Universidad de Carabobo

Computación

Prestaciones de la CPU

Métricas, Fórmulas Fundamentales y la Ecuación Clásica

Materia:
Arquitectura del Computador

Elaborado por:
Alessandro Ocanto
C.I: 31.467.550

14 de diciembre de 2025

1. Introducción

La evaluación de las prestaciones (performance) es un pilar fundamental en la arquitectura de computadores. Para diseñadores y usuarios, no basta con que un sistema funcione; debe hacerlo de manera eficiente.

Aunque existen diversas métricas, la medida base más objetiva es el **tiempo de ejecución**. Este documento formaliza las relaciones matemáticas entre el tiempo de respuesta, la frecuencia de reloj, los ciclos de ejecución y el conteo de instrucciones. Estas fórmulas permiten cuantificar cuánto más rápida es una máquina respecto a otra y entender cómo las decisiones de diseño (como el repertorio de instrucciones) impactan en el rendimiento final.

2. Prestaciones Relativas y Tiempo de Ejecución

Para maximizar las prestaciones, el objetivo principal es minimizar el tiempo de ejecución. Por definición, las prestaciones son inversamente proporcionales al tiempo de ejecución.

$$\text{Prestaciones}_X = \frac{1}{\text{Tiempo de ejecución}_X} \tag{1}$$

Cuando comparamos dos computadores, X e Y, decimos que "X es más rápido que Y"si las prestaciones de X son mayores. Para cuantificar esta diferencia, utilizamos la relación "X es n veces más rápida que Y":

$$\frac{\text{Prestaciones}_X}{\text{Prestaciones}_Y} = n \tag{2}$$

Sustituyendo la definición de prestaciones, obtenemos la relación fundamental para comparar tiempos:

$$\frac{\text{Tiempo de ejecución}_Y}{\text{Tiempo de ejecución}_X} = n \tag{3}$$

Esto implica que si el computador X es n veces más rápido, tardará n veces menos tiempo en ejecutar el mismo programa que el computador Y.

3. Prestaciones de la CPU y sus Factores

La métrica más básica para el hardware es el ciclo de reloj. Un programa requiere una cantidad determinada de ciclos para completarse. Podemos expresar el tiempo de ejecución de la CPU relacionando los ciclos con la duración de cada uno.

3.1. Fórmulas Básicas del Tiempo de CPU

$$\text{Tiempo CPU} = \text{Ciclos de reloj CPU} \times \text{Tiempo del ciclo de reloj} \tag{4}$$

Alternativamente, dado que la frecuencia de reloj (f) es la inversa del tiempo de ciclo (T), es decir $f = 1/T$, podemos expresar la fórmula como:

$$\text{Tiempo CPU} = \frac{\text{Ciclos de reloj CPU}}{\text{Frecuencia de reloj}} \tag{5}$$

3.2. Implicaciones de Diseño

Estas ecuaciones ponen de manifiesto que un diseñador de hardware tiene dos vías principales para mejorar las prestaciones:

- Reducir la longitud del ciclo de reloj (aumentar la frecuencia).
- Reducir el número total de ciclos requeridos para un programa.

Existe a menudo un compromiso de diseño: técnicas que reducen el número de ciclos pueden, inadvertidamente, aumentar la longitud del ciclo o viceversa.

4. La Ecuación Clásica de las Prestaciones

La mayoría de los computadores ejecutan instrucciones que pueden tardar diferentes cantidades de tiempo. Para capturar este comportamiento, introducimos dos conceptos nuevos:

1. **Número de Instrucciones (IC):** La cantidad total de instrucciones ejecutadas por el programa.
2. **CPI (Cycles Per Instruction):** El número promedio de ciclos de reloj necesarios para ejecutar una instrucción.

4.1. Ecuación General

Al expandir el total de ciclos como $IC \times CPI$, obtenemos la ecuación clásica:

$$\text{Tiempo de ejecución} = \text{Nº Instrucciones} \times \text{CPI} \times \text{Tiempo de ciclo} \tag{6}$$

O en función de la frecuencia:

$$\text{Tiempo de ejecución} = \frac{\text{Nº Instrucciones} \times \text{CPI}}{\text{Frecuencia de reloj}} \tag{7}$$

4.2. Cálculo del CPI Promedio

Cuando un programa contiene diferentes tipos de instrucciones (por ejemplo, aritméticas, de carga/almacenamiento, saltos), cada tipo puede tener un CPI distinto. El CPI global se calcula como un promedio ponderado:

$$\text{Ciclos CPU} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i) \tag{8}$$

Donde CPI_i es el ciclos por instrucción para la clase i, y C_i es el número de veces que se ejecuta esa instrucción.

Esta descomposición permite a los diseñadores de compiladores y hardware evaluar comparaciones complejas. Por ejemplo, una secuencia de código con más instrucciones podría ser más rápida si utiliza instrucciones con un CPI mucho menor.