Technology Trends (Annual)

- Processor
  - Density & cost: ~+30%, Speed: ~+20%
- Memory
  - Density & cost: ~+60%, Speed: ~+4%
- Disk
  - Density: ~+60%, Speed: ~+10%

- Changing quickly and with respect to each other!!
  - Things that used to be bad ideas are now good
  - Things that used to be good ideas are now bad
  - Constant re-evaluation and re-design

Moore’s Law

- [material in class]

Processor Performance in Perspective

- What if cars improved at the same rate as processors did?
  - 1985
    - Processors: 10 MHz, 1 MIPS
    - Cars: 60 MPH, 8 MPG
  - 2000
    - Processors: 1 GHz, 500 MIPS
    - Cars: 6000 MPH, 4000 MPG

- What if cars crashed as often as computers did?
Empirical Evaluation

- Metrics
  - Performance
  - Cost
  - Power consumption
    - Important these days, but we won’t talk about it much
- Basis for
  - Design decisions
  - Purchasing decisions

Performance

- Two definitions
  - Latency (execution time): time to finish a fixed task
  - Throughput (bandwidth): # of tasks that finish in fixed time
  - Very different: throughput can exploit task parallelism, latency cannot
  - Often contradictory
  - Choose definition that matches goals (most frequently throughput)
- Example: move people from A to B, distance=10 miles
  - Car: capacity = 5, speed = 60MPH
  - Bus: capacity = 60, speed = 20MPH
  - One-way latency: car = 10 min, bus = 30 min
  - Throughput: car = 15 PPH, bus = 60 PPH (counting round-trip)

Performance Improvement

- Processor A is X times faster than processor B if
  - Latency(Program P,A) = Latency(Program P,B) / X
  - Throughput(Program P,A) = Throughput(Program P,B) * X
- Processor A is X% faster than processor B if
  - Latency(Program P,A) = Latency(Program P,B) / (1+X/100)
  - Throughput(Program P,A) = Throughput(Program P,B) * (1+X/100)
- Car/bus example
  - Latency? Car is 3 times (and 200%) faster than bus
  - Throughput? Bus is 4 times (and 300%) faster than car

What Is ‘P’ in Latency(P,A)?

- Program
  - Latency(A) makes no sense, processor executes some program
  - But which one?
- Actual target workload?
  - Accurate
    - Not portable, overly specific, hard to pinpoint problems
- Some representative benchmark program(s)?
  - Portable, pretty accurate
    - May be hard to pinpoint problems, may not be exactly what you run
- Some small kernel benchmarks (micro-benchmarks)
  - Portable, easy to run, easy to pinpoint problems
  - Not representative of complex behaviors of real programs
SPEC Benchmarks

- SPEC (Standard Performance Evaluation Corporation)
  - http://www.spec.org/
  - Consortium of companies that collects, standardizes, and distributes benchmark programs
  - Post SPECmark results for different processors
  - Benchmark suites for CPU, I/O, Web, Mail, etc.
  - Suites updated every few years
    - Prevents companies from targeting only benchmarks

- SPEC CPU 2000
  - 12 "integer": gzip, gcc, perl, crafty (chess), vortex (DB), etc.
  - 14 "floating point": mesa (openGL), equake, facerec, etc.

It Goes Without Saying

- [material in class]

CPU Performance Equation

- Latency(P,A) = seconds / program =
  - (instructions / program) * (cycles / instruction) * (seconds / cycle)
- Why this division?
  - Separates responsibility
  - Instructions / program: dynamic instruction count
    - Function of program, compiler, ISA
  - Cycles / instruction: CPI
    - Function of program, compiler, ISA, microarchitecture
  - Seconds / cycle: clock period
    - Function of microarchitecture, technology parameters
- For low latency (better performance), minimize all three
  - Hard to do, typically pull design in opposite directions

CPI

- [material in class]
Another 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. A better data cache to reduce load cost to 3 cycles?

- Compute CPI
  - Base = 0.5*1 + 0.2*5 + 0.1*1 + 0.2*2 = 2
  - A = 0.5*1 + 0.2*5 + 0.1*1 + 0.2*1 = 1.8
  - B = 0.5*1 + 0.2*3 + 0.1*1 + 0.2*2 = 1.6 (winner)

IPC, MHz, and MIPS

- Ignoring (instructions/program) for now
  - Once ISA is fixed (which it is), not much we can do about this part

- CPU performance equation becomes
  - seconds / instruction = (cycles / instruction) * (seconds / cycle)
  - This is a latency measure, if we care about throughput ...

- Instructions / second = (instructions / cycle) * (cycles / second)

- MIPS (millions of instructions per second)
  - Instructions / second * 10^6
  - Cycles / second: clock speed (in MHz, GHz)

- Example: CPI = 2, clock = 500MHz, what is MIPS?
  - MIPS = 0.5 * 500 MHz * 10^-6 = 250 MIPS
  - Notice M = 10^6

Performance Example

- Which is faster, processor A or B?
  - Have the same ISA, same compiler
  - Processor A has CPI = 2 and clock speed = 500 MHz
  - Processor B has CPI = 1 and clock speed = 300 MHz

- Processor A: 0.5 * 500 MHz * 10^-6 = 250 MIPS
  - Processor B: 1 * 300 MHz * 10^-6 = 300 MIPS

- But which do you think people would buy?

- What if A and B had different ISAs or compilers?
  - Comparing apples and oranges? Apples and chickens??

Performance Rules of Thumb

- Make common case fast
  - Sometimes called Amdahl's Law
  - Corollary: don't optimize 1% to the detriment of other 99%

- Build a balanced system
  - Don't over-engineer capabilities that cannot be utilized

- Design for actual, not peak, performance
  - For actual performance X, machine capability must be > X