Computer Organization

Lecture 2 - Performance

Reading: 1.4

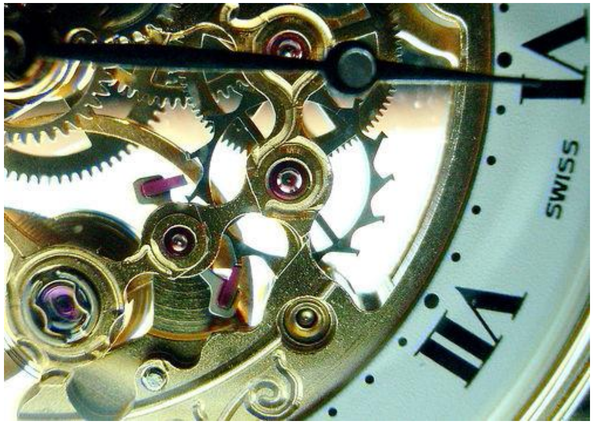


Photo credit:
Jurvetson via flickr

Roadmap for the term: major topics

- Computer Systems Overview
- **Performance**
- Assembly Language
- Instruction sets (and Software)
- Logic & Arithmetic
- Processor Implementation
- Memory Systems

Performance Outline

- **Motivation 3**
- **Defining Performance**
- **Common Performance Metrics**
- **Benchmarks**
- **Amdahl's Law**

Motivation

- Goal: Learn to "measure, summarize and report " performance of a computer system

- Why study performance?
  - To make intelligent decisions when choosing a system
  - To make intelligent decisions when designing a system
  - Understand impact of implementation decisions

- Challenges
  - How do we measure performance accurately?
  - How do we compare performance fairly?

What's a good measure of performance?

- **Response Time**
  - **How long does it take to complete a single task?**
- **Throughput**
  - **How many tasks are completed per unit time**
- **The measure we use depends on the application**

Execution Time vs. Throughput

- **Analogy: passenger airplanes (book Figure 1.13)**
  - **Concorde - fastest "response time" for an individual user**
  - **Boeing 747 - highest passenger throughput**

Airplane	Passenger Capacity	Cruising Range (miles)	Cruising Speed (mph)	Passenger Throughput (passengers x mph)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
Concorde	132	4000	1350	178,200
DC-8-50	146	8720	544	79,424

Performance Outline

- Motivation
- **Defining Performance**
- Common Performance Metrics
- Benchmarks
- Amdahl's Law

Relative Performance

- **For a given program on machine X:**

$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

- **Comparing performance of machines:**

$$\text{Performance}_X > \text{Performance}_Y \text{ if}$$
$$\text{Execution Time}_X < \text{Execution Time}_Y$$

Comparing Performance

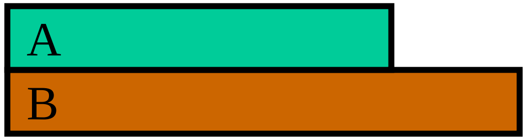
- **We say "X is *n* times faster than Y" if:**

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n$$

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

Example - Performance

Machine	Execution Time
A	15 seconds
B	20 seconds



- **Which machine is faster?**

- **By how much?**

Performance Outline

- Motivation
- Defining Performance
- **Common Performance Metrics**
- Benchmarks
- Amdahl's Law

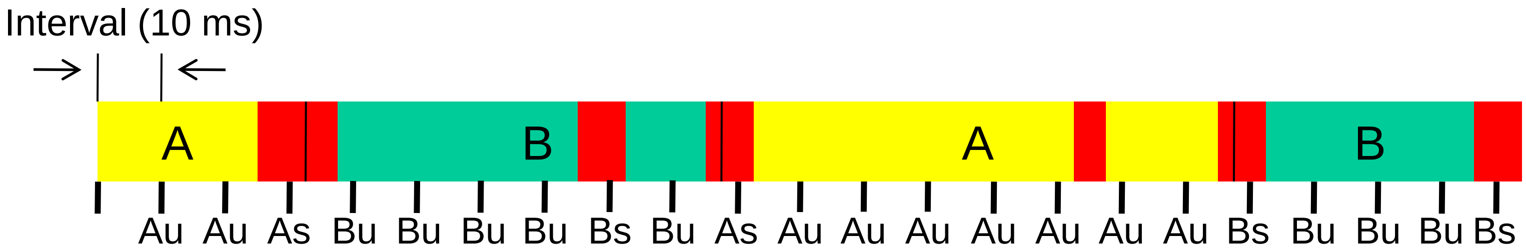
Elapsed Time / Execution Time

- Elapsed Time ("Wall Clock" Time)
  - How long does it take the program to complete?
  - System Performance – based on elapsed time

- Execution Time (CPU Time)
  - How much time the CPU spent executing the program
    - User time – time CPU spends executing program instructions
    - System time – time CPU spends in OS on behalf of program
  - CPU Performance – based on CPU time

Measuring CPU Time with the OS

- **OS runs multiple programs**
  - **Timer interrupts programs at fixed intervals (e.g. 10ms)**
  - **OS decides whether to switch which program runs**
  - **Interval counter – counts intervals when program is running in user mode (u) and system mode (s)**
- **Issues**
  - **Inaccuracy – interval counter can miss changes (cancels out for long programs)**



Program A:

Program B:

Measuring CPU Time with Unix/Linux

- Interval-based timing in the "real world"
- Linux/Unix `time` command

```
% time A
        ...
90.7u 12.9s 2:39 65%
```

User Time (sec)

System Time (sec)

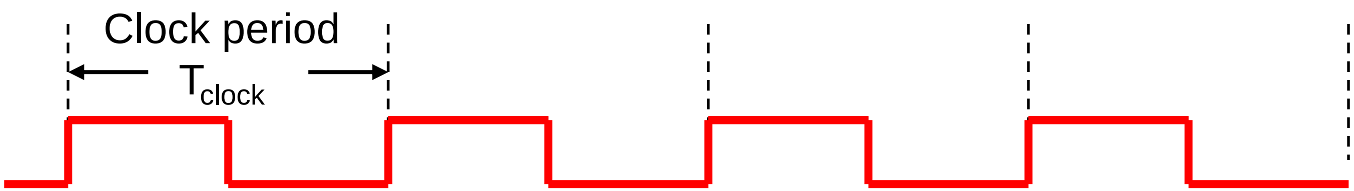
Wall-clock Time

CPU Utilization = (UT+ST) / WCT

103.6/159 = 65%

Clocks and Performance

- **Clock signal - controls sequential circuit operation**



Clock frequency $f_{clock} = 1/T_{clock}$

초당 클락이
몇 번 바뀌는가?

Clocks and Performance - CPU Time

- **How do we relate clock to CPU Time?**

$$f_{clock} = 4\ \text{GHz}$$

$$\text{CPU time} = \frac{\text{clock cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{clock cycle}}$$

- $1\text{GHz} = 10^{9}\text{Hz}$
- $1\text{ps} = 10^{-12}\text{s}$
- $1\text{ns} = 10^{-9}\text{s}$

Measuring Elapsed Time with Counters

- **Modern microprocessors include cycle counters**

  Ex) Intel Pentium & Later Processors
  - **64-bit Time Stamp Counter (TSC)**
  - **Counts clock cycles since reset**
  - **`rdtsc` instruction – moves TSC to `{edx,eax}` registers**
  - **Measure elapsed time by**
    - ✓ **Reading TSC at start**
    - ✓ **Reading TSC at end**
    - ✓ **Subtract & multiply by clock period**

Get the number of clocks during program execution

- **Issues**
  - **Context changes**
  - **Impact of caching**

Example - Clock Cycles

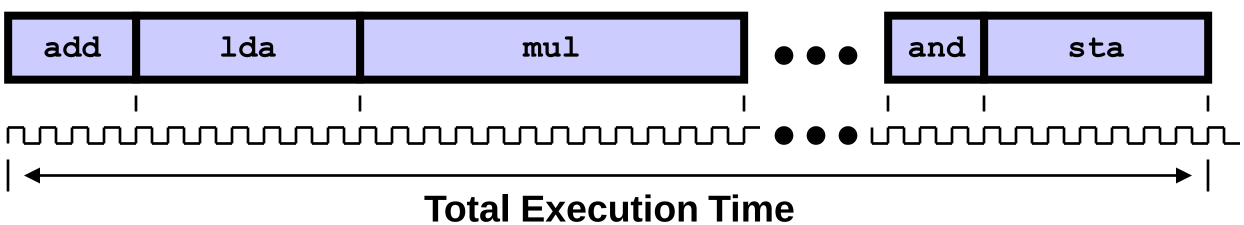
Machine	Execution Time	Clock Freq.
A	15 seconds	3 GHz
B	20 seconds	2.5 GHz

- **How many clock cycles does A execute?**

- **How many clock cycles does B execute?**

Clock Cycles per Instruction (CPI)

- Consider the 68HC11 …
  - ADDA - 3 cycles (IMM) -> 5 cycles (IND, Y)
  - MUL - 10 cycles
  - IDIV - 41 cycles
- More complex processors have other issues…
  - Pipelining - parallel execution (CPI=1!), but stalls occur
  - Memory system: cache misses, page faults, etc.
- How can we combine these into an overall metric?



Clock Cycles per Instruction (CPI)

- **Average number of clock cycles per instruction**
- **Measured for an entire program**

$$\text{CPU Clock Cycles} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Average Clock Cycles}}{\text{Instruction}}$$

$$\text{CPI} = \frac{\text{Average Clock Cycles}}{\text{Instruction}} = \frac{\text{CPU Clock Cycles/Program}}{\text{Instructions/Program}}$$

Example - CPI

Machine	Clock Cycles	Instructions
A	1.35×10^{10}	2.6×10^{9}
B	1.2×10^{10}	3.0×10^{9}

- **What is the CPI of A?**

- **What is the CPI of B?**

CPI Example

- **Computer A: Cycle Time = 250ps, CPI = 2.0**
- **Computer B: Cycle Time = 500ps, CPI = 1.2 (Same ISA)**
- **Which is faster, and by how much?**

$$\text{CPU Time}_A = \underline{\text{Instruction Count}} \times \text{CPI}_A \times \text{Cycle Time}_A$$

$$= \text{IC} \times 2.0 \times 250\text{ps} = \text{IC} \times 500\text{ps}$$

A is faster…

$$\text{CPU Time}_B = \underline{\text{Instruction Count}} \times \text{CPI}_B \times \text{Cycle Time}_B$$

$$= \text{IC} \times 1.2 \times 500\text{ps} = \text{IC} \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{\text{IC} \times 600\text{ps}}{\text{IC} \times 500\text{ps}} = 1.2$$

…by this much

The Performance Equation

- **The "Iron Law" of Performance**

$$\boxed{\text{CPU time}} = \frac{\#\,\text{instructions}}{\text{program}} \times \frac{\text{clock cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{clock cycle}}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

CPI in More Detail

Class 1: CPI = 3

10 Ins

Class 2: CPI = 5

30 Ins

- **If *n* different instruction classes take different numbers of cycles per instruction (CPI)**

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

CC = 10 x 3 + 30 x 5 = 180
CPI = 180/40 = 4.5

- **Weighted average CPI**

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

CPI = 3 x 10/40 + 5 x 30/40 = 180/40 = 4.5

Another CPI Example

- Consider <u>two compiled code sequences</u> using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	20	10	20
IC in sequence 2	40	10	10

- Which sequence executes more instructions?
- Which sequence is faster?
- What is the CPI for each sequence?

CPI Example

- **Solution: calculate total cycles for each sequence, then CPI**

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instrunction Count}_i)$$

Class	**A**	**B**	**C**
CPI for class	**1**	**2**	**3**
IC in sequence 1	**20**	**10**	**20**
IC in sequence 2	**40**	**10**	**10**

- **Sequence 1: IC = 50**
  - **Clock Cycles**
 = 20 × 1 + 10 × 2 + 20 × 3 = 100
  - **Avg. CPI = 100/50 = 2.0**
- **Sequence 2: IC = 60**
  - **Clock Cycles**
 = 40 × 1 + 10 × 2 + 10 × 3 = 90
  - **Avg. CPI = 90/60 = 1.5**

Performance Summary

- The "BIG PICTURE"

$$\text{CPU time} = \frac{\#\,\text{instructions}}{\text{program}} \times \frac{\text{clock cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{clock cycle}}$$

- **Performance depends on**
  - **Algorithm: affects Instruction Count (IC), possibly CPI**
  - **Programming language: affects IC, CPI**
  - **Compiler: affects IC, CPI**
  - **Instruction set architecture: affects IC, CPI, T_{clock}**
  - **Implementation Technology and Design: affects CPI, T_{clock}**

Clock Cycles and Performance - Example

- **Program runs on Computer A:**
  - **CPU Time: 10 seconds**
  - **Clock (rate): 400MHz = 400×10^6 cycles/sec**

- **Computer B can run clock faster**
  - **But, requires 1.2 X clock cycles to perform same task**
  - **Desired CPU Time: 6 Seconds**

- **What should the clock frequency of Computer B be to reach this target?**
- **Key to approach: Performance equation**

$$\text{CPU time} = \frac{\text{clock cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{clock cycle}} = \frac{\text{clock cycles}}{\text{clock frequency}}$$

Clock Cycles and Performance - Example (cont’d)

- **First step: find clock cycles executed by Computer A**

- **Second step: find clock cycles executed by Computer B**

Clock Cycles and Performance - Example (cont’d)

- **Third step: given clock cycles and CPU time, solve for clock rate of Computer B**

Summary

- Response time vs. Throughput
- Elapsed time vs. execution time (CPU time)
- How does OS measure CPU time?
- What is the problem if interrupt interval is too short in measuring CPU time?
- Why is it difficult to measure the execution time of a program?
- Why do we use the number of clock cycles to measure the elapsed time?
- Why do we use CPI to measure CPU time?