Performance

CS 365 Lecture 10
Prof. Yih Huang

Computer Performance:
TIME, TIME, TIME

- **Response Time** (latency)
  - How long does it take for my job to run?
- **Throughput**
  - How many jobs can a system supports at once?
  - What is the average execution rate?
- If we upgrade a machine with a new processor what do we increase?
- If we add a new machine to the lab what do we increase?
Execution Time

☐ Elapsed Time
– counts everything (disk and memory accesses, I/O, etc.)

☐ CPU time
– doesn't count I/O or time spent running other programs
– can be broken up into system time, and user time

Program Performance

\[
\text{cpu time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}
\]

**Instruction Count**: the # of instructions to execute the program

**CPI**: average # of clock cycles per instruction

Clock period: length of one clock cycle in second
Discussion

- When a processor vendor advertises GHz, what is the factor of focus?

- Instruction count depends on particular program and input data.

- CPI is also program dependent.
  - Different programs have different instruction mixes.

CPI

\[ CPI = \sum_{j=1}^{n} CPI_j \times F_j \] where \( F_j = \frac{I_j}{\text{Instruction Count}} \)

- # of cycles to execute instruction \( j \), determined by CPU architecture
- Frequency of instruction \( j \) in the program, determined by the given program
CPI Example, using MIPS Lite

- 50% of the instructions of Program A is type R, 30% branches, 10% load, 10% store. Compute the CPI of Program A.

- 30% of the instructions of Program B is type R, 10% branches, 40% load, 20% store. Compute the CPI of Program B.

CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).
- For some program with IC = x,
  - 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
- What machine is faster for this program, and by how much?
Instruction Count Example

- A compiler is trying to decide between two code sequences. There are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles respectively.
- The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
- The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

- CPI for sequence I:

- CPU cycles for sequence I:

- CPI for sequence II:

- CPU cycles for sequence II:
MIPS

- **MIPS**: Millions of instructions per second
- Another performance indicator used by many processor vendors.
- Intuitively, the more instructions a processor can execute per second, the better its performance.
- Or is it?

MIPS example

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles respectively.
- For a given program,
  - The first compiler uses 5 million Class A instructions, 1 million Class B, and 1 million Class C instructions.
  - The second uses 10 million Class A, 1 million Class B, and 1 million Class C.
Which sequence will be faster according to MIPS?

Which sequence will be faster according to execution time?

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.

- Small benchmarks
  - nice for architects and designers
  - easy to standardize
  - can be abused
SPEC

- SPEC (System Performance Evaluation Cooperative)
  - companies have agreed on a set of real program and inputs
  - can still be abused
  - valuable indicator of performance (and compiler technology)

### SPEC ‘95

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>go</strong></td>
<td>Artificial intelligence; plays the game of Go</td>
</tr>
<tr>
<td><strong>m88ksim</strong></td>
<td>Motorola 88K chip simulator; runs test program</td>
</tr>
<tr>
<td><strong>gcc</strong></td>
<td>The Gnu C compiler generating SPARC code</td>
</tr>
<tr>
<td><strong>compress</strong></td>
<td>Compresses and decompresses file in memory</td>
</tr>
<tr>
<td><strong>li</strong></td>
<td>Lisp interpreter</td>
</tr>
<tr>
<td><strong>ijpeg</strong></td>
<td>Graphic compression and decompression</td>
</tr>
<tr>
<td><strong>perl</strong></td>
<td>Manipulates strings and prime numbers in the special-purpose programming language Perl</td>
</tr>
<tr>
<td><strong>vortex</strong></td>
<td>A database program</td>
</tr>
<tr>
<td><strong>tomcatv</strong></td>
<td>A mesh generation program</td>
</tr>
<tr>
<td><strong>swim</strong></td>
<td>Shallow water model with 513 x 513 grid</td>
</tr>
<tr>
<td><strong>su2cor</strong></td>
<td>Quantum physics; Monte Carlo simulation</td>
</tr>
<tr>
<td><strong>hydro2d</strong></td>
<td>Astrophysics; Hydrodynamic Naïve Stokes equations</td>
</tr>
<tr>
<td><strong>mgrid</strong></td>
<td>Multigrid solver in 3-D potential field</td>
</tr>
<tr>
<td><strong>applu</strong></td>
<td>Parabolic/elliptic partial differential equations</td>
</tr>
<tr>
<td><strong>trub3d</strong></td>
<td>Simulates isotropic, homogeneous turbulence in a cube</td>
</tr>
<tr>
<td><strong>apsi</strong></td>
<td>Solves problems regarding temperature, wind velocity, and distribution of pollutant</td>
</tr>
<tr>
<td><strong>fpppp</strong></td>
<td>Quantum chemistry</td>
</tr>
<tr>
<td><strong>wave5</strong></td>
<td>Plasma physics; electromagnetic particle simulation</td>
</tr>
</tbody>
</table>
Does doubling the clock rate double the performance?

SPECint

SPECfp
Amdahl's Law

- Execution Time After Improvement = Execution Time Unaffected + (Execution Time Affected / Amount of Improvement)

- Example: Machines A and B use the same implementation design. The only difference between is clock rate.

- Running on Machine A, program X spends 20% of its execution times accessing memory and 80% doing computations.
- Machine B’s clock rate is twice of that of A.
  - How much faster X will be on B?

  - How about 4 times faster?
  - 8 times faster?
  - 16 times faster?
**Discussion**

- There is no absolute way to say one processor is faster than another.
- Performance numbers are always specific to particular programs.
- For a given architecture, performance can be increased by
  - Increases in clock rate
  - New datapaths that lower CPI
  - Smarter compiler technologies

**Increasing Clock Rate**

- Achieved by the (constantly advancing) VLSI technologies
- Also can be made possible by reorganizing the datapath so that each cycle performs less tasks.
  - Recall the single and multi cycle datapaths of MIPS Lite.
- May have adverse effects on CPI
Reorganizing the Datapath

- The aim is to
  - Decrease CPI
  - Facilitate faster clock rate
- To lower CPI, we add more hardware so that more parallel tasks can be done in one cycle.
- To reduce clock rate, we want less sequential tasks per cycle.

VLSI Factors

- Give the same VLSI technology, there are limits in
  - how much gates one can have in a chip
  - How fast the clock can be
- Moreover, the two factors are intertwined.
  - Less gates produce faster clocks
  - The relationship is by no means linear.
Reorganization to Increase Clock

- Consider the floating point adder in the previous lecture.
  - One clock cycle has to accommodate the delays of two ALU.
- Special registers can be added between the two so that one clock cycle is only one ALU
  - Faster clock rates are now possible
  - The effect on CPI?

Reorganization to Lower CPI

- Consider the BEQ instruction of YH16.
- We can merge Cycle 4 and 5 by adding a second ALU to the datapath.
- The effect on clock rate?
  - Good news: the new adder operates in parallel with the other.
  - Bad news: more gates always drag down the system to some extent.
### BEQ

- **Cycle 0:**
  - IR ← Memory[PC]
  - C ← PC + 1

- **Cycle 1:**
  - A ← Reg[rs]
  - B ← Reg[rt]
  - MDR ← Mem[C]
  - PC ← C

- **Cycle 2:** C ← PC + 1
- **Cycle 3:** PC ← C

- **Cycle 4:**
  - C ← PC + MDR

- **Cycle 5:**
  - Z ← (A==B)

- **Cycle 6:**
  - PC ← C, if Z

### Modify the Data Path

![Data Path Diagram]
BEQ with a Separate PC+1 Adder

- **Cycle 0:**
  - IR ← Memory[PC]
  - PC ← PC + 1

- **Cycle 1:**
  - A ← Reg[rs]
  - B ← Reg[rt]
  - MDR ← Mem[PC]

- **Cycle 2:**
  - C ← PC + MDR

- **Cycle 3:**
  - Z ← (A==B)

- **Cycle 4:**
  - PC ← C, if Z

---

My Observations

- Until very recently, Intel generally goes for fast clock rates, willing to sacrifice CPI.
  - Pentium III 400MHz was actually slower than Pentium II 400MHz.

- AMD prefers higher CPI with relatively slow clocks.

- PowerPC also emphasizes CPI more than clock rates.
  - Do not compare Mac Hz with PC Hz.