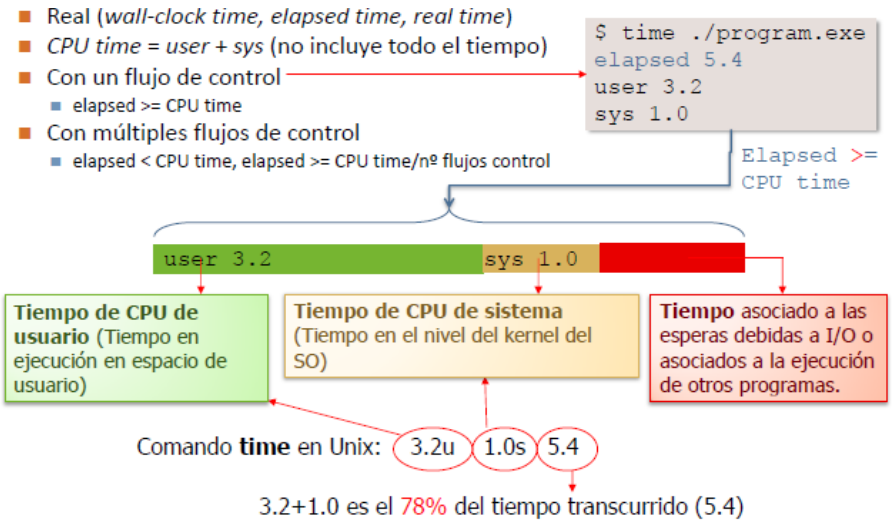
Arquitectura de Computadores. Tema 1

Arquitecturas paralelas: clasificación y prestaciones

Lección3: Evaluación de prestaciones de una arquitectura

Tiempo de respuesta de un programa es una arquitectura



Alternativas para obtener tiempos

Función	Fuente	Tipo	Resolución aprox. (microsegundos)
time	SO (/usr/bin/time)	elapsed, user, system	10000
clock()	SO (time.h)	CPU	10000
gettimeofday()	SO (sys/time.h)	elapsed	1
clock_gettime()/clock_getres()	SO (time.h)	elapsed	0.001
omp_get_wtime()/ omp_get_wtick()	OpenMP (omp.h)	elapsed	0.001
SYSTEM_CLOCK()	Fortran	elapsed	1

Tiempo de CPU

$$\text{Tiempo de CPU } (T_{CPU}) = \text{Ciclos\_del\_Programa} \times T_{CICLO} = \frac{\text{Ciclos\_del\_Programa}}{\text{Frecuencia\_de\_Reloj}}$$

$$\text{Ciclos por Instrucción (CPI)} = \frac{Ciclos\_del\_Programa}{Numero\_de\_Instrucciones(NI)}$$

$$T_{CPU} = NI \times CPI \times T_{CICLO}$$

$$\text{Ciclos\_del\_Programa} = \sum_{i=1}^{n} CPI_i \times I_i$$

$$CPI = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{NI}$$

En el programa hay I_i instrucciones del tipo i (i=1,...n)

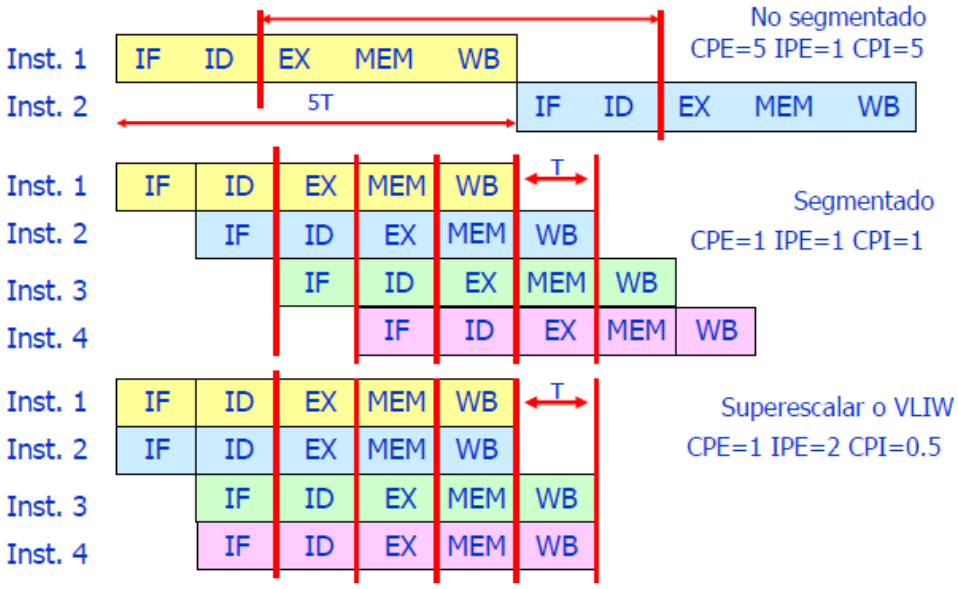
Cada instrucción del tipo i consume CPI_i ciclos

Hay n tipos de instrucciones distintos.

$$T_{CPU} = NI \times \underbrace{(CPE / IPE)}_{CPI} \times T_{ciclo}$$

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

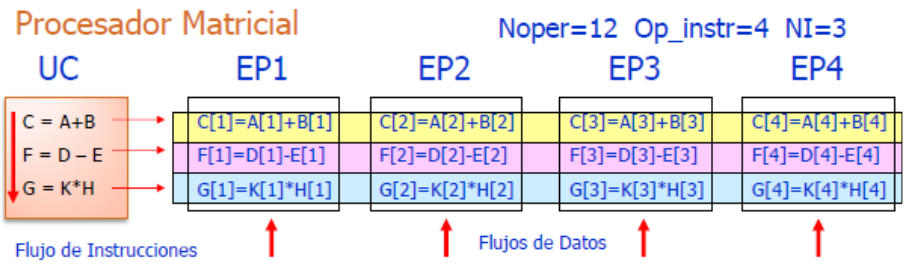
- CPE: número mínimo de ciclos transcurridos entre los instantes en que el procesador puede emitir instrucciones
- IPE: instrucciones que pueden emitirse (para empezar su ejecución) cada vez que se produce dicha emisión.

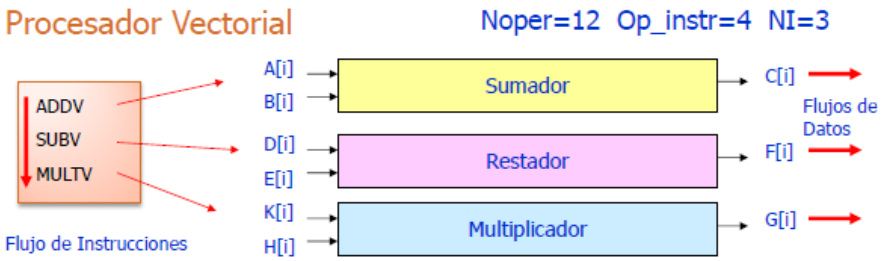


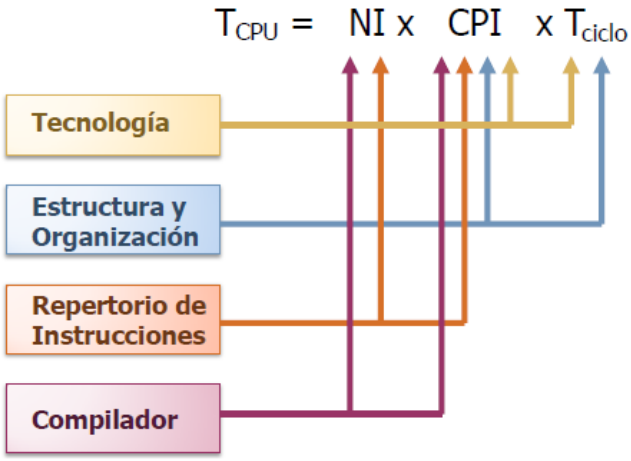
$$T_{CPU} = \underbrace{(Noper/Op\_instr)}_{NI} \times CPI \times T_{ciclo}$$

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







Productividad: MIPS Y MFLOPS

MIPS: Millones de Instrucciones por segundo

$$MIPS = \frac{NI}{T_{CPU} x 10^6} = \frac{F(frecuencia)}{CPI x 10^6}$$

- 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)

MFLOPS: Millones de operaciones en coma flotante por segundo

$$MFLOPS = \frac{Operaciones\_en\_Coma\_Flotante}{T_{CPU} x 10^6}$$

- 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

Conjunto de programas de prueba (Benchmark)

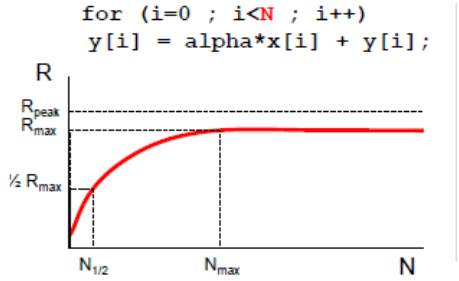
- Propiedades exigidas a medidas de prestaciones
    - Fiabilidad: representativas, evaluar diferentes componentes del sistema y reproducibles
- Permitir comparar diferentes realizaciones de un sistema o diferentes sistemas: aceptadas por los interesados (usuarios, fabricantes, vendedores)
- Interesados:
    - Vendedores y fabricantes de hardware o software
    - Investigadores de hardware o software
    - Compradores de hardware o software
- Tipos de Benchmarks
    - De bajo nivel o microbenchmark
        - Test ping-pong, evaluación de las operaciones con enteros o con flotantes
    - Kernels
        - Resolución de sistemas de ecuaciones, multiplicación de matrices, FFT, descomposición LU
    - Sintéticos
        - Dhrystone, Whetstone
    - Programas reales
        - SPEC CPU2006: enteros (gcc, gzip…)
    - Aplicaciones diseñadas
        - Predicción de tiempo, simulación de terremotos
- Benchmark suites
    - SPEC CPU2006
        - Evaluación de operaciones con enteros (CINT2006) y con punto flotante (CFP2006) en un core
        - Tipo: aplicaciones reales
        - Herramientas: C, C++ y Fortran
    - SPEC OMP 2001
        - Científico
        - Variables compartidas
        - Aplicaciones diseñadas. Basado en SPEC CPU2000. Evalúa procesador, memoria, SO y herramientas de programación
        - Herramientas: OpenMP

- SPEC HPC2002
  - Científico
  - Variables compartidas, paso de mensajes y combinación de ambos
  - Tipo: basados en aplicaciones HPC diseñadas reales. Evalúa procesador, comunicación, E/S, compilador y bibliotecas
  - Herramientas: Serie, OpenMP, MPI, combinación MPI-OpenMP
- TPC (transaction processing performance council)
  - Procesamiento de transacciones o OLTP; sistemas de soporte de decisiones o DSS; comercio electrónico, servidores web y de aplicaciones
  - Tipo: entradas software comercial (bases de datos, servidores de internet) y carga de trabajo diseñada
- NPB2, NPB3 (NAS Parallel Benchmark)
  - Científico
  - Paso de mensajes, variables compartidas
  - Tipo: núcleos y aplicaciones diseñadas
  - Herramientas: NPB2 (MPI), NPB3 (OpenMP, java, HPF)
- Implementaciones de la biblioteca BLAS (basic linear algebra subprograms)
  - Tipo: núcleos con operaciones del álgebra lineal
  - Herramientas: hay implementaciones con diferentes herramientas de programación (Fortran, C, C++…)

➢ LINPACK

- El núcleo de este programa es una rutina denominada DAXPY (doublé alpha X plus Y) que multiplica un vector por una constante y lo suma a otro vector. Las prestaciones obtenidas se escalan con el valor de N (tamaño del vector).

```
for (i=0 ; i<N ; i++)
 y[i] = alpha*x[i] + y[i];
```



Mejora o ganancia de prestaciones (speed-up o ganancia en velocidad)

Si en un computador se incrementan las prestaciones de un recurso haciendo que su velocidad sea p veces mayor (Ej: 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, S_p:

$$S_p = \frac{V_p}{V_1} = \frac{T_1}{T_p}$$

V_1 Velocidad de la máquina base
V_p Velocidad de la máquina mejorada (un factor p en uno de sus componentes)
T_1 Tiempo de ejecución en la máquina base
T_p Tiempo de ejecución en la máquina mejorada

La 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:

$$S \leq \frac{p}{1 + f(p-1)}$$

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**

Hay que mejorar el caso más frecuente (lo que más se usa)