EE350 lecture 2

Overview of Today’s lecture

- Review of last lecture
- Yields & costs of ICs
- Performance evaluation - Throughput vs. response time
- Types of benchmarks
- Break
- Some existing benchmarks
- Reading Chapters 1, 2

5 components of Computer
forces on computer architecture
Costs of Integrated Circuits

- Die Cost = Wafer Cost/(Dies per wafer * Die yield)
- Dies per wafer = Wafer area/Die area
- From empirical observation: Die yield = wafer yield * \{1 + (Defects per unit area * Die area/\alpha)}^{-\alpha} \alpha = 2.
- Die cost proportional to roughly the cube of the area.

### Real World Examples

<table>
<thead>
<tr>
<th>Chip</th>
<th>Wafer cost</th>
<th>Defect/cm²</th>
<th>Area/mm²</th>
<th>Dies/wafer</th>
<th>Yield</th>
<th>Die cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>386 DX</td>
<td>$900</td>
<td>1.0</td>
<td>43</td>
<td>360</td>
<td>71%</td>
<td>$4</td>
</tr>
<tr>
<td>486DX2</td>
<td>$1200</td>
<td>1.0</td>
<td>81</td>
<td>181</td>
<td>54%</td>
<td>$12</td>
</tr>
<tr>
<td>PowerPC 601</td>
<td>$1700</td>
<td>1.3</td>
<td>121</td>
<td>115</td>
<td>28%</td>
<td>$53</td>
</tr>
<tr>
<td>HP PA 7100</td>
<td>$1300</td>
<td>1.0</td>
<td>196</td>
<td>66</td>
<td>27%</td>
<td>$73</td>
</tr>
<tr>
<td>DEC Alpha</td>
<td>$1500</td>
<td>1.2</td>
<td>234</td>
<td>53</td>
<td>19%</td>
<td>$149</td>
</tr>
<tr>
<td>SuperSPARC</td>
<td>$1700</td>
<td>1.6</td>
<td>256</td>
<td>48</td>
<td>13%</td>
<td>$272</td>
</tr>
<tr>
<td>Pentium</td>
<td>$1500</td>
<td>1.5</td>
<td>296</td>
<td>40</td>
<td>9%</td>
<td>$417</td>
</tr>
</tbody>
</table>
Other Costs

- IC cost = (Die cost + Testing cost + Packaging cost )/final test yield
- Packaging test depends on pins, heat dissipation, ...
- Testing cost depends on complexity of the chip, design philosophy,...

<table>
<thead>
<tr>
<th>Chip</th>
<th>Die</th>
<th>Package</th>
<th>Test</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>386DX</td>
<td>$4</td>
<td>$1</td>
<td>$4</td>
<td>$9</td>
</tr>
<tr>
<td>486DX2</td>
<td>$12</td>
<td>$11</td>
<td>$12</td>
<td>$35</td>
</tr>
<tr>
<td>PowerPC 601</td>
<td>$53</td>
<td>$3</td>
<td>$21</td>
<td>$77</td>
</tr>
<tr>
<td>HP PA 7100</td>
<td>$73</td>
<td>$35</td>
<td>$16</td>
<td>$124</td>
</tr>
<tr>
<td>DEC Alpha</td>
<td>$149</td>
<td>$30</td>
<td>$23</td>
<td>$202</td>
</tr>
<tr>
<td>SuperSPARC</td>
<td>$272</td>
<td>$20</td>
<td>$34</td>
<td>$326</td>
</tr>
<tr>
<td>Pentium</td>
<td>$417</td>
<td>$19</td>
<td>$37</td>
<td>$473</td>
</tr>
</tbody>
</table>

Technology Trends

- Die size 2x/3 years, line widths halve/7 years
- DRAM capacity 4x/3 years, speed 1.4x/10 years
- Logic 2x/3 years, speed 2x/3 years
- Disk 4x/3 years, speed 1.4x/10 years
- Processor performance 1.54x/year
Performance measures

- **Throughput**: number of "tasks" completed in unit time.
- **Response time**: time to complete a given task.
- **Service time**: time to complete a given task under zero load.
- Performance/Cost Analysis is the main driving force of engineering.

<table>
<thead>
<tr>
<th>Plane</th>
<th>Time (hrs)</th>
<th>Speed (mph)</th>
<th>Passengers (1000)</th>
<th>Throughput (pmph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing</td>
<td>6.5</td>
<td>610</td>
<td>470</td>
<td>286.70</td>
</tr>
<tr>
<td>Concorde</td>
<td>3.0</td>
<td>1350</td>
<td>132</td>
<td>178.00</td>
</tr>
</tbody>
</table>

- Which plane is better?
- Response time of Concorde is better
- Throughput of Boeing is better
- What about cost of a ticket
- Throughput/$ of Boeing even better
- Choice of performance metric is important
- Reducing response time is normally harder than increasing throughput.
Performance metrics

- Application: execution time, operations/second
- ProcessorISA: MIPS, MFLOPS
- Datapath: Megabytes/second
- Hardware/Implementation: clock rate (MHz)

Performance metrics (cont’d)

- CPU execution time = CPU clock cycles/program x clock cycle time
- CPU clock cycles/program = Instructions/program x avg. clk cycles/instr.
- CPI = (CPU clock cycles/program)/(Instructions/program)
- CPU time = seconds/program = (Instructions/program) * (cycles/instruction) * (seconds/cycle)
**CPU performance**

- CPU time = (Instr.count) * CPI / (clock rate)

<table>
<thead>
<tr>
<th></th>
<th>instr. count</th>
<th>CPI</th>
<th>clock rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compiler</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CPU performance**

<table>
<thead>
<tr>
<th></th>
<th>instr. count</th>
<th>CPI</th>
<th>clock rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compiler</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>ISA</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></td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>
CPI

- CPU time = (Instr.count) * CPI / (clock rate)
- CPI = CPU time * clock rate / (instr. count)
- CPI = \( \sum cpi_i \times f_i \), where \( cpi_i \) = CPI of instruction \( i \) and \( f_i \) = frequency of instruction \( i \)

Example for calculating CPI

- For a typical program

<table>
<thead>
<tr>
<th>Operation</th>
<th>Freq</th>
<th>Cycles</th>
<th>( cpi_i )</th>
<th>% Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>50%</td>
<td>1</td>
<td>0.5</td>
<td>29%</td>
</tr>
<tr>
<td>LOAD</td>
<td>20%</td>
<td>2</td>
<td>0.4</td>
<td>24%</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>2</td>
<td>0.2</td>
<td>12%</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>3</td>
<td>0.6</td>
<td>35%</td>
</tr>
<tr>
<td>CPI</td>
<td></td>
<td></td>
<td>1.7</td>
<td></td>
</tr>
</tbody>
</table>

- CPI brings out the relative advantages of improving different instructions
- Spend resources where programs spend most time.
MIPS, MFLOPS

- Different machines have different instruction sets
- Same program $\Rightarrow$ different instruction counts on diff. machines
- "Power" of instructions different
- MIPS can be higher if little is done in a short time
- MFLOPS has similar problem + program may not spend anytime
- Not good measures to compare machines
- Remember execution time is the primary measure.

Benchmarks

- Way of comparing different systems.
- Good benchmarks represent a large class of programs
- Improving benchmark performance $\Rightarrow$ improved performance for large number of programs
- Good benchmarks provide good targets
- Performance metrics used to sell machines
- Good benchmarks will focus energy in right direction
- Bad benchmarks only help sell machines
- Bad benchmarks may hurt real programs
Some Successful benchmarks

- SPEC -CPU performance
- TPC -database (I/O) performance

SPEC benchmark

- Systems Performance Evaluation Committee formed by Sun, MIPS, HP, Apollo, DEC and EE Times magazine.
- Benchmark is a set of real programs, inputs.
- Included a standard for reporting results – processor used, memory system parameters, etc.
- Allowed compiler optimizations – need to be reported.
TPC benchmark

- Tandem and other companies created Transaction Processing Council
- Created TPC benchmark to measure systems performance in database applications
- Specifies the size of the database, configurations to be used for testing
- Specifies rules for scaling the system/benchmark tests
- Synthetic workload mimicking typical transaction systems.
- Performance metrics: "Transactions per second" and "Transactions per second/\$."
- Measures database software performance and system performance.
- Widely used by industry.
- Several versions of benchmark: TPC-A, TPC-B, TPC-C and TPC-D.

Improving Performance - Amdahl’s law

- Let T’ be the fraction of program that can’t be improved by new enhancements.
- New execution time = (1-f)/S + f, compared to 1.
- Performance Improvement or Speedup = (Old execution time)/(New execution time)
- Speedup = 1/(f + (1-f))/S \neq S.
- An improvement of S in one part doesn’t improve the whole program execution time by S.
- Example: 90% of a program can be run 10 times faster on a new machine, how much faster is the new machine on the whole program?
- Speedup = 1/(0.9/10+0.1) = 1/0.19 = 5.26.