# CMPE 110: Computer Architecture

## Week 2

## Performance

Jishen Zhao (http://users.soe.ucsc.edu/~jzhao/)

[Adapted in part from Jose Renau, Mary Jane Irwin, Joe Devietti, Onur Mutlu, and others]

## Reminder

- The class Google site is public
  - No need to request access permission

# Review: Performance Metrics

- Metrics
  - Latency and throughput
  - Speedup
  - Averaging
- CPU Performance
  - CPI
  - IPC




# Cycles per Instruction (CPI)

- **CPI**: Cycles/instruction
  - **IPC** = 1/CPI
    - Used more frequently than CPI
    - Favored because "bigger is better", but harder to compute with
  - Different instructions have different cycle costs
    - E.g., "add" typically takes 1 cycle, "divide" takes >10 cycles
  - Depends on relative instruction frequencies (what if idle)
- CPI example
  - A program executes equal: integer, floating point (FP), memory ops
  - Cycles per instruction type: integer = 1, memory = 2, FP = 3
  - What is the CPI? (1 cycle+2 cycles +3 cycles) / 3 instrs= 2
    Calculated in another way: 33% x 1 + 33% x 2 + 33% x 3 = 2

# CPI Example

- Assume a processor with instruction frequencies and costs
  - Integer ALU: 50%, 1 cycle
  - Load: 20%, 5 cycle
  - Store: 10%, 1 cycle
  - Branch: 20%, 2 cycle

- Which change would improve performance more?
  - A. "Branch prediction" to reduce branch cost to 1 cycle?
  - B. Faster data memory to reduce load cost to 3 cycles?
- Compute CPI
  - Base = 0.5*1 + 0.2*5 + 0.1*1 + 0.2*2 = 2 CPI
  - A = 0.5*1 + 0.2*5 + 0.1*1+ 0.2*1 = 1.8 CPI (1.11x or 11% faster)
  - B = 0.5*1 + 0.2*3 + 0.1*1 + 0.2*2 = 1.6 CPI (1.25x or 25% faster)
    - **B is the winner**



# Measuring CPI

- How are CPI and execution-time actually measured?
  - Execution time? stopwatch timer (Unix "time" command)
  - Cycles = execution time * clock frequency
  - How is instruction count measured? → truly executed insn
- Hardware event counters
  - Available in most processors today
  - One way to measure instruction count
- Cycle-level micro-architecture simulation
  - \+ Measure exactly what you want … and impact of potential fixes!
  - Method of choice for many micro-architects
- More useful is CPI breakdown ($CPI_{CPU}$, $CPI_{MEM}$, etc.)
  - So we know what performance problems are and what to fix

# Summary: Performance Metrics

- Performance metrics: Latency & throughput
- Comparing performance: Speedup
- Averaging performance:
  - Arithmetic mean
  - Harmonic mean
  - Geometric mean
- Measuring CPU performance: CPI (IPC)




# Today: Performance (cont.)

- Other CPU performance metrics
- Amdahl's law
- Benchmarking

# Other CPU performance metrics

- MIPS
  - Million instructions per second <- IPC
- FLOPS
  - The number of floating point instructions per second
  - GFLOPS
  - Will visit again when we discuss GPU architectures



# Example

| Instruction | Percentage | CPI_M1 | CPI_M2 |
|---|---|---|---|
| ALU | 50% | 1 | 2 |
| Branches | 15% | 2 | 1 |
| Loads | 20% | 2 | 1 |
| Stores | 15% | 1 | 1 |

- What is the average MIPS for M1 and M2?
  - Clock frequency is 1GHz →1 cycle is 1ns

# Answer

| Instruction | Percentage | CPI_M1 | CPI_M2 |
|---|---|---|---|
| ALU | 50% | 1 | 2 |
| Branches | 15% | 2 | 1 |
| Loads | 20% | 2 | 1 |
| Stores | 15% | 1 | 1 |

- What is the average MIPS for M1 and M2?
  - Clock frequency is 1GHz →1 cycle is 1ns
  - M1: CPI = 0.5*1+0.15*2+0.2*2+0.15*1 = 1.35
    - 1 million instructions take $1.35*10^6$ ns, i.e., $1.35 * 10^{-3}$ s, to run
    - MIPS = $1 / (1.35*10^{-3}) = 741$
  - M2: CPI = 0.5*2 + 0.15*1 + 0.2*1+ 0.15*1 = 1.5
    - MIPS = $1 / (1.5*10^{-3}) = 667$



# Example

| Instruction | Percentage | CPI_M1 | CPI_M2 |
|---|---|---|---|
| ALU | 50% | 1 | 2 |
| Branches | 15% | 2 | 1 |
| Loads | 20% | 2 | 1 |
| Stores | 15% | 1 | 1 |

- What is the peak MIPS for M1 and M2?
  - Clock frequency is 1GHz →1 cycle is 1ns

# Example

| Instruction | Percentage | CPI_M1 | CPI_M2 |
|---|---|---|---|
| ALU | 50% | 1 | 2 |
| Branches | 15% | 2 | 1 |
| Loads | 20% | 2 | 1 |
| Stores | 15% | 1 | 1 |

- What is the peak MIPS for M1 and M2?
  - Clock frequency is 1GHz →1 cycle is 1ns
  - Both M1 and M2:
    - $CPI_{peak}$ = 1
    - 1 instruction takes 1 cycle, i.e., 1ns = $10^{-9}$ s, to run
    - 1 million instructions take $10^{-3}$ s to run
    - MIPS = $(1*10^{-6})/(1*10^{-9})$ = 1000



# Summary: Performance Metrics

- Performance metrics
  - Latency & throughput
- Comparing performance
  - Speedup
- Averaging performance
  - Arithmetic mean
  - Harmonic mean
  - Geometric mean
- Measuring CPU performance
  - CPI (IPC), MIPS, FLOPS, etc.

# Amdahl's Law



## Amdahl's Law

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

How much will an optimization improve performance?

*P* = proportion of running time affected by optimization

*S* = speedup

What if I speedup 25% of a program's execution by 2 times?

1.14x speedup

What if I speedup 25% of a program's execution by ∞?

1.33x speedup

# Amdahl's Law for Parallelization

$$\frac{1}{(1-P)+\frac{P}{N}}$$

How much will parallelization improve performance?

$P$ = proportion of parallel code

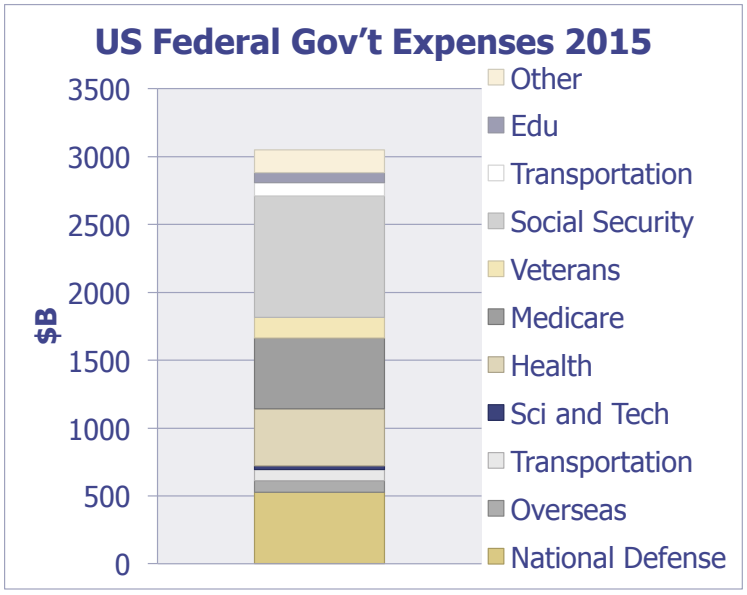
$N$ = threads

"Max speedup", so N = ∞

- What is the max speedup for a program that's 10% serial? 10x speedup
- What about 1% serial? 100x speedup



# Amdahl's Law for the US Budget




https://en.wikipedia.org/wiki/2015_United_States_federal_budget

Scrapping Dept of Transportation ($81B) cuts budget by 2.6%

# Example: Forget about the formula for a while

- Assume a program has the following instruction type breakdown:
  - 40% memory, latency is 4 cycles
  - 50% adds, latency is 2 cycles
  - 10% multiplies, latency is 16 cycles
- If you could pick one type of instructions to make twice as fast (half of latency) in the next-generation of this processor, which instruction type would you pick?



# Answer

- Assume a program has the following instruction type breakdown:
  - 40% memory, latency is 4 cycles
  - 50% adds, latency is 2 cycles
  - 10% multiplies, latency is 16 cycles
- Pick one type of instructions to make twice as fast (half of latency)
  - Base CPI: 0.4*4 + 0.5*2 + 0.1*16 = 4.2
  - Make memory instructions 2x faster:
    - $CPI_1$ = 0.4*2 + 0.5*2 + 0.1*16 = 3.4
  - Make adds 2x faster: $CPI_2$ = (0.4*4 + 0.5*1 + 0.1*16 = 3.7
  - Make multiplies 2x faster: $CPI_3$ = (0.4*4 + 0.5*2 + 0.1*8 = 3.4

## What we learned

- Amdahl's Law:

How much does
an optimization improve performance?

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



# Benchmarking

# Processor Performance and Workloads

- Q: what does performance of a chip mean?
- A: nothing, there must be some associated workload
  - **Workload**: set of tasks someone (you) cares about

- **Benchmarks**: standard workloads
  - Used to compare performance across machines
  - Either are or highly representative of actual programs people run

- **Micro-benchmarks**: non-standard workloads
  - Tiny programs used to evaluate certain aspects of performance
  - Not representative of complex behaviors of real applications
  - Examples: binary tree search, towers-of-hanoi, 8-queens, etc.