Computer Organization

The Role of Performance

Need

- Hardware performance is the primary key for examining the performance
- The way to measure and summarize performance
- To determine Major Factors that affect the performance of a computer

Performance

- Purchasing perspective
  - Given a collection of machines, which has the
    - Best performance ?
    - Least cost ?
    - Best performance/cost?
- Design perspective
  - Faced with design options, which has the
    - Best performance ?
    - Least cost ?
    - Best performance/cost?
- Both require
  - Basis for comparison
  - Metric for evaluation

Performance

Goal:

- To understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors

Performance

- *Performance is the key to understand underlying motivation for the hardware and its organization*

- *Why is some hardware better than others for different programs?*
- *What factors of system performance are hardware related?*
  (e.g., do we need a new machine, or a new operating system?)
- *How does the machine's instruction set affect performance?*

Computer Performance

?

TIME

Computer Performance: TIME

- *Response Time* (*elapsed time*, *latency*):
  - Time to Respond (Complete an operation)

- Other way:
  - How long does it take for *my* job to run?
  - How long does it take to execute (start to finish) *my* job?
  - How long must *I* wait for the database query?

  - Also referred as Execution Time…

Individual user concerns…

Desktops
Or
Embedded Systems

Computer Performance: TIME

- *Throughput*:
  - Jobs completed per unit time

- Other way:
  - How *many* jobs can the machine run at once?
  - What is the *average* execution rate?
  - How *much* work is getting done?

Systems manager concerns…

Servers

Computer Performance: TIME

- *Response Time* (*elapsed time*, *latency*):
  - How long does it take for *my* job to run?
  - How long does it take to execute (start to finish) *my* job?
  - How long must *I* wait for the database query?
- *Throughput*:
  - How *many* jobs can the machine run at once?
  - What is the *average* execution rate?
  - How *much* work is getting done?

Individual user concerns…

Desktops Or Embedded Systems

Systems manager concerns…

Servers

To maximize the performance, Minimize the execution time

Execution Time

- *Elapsed Time*
  - Counts everything from start to finish
    - *Disk and memory accesses*
    - *Waiting for I/O*
    - *Running other programs, etc.*
  - A useful number, but often not good for comparison purposes

Elapsed time = CPU time + wait time (I/O, other programs, etc.)

Execution Time

- *CPU time*
  - Doesn't count waiting for I/O or time spent running other programs
  - Can be divided into *user CPU time* and *system CPU time* (OS calls on behalf of the program)

- *User CPU Time* *The CPU Time spent in a program itself*
- *System CPU Time* *The CPU Time spent in the operating system performing tasks on behalf of the program*

CPU time = User CPU time + System CPU time

⇒ Elapsed time = User CPU time + System CPU time + wait time

- Our focus:
- *User CPU time* (*CPU execution time* or, simply, *execution time*)

Defining (Speed) Performance

Performance = 1/Execution Time

Lesser the time, Higher the Performance

“Machine X is *n* time faster than Machine Y”, Means ?

$$Performance_X \ / \ Performance_Y = n$$

Performance Example:

Time taken to run a program 10s on machine A and 15s on machine B, how much time, the machine A is faster than machine B?

Solution:

A is n times faster than B, if

$$\text{Performance}_A \ / \ \text{Performance}_B = n$$

$$\text{Execution Time}_B \ / \ \text{Execution Time}_A = n$$

$$n = 15s \ / \ 10s = 1.5$$

- So, A is 1.5 times faster than B
- Decreasing response time almost always improves throughput

Clock Cycles

- Instead of reporting execution time in seconds, often uses the term *cycles*
- In modern computers hardware events progress cycle by cycle: in other words, each event, e.g., multiplication, addition, etc., is a sequence of cycles

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

- Also known as tick, clock period, clocks, cycle

Clock Cycles

- *cycle time* = time between ticks
  = seconds per cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s

- *clock rate* = the inverse of the clock cycle (*frequency*) (1 Hz. = 1 cycle/sec, 1 MHz. = 10^6 cycles/sec)
  - e.g., 4.0GHz = 4000MHz = 4.0×10^9Hz

Performance Equation I

CPU execution time = $\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$
For a program

equivalently

CPU execution time for a program = CPU clock cycles for a program × Clock cycle time

- So, to improve performance one can either:
  - Reduce the number of cycles for a program, or
  - Reduce the clock cycle time, or, equivalently,
  - Increase the clock rate (frequency)

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400Mhz clock.
- We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.
- *What clock rate should we tell the designer to target?*

CPU Time Example

Our favorite program runs in 10 seconds on computer A, which has a 400Mhz cloc

$$\text{CPU time}_A = \text{CPU clock cycles}_A / \text{Clock Rate}_A$$

$$10 \text{ Seconds} = \text{CPU clock cycles}_A / 400 \times 10^6 \text{ Cycles/Second}$$

$$\text{CPU clock cycles}_A = 10 \text{ Seconds} \times 400 \times 10^6 \text{ Cycles/Second}$$

$$\text{CPU clock cycles}_A = 4000 \times 10^6 \text{ Cycles}$$

CPU Time Example

⋯ a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program.

What clock rate should we tell the designer to target?

$$\text{CPU time}_B = 1.2 \times \text{CPU clock cycles}_B \;/\; \text{Clock Rate}_B$$

$$6 \text{ Seconds} = 1.2 \times 4000 \times 10^6 \text{ Cycles} \;/\; \text{Clock Rate}_B$$

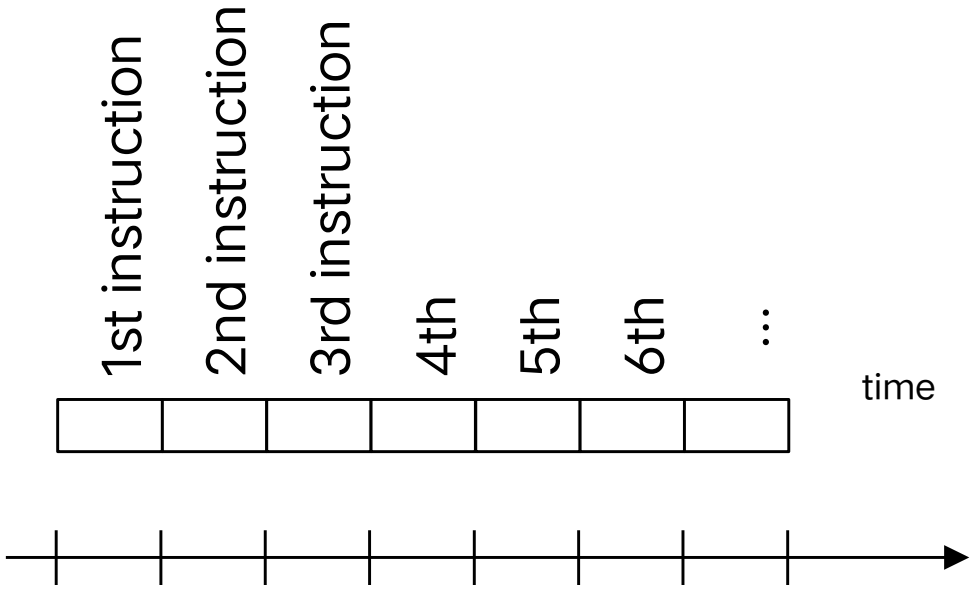
$$\text{Clock Rate}_B = 1.2 \times 4000 \times 10^6 \text{ Cycles} \;/\; 6 \text{ Seconds}$$

$$= 800 \times 10^6 \text{ Cycles / Second}$$

$$= 800 \text{ MHz}$$

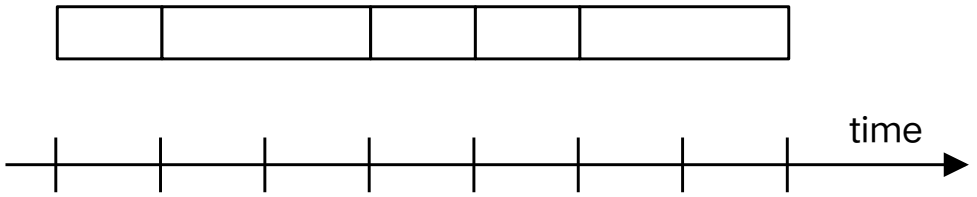
How many cycles for a program?

- Could assume that # of cycles = # of instructions



- *This assumption is incorrect!* Because:
  - Different instructions take different amounts of time (cycles)
  - Why…?

How many cycles for a program?



- Multiplication takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers
- *Important point*:
  - Changing the cycle time often changes the number of cycles required for various instructions because it means changing the hardware design

Performance Equation II

- Performance Equation I

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

- Do not include any reference to the number of instructions needed for the program
- But, the execution time must depend on the number of instructions in a program

Performance Equation II

CPU Clock cycles = Instructions x Average Clock cycles per
for a program Instruction

CPU execution time = Instruction count x Avg CPI x Clock cycle time
for a program for a program

Useful in evaluation of various implementations of the same instruction set architecture (ISA) for the same program

CPI Example I

- Suppose we have two implementations of the same instruction set architecture (ISA). For some program:
  - machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  - machine B has a clock cycle time of 20 ns. and a CPI of 1.2

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

CPI Example I

- machine A has a clock cycle time of 10 ns. and a CPI of 2.0
- machine B has a clock cycle time of 20 ns. and a CPI of 1.2

$$\text{CPU clock cycles}_A = I \times 2.0$$
$$\text{CPU clock cycles}_B = I \times 1.2$$

Where, I = no. of instructions

$$\text{CPU time}_A = \text{CPU clock cycles}_A \times \text{Clock cycle time}_A$$
$$= I \times 2.0 \times 10 \text{ ns} = 20 \times I \text{ ns}$$
$$\text{CPU time}_B = \text{CPU clock cycles}_B \times \text{Clock cycle time}_B$$
$$= I \times 1.2 \times 20 \text{ ns} = 24 \times I \text{ ns}$$

$$\text{Therefore, Performance}_A / \text{Performance}_B$$
$$= \text{Execution Time}_B / \text{Execution Time}_A$$
$$= 24 * I \text{ ns} / 20 * I \text{ ns}$$
$$= 1.2$$

A is 1.2 times faster than B

CPI in More Detail

- Overall CPI for a program depends on both the number of cycles for each instruction type and the frequency of each instruction type in the program execution (If different instruction classes take different numbers of cycles)

- $\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{InstructionCount}_i)$

CPI Example II

- A compiler designer is trying to decide between two code sequences for a particular machine.
- Based on the hardware implementation, there are three different classes of instructions: Class A, Class B and Class C

  and they require 1, 2 and 3 cycles resp.

  Instruction counts for instruction class
- Code sequence A B C

  1 2 1 2

  2 4 1 1

- *Which sequence will be faster?*
- *How much?*
- *What is the CPI for each sequence?*

CPI Example II

Code Sequence Total instructions

1	5
2	6

- Clock Cycles = $\sum_{i=1}^{n}$ (CPI_i x $InstructionCount_i$)

CPU clock cycles1 = (2x1)+(1x2)+(2x3)=10 cycles

CPU clock cycles2 = 9 cycles

So, code sequence 2 is faster, even though it actually executes one extra instruction.

CPI= CPU Clock Cycles/Instruction Count

CPI1 = 10/5 = 2

CPI2 = 9/6 = 1.5

Factors in the Performance Equation

- CPU Execution time
  - Running the program (Seconds for the program)
- Clock Cycle Time
  - Published as part of the documentation for a computer (Seconds per clock cycle)
- Instruction Count
  - Depends on the architecture (Instructions executed for the program)
- CPI
  - Varies by application, as well as among implementations with the same Instruction set (Average number of clock cycles per instruction)

Benchmarks

- Performance best determined by running a real application
  - Use programs typical of expected workload
  - or, typical of expected class of applications
    e.g., compilers/editors, scientific applications, graphics, etc.

- Benchmark suites
  - Perfect Club: set of application codes
  - Livermore Loops: 24 loop kernels
  - Linpack: linear algebra package
  - SPEC: mix of code from industry organization

SPEC

- Programs used to measure performance
  - Specialized benchmarks for particular classes of applications
- Standard Performance Evaluation Corporation (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- Elapsed time to execute a selection of programs
  - Negligible I/O, so focuses on CPU performance
- Normalize relative to reference machine
- Summarize as geometric mean of performance ratios
  - SPEC CPU2006
    - CINT2006 (integer) and CFP2006 (floating-point)

Specialized SPEC Benchmarks

- I/O
- Network
- Graphics
- Java
- Web server
- Transaction processing (databases)

Other Performance Metrics

- Power consumption
  - Especially in the embedded market where battery life is important
    - For power-limited applications, the most important metric is energy efficiency
  - Clock Rate and Power both are co-related
  - So, both increased rapidly for decades and then flattened off
  - And now there is a practical power limit for cooling commodity microprocessors

Other Performance Metrics

- **Design of Microprocessors**
  - Power limit forced the drastic change
  - Rather than to decrease the response time of a single program running on a single microprocessor
    - Adapt the microprocessors with multiple processors per chip
      - Where the benefit is often more on throughput than on response time

Other Performance Metrics

- **Amdahl's Law**
  - A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used

- Execution Time After Improvement =

$$\text{Execution Time Unaffected} + \left(\frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} \right)$$

Amdahl's Law

- Suppose a program runs in 100 seconds on a machine, with multiply operations responsible for 80 seconds of this time.
  - How much do I have to improve the speed of multiplication if I want my program to run 5 times faster?

- Execution Time After Improvement =

$$\text{Execution Time Unaffected} + \left(\frac{\text{Execution Time Affected}}{\text{Amount of Improvement}} \right)$$

Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

Improved part of code

- Execution Time After Improvement =
  Execution Time Unaffected + $\frac{\text{Execution Time Affected}}{\text{Amount of Improvement}}$

- Execution Time After Improvement = 20+(80/n)
- 100/5 = 20 = 20 + (80/n)
- 0 = 80/n Can’t be done!

Other Performance Metrics

- **MIPS (**Millions of Instructions Per Second**)**
  - It is an instruction execution rate
  - Specifies performance inversely to execution time
  - Faster computers have a higher MIPS rating
  - Easy to understand

MIPS

- MIPS = $\frac{\text{Instruction Count}}{\text{Execution time} \times 10^6}$

$$= \frac{\text{Instruction Count}}{\frac{\text{CPU Clock Cycles}}{\text{Clock Rate}} \times 10^6} = \frac{\text{Clock Rate}}{\text{CPI} \times 10^6}$$

(As, CPU Clock cycles = Instructions count x Average Clock cycles per Instruction)

- CPI varies between programs on a given CPU

MIPS Example

- Two different compilers are being tested for a 500 MHz. Machine with three different classes of instructions: Class A, Class B, and Class C, which require 1, 2 and 3 cycles (respectively). Both compilers are used to produce code for a large piece of software.
- Compiler 1 generates code with 5 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.
- Compiler 2 generates code with 10 billion Class A instructions, 1 billion Class B instructions, and 1 billion Class C instructions.

- *Which sequence will be faster according to execution time?*
- *Which sequence will be faster according to MIPS?*

MIPS Example

- Execution Time = $\frac{\text{CPU clock cycles}}{\text{Clock Rate}}$

$$\text{CPU Clock cycles} = \sum_{i=1}^{n} (CPI_i \times \text{InstructionCount}_i)$$

CPU Clock cycles**1**= (5x1+1x2+1x3) x 10^9 = 10 x 10^9

CPU Clock cycles**2**= (10x1+1x2+1x3) x 10^9 = 15 x 10^9

So, Execution Time**1**= (10 x 10^9) / 500 x 10^6 = 20 Seconds

Execution Time**2**= (15 x 10^9) / 500 x 10^6 = 30 Seconds

So, Compiler 1 generates the faster program

MIPS Example

- MIPS = $\frac{\text{Instruction Count}}{\text{Execution time} \times 10^6}$
- MIPS1 = $(5+1+1) \times 10^9 / 20 \times 10^9 = 350$
- MIPS2 = $(10+1+1) \times 10^9 / 30 \times 10^9 = 400$

So, the code from compiler 2 has a higher MIPS rating, but the compiler 1 runs faster!

MIPS can fail to give a true picture of performance even when comparing two versions of the same program on the same machine

- MIPS doesn't account for
  - Differences in ISAs between computers
  - Differences in complexity between instructions

Terminology

- A given program will require:
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- A vocabulary that relates these quantities:
  - *cycle time* (seconds per cycle)
  - *clock rate* (cycles per second)
  - (*average*) *CPI* (cycles per instruction)
    - a floating point intensive application might have a higher average CPI
  - *MIPS* (millions of instructions per second)
    - this would be higher for a program using simple instructions

Summary

- Depends on …
  - The instructions set
  - The Processor organization
  - Compilation techniques