Performance

- Performance
  - time to do the task, i.e., execution time, response time, latency, etc.
  - tasks finished per day, hour, week, sec, ms, ns, etc
    » throughput, bandwidth

X is n times faster than Y means that

\[
\frac{\text{ExTime}(Y)}{\text{ExTime}(X)} = \frac{\text{Performance}(X)}{\text{Performance}(Y)} = n
\]

Performance Metrics

MIPS
- millions of instructions per second

MFLOPS
- millions of (F.P.) operations per second

Others
- Megabytes per second
- Cycles per second (clock rate)

CPU Performance

CPU time = seconds/program
= instructions/program \times\ cycles/instruction \times\ seconds/cycle

<table>
<thead>
<tr>
<th>Program</th>
<th>instr. count</th>
<th>CPI</th>
<th>clock rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>compiler</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>inst. set</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>organization</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>technology</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CPI

- CPI: average clock cycles per instruction

\[
CPI = \frac{\text{cpu time} \times \text{clock rate}}{\text{instruction count}}
\]

- CPI = clock cycles / instruction count

- CPU time = clock cycle time \times \sum_{i=1}^{n} CPI_i \times I_i

Instruction frequency = \( F_i = \frac{\text{No. of the ith instr.}}{\text{total inst. count}} \)

Invest resources where most time is spent

Make frequently used components run fast

RISC Processor

A simple RISC machine instr. frequency mix

<table>
<thead>
<tr>
<th>Op</th>
<th>Freq</th>
<th>Cycles</th>
<th>CPI</th>
<th>% time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>50%</td>
<td>1</td>
<td>.5</td>
<td>33%</td>
</tr>
<tr>
<td>Load</td>
<td>20%</td>
<td>2</td>
<td>.4</td>
<td>27%</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>2</td>
<td>.2</td>
<td>13%</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2</td>
<td>.4</td>
<td>27%</td>
</tr>
</tbody>
</table>

Amdahl’s Law

Speedup due to enhancement E

\[
\text{Speedup(E)} = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}
\]

Suppose a fraction F of the task is accelerated by a factor S

\[
\text{Speedup(E)} = \frac{1}{(1-F) + \frac{F}{S}}
\]

If a program has two processes to run, and we double the speed of one process, what is the speedup?

Processor Metrics

- Processor Metrics
  - CPU execution time
  - CPU clock cycles per program
  - CPI (this is an average value)
  - IPC = 1/CPI, instruction per clock cycle
  - CPI is closely related with
    - Instruction set architecture
    - Implementation method
    - program measured
Amdahl’s Law

make the common case fast
- examples used in modern computers
  » registers: keep frequently used values in register
    files (register file is small in size, small access time)
  » memory hierarchy: cache memory (temporal locality,
    spatial locality, save recently used data in cache.
  » simple addressing modes

MIPS

- MIPS calculation
  \[ \text{MIPS} = \frac{\text{instruction count}}{\text{execution time} \times 10^6} \]
- MIPS (millions of instructions per second)
  MIPS would be higher for a program using simple instructions.
  \[ \text{MIPS} = \frac{\text{MHz}}{\text{CPI}} \]
  A processor running at 100 MHz and CPI = 2,
  MIPS of this processor = 100/2 = 50

MFLOPS

MFLOPS
\[ \text{MFLOPS} = \frac{\text{FP operations}}{\text{(time} \times 10^6)} \]

What’s wrong with MFLOPS?
- Machine dependent
- Program dependent
  The execution times of FP multiply, FP divide
  FP add, FP subtraction are different.

Evaluate Performance

Benchmarks
- Benchmarks should represent a large class of important
  programs
- improving benchmark performance should help many
  programs
- for better or worse, benchmarks shape a field
- bad benchmarks hurt progress
- good benchmarks accelerate progress
Evaluate Processor Performance

- Toy benchmarks
  - 100 line code, such as puzzle, quicksort, bubble sort, etc.
- Synthetic Benchmarks
  - attempt to match average frequencies of real workloads
  - such as Whetstone, Dhrystone
- Microbenchmarks
  - collection of small tests that cover machine primitives
- Kernels
  - time critical excerpts of real programs
  - such as Linpack, Livermore loops
- Real Programs
  - such as gcc, spice, database, stock trading

SPEC Benchmark

- SPEC : System Performance Evaluation Committee
- EE Times + 5 companies (Sun, MIPS, HP, Apollo, DEC) join together to perform SPEC in 1988.
- SPEC creates standard list of programs, inputs, reports, (some real programs, includes OS calls, some I/O)

SPEC Benchmark

- First round 1989 :
  - 10 programs
- Second round 1992 :
  - Specint92 (6 integer programs), SpecFP92 (14 floating point programs)
- Third round 1995 :
  - new sets of programs
- Benchmarks useful for 3 years!

Comparing and Summarizing Performance

- Arithmetic Mean (AM)

\[ AM = \frac{\sum_{i=1}^{n} \text{Time}_i}{n} \]
Comparing and Summarizing Performance

• Harmonic Mean (HM)

\[
HM = \frac{n}{\sum_{i=1}^{n} \frac{1}{\text{rate}_i}}
\]

rate = 1 / execution time
n programs in the workload

• Geometric Mean (GM)

\[
GM = \sqrt[n]{\prod_{i=1}^{n} \text{execution time ratio}_i}
\]

execution time ratio, is the execution time normalized to the reference machine for the ith program of a total of n programs

• Arithmetic Mean
  – AM tracks relative execution time

• Harmonic Mean
  – HM tracks total execution time

• Geometric Mean
  – GM rewards all improvements equally

Performance Summary

• CPU time
  – time on workload is the final measure of computer performance

• Benchmarks
  – good benchmarks, good ways to summarize performance

• Amdahl's Law
  – invest your effort where the time spent most
  – remaining unimproved parts also count
CPI Example

• 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 CPI of 2.0
  Machine B has a clock cycle time of 20 ns. and CPI of 1.2

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

• If two machines have the same ISA which of our quantities (clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?

No. of Instructions Example

• 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 respectively
  • The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
  • The second code sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

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

MIPS Example

• Two different compilers are being tested for a 100MHz machine with three different classes of instructions: Class A, Class B, and Class C which require one, two, and three cycles respectively. Both compilers are used to produce code for a large piece of software.
  • The first compiler’s code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions
  • The second compiler’s code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

Which sequence will be faster according to MIPS?
• Which sequence will be faster according to execution time?

Amdahl’s Law Example

• Suppose we enhance a machine making all floating-point instructions run five times faster. If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?
Some Concepts to Remember

• For a given architecture, performance increase come from
  – Increases in clock rate
  – Lowers CPI (through improvements in processor organization)
  – Lowers CPI or instruction count (through compiler enhancements)