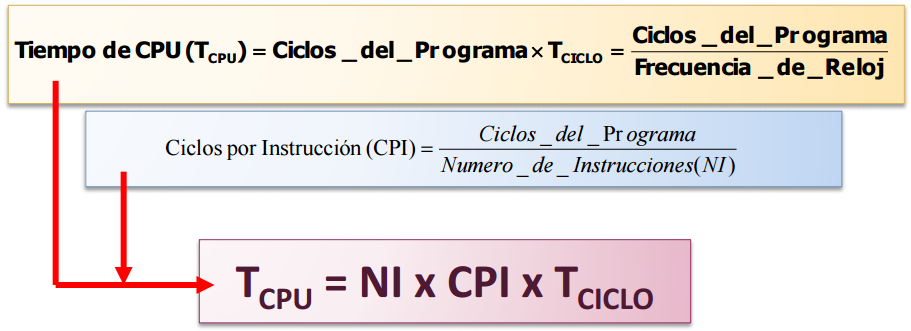
**Lección 3. Evaluación de prestaciones de una arquitectura**

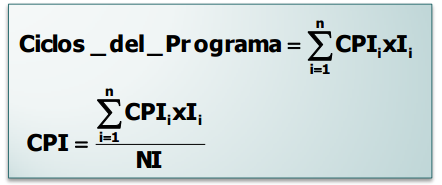
**|** **Tiempo de respuesta de un programa en una arquitectura |**

* Real (wall-clock time, elapsed time, real time)
* CPU time = user + sys (no incluye todo el tiempo)
* Con un flujo de controlν
  + Elapsed >= CPU time
* Con múltiples flujos de controlν
* elapsedν < CPU time, elapsed >= CPU time/nº flujos control



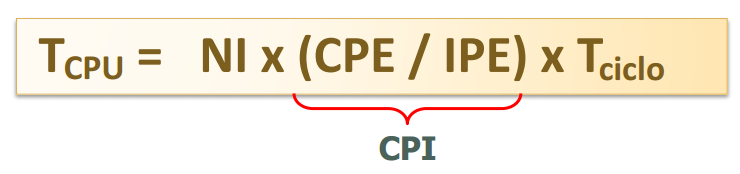
**|Tiempo de CPU|**

****



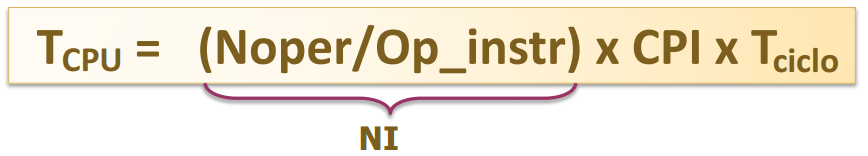
En el programa hay I i instrucciones del tipo i (i=1,…n)

Cada instrucción del tipo i consume CPIi ciclos Hay n tipos de instrucciones distinto

****

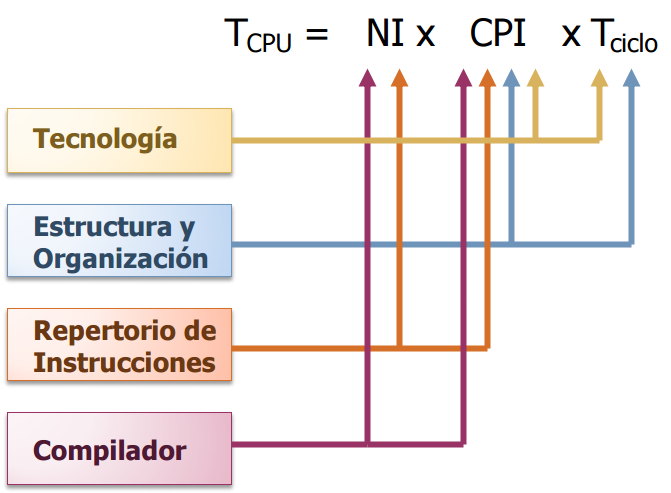
Hay procesadores que pueden lanzar para que empiecen a ejecutarse (emitir) varias instrucciones al mismo tiempo.

* CPE: Número mínimo de ciclos transcurridos entre los instantes en que el procesador puede emitir instrucciones
* IPE: Instrucciones que pueden emitirse (para empezar su ejecución) cada vez que se produce dicha emisión.

****

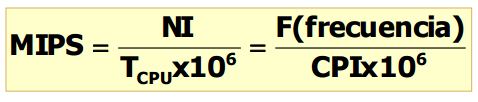
Hay procesadores que pueden codificar varias operaciones en una instrucción.

* Noper: Número de operaciones que realiza el programa
* Op\_instr: Número de operaciones que puede codificar una instrucción

****

**|MIPS|**

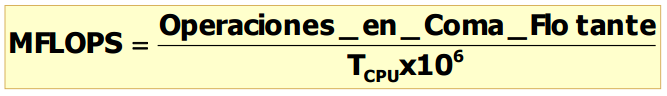
MIPS: Millones de Instrucciones por segundo



* Depende del repertorio de instrucciones (difícil la comparación de máquinas con repertorios distintos)
* Puede variar con el programa (no sirve para caracterizar la máquina)
* Puede variar inversamente con las prestaciones (mayor valor de MIPS corresponde a peores prestaciones) MIPS (Meaningless Indication of Processor Speed)

**|MFLOPS|**

MFLOPS: Millones de operaciones en coma flotante por segundo



* No es una medida adecuada para todos los programas (sólo tiene en cuenta las operaciones en coma flotante del programa)
* El conjunto de operaciones en coma flotante no es constante en máquinas diferentes y la potencia de las operaciones en coma flotante no es igual para todas las operaciones (por ejemplo, con diferente precisión, no es igual una suma que una multiplicación..)

Es necesaria una normalización de las instrucciones en coma flotante.

**|Benchmarks|**

* Propiedades exigidas a medidas de prestaciones:
  + Fiabilidad => Representativas, evaluar diferentes componentes del sistema y reproducibles
  + Permitir comparar diferentes realizaciones de un sistema o diferentes sistemas=> Aceptadas por todos los interesados (usuarios, fabricantes, vendedores)
* Interesados:
  + Vendedores y fabricantes de hardware o software.
  + Investigadores de hardware o software.
  + Compradores de hardware o software.

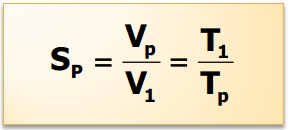
Tipos de Benchmark:

* De bajo nivel o microbenchmark test ping-pong, evaluación de las operaciones con enteros o conν flotantes
* Kernels resolución de sistemas de ecuaciones, multiplicación de matrices,ν FFT, descomposición LU
* Sintéticos Dhrystone, Whetstone
* Programas reales SPEC CPU2006: enteros (gcc, gzip, perlbmk)
* Aplicaciones diseñadas Predicción de tiempo, simulación de terremotos.

**|Mejora o Ganancia de Prestaciones (Speedup o ganancia en velocidad)|**

Si en un computador se incrementan las prestaciones de un recurso haciendo que su velocidad sea p veces mayor (ejemplos: se utilizan p procesadores en lugar de uno, la ALU realiza las operaciones en un tiempo p veces menor,..):

El incremento de velocidad que se consigue en la nueva situación con respecto a la previa (máquina base) se expresa mediante la ganancia de velocidad o speed-up, Sp



V1 Velocidad de la máquina base

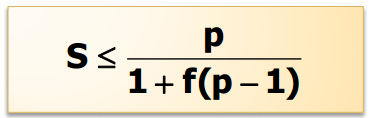
Vp Velocidad de la máquina mejorada (un factor p en uno de sus componentes)

T1 Tiempo de ejecución en la máquina base

Tp Tiempo de ejecución en la máquina mejorada

Ley de Amdahl

La mejora de velocidad, S, que se puede obtener cuando se mejora un recurso de una máquina en un factor p está limitada por:



Donde f es la fracción del tiempo de ejecución en la máquina sin la mejora durante el que no se puede aplicar esa mejora.

Ejemplo: Si un programa pasa un 25% de su tiempo de ejecución en una máquina realizando instrucciones de coma flotante, y se mejora la máquina haciendo que estas instrucciones se ejecuten en la mitad de tiempo, entonces p=2; f=0.75; y S≤2/(1+0.75)=1.14