Trends in High-Performance Computer Architecture

David J. Lilja

Department of Electrical Engineering
Center for Parallel Computing
University of Minnesota
Minneapolis

E-mail: lilja@ee.umn.edu
Phone: 625-5007
FAX: 625-4583
Trends and Predictions

**Trend, n.**

1: direction of movement: *FLOW*
2 a: a prevailing tendency or inclination: *DRIFT*
2 b: a general movement: *SWING*
2 c: a current style or preference: *VOGUE*
2 d: a line of development: *APPROACH*

*Webster’s Dictionary*

It is very difficult to make an accurate prediction, especially about the future.

*Niels Bohr*
Historical Trends and Perspective

- pre-WW II: Mechanical calculating machines
- WW II - 50’s: Technology improvement
  relays → vacuum tubes
  high-level languages
- 60’s: Miniaturization/packaging
  transistors
  integrated circuits
- 70’s: Semantic gap
  complex instruction sets
  language support in hardware
  microcoding
- 80’s: Keep It Simple, Stupid
  RISC vs CISC debate
  shift complexity to software
- 90’s: What to do with all of these transistors?
  large on-chip caches
  prefetching hardware
  speculative execution
  special-purpose instructions
  multiple processors on-a-chip
What is Computer Architecture?

- It has nothing to do with buildings.
- Goals of a computer designer
  - control complexity
  - maximize performance
  - minimize cost?
- Use levels of abstraction
  
  silicon and metal
  → transistors
  → gates
  → flip-flops
  → registers
  → functional units
  → processors
  → systems

- Architecture
  - defines interface between higher levels and software
  - requires close interaction between
    * HW designer
    * SW designer
Performance Metrics

- **System throughput**
  - work per unit time → rate
  - used by system managers

- **Execution time**
  - how long to execute your application
  - used by system designers and users

\[
T_{exec} = n \text{ instrs} \times \frac{\# \text{ cycles}}{\# \text{ instrs}} \times \frac{\text{seconds}}{\text{cycle}}
\]

\[
= n \times \text{CPI} \times T_{\text{clock}}
\]

- Example

\[
T_{exec} = 900 \ M \text{ instrs} \times \frac{1.8 \text{ cycles}}{\text{instr}} \times \frac{10 \text{ ns}}{\text{cycle}} = 16.2 \text{ sec}
\]
Improving Performance

\[ T_{exec} = T_{clock} \times n \times CPI \]

- Improve clock rate, \( T_{clock} \)
- Reduce total number of instructions executed, \( n \)
- Reduce average number of cycles per instruction, \( CPI \)
1) Improving the Clock Rate

- Use faster technology
  - BiCMOS, ECL, etc
  - smaller features to reduce propagation delay
- Pipelining
  - reduce the amount of work per clock cycle

- Performance improvement
  - reduces $T_{\text{clock}}$
  - overlaps execution of instructions
    $\rightarrow$ parallelism
- Maximum speedup $\leq$ pipeline depth
Cost of Pipelining

- More hardware
  - need registers between each pipe segment
- Data hazards
  - data needed by instr $i+x$ from instr $i$ has not been calculated
- Branch hazards
  - began executing instrs from wrong branch path
Branch Penalty

- Instruction $i+2$ branches to instr $j$
  - branch resolved in stage 5

<table>
<thead>
<tr>
<th>cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$i$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>$i+1$</td>
<td>$i$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>$i+2$</td>
<td>$i+1$</td>
<td>$i$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>$i+3$</td>
<td>$i+2$</td>
<td>$i+1$</td>
<td>$i$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>$i+4$</td>
<td>$i+3$</td>
<td>$i+2$</td>
<td>$i+1$</td>
<td>$i$</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>$i+5$</td>
<td>$i+4$</td>
<td>$i+3$</td>
<td>$i+2$</td>
<td>$i+1$</td>
<td>$i$</td>
</tr>
<tr>
<td>7</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
<td>$i+2$</td>
<td>$i+1$</td>
</tr>
<tr>
<td>8</td>
<td>$j$</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
<td>$i+2$</td>
</tr>
<tr>
<td>9</td>
<td>$j+1$</td>
<td>$j$</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
</tr>
<tr>
<td>10</td>
<td>$j+2$</td>
<td>$j+1$</td>
<td>$j$</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
</tr>
<tr>
<td>11</td>
<td>$j+3$</td>
<td>$j+2$</td>
<td>$j+1$</td>
<td>$j$</td>
<td>$X$</td>
<td>$X$</td>
</tr>
<tr>
<td>12</td>
<td>$j+4$</td>
<td>$j+3$</td>
<td>$j+2$</td>
<td>$j+1$</td>
<td>$j$</td>
<td>$X$</td>
</tr>
<tr>
<td>13</td>
<td>$j+5$</td>
<td>$j+4$</td>
<td>$j+3$</td>
<td>$j+2$</td>
<td>$j+1$</td>
<td>$j$</td>
</tr>
<tr>
<td>14</td>
<td>$j+6$</td>
<td>$j+5$</td>
<td>$j+4$</td>
<td>$j+3$</td>
<td>$j+2$</td>
<td>$j+1$</td>
</tr>
</tbody>
</table>

- Data hazards produce similar pipeline *bubbles*
  - $i+3$ needs data generated by $i+2$
  - $i+3$ stalled until $i+2$ in stage 5

- Solutions to hazards
  - data bypassing
  - instruction reordering
  - branch prediction
  - delayed branch
2) Reduce Number of Instructions Executed

\[ T_{\text{exec}} = T_{\text{clock}} \times n \times \text{CPI} \]

- **CISC -- Complex Instruction Set Computer**
  - powerful instrs to reduce instr count
  - complex addressing modes
  - complex loop, move instructions
  - But may increase cycle time, \( T_{\text{clock}} \)

- **RISC -- Reduced Instruction Set Computer**
  - small, simple instruction set
  - simpler implementation
  - \( \rightarrow \) faster clock
  - But must execute more instructions for same work
**RISC vs CISC Debate**

- Pentium, Pentium-Pro, Motorola 68xxx *vs* MIPS (SGI), PowerPC, Cray

<table>
<thead>
<tr>
<th></th>
<th>$T_{clock}$</th>
<th>$n$ (instrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RISC</td>
<td>↓</td>
<td>↑</td>
</tr>
<tr>
<td>CISC</td>
<td>↑</td>
<td>↓</td>
</tr>
</tbody>
</table>

- RISC tends to win
  - simple instructions $\rightarrow$ easier pipelining
- But trade-off is technology dependent
- Market considerations determine actual winner
- Special purpose instructions
  - HP PA-7100LC has special multimedia instructions
    - reduce total instruction count for MPEG encode/decode
    - exploit pixels $<$ full word width
3) Reduce Average Cycles per Instruction

\[ T_{exec} = T_{\text{clock}} \times n \times \text{CPI} \]

- Decreasing CPI \equiv increasing \text{ Instructions Per Cycle (IPC)}

\[ T_{exec} = T_{\text{clock}} \times n \times \frac{1}{\text{IPC}} \]

- CPI < 1 \rightarrow parallelism
  - instruction-level
  - processor-level
Superscalar Processors

- Almost all microprocessors today use superscalar
- Use hardware to check for instruction dependences
- Issue multiple instructions simultaneously
- Instruction window

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>r1, r2, r3</td>
</tr>
<tr>
<td>sub</td>
<td>r3, r4, r5</td>
</tr>
<tr>
<td>mult</td>
<td>r6, r7, r6</td>
</tr>
<tr>
<td>store</td>
<td>r6, y</td>
</tr>
<tr>
<td>cmp</td>
<td>r6, #5</td>
</tr>
<tr>
<td>load</td>
<td>x, r8</td>
</tr>
</tbody>
</table>

- DEC Alpha 21164
VLIW -- Very Long Instruction Word

- Rely on compiler to detect parallel instructions
  - pack independent instructions into one long instruction
  - ≈ microcode compaction
- Simplifies hardware compared to superscalar
- But
  - compile-time information is incomplete
    conservatively assume not parallel
  - code explosion
  - execution stalls
Amdahl’s Law

- Limits maximum performance improvement

\[ \text{Perf Improvement} = \frac{\text{Part affected}}{\text{Improvement factor}} + \text{Part unaffected} \]

- Travel from Minneapolis to Chicago

  By car

  \[ \frac{420 \text{ miles}}{60 \text{ mi/hr}} = 7 \text{ hr} \]

  By taxi + plane + taxi

  \[ \frac{30 \text{ miles}}{20 \text{ mi/hr}} + \frac{360 \text{ miles}}{360 \text{ mi/hr}} + \frac{30 \text{ miles}}{20 \text{ mi/hr}} = 4 \text{ hr} \]

  ⇒ Plane is 6× faster, but net improvement = 1.75×

- Limited by slowest component

- Corollary: Focus on part that produces biggest bang per buck.

- Corollary: Make the most common case fast.
Processor-Memory Speed Gap

- "But a processor doth not a system make."

- Relative performance improvement of CPU and DRAM.
  - CPU ≈ 25%- 50% per year.
  - DRAM ≈ 7% per year.
Memory Delay is the Killer

- Speed ratio of memory to CPU → 100×
- $T_{exec} = T_{CPU} + T_{memory}$
- Faster processors reduce only $T_{CPU}$
- Memory instructions ≈ 20% of instructions executed
- Amdahl’s Law
  - If $T_{CPU} \rightarrow 0$, System speedup $\leq 5\times$
Reducing Memory Delay

- Amortize delay over many references
  - exploit *locality* of references
  - caches
  - vector operations $\approx$ pipelining memory

- Hide the delay
  - data prefetching
  - context-switching with multiple independent threads
I/O is the Killer

- $T_{exec} = T_{CPU} + T_{memory} + T_{I/O}$
- I/O delay worse than memory
  - video-on-demand
  - multimedia
  - network computing
- Merging of *intrasystem* and *intersystem* communication
  - FDDI, ATM, Fibre Channel, ISDN, etc.
    - WAN: wide-area network
    - LAN: local-area network
    - PAN: processor-area network
  - network-connected I/O devices
## Contemporary Microprocessors

<table>
<thead>
<tr>
<th></th>
<th>DEC Alpha 21164</th>
<th>Sun Ultra-SPARC-1</th>
<th>SGI MIPS R10000</th>
<th>HP PA-8000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avail</td>
<td>1Q95</td>
<td>1Q96</td>
<td>1Q96</td>
<td>4Q95</td>
</tr>
<tr>
<td>Tech</td>
<td>0.5μm</td>
<td>0.5μm</td>
<td>0.35μm</td>
<td>0.55μm</td>
</tr>
<tr>
<td>Clock</td>
<td>300 MHz</td>
<td>182 MHz</td>
<td>200 MHz</td>
<td>200 MHz</td>
</tr>
<tr>
<td>Trans</td>
<td>9.3 M</td>
<td>5.2 M</td>
<td>6.8 M</td>
<td></td>
</tr>
<tr>
<td>S’scalar</td>
<td>4-way</td>
<td>4-way</td>
<td>4-way</td>
<td>4-way</td>
</tr>
<tr>
<td>On-chip cache</td>
<td>8K I and D + 96K 2nd-level</td>
<td>16K I and D</td>
<td>32K I and D</td>
<td>NONE</td>
</tr>
<tr>
<td>SPECint92</td>
<td>345</td>
<td>260</td>
<td>300</td>
<td>360</td>
</tr>
<tr>
<td>SPECfp92</td>
<td>505</td>
<td>410</td>
<td>600</td>
<td>550</td>
</tr>
<tr>
<td>Power</td>
<td>50W</td>
<td>25W</td>
<td>30W</td>
<td></td>
</tr>
</tbody>
</table>
Trends in Clock Cycle Times

- Cray vs microprocessors
- Increase IPC
  - fine-grained parallelism
- Increase number of processors
  - coarse-grained parallelism
Data- vs Control-Parallelism

- Data-parallel
  - Single Instruction, Multiple Data (SIMD)

- Control-parallel
  - Multiple Instruction, Multiple Data (MIMD)
Multiprocessor Systems

- Parallelism is commonplace
  - desktop multiprocessors
  - networks of workstations
  - superscalar
- Applications
  - relational database servers
  - decision support
  - data mining
  - transaction processing
  - scientific/engineering
    - crash simulation
    - weather modeling
    - oceanography
    - radar
  - medical imaging
- Manufacturers
  - Sun Microsystems, Silicon Graphics, Intel, Hewlett-Packard, Compaq, Cray, Convex, IBM, Tandem, Pyramid, ...
Multiprocessor Design Issues

- Interconnection network
  - latency and bandwidth
  - topology
- Memory delay
  - network delay
  - cache coherence problem
- Task granularity
  - small tasks → more parallelism, but more synch
  - large tasks → less synch, but less parallelism
- Programming complexity
  - shared-memory
  - message-passing
  - automatic compiler parallelization
Improving Computer Performance: Summary

\[ T_{\text{exec}} = T_{\text{clock}} \times n \times \frac{1}{IPC} + T_{\text{memory}} + T_{I/O} \]

1) Improve the clock rate, \( T_{\text{clock}} \)
   - faster technology
   - pipelining

2) Reduce the total number of instructions executed, \( n \)
   - CISC vs RISC debate
   - specialized instructions
     e.g. multimedia support

3) Increase the parallelism, \( IPC \)
   - superscalar
   - VLIW
   - multiple processors
   - speculative execution

→ But, memory delay is **the** killer!

→ But, I/O delay is **the** killer!
Parting Thoughts

- “We haven’t much money so we must use our brains.”
  *Lord Rutherford, Cavendish Laboratory*
  - technology driven by low-cost, high-volume devices
- “Even if you are on the right track, you’ll get run over if you just sit there.”
  *Will Rogers*
  - the pace of technology is brutal
- “A distributed system is one in which I cannot get something done because a machine I’ve never heard of is down.”
  *Leslie Lamport*
  - the processor is becoming secondary to the network
- “There are 3 types of mathematicians. Those who can count, and those who cannot.”
  *Robert Arthur*
  - parallel software is hard
Parting Thoughts

- “You know you have achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away.”
  
  *Antoine de Saint Exupery*

- “Everything should be made as simple as possible, but no simpler.”
  
  *Albert Einstein*

- High-performance requires elegant design