**Review: What is “Computer Architecture”**

- Co-ordination of *levels of abstraction*

![Diagram](image1)

- Under a set of rapidly changing Forces
- CA = IS + CO

**Review: Levels of Representation**

<table>
<thead>
<tr>
<th>High Level Language Program</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly Language Program</td>
<td>Assembler</td>
</tr>
<tr>
<td>Machine Language Program</td>
<td>Machine Interpretation</td>
</tr>
</tbody>
</table>

- `temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;`

- `lw $15, 0($2)
  lw $16, 4($2)
  sw $16, 0($2)
  sw $15, 4($2)`

- `0000 1001 1100 0110 1010 1111 0101 1000
  1010 1111 0101 1000 0000 1001 1100 0110
  1100 0110 1010 1111 0101 1000 0000 1001
  0101 1000 0000 1001 1100 0110 1010 1111`

**Review: Levels of Organization**

- SPARCstation 20

- Computer
  - SPARC Processor
  - Memory
  - Devices
  - Datapath
  - Control
  - Input
  - Output

**Summary: Computer System Components**

- Proc
- Caches
- Busses
- Memory
- Controllers
- I/O Devices: Disks, Displays, Keyboards, Networks

- All have interfaces & organizations

**Review: Summary from Last Lecture**

- All computers consist of five components
  - Processor: (1) datapath and (2) control
  - (3) Memory
  - (4) Input devices and (5) Output devices

- Not all “memory” are created equally
  - Cache: fast (expensive) memory are placed closer to the processor
  - Main memory: less expensive memory—we can have more

- Least amount of research (so far)
Integrated Circuits Costs --- manufacturing process

Click "how chips are made" on the website.

Integrated Circuits Costs --- formula

Die cost = \( \frac{\text{Cost per wafer}}{\text{Die cost per wafer}} \times \text{Yield} \)

Dies per wafer = \( \frac{\text{wafer area}}{\text{Die area}} \)

Die Yield = \( \frac{1}{1 + \left( \frac{\text{Defect per area}}{\text{Die area}} \right)^2} \)

Die Cost is goes roughly with the cube of the area.

Real World Examples

<table>
<thead>
<tr>
<th>Chip</th>
<th>Metal layers</th>
<th>Line width</th>
<th>Wafer cost</th>
<th>Defect /cm²</th>
<th>Area mm²</th>
<th>Dies per wafer</th>
<th>Yield</th>
<th>Die Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>386DX</td>
<td>2</td>
<td>0.90</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>3</td>
<td>0.80</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>4</td>
<td>0.80</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>3</td>
<td>0.70</td>
<td>$1500</td>
<td>1.2</td>
<td>234</td>
<td>53</td>
<td>19%</td>
<td>$149</td>
</tr>
<tr>
<td>DEC Alpha</td>
<td>3</td>
<td>0.70</td>
<td>$1700</td>
<td>1.6</td>
<td>256</td>
<td>48</td>
<td>13%</td>
<td>$272</td>
</tr>
<tr>
<td>SuperSPARC</td>
<td>3</td>
<td>0.70</td>
<td>$1500</td>
<td>1.5</td>
<td>296</td>
<td>40</td>
<td>9%</td>
<td>$417</td>
</tr>
</tbody>
</table>


IC cost = Die cost + Testing cost + Packaging cost

Final test yield

Packaging Cost: depends on pins, heat dissipation

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

Other Costs

CMOS improvements

Die size 2X / 3 years; Line widths halve / 7 years

<table>
<thead>
<tr>
<th>Capacity</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>2x in 3 years</td>
</tr>
<tr>
<td>DRAM</td>
<td>4x in 3 years</td>
</tr>
<tr>
<td>disk</td>
<td>4x in 3 years</td>
</tr>
</tbody>
</table>

Processor Performance


IBM Power 2/790
DEC AXP 3100
IBM RS/6000/710
Sun-4/260
MIPS M/120
MIPS M2000
HP 9000/750
IBM Power 2/600
MIPS R10000
DEC 3170A
IBM 4300
IBM 5400/5450/5470
IBM 5900/790
The bottom line: Performance (and cost)

- Different measurements lead to different results
- Time to do the task (Execution Time)
  - execution time, response time, latency
- Tasks per day, hour, week, sec, ns... (Performance)
  - throughput, bandwidth
- Cost renders the measurement more complex
- The bottom-line performance measurement is CPU execution time.

<table>
<thead>
<tr>
<th>Airplane</th>
<th>DC to Paris</th>
<th>Range</th>
<th>Speed (m.p.h.)</th>
<th>Passengers</th>
<th>Throughput (p x m.p.h.)</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 747</td>
<td>6.5 hours</td>
<td>4150</td>
<td>610</td>
<td>470</td>
<td>286,700</td>
<td>???</td>
</tr>
<tr>
<td>BAC/Sud Concorde</td>
<td>3 hours</td>
<td>4000</td>
<td>1350</td>
<td>132</td>
<td>178200</td>
<td>???</td>
</tr>
<tr>
<td>Douglas DC-8</td>
<td>7.3 hours</td>
<td>9720</td>
<td>544</td>
<td>146</td>
<td>75,424</td>
<td>???</td>
</tr>
</tbody>
</table>

"X is n times faster than Y" means

\[
\text{ExTime}(Y) \quad \text{Performance}(X) \\
\frac{}{\text{ExTime}(X) \quad \text{Performance}(Y)} \\
\text{Time of Concorde vs. Boeing 747?} \\
\text{Throughput of Boeing 747 vs. Concorde?}
\]

Metrics of performance

- Application
  - Answers per month
  - Operations per second
- Programming Language
- Compiler
- (millions) of instructions per second – MIPS
- (millions) of (F.P.) operations per second – MFLOP/s
- Datapath
- Control
- Function Units
- Transistors
- Wires
- Pins
- Cycles per second (clock rate)
- Megabytes per second

Aspects of CPU Performance

<table>
<thead>
<tr>
<th>Aspects of CPU Performance</th>
<th>Program</th>
<th>Compiler</th>
<th>Instr. Set Arch.</th>
<th>Organization</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>instr. count</td>
<td>CPI</td>
<td>clock rate</td>
<td>CPI</td>
<td>clock rate</td>
<td>CPI</td>
</tr>
</tbody>
</table>

Relating Processor Metrics (pp60-66)

- CPU execution time = CPU clock cycles/pgm ÷ clock rate
- or CPU execution time = CPU clock cycles/pgm X clock cycle time
- CPU clock cycles/pgm = Instructions/pgm X avg. clock cycles per instr.
- or CPI = CPU clock cycles/pgm ÷ Instructions/pgm
- CPI tells us something about the Instruction Set Architecture, the Implementation of that architecture, and the program measured
Organizational Trade-offs

3 factors:
Where are they?
How are they related?

CPI – How to compute?

"Average cycles per instruction"

\[
\text{CPI} = \frac{(\text{CPU Time} \times \text{Clock Rate})}{\text{Instruction Count}} = \frac{\text{Clock Cycles}}{\text{Instruction Count}}
\]

"CPU clock cycles summed up"

\[
\text{CPU time} = \sum_{i=1}^{n} \text{CPI}_i \times C_i
\]

"Instruction frequency"

\[
\text{CPI} = \frac{\sum_{i=1}^{n} \text{CPI}_i \times F_i}{\text{Instruction Count}}
\]

Invest Resources where time is Spent!

Example (page 60)

Our favorite program runs in 10 sec on machine A, which has a 400MHz clock. We are trying to design a machine B with faster clock rate so as to reduce the execution time to 6 sec.

The increase of clock rate will affect the rest of the CPU design, causing B to require 1.2 times as many clock cycles as machine A for this program. What clock rate should be?

Answer:

\[
\text{CPU time A} = \frac{\text{CPU clock cycle A}}{\text{clock rate A}}
\]

\[
= \frac{10 \text{ sec} \times 400 \times 10^6}{1.2 \times 400 \times 10^6} = 6 \text{ sec}
\]

Clock rate B = CPU clock cycle B / CPU time B

\[
= \frac{1.2 \times 400 \times 10^6}{6} = 800 \text{ MHz}
\]

B’s clock is 2 times as fast as A’s clock, but B is not 2 times as fast as A.

Example (page 62)

Base Machine (Reg / Reg) and Instruction frequencies in the execution of a program:

<table>
<thead>
<tr>
<th>Op</th>
<th>Freq</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>43%</td>
<td>1</td>
</tr>
<tr>
<td>Load</td>
<td>21%</td>
<td>2</td>
</tr>
<tr>
<td>Store</td>
<td>12%</td>
<td>2</td>
</tr>
<tr>
<td>Branch</td>
<td>24%</td>
<td>2</td>
</tr>
</tbody>
</table>

Question: What is the average CPI of the machine?

\[
\text{CPI} = 1 \times 43\% + 2 \times 21\% + 2 \times 12\% + 2 \times 24\% = 1.57
\]

Example (page 62)

Suppose we have two implementations of the same instruction set. Machine A has a clock cycle time of 10 ns and an average CPI of 2.0 for some program.

Machine B has a clock cycle time of 20 ns and an average CPI of 1.2 for the same program.

Which is faster? And by how much?

Let I denote the number of instructions of the program

\[
\text{CPU time A} = I \times 2.0 \times 10 \text{ ns} = 20 \text{ I}
\]

\[
\text{CPU time B} = I \times 1.2 \times 20 \text{ ns} = 24 \text{ I}
\]

Machine A is 1.2 faster than B.

Again we see 3 factors are related!
### Example (pp65-66)

ISA has 3 kinds of instructions:

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

One program has 2 code sequences:

<table>
<thead>
<tr>
<th>Code Sequence</th>
<th>Instruction counts for instruction class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A 2 B 3 C 1 2</td>
</tr>
<tr>
<td>2</td>
<td>4 1 2</td>
</tr>
</tbody>
</table>

Which code sequence has more instructions? Which will be faster? What is the CPI for each sequence?

- S.1 has 5 instructions; S.2 has 6.
- S.1 needs $2x1+1x2+2x3=9$ cycles; S.2 needs $4x1+1x2+1x3=9$ cycles.
- S.1 has CPI=10/5=2; S.2 has CPI=9/6=1.5

### Marketing Metrics

- MIPS = Instruction Count / Time * 10^6
- = Clock Rate / CPI * 10^6
- Million Instructions Per Seconds
- machines with different instruction sets?
- programs with different instruction sets?
- dynamic frequency of instructions
- Peak MIPS: impractical
- uncorrelated with performance. (see the next example)

### Why Do Benchmarks? Or How to evaluate an athlete?

- Triathlon (3 sports)
  - swimming
  - bicycling
  - running

- Pentathlon (5 sports)
  - sprinting
  - hurdles
  - long jumping
  - discus
  - javelin

- Decathlon (10 sports)
  - 100-meter
  - 400-meter
  - 1,500-meter runs
  - 110-meter high hurdle
  - discus
  - javelin throws
  - shot-put
  - pole vault
  - high jump
  - long jump

### Programs to Evaluate Processor Performance (pp86-87)

- (Toy) Benchmarks
  - Small but easy to compile and run on simulators, convenient in early designing stage of a new machine. No compiler for novel machines.
  - 10-100 line
  - e.g.; sieve, puzzle, quicksort

- Synthetic Benchmarks
  - attempt to use a single benchmark to match average frequencies of real workloads or a set of benchmarks
  - e.g., Whetstone(Algol60/Fortran), Dhrystone(Ada/C)

- Kernels
  - Time critical excerpts of Real programs
  - Popular in scientific computing to illuminate performance of individual features of a machine.
  - e.g., Livermore loops(21 loops), Linpack(linear algebra)

- Real programs
  - e.g., gcc, spice
Successful Benchmark: SPEC (pp87-89)

° 1987 RISC industry mired in “bench marketing”: (“That is 8 MIPS machine, but they claim 10 MIPS!”)
° EETimes (http://www.eet.com/) + 5 companies band together to perform Systems Performance Evaluation Committee (SPEC) in 1988: Sun, MIPS, HP, Apollo, DEC
° Create standard list of programs, inputs, reporting: some real programs, includes OS calls, some I/O

SPEC first round

° First round 1989; 10 programs = 4 for integer + 6 for FP, single number to summarize performance
° One program (matrix300): 99% of time in single line of code
° New front-end compiler could improve dramatically (Fig. 2.3)

SPEC Evolution

° Second round; SPECInt92 (6 integer programs) and SPECfp92 (14 floating point programs)
  Compiler Flags unlimited. March 93 of DEC 4000 Model 610:
  spice:
  unix.c:/def=(sysv, has_bcopy, “bcopy(a,b,c)=memcpy(b,a,c)"
  wave5:/ali=(all, dcom=nat)/ag=a/ur=4/ur=200
  nasa7:/nosecu/ag=a/ur=4/ur=200/loc=bias
° Add SPECbase: don’t allow program-specific optimization flags.
° Third round; 1995; new set of programs(8 int + 10fp) (fig. 2.6, p72)
  “benchmarks useful for 3 years”
  • Base machine is changed from VAX-11/780 to Sun SPARC 10/40

How to Summarize Results?

Program 1: 1 sec on machine A, 10 sec on machine B
Program 2: 1000 sec 100 sec

What are your conclusions?

• A is 10 times faster than B for program1.
• B is 10 times faster than A for Program2.
• Total execution time: a consistent summary measure
  • B is 1001/110=9.1 times faster than A.
• Workload: need to consider the percentage/frequency of each program in the total job.

How to Summarize Performance

° Suppose n programs have execution time \( t_j \) where \( j = 1, 2, \ldots, n \).
° Suppose the workload is \( \sum w_j t_j \) where \( \sum w_j = 1 \).
° Arithmetic Mean \( AM(t) = \frac{1}{n} \sum_{j=1}^{n} t_j \)
° Weighted Arithmetic mean \( WAM(t) = \frac{\sum_{j=1}^{n} w_j t_j}{\sum_{j=1}^{n} w_j} \)
° Geometric Mean \( GM(t) = \left( \prod_{j=1}^{n} t_j \right)^{\frac{1}{n}} = \sqrt[n]{t_1 t_2 \cdots t_n} \)
° Weighted Geometric Mean \( WGM(t) = \left( \prod_{j=1}^{n} t_j^{w_j} \right)^{\frac{1}{\sum_{j=1}^{n} w_j}} \)
° Harmonic Mean \( HM(t) = \frac{n}{\sum_{j=1}^{n} \frac{1}{t_j}} \)
° Weighted Harmonic Mean \( WHM(t) = \frac{\sum_{j=1}^{n} w_j \left( \frac{1}{t_j} \right)}{\sum_{j=1}^{n} w_j} \)
Impact of Means on SPECmark89 for IBM 550

<table>
<thead>
<tr>
<th>Program</th>
<th>Before</th>
<th>After</th>
<th>Before</th>
<th>After</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc</td>
<td>30</td>
<td>29</td>
<td>49</td>
<td>51</td>
<td>8.91</td>
<td>9.22</td>
</tr>
<tr>
<td>espresso</td>
<td>35</td>
<td>34</td>
<td>65</td>
<td>67</td>
<td>7.64</td>
<td>7.86</td>
</tr>
<tr>
<td>spice</td>
<td>47</td>
<td>47</td>
<td>510</td>
<td>510</td>
<td>5.69</td>
<td>5.69</td>
</tr>
<tr>
<td>doduc</td>
<td>46</td>
<td>49</td>
<td>41</td>
<td>38</td>
<td>5.81</td>
<td>5.45</td>
</tr>
<tr>
<td>nasa7</td>
<td>78</td>
<td>144</td>
<td>258</td>
<td>140</td>
<td>3.43</td>
<td>1.86</td>
</tr>
<tr>
<td>li</td>
<td>34</td>
<td>34</td>
<td>183</td>
<td>183</td>
<td>7.86</td>
<td>7.86</td>
</tr>
<tr>
<td>eqntott</td>
<td>40</td>
<td>40</td>
<td>28</td>
<td>28</td>
<td>6.68</td>
<td>6.68</td>
</tr>
<tr>
<td>matrix300</td>
<td>78</td>
<td>730</td>
<td>58</td>
<td>6</td>
<td>3.43</td>
<td>0.37</td>
</tr>
<tr>
<td>fpppp</td>
<td>90</td>
<td>87</td>
<td>34</td>
<td>35</td>
<td>2.97</td>
<td>3.07</td>
</tr>
<tr>
<td>tomcatv</td>
<td>133</td>
<td>138</td>
<td>20</td>
<td>19</td>
<td>2.01</td>
<td>1.94</td>
</tr>
<tr>
<td>Mean</td>
<td>54</td>
<td>72</td>
<td>124</td>
<td>108</td>
<td>54.42</td>
<td>49.99</td>
</tr>
<tr>
<td>Geometric Mean</td>
<td>1.33</td>
<td>Ratio</td>
<td>1.16</td>
<td>Ratio</td>
<td>1.09</td>
<td></td>
</tr>
</tbody>
</table>

Example (part 1)

<table>
<thead>
<tr>
<th>Program</th>
<th>Time on A</th>
<th>Time on B</th>
<th>Normalized to A</th>
<th>Normalized to B</th>
</tr>
</thead>
<tbody>
<tr>
<td>program1</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>program2</td>
<td>1000</td>
<td>100</td>
<td>0.1</td>
<td>10</td>
</tr>
</tbody>
</table>

- The difficulty arises from the use of arithmetic mean of normalized time.
- Geometric mean is independent of which data series we use for normalized time because 
  \[
  \frac{GM(y_i)}{GM(y_j)} = GM\left(\frac{y_i}{y_j}\right)
  \]

Amdahl's Law

Suppose that enhancement E accelerates a fraction F of the task by a factor S and the remainder of the task is unaffected then,

\[
\text{Speedup(E)} = \frac{\text{ExTime(without E)}}{\text{ExTime(with E)} + \frac{F}{S} \times \text{ExTime(without E)}}
\]

Example:

Suppose a person wants to travel from city A to city B by city C. The routes from A to C are in mountains and the routes from C to B are in desert. The distances from A to C, and from C to B are 80 miles and 200 miles, respectively.

From A to C, walk at speed of 4 mph
From C to B, walk or drive (at speed of 100 mph)

Question: How long will it take for the entire trip
How much faster from A to B by a car as opposed to walk

Example:

Suppose an enhancement runs 10 times faster than the original machine, but is only usable 40% of the time.

Question: what is the overall speedup?

Answer: Fraction_enhance = 0.4

\[
\text{Speedup_enhanced} = 10
\]

\[
\text{Speedup_overall} = 1/(0.6+0.4/10) = 1.56
\]

Cost Summary

- Integrated circuits driving computer industry
- Die costs goes up with the cube of die area
**Performance Evaluation Summary**

- Time is the measure of computer performance!
- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- If not good benchmarks and summary, then choice between improving product for real programs vs. improving product to get more sales=> sales almost always wins.
- Remember Amdahl's Law: Speedup is limited by unimproved part of program.

**Homework, due Feb. 3 class time (Monday)**

Question 1.1 through 1.26
Question 2.1 through 2.4
Question 2.10 through 2.13
Question 2.18 through 2.20
Question 2.41