Lecture 3

In this lecture we discussed why the clock rate of a computer doesn't necessarily correspond to its speed. We also looked at ways we can compare / discuss the performances of multiple computers, culminating in benchmarking and practice problems on performance from the textbook.

Why does a clock rate not correspond directly to the speed of a computer?

- Computers can increase the clock rate at high loads (dynamic clock speed adjustments)
  - Only temporarily, though, cause the CPU can overheat
  - Base clock speed / base frequency is the default speed

- The existence of multicore CPUs (multiple CPUs / processors in one)
  - e.g. `8 * 2.3 GHz > 3 * 3.6 GHz`
  - Note: The system is *potentially* more powerful, but many programs only utilize 1 core, so realistically a program might run fatser on the latter computer.

- The same instruction set on different CPUs may be implemented different in hardware (architecture implementations)

  Example

  ```
  Comp A: 2 GHz, 1 cycles per instruction (CPI)

  Comp B: 3 GHz, 3 cycles per instruction
  ```

  Comp A executes 2 billion instructions per second, Comp B executes only 1 billion instructions per second. Comp A is faster despite having a slower clock rate (this could be due to architecture implementation differences)

- Other parts of the computer exist besides the CPU
  - RAM (a bottleneck)
  - If the RAM sucks, that significantly affects the operation of the computer system (constant memory stalls)

Takeaway: No single piece of information can define / describe the computer, all aspects / components affect performance

How can we compare?

Benchmarks

- consists of test program(s) to execute
- compare benchmark performance between multiple computers
- downsides: some computers have parts specialized for certain benchmarks, so you may not get an accurate result (solution: use many varied benchmarks)

Some formulas and definitions

$$CYCLES = TIME\ (seconds) \times FREQUENCY\ (Hz)$$

$$PERFORMANCE = \frac{1}{EXECUTION\ TIME}$$

- Execution time is the pure time spent executing a program (assuming no interruptions)"
- You can measure this time in either seconds or cycles"

$$EXECUTION\ TIME = \#\ INSTRUCTIONS \times CPI \times CLOCK\ CYCLE\ TIME = \frac{\#\ INSTRUCTIONS \times CPI}{FREQUENCY}$$

$$CLOCK\ CYCLE\ TIME = \frac{1}{FREQUENCY}$$

CPI - Cycles per Instruction

- The average number of clock cycles per instruction for a program
- Each instruction takes integer number of cycles, but on average it can be a floating point
- The CPI can be < 1 (e.g. 0.5) if multiple cores of a CPU execute multiple instructions in one cycle

Dynamic instruction count - number of instructions a program has, dynamic meaning that loops and etc are taken into account

Textbook Performance Problems

Example 1

Page 34

Our favorite program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?

	Computer A	Computer B
TIME	10 seconds	6 seconds
FREQ	2 GHz	?
CYCLES	?	?

STEPS

1 Count cycles for A (**2 GHz x 10 seconds = 20 billion cycles**)

2 Using this info, get the number of cycles for B. We're told that Computer B will require 1.2x more clock cycles than computer A. **20 billion x 1.2 = 24 billion cycles**.

3 Plug in to **CYCLES = TIME x FREQ**

24 billion cycles = 6 seconds x ? GHz = 6 seconds x 4 GHz (giga = billion)

4 The designer should aim for a frequency of 4 GHz

Example 2

Processor	A	B
CPI	2	2.5

Program X - dynamic instruction count of 1 billion

Program X runs on processor B, how many cycles does it take to execute?

$$1\ billion\ instructions \times 2.5\ \frac{cycles}{instructions} = 2.5\ billion\ cycles$$

Example 3

Instruction Type	A	B
Cycles Per Instruction	2	3

Program X:

- 65% instructions of type A
- 35% instructions of type B

Find the CPI:

(0.65 * 2) + (0.35 * 3) = 1.3 + 1.05 = 2.35

Alternatively, imagine there are 100 instructions. Then, 65 are of type A, and 35 are of type B. Type A instructions will take 65 * 2, or 130 cycles in total, and type B instructions will take 35 * 3, or 105 cycles in total. So, for a 100 instructions there will be 130 + 105, or 235 cycles used. Per each instruction, simply divide total cycles by the amount of instructions -- 235 / 100 = 2.35.