# Lecture 2: Performance

# Performance Metrics

- Possible measures:
  - response time (Latency) – time elapsed between start and end of a program
  - throughput – amount of work done in a fixed time

- The two measures are usually linked
  - A faster processor will improve both
  - More processors will likely only improve throughput
  - Some policies will improve throughput and worsen response time

- What influences performance?



# Execution Time

Consider a system “X” executing a fixed workload “W”

$Performance_X = 1 / Execution\ time_X$

Execution time = response time
- Note that this includes time to execute the workload as well as time spent by the operating system co-ordinating various events.



# Speedup and Improvement Representation

- System X executes a program in 10 seconds, system Y executes the same program in 15 seconds
- System X is 1.5 times faster than system Y
- The speedup of system X over system Y is 1.5 (the ratio)
- The performance improvement of X over Y is
  1.5 -1 = 0.5 = 50%
- The execution time reduction for the program, compared to Y is (15-10) / 15 = 33%
  The execution time increase, compared to X is
  (15-10) / 10 = 50%



# Performance Equation - I

CPU execution time = CPU clock cycles
(required for executing a program) x Clock cycle time
Clock cycle time = 1 / Clock speed

If a processor has a frequency of 3 GHz, the clock ticks 3 billion times in a second – as we'll soon see, with each clock tick, one or more/less instructions may complete

If a program runs for 10 seconds on a 3 GHz processor, how many clock cycles did it run for?

If a program runs for 2 billion clock cycles on a 1.5 GHz processor, what is the execution time in seconds?



# Performance Equation - II

The number of clock cycles for a program
#CPU clock cycles = #instructions in the program x
avg clock cycles per instruction (CPI) given for a CPU

Substituting in previous equation,

Execution time = clock cycle time x # instructions x avg CPI

If a 2 GHz processor graduates an instruction every third cycle, how many instructions are there in a program that runs for 10 seconds?



# Factors Influencing Performance

Execution time = clock cycle time x number of instrs x avg CPI

- Clock cycle time: manufacturing process (how fast is each transistor), how much work gets done in each pipeline stage (more on this later)
- Number of instrs: the quality of the compiler, the instruction set architecture
- CPI: the nature of each instruction and the quality of the architecture implementation



# Example (Quiz)

Execution time = clock cycle time x number of instrs x avg CPI

Which of the following two systems is better?

- A program is converted into 4 billion MIPS instructions by a compiler ; the MIPS processor is implemented such that each instruction completes in an average of 1.5 cycles and the clock speed is 1 GHz

- The same program is converted into 2 billion x86 instructions; the x86 processor is implemented such that each instruction completes in an average of 6 cycles and the clock speed is 1.5 GHz



# Benchmark Suites

- Each CPU vendor announces a SPEC rating for their system
  - a measure of execution time for a fixed collection of programs
  - is a function of a specific CPU, memory system, IO system, operating system, compiler
  - enables easy comparison of different systems

The key is coming up with a collection of relevant programs



# SPEC CPU

- SPEC: System Performance Evaluation Corporation**, an industry consortium that creates a collection of relevant programs**
- The 2006 version includes 12 integer and 17 floating-point applications
- The SPEC rating specifies how much faster a system is, compared to a baseline machine – a system with SPEC rating 600 is 1.5 times faster than a system with SPEC rating 400
- Note that this rating incorporates the behavior of all 29 programs – this may not necessarily predict performance for your favorite program!



# Deriving a Single Performance Number

How is the performance of 29 different apps compressed into a single performance number?

- SPEC uses arithmetic mean (AM) – the average of each program's execution time

- Weighted arithmetic mean – the execution times of some programs are weighted to balance priorities



# Common Principles

- Energy: systems leak energy even when idle
- Energy: performance improvements typically also result in energy improvements
- **90-10 rule**: 10% of the program accounts for 90% of execution time
- **Principle of locality**: the same data/code will be used again (temporal locality), nearby data/code will be touched next (spatial locality)



# Example Problem (Quiz)

- A 1 GHz processor takes 100 seconds to execute a program, while consuming 70 W of dynamic power and 30 W of leakage power. Does the program consume less energy in Turbo boost mode when the frequency is increased to 1.2 GHz?



# Example Problem

- A 1 GHz processor takes 100 seconds to execute a program, while consuming 70 W of dynamic power and 30 W of leakage power. Does the program consume less energy in Turbo boost mode when the frequency is increased to 1.2 GHz?

Normal mode energy = 100 W x 100 s = 10,000 J
Turbo mode energy = (70 x 1.2 + 30) x 100/1.2 = 9,500 J

Note:

Frequency only impacts dynamic power, not leakage power.

We assume that the program's CPI is unchanged when frequency is changed, i.e., exec time varies linearly with cycle time.



# Recap

- Knowledge of hardware improves software quality: compilers, OS, threaded programs, memory management
- Important trends: growing transistors, move to multi-core and accelerators, slowing rate of performance improvement, power/thermal constraints, long memory/disk latencies
- Reasoning about performance: clock speeds, CPI, benchmark suites, performance equations
- Next: assembly instructions