CS2100 - L15 - Performance

Week 9

15.1 - Key terms

- Performance & response time
- Execution time
- Average CPI

15.2 - Factors affecting performance

- Compiler
- Hardware implementation

15.3 - Amdahl's law

15.1 - Key terms
Performance & response time

Performance: Definitions

$$Performance = \frac{1}{ResponseTime}$$

Performance	Response time
• "***units of things-per-second***"	• "***number of seconds***"
• Bigger is better	• Smaller is better

Speedup ***n***, between ***x*** and ***y*** is

$$Speedup = \frac{Performance_x}{Performance_y} = \frac{ResponseTime_y}{ResponseTime_x}$$

Execution time

- We define execution time to focus on **User CPU Time:**
  - Time spent executing the lines of code in the program
  - Excluding time spent on input / output, task switching etc

$$CPU\ Time = \frac{Seconds}{Program}$$

$$CPU\ Time = \frac{Cycles}{Program} \times \frac{Seconds}{Cycle}$$

$$CPU\ Time = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

Average CPI

$$CPU\ Time = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

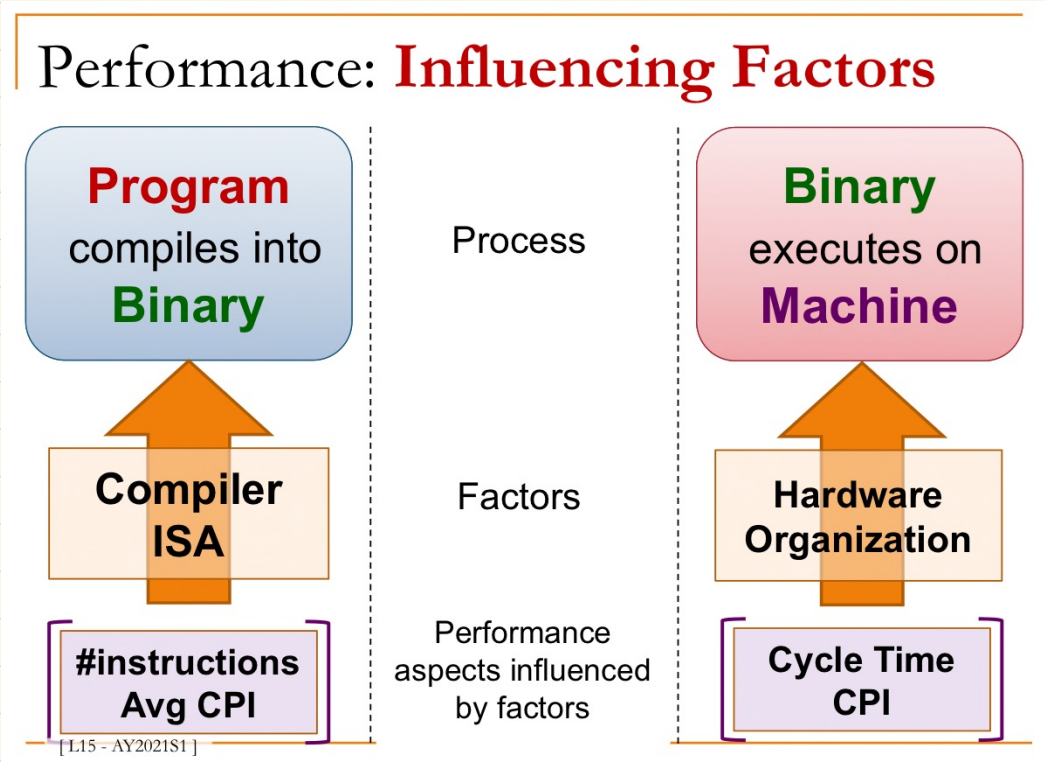
- Average Cycle Per Instruction (CPI)

CPI = (CPU time × Clock rate) / Instruction count
= Clock cycles / Instruction count

$$\text{CPI} = \sum_{k=1}^{n} \text{CPI}_k \times F_k \quad \text{where} \quad F_k = \frac{I_k}{\text{Instruction count}}$$

I_k = Instruction frequency

15.2 - Factors affecting performance



Factors: program → binary (Compiler)

- **Compiler:**
  - Different compiler may generate different binary
    - e.g. gnu vs intel c/c++ compiler
  - Different optimization may generate different binary
    - e.g. different optimization level in ***gnu c compiler***
- **Instruction Set Architecture:**
  - The same high level statement can be translated differently depending on the ISA
    - e.g. same C program under ***Intel*** machine vs ***Sunfire*** server

Factors: execution of binary (Hardware implementation)

- **Machine**
  - More accurately the hardware implementation
  - Determine **cycle time** and **cycle per instruction**
- **Cycle Time**:
  - Different clock frequency (e.g. 2ghz vs 3.6ghz)
- **Cycle Per Instruction:**
  - Design of internal mechanism (e.g. specific accelerator to improve floating point performance)

15.3 - Amdahl's law

Summary: Key Concepts

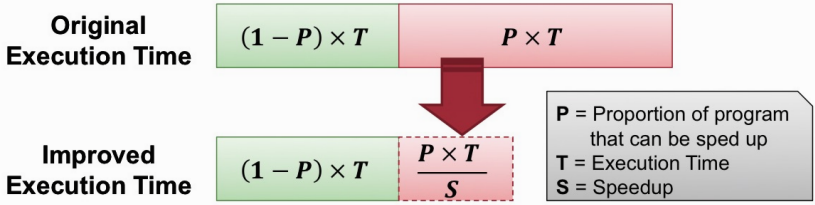
- Performance is specific to a particular program on a specific machine
  - Total execution time is a consistent summary of performance
- For a given architecture, performance increase comes from:
  - Increase in clock rate (without adverse CPI effects)
  - Improvement in processor organization that lowers CPI
  - Compiler enhancement that lowers CPI and/or instruction count

Pitfall:
Expecting improvement in one aspect of a machine's performance to affect the total performance.

Amdahl's Law: **Definition**

Amdahl's Law

Performance is limited to the non-speedup portion of the program

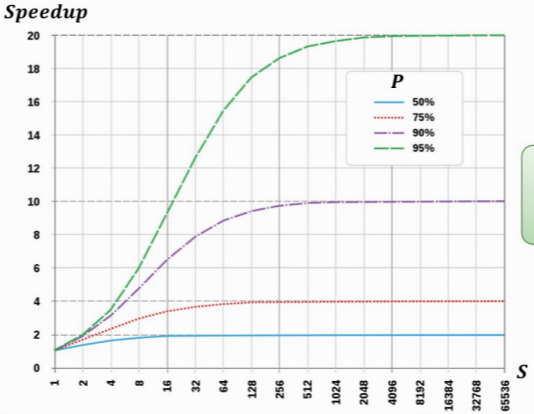


Amdahl's Law Corollary

Optimize the common case first!

Amdahl's Law: **Definition**

- The proportion of ***non-affected portion (1-P)*** effectively place an upper limit of overall speedup:



$$Speedup = \frac{1}{(1-P)+\frac{P}{S}}$$

Image modified from: https://en.wikipedia.org/wiki/Amdahl%27s_law