In the News

- "Intel shows off Penryn chips" CNET News Jan. 26th 2007
- New 45-nanometer generation (smaller Core 2 Duo with enhancements)
- New extensions to the instruction set
  - SSE4: fourth generation of Streaming SIMD (single instruction, multiple data) Extensions for multimedia applications and technical computing
- Planning for a more powerful chip when the 45-nanometer technology matures
- AMD, IBM, and others to follow

Performance Metrics

- Purchasing perspective
  - given a collection of machines, which has the
    - best performance?
    - least cost?
    - best cost/performance?
- Design perspective
  - faced with design options, which has the
    - best performance improvement?
    - least cost?
    - best cost/performance?
- Both require
  - basis for comparison
  - metric for evaluation

Our goal is to understand what factors in the architecture contribute to overall system performance and the relative importance (and cost) of these factors

Defining (Speed) Performance

- Normally interested in reducing
  - Response time (aka execution time) – the time between the start and the completion of a task
    - Important to individual users
  - Thus, to maximize performance, need to minimize execution time
    \[
    \text{performance}_X = \frac{1}{\text{execution time}_X}
    \]
    If X is n times faster than Y, then
    \[
    \text{performance}_X = \frac{\text{execution time}_Y}{\text{execution time}_X} = n
    \]
- Throughput – the total amount of work done in a given time
  - Important to data center managers
  - Decreasing response time almost always improves throughput

Performance Factors

- Want to distinguish elapsed time and the time spent on our task
- CPU execution time (CPU time) – time the CPU spends working on a task
  - Does not include time waiting for I/O or running other programs

CPU execution time

\[
\text{CPU execution time} = \frac{\# \text{CPU clock cycles}}{\text{clock cycle time}}
\]

or

CPU execution time

\[
\text{CPU execution time} = \frac{\# \text{CPU clock cycles for a program}}{\text{clock cycle time}}
\]

Can improve performance by reducing either the length of the clock cycle or the number of clock cycles required for a program

Review: Machine Clock Rate

- Clock rate (MHz, GHz) is inverse of clock cycle time (clock period)

\[
\text{CC} = \frac{1}{\text{CR}}
\]

10 nsec clock cycle ⇒ 100 MHz clock rate
5 nsec clock cycle ⇒ 200 MHz clock rate
2 nsec clock cycle ⇒ 500 MHz clock rate
1 nsec clock cycle ⇒ 1 GHz clock rate
500 psec clock cycle ⇒ 2 GHz clock rate
250 psec clock cycle ⇒ 4 GHz clock rate
200 psec clock cycle ⇒ 5 GHz clock rate
Clock Cycles per Instruction

- Not all instructions take the same amount of time to execute.
- One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction.

\[ \text{# CPU clock cycles} = \text{# Instructions} \times \text{Average clock cycles per program per instruction} \]

- Clock cycles per instruction (CPI) – the average number of clock cycles each instruction takes to execute.
- A way to compare two different implementations of the same ISA.

<table>
<thead>
<tr>
<th>CPI for this instruction class</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPI</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Effective CPI

- Computing the overall effective CPI is done by looking at the different types of instructions and their individual cycle counts and averaging.

\[
\text{Overall effective CPI} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i)
\]

- Where IC is the count (percentage) of the number of instructions of class i executed.
- CPI is the (average) number of clock cycles per instruction for that instruction class.
- n is the number of instruction classes.

- The overall effective CPI varies by instruction mix – a measure of the dynamic frequency of instructions across one or many programs.

THE Performance Equation

- Our basic performance equation is then

\[
\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{clock cycle}
\]

- These equations separate the three key factors that affect performance:
  - Can measure the CPU execution time by running the program.
  - The clock rate is usually given.
  - Can measure overall instruction count by using profilers/simulators without knowing all of the implementation details.
  - CPI varies by instruction type and ISA implementation for which we must know the implementation details.

Determinates of CPU Performance

<table>
<thead>
<tr>
<th>CPU time = Instruction_count \times CPI \times clock_cycle</th>
</tr>
</thead>
</table>

- Algorithm
- Programming language
- Compiler
- ISA
- Processor organization
- Technology

A Simple Example

<table>
<thead>
<tr>
<th>Op</th>
<th>Freq</th>
<th>CPI</th>
<th>Freq x CPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>50%</td>
<td>1</td>
<td>.5</td>
</tr>
<tr>
<td>Load</td>
<td>20%</td>
<td>5</td>
<td>1.0</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>3</td>
<td>.3</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>4</td>
<td>.8</td>
</tr>
</tbody>
</table>

\[
\Sigma = 2.2
\]

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
- CPU time new = 1.6 x IC x CC so 2.2/1.6 means 37.5% faster.
- How does this compare with using branch prediction to shave a cycle off the branch time?
- CPU time new = 2.0 x IC x CC so 2.2/2.0 means 10% faster.
- What if two ALU instructions could be executed at once?
- CPU time new = 1.95 x IC x CC so 2.2/1.95 means 12.8% faster.
Comparing and Summarizing Performance

- How do we summarize the performance for benchmark set with a single number?
  - The average of execution times that is directly proportional to total execution time is the arithmetic mean (AM)
  
  \[ AM = \frac{1}{n} \sum_{i=1}^{n} Time_i \]
  
  - Where Time is the execution time for the i-th program of a total of n programs in the workload
  - A smaller mean indicates a smaller average execution time and thus improved performance

- Guiding principle in reporting performance measurements is reproducibility – list everything another experimenter would need to duplicate the experiment (version of the operating system, compiler settings, input set used, specific computer configuration (clock rate, cache sizes and speed, memory size and speed, etc.))

Example SPEC Ratings

- SPEC Benchmarks www.spec.org

<table>
<thead>
<tr>
<th>Integer benchmarks</th>
<th>FP benchmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>gzip</td>
<td>ew</td>
</tr>
<tr>
<td>vpr</td>
<td>fpp</td>
</tr>
<tr>
<td>gcc</td>
<td>mgdr</td>
</tr>
<tr>
<td>mcf</td>
<td>aplp</td>
</tr>
<tr>
<td>crafty</td>
<td>fisf</td>
</tr>
<tr>
<td>parser</td>
<td>galg</td>
</tr>
<tr>
<td>eon</td>
<td>art</td>
</tr>
<tr>
<td>perfibk</td>
<td>equake</td>
</tr>
<tr>
<td>gap</td>
<td>facnom</td>
</tr>
<tr>
<td>vortex</td>
<td>ammp</td>
</tr>
<tr>
<td>bzip2</td>
<td>lusac</td>
</tr>
<tr>
<td>twolf</td>
<td>trac</td>
</tr>
<tr>
<td>sixtrack</td>
<td>sixtrack</td>
</tr>
<tr>
<td>bzip2</td>
<td>sixtrack</td>
</tr>
<tr>
<td>primality testing</td>
<td>sixtrack</td>
</tr>
<tr>
<td>Image recognition</td>
<td>sixtrack</td>
</tr>
<tr>
<td>SPEC Benchmarks</td>
<td>sixtrack</td>
</tr>
</tbody>
</table>

Other Performance Metrics

- Power consumption – especially in the embedded market where battery life is important (and passive cooling)
  - For power-limited applications, the most important metric is energy efficiency: take performance divide by average power consumption while running the benchmark

Summary: Evaluating ISAs

- Design-time metrics:
  - Can it be implemented, in how long, at what cost?
  - Can it be programmed? Ease of compilation?

- Static Metrics:
  - How many bytes does the program occupy in memory?

- Dynamic Metrics:
  - How many instructions are executed? How many bytes does the processor fetch to execute the program?
  - How many clock cycles are required per instruction?
  - How "lean" a clock is practical?

  Best Metric: Time to execute the program!
  
  \[ CPI = \frac{Inst. Count}{Cycle Time} \]

Next Lecture and Reminders

- Next lecture
  - MIPS non-pipelined datapath/control path review

- Reminders
  - HW3 due February 7th