0x10 - CPU Performance Metrics

ENGR 3410: Computer Architecture

Jon Tse

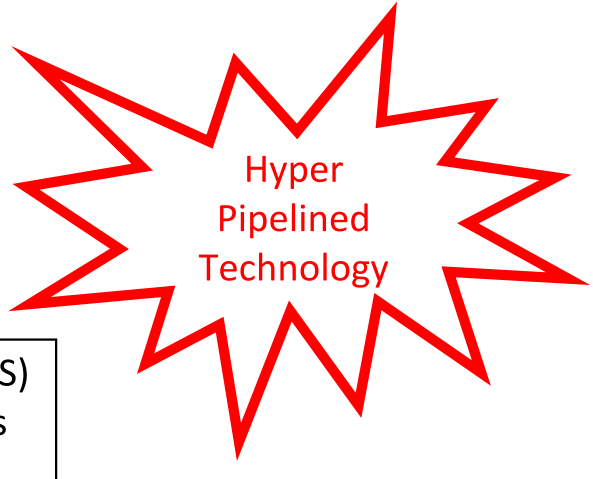
Fall 2020

Housekeeping

- Any questions on Lab 4?
- Final Project teams due tomorrow!

Computer “Performance”

- MIPS (Million Instructions Per Second) vs. MHz (Million Cycles Per Second)
- Throughput (jobs/seconds) vs. Latency (time to complete a job)
- Measuring, Metrics, Evaluation – what is “best”?



Apple A12 processor (iPhone XS)
20% faster than A11, 40% less power

Performance Example: Planes

Airplane	Passenger Capacity	Cruising Range (miles)	Cruising Speed (mph)	Passenger Throughput (passengermile/hour)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
Concorde	132	4000	1350	178,200
Douglas DC8	146	8720	544	79,424

- Which is the "best" plane?
  - Which gets one passenger to the destination first?
  - Which moves the most passengers?
  - Which goes the furthest?

- Which is the "speediest" plane (between Seattle and NY)?
  - Latency: how fast is one person moved?
  - Throughput: number of people per time moved?

Computer Performance

- Primary goal: execution time (time from program start to program completion)

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

- To compare machines, we say "X is n times faster than Y"

$$n = \frac{Performance_x}{Performance_y} = \frac{ExecutionTime_y}{ExecutionTime_x}$$

- Example: Machine *Orange* and *Grape* run a program
  Orange takes 5 seconds, Grape takes 10 seconds

- Orange is 2 times faster than Grape

Execution Time

- Elapsed Time
  - counts everything *(disk and memory accesses, I/O , etc.)*
  - a useful number, but often not good for comparison purposes
- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into system time, and user time

- Example: Unix “time” command

```
fpga.olin.edu> time javac CircuitViewer.java
3.370u 0.570s 0:12.44 31.6%
```

- Our focus: user CPU time
  - time spent executing the lines of code that are "in" our program

CPU Time

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} * \text{Clock period}$$

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} * \frac{1}{\text{Clock rate}}$$

- Application example:

  A program takes 10 seconds on computer *Orange*, with a 400MHz clock. Our design team is developing a machine *Grape* with a much higher clock rate, but it will require 1.2 times as many clock cycles. If we want to be able to run the program in 6 seconds, how fast must the clock rate be?

$$\frac{10}{400\text{MHz}}$$

CPI

- How do the # of instructions in a program relate to the execution time?

CPU clock cycles for a program = Instructions for a program * Average Clock Cycles per Instruction **(CPI)**

CPU execution time for a program = Instructions for a program * CPI * $\frac{1}{\text{Clock rate}}$

CPI Example

- Suppose we have two implementations of the same instruction set (ISA).
- For some program

Machine A:
- clock cycle time of 10 ns
- CPI of 2.0

Machine B:
- clock cycle time of 20 ns
- CPI of 1.2

- What machine is faster for this program, and by how much?

Computing CPI

$$CPI = \sum_{types} \left(Cycles_{type} * Frequency_{type} \right)$$

- Different types of instructions can take very different amounts of cycles
- Memory accesses, integer math, floating point, control flow

fixed

program based

Instruction Type	Type Cycles	Type Frequency	Cycles * Freq
ALU	1	50%	
Load	5	20%	
Store	3	10%	
Branch	2	20%	
		CPI:	

Computing CPI

$$CPI = \sum_{types} \left(Cycles_{type} * Frequency_{type} \right)$$

- Different types of instructions can take very different amounts of cycles
- Memory accesses, integer math, floating point, control flow

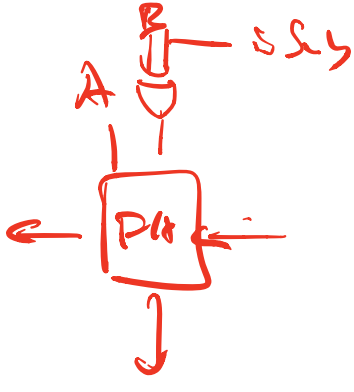
Instruction Type	Type Cycles	Type Frequency	Cycles * Freq
ALU	1	50%	.5
Load	5	20%	1
Store	3	10%	.3
Branch	2	20%	.4
		CPI:	2.2

CPI & Processor Tradeoffs

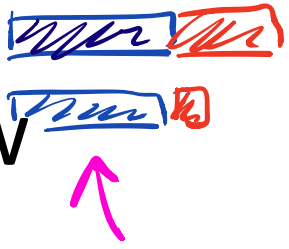
Instruction Type	Type Cycles	Type Frequency
ALU	1	50%
Load	5	20%
Store	3	10%
Branch	2	20%

How much faster would the machine be if:

1. A data cache reduced the average load time to 2 cycles?
2. Branch prediction shaved a cycle off the branch time?
3. Two ALU instructions could be executed at once?



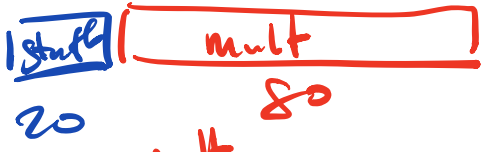
Warning: Amdahl's Law



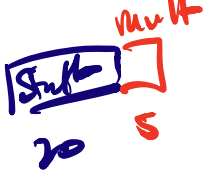
- The impact of a performance improvement is limited by what is NOT improved:

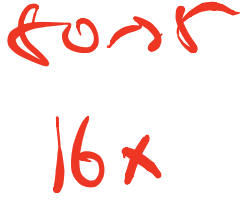
$$\text{Execution time after improvement} = \text{Execution time of unaffected} + \text{Execution time affected} * \frac{1}{\text{Amount of improvement}}$$

- Example: Assume a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to speed up multiply to make the program run 4 times faster?



- 5 times faster?





Warning 2: MIPs, MHz ≠ Performance

- Higher MHz (clock rate) doesn't always mean better CPU
  Orange computer: 1000 MHz, CPI: 2.5, 1 billion instruction program

  Grape computer: 500MHz, CPI: 1.1, 1 billion instruction program

- Higher MIPs (million instructions per second) doesn't always mean better CPU
  1 MHz machine, with two different compilers/instruction sets
  Compiler A on program X: 10M ALU, 1M Load
  Compiler B on program X: 5M ALU, 1.5M Load

  Execution Time: A _____ B _____

  MIPS: A _____ B _____

Instruction Type	Type Cycles
ALU	1
Load	5
Store	3
Branch	2

Warning 2: MIPs, MHz ≠ Performance

- Higher MHz (clock rate) doesn't always mean better CPU
  Orange computer: 1000 MHz, CPI: 2.5, 1 billion instruction program -> 2.5 s

  Grape computer: 500MHz, CPI: 1.1, 1 billion instruction program -> 2.2 s

- Higher MIPs (million instructions per second) doesn't always mean better CPU
  1 MHz machine, with two different compilers/instruction sets
  Compiler A on program X: 10M ALU, 1M Load
  Compiler B on program X: 5M ALU, 1.5M Load

  Execution Time: A 15s B 12.5s

  MIPS: A 1MIPs B 1MIPs

Instruction Type	Type Cycles
ALU	1
Load	5
Store	3
Branch	2

Processor Performance Summary

- Machine performance:

$$\text{CPU execution time for a program} = \text{Instructions for a program} * \text{CPI} * \frac{1}{\text{Clock rate}}$$

- Better performance:
  - ↓ number of instructions to implement computations
  - ↓ CPI
  - ↑ Clock rate

- Improving performance must balance each constraint
  Example: RISC vs. CISC

Common Comparison Metrics

- SPEC CPU
  - Integer and floating point benchmark sets
  - Representative program mix – gcc, zip, h264
- Whetstone MWIPS
  - Specific program profile stressing Floating Point
  - Minimal memory stress
  - AM386 developed 5.68MWIPS @ 40MHz (1991)
  - I7 930 develops 2496MWIPS @ 2800MHz

Common Comparison Metrics

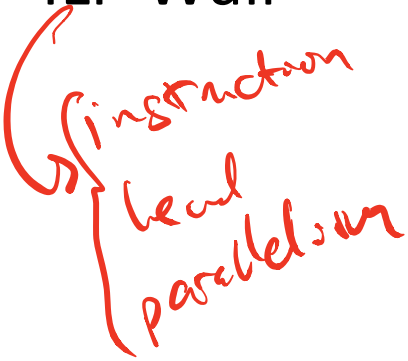
- Dhrystone
  - Specific program stressing integer and string ops
- LINPACK
  - Solve linear equation $Ax=b$
  - Common calculation in engineering

Performance

- What is the task?
- What is the hardware/ISA?
- What is the input set?

The Walls

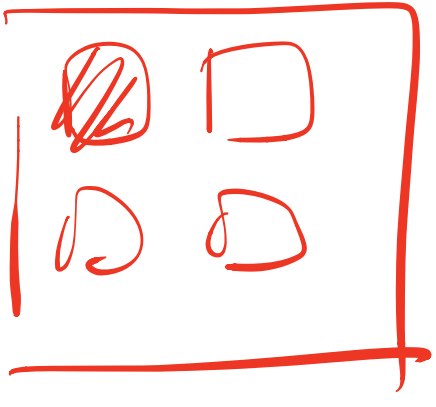
- Power Wall
- Memory Wall
- ILP Wall



Power Wall

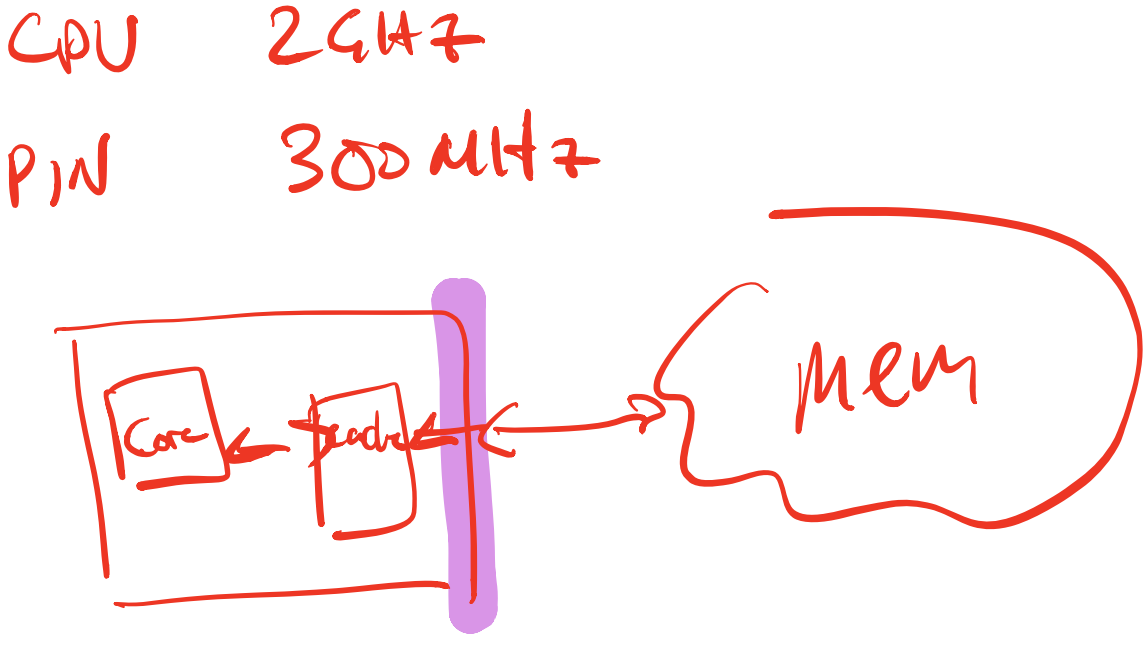
Power used to be free…

$$P \sim \delta \frac{1}{2} C V^2$$



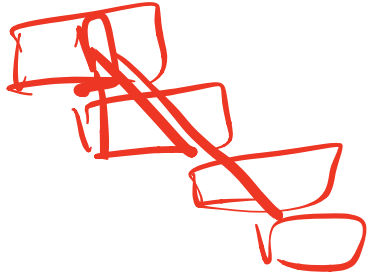
Memory Wall

Feeding multiple compute engines is hard...



ILP Wall

- Pipelining
- Speculative Execution
- Register Renaming
- Out of Order Execution



add $t0, $t1, $t2
sub $s0, $t1, $s2

It Depends

It’s all related, it all depends…