Medición del Rendimiento de Computadores

Motivación:

- ✓ ¿Qué medidas de tiempo se utilizan para evaluar el rendimiento?
- ✓ ¿Cómo definimos el tiempo de cpu y el rendimiento del procesador?
- ✓ ¿Qué parámetros condicionan el tiempo de cpu?
- ✓ ¿Qué medidas de productividad se utilizan para evaluar el rendimiento?
- ✓ ¿Qué ventajas e inconvenientes presentan dichas medidas?
- ✓ ¿Qué tipos de programas se utilizan para evaluar el rendimiento?

Medición del Rendimiento de Computadores

- Introducción
- Tiempo de ejecución. Rendimiento del Procesador
- Medidas de Productividad
  - ✓ MIPS
  - ✓ MFLOPS
- Programas de evaluación *(Benchmarks)*

Medición del Rendimiento

- ¿Por qué se necesita medir el rendimiento?
  - ✓ comparación del hardware de las máquinas
  - ✓ comparación del software de las máquinas (compiladores)
- Propósito final:
  - ✓ tomar decisiones de compra
  - ✓ desarrollar nuevas arquitecturas
- ¿Basta con una sola medida?
  - ✓ ¿es más rápida una máquina de 2 GHz que otra de 1,5 GHz?
- Medidas utilizadas:
  - ✓
  - ✓

Tiempo de Ejecución

- Tiempo de Respuesta (transcurrido, de reloj, *elapsed time*)
  - ✓ incluye todo (procesador, accesos a memoria, E/S, S.O., multiprogramación)
  - ✓ en un sistema descargado da idea del rendimiento del computador o de sistema
  - ✓ útil, pero no muy adecuado para efectuar comparaciones
- Tiempo de CPU
  - ✓ no incluye E/S ni tiempo ejecutando otros programas (multiprogramación)
  - ✓ Tcpu = Tcpu (usuario) + Tcpu (sistema)
  - ✓ Tcpu (sistema) = tiempo ejecutando código del S.O.
- Tcpu (usuario)
  - ✓ tiempo dedicado a la ejecución de código de “nuestro” programa
  - ✓ da idea del Rendimiento del Procesador o Rendimiento de CPU
  - ✓ esta influido por

Ciclo de Reloj

- El rendimiento se calcula en función del ciclo de reloj

$$\frac{\text{segundos}}{\text{programa}} = \frac{\text{ciclos}}{\text{programa}} \times \frac{\text{segundos}}{\text{ciclo}}$$

- El tiempo se puede medir en ciclos de reloj

- ✓ Tiempo de ciclo (**T**) = tiempo entre dos eventos de reloj (período de reloj)
- ✓ Frecuencia de reloj (**f**) = ciclos / segundo
- ✓
- ✓

Ciclos por Instrucción

- ¿Son iguales el número de instrucciones y el de ciclos?, **NO**
  - ✓ el número de ciclos depende de la organización hardware
  - ✓ el número cambia para cada procesador
- ¿Consumen todas las instrucciones igual nº de ciclos?, **NO**
  - ✓ las multiplicaciones/divisiones consumen mas que las sumas/restas
  - ✓ las instrucciones flotantes consumen mas que las enteras
  - ✓ las que acceden a memoria consumen mas que las que acceden a registros
- Ciclos por Instrucción (**CPI**)
  - ✓ número de ciclos promedio consumido por las instrucciones de un programa
  - ✓ permite comparar organizaciones alternativas que soportan una misma ISA
  - ✓ Si NI = nº de instrucciones del programa ⇒

Rendimiento del Procesador

NI = nº de instrucciones del programa

CPI = ciclos por Instrucción

T = tiempo de ciclo

nº total de ciclos = NI x CPI

T_{CPU} (usuario) = nº total de ciclos x tiempo de ciclo = (NI x CPI) x T

$$\mathbf{T_{CPU} = T \times \sum NI_i \times CPI_i} \quad \text{(i = tipo de instrucción)}$$

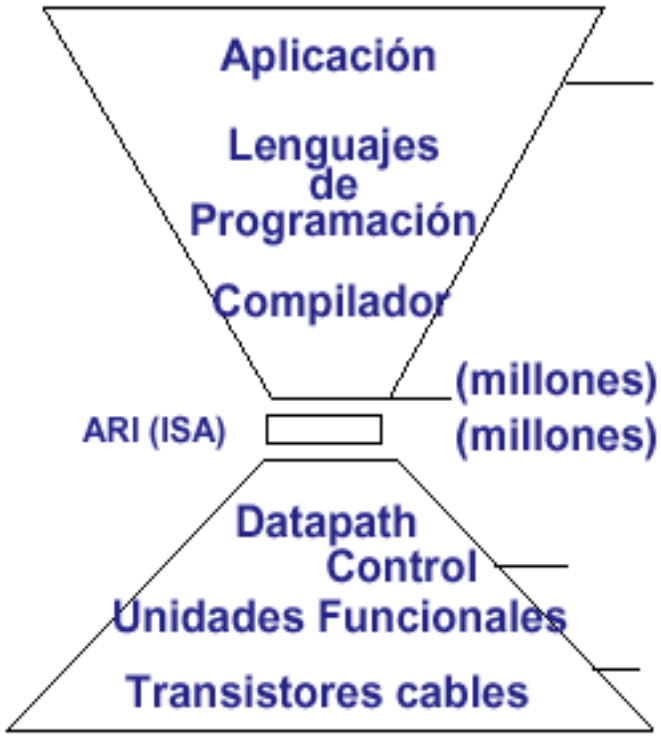
$$\mathbf{CPI = \sum CPI_i \times F_i} \quad \text{(Fi = frecuencia de aparición de instrucciones i)}$$

NI = función(Compilador, Lenguaje máquina)

CPI = función(Lenguaje máquina, Organización)

T = función(Organización, Tecnología)

Medidas de Productividad



Respuestas por mes, hora, segundo
Operaciones por segundo

(millones) de Instrucciones por segundo: MIPS
(millones) de (FP) operaciones por segundo: MFLOPS

Megabytes por segundo

Ciclos por segundo (frecuencia de reloj)

La única medida fiable es el tiempo de ejecución programas reales
Dos aspectos: Rendimiento del computador, *Rendimiento del procesador*

MIPS

- **M**illones de **I**nstrucciones **P**or **S**egundo

$$MIPS = \frac{NI}{Tejecucion \times 10^6} = \frac{NI}{TCPU \times 10^6} = \frac{1}{CPI \times T \times 10^6} = \frac{f}{CPI \times 10^6}$$

- Mayor velocidad implica mayor valor de MIPS *(a priori)*

$$Tejecucion = \frac{NI}{MIPS \times 10^6}$$

MIPS

- **Inconvenientes:**
  - ✓ MIPS depende del juego de instrucciones
    ⇒ difícil comparar máquinas con JI diferentes
  - ✓ MIPS depende del programa (de su mezcla de instrucciones)
  - ✓ MIPS puede llegar a variar inversamente al rendimiento
    ⇒ medida no consistente (
    solución:

MFLOPS

- ***M**illions of **FL**oating Point **O**perations **P**er **S**econd*

$$\text{MFLOPS} = \frac{\text{Nº Operaciones Flotantes}}{\text{Tejecucion} \times 10^6}$$

- Aplicable a programas con Operaciones Flotantes:
  - ✓ sumas, restas, multiplicaciones, divisiones
- Simple precisión o Doble precisión

MFLOPS

- **Inconvenientes:**
  - ✓ MFLOPS dependen de si se dispone o no de FPU *(Floating Point Unit)*
  - ✓ MFLOPS dependen de si están o no soportadas funciones complejas (multiplicación, división, raíz cuadrada, seno, coseno, etc.)
  - ✓ MFLOPS dependen de la mezcla de I. enteras y flotantes del programa
  - ✓ MFLOPS dependen de la mezcla de operaciones flotantes (operaciones mas complejas tienen un mayor tiempo de ejecución)
    solución:

Medición del Rendimiento

- **La única medida objetiva es el Tiempo de Ejecución**
- **Cualquier otra medida ofrece una visión parcial**
  - ✓ nº de instrucciones por programa
  - ✓ nº de ciclos por programa
  - ✓ nº de ciclos por instrucción
  - ✓ nº de segundos por ciclo
  - ✓ nº de ciclos por segundo
  - ✓ nº de instrucciones por segundo

Resumen de Vocabulario

NI	número de instrucciones *(máquina)*
CPI	*(ciclos por instrucción)*
T*ejecucion*	tiempo de ejecución *(segundos por programa)*
T	tiempo de ciclo *(segundos por ciclo)*
f	frecuencia de reloj *(ciclos por segundo)*
MIPS	*(millones de instrucciones por segundo)*
MFLOPS	*(millones de operaciones flotantes por segundo)*

Programas de Evaluación *(Benchmarks)*

- Carga ideal para la evaluación:
  - ✓ Programas Usuario + Comandos S.O. (Carga Trabajo, *Workload*)
- Problema:
  - ✓ No disponibilidad de los computadores para correr la Carga de Trabajo
- Alternativa:
  - ✓ Cargas neutrales ejecutadas por los fabricantes u organismos evaluadores
- Tipos de programas de evaluación *(benchmarks):*
  - ✓
  - ✓
  - ✓

Benchmarks sintéticos

- Simulan la frecuencia de instrucciones y operandos de un abanico de programas reales (código artificial)
- Ejemplos:
  - ✓ **Dhrystone** (código entero)
  - ✓ **Whetstone** (código flotante)
- Inconvenientes:
  - ✓ optimizaciones del fabricante solo para la mejora de ese tipo de programas
  - ✓ compiladores detectores de benchmarks

    ⇒
  - ✓ su pequeño tamaño da lugar a un rendimiento óptimo del sistema de memoria

    ⇒

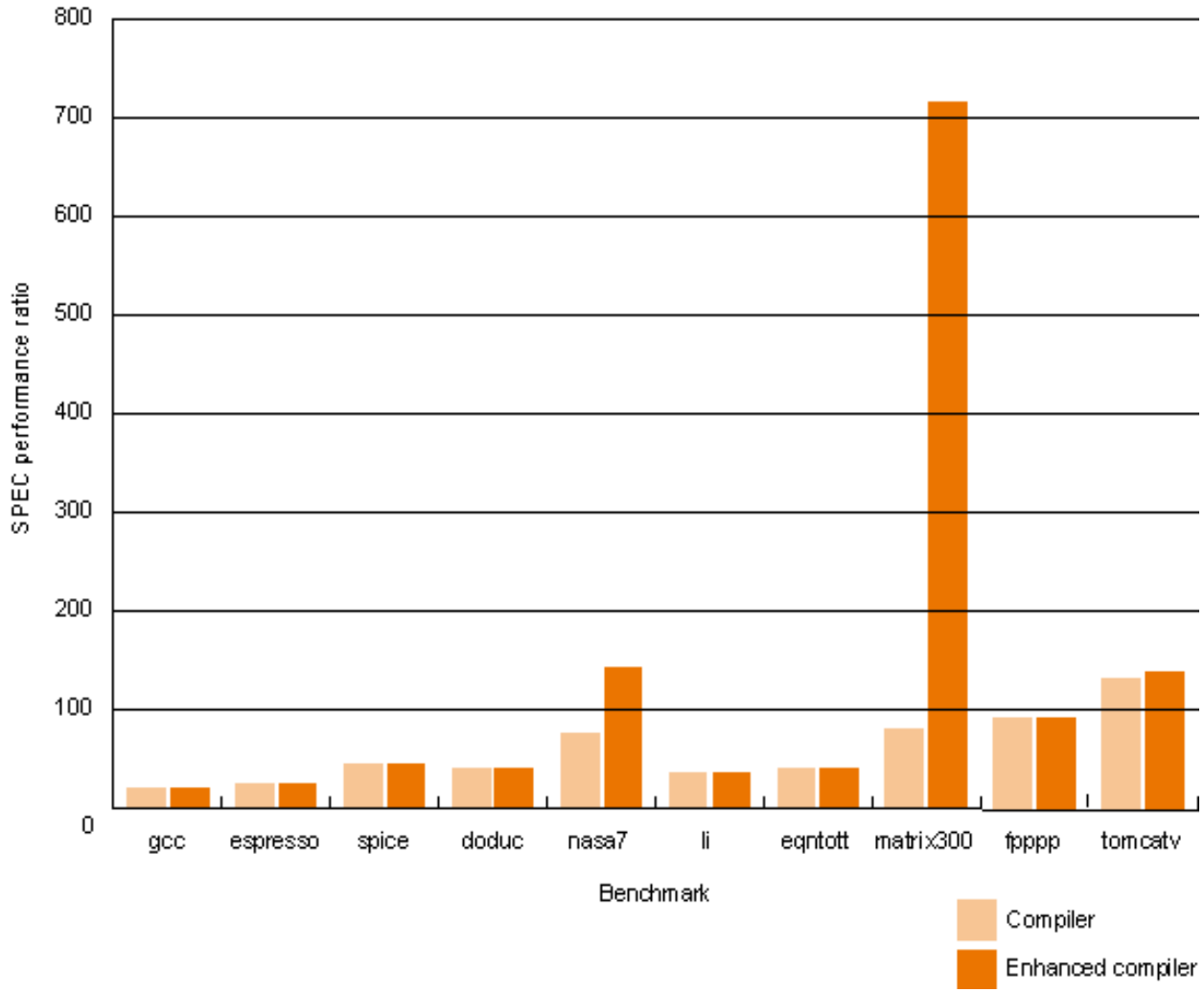
Programas de tamaño reducido

- Pequeños programas de entre 10 y 100 líneas con resultado conocido
- Incluyen normalmente bucles, repitiéndose mucho algunas instrucciones
- Fáciles de simular durante el diseño de la máquina (compilador no disponible)
- Fáciles de estandarizar

- Ejemplos:
  - ✓ programas de ordenación
  - ✓ multiplicación de matrices, etc.
  - ✓ **Flops**: resolución de integrales
  - ✓ **Linpack**: resolución de sistemas de ecuaciones lineales

- Inconvenientes:
  - ✓ optimizaciones del fabricante solo para la mejora de ese tipo de programas
  - ✓ rendimiento óptimo del sistema de memoria ⇒

Optimizaciones específicas para *Benchmarks*



Aplicaciones reales

- Programas de uso común que forman parte de las cargas de trabajo
  - ✓ Compiladores y Editores de texto
  - ✓ Bases de datos y Hojas de cálculo
  - ✓ Programas científicos

- Ejemplos: Conjunto de programas **SPEC** y *benchmark* **SYSmark** (PCs)
  - ✓ SPEC =
  - ✓ Programas consensuados por el conjunto de fabricantes
  - ✓ Los mas utilizados para medir rendimiento y eficiencia del compilador
  - ✓ Actualizaciones continuas: SPEC 92, SPEC 95, SPEC 2000, SPEC 2006
  - ✓ SYSmark:
  - ✓

Problema 2.2

1) MIPS con copro = f_{CPU} / (CPI con copro x 10E6) =
   MIPS sin copro = f_{CPU} / (CPI sin copro x 10E6) =

2) NI con copro = MIPS con copro x 10E6 x Tej con copro =
   NI sin copro = MIPS sin copro x 10E6 x Tej sin copro =

3) NI con copro = = N Iter x (NI enteras básicas + NI flotantes)
   ➔ NI enteras básicas =
   NI sin copro = = N Iter x (NI enteras básicas + NI enteras emulación)
   ➔ NI enteras emulación =
   NI medio enteras / flotante = NI enteras emulación / NI flotantes =

4) MFLOPS nativos = Nº Operaciones flotantes x NIter / (Tej con copro x 10E6) =
   Nº Op. Normalizadas = (nº sum + nº res + nº mul + nº conv + nº comp) x 1 + nº div x 4 =
   MFLOPS normalizados = Nº Op. Normalizadas x NIter / (Tej con copro x 10E6) =

5) Nº Op. en funciones = Nº total de Op. flotantes - Nº basico de Op. flotantes =
   Nº invocaciones función = nº atan + nº sin + nº cos + nº sqrt + nº exp + nº log =
   Nº medio Op. / función = Nº Op. en funciones / Nº invocaciones función =
   ➔