CS 271
Computer Architecture & Assembly Language

Lecture 20
Parallelism
Research & Innovation
Closing Remarks
3/10/22, Thursday
Lecture Topics:

• Parallelism
• Research & Innovation
• Closing remarks
Parallelism
Hardware Parallelism (overview)

• Instruction-level parallelism
  • Pipeline
  • Cache

• Processor-level parallelism
  • Multiprocessor (multiple CPUs, common memory)
  • Multicomputer (multiple CPUs, each with own memory)
# Pipelining

<table>
<thead>
<tr>
<th></th>
<th>U-1</th>
<th>U-2</th>
<th>U-3</th>
<th>U-4</th>
<th>U-5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Instruction Fetch</td>
<td>Instruction Decode</td>
<td>Operand Fetch</td>
<td>Instruction Execute</td>
<td>Operand Store</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>U-2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>U-3</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>U-4</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>U-5</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>T-1</td>
</tr>
<tr>
<td>T-2</td>
</tr>
<tr>
<td>T-3</td>
</tr>
<tr>
<td>T-4</td>
</tr>
<tr>
<td>T-5</td>
</tr>
<tr>
<td>T-6</td>
</tr>
<tr>
<td>T-7</td>
</tr>
<tr>
<td>T-8</td>
</tr>
<tr>
<td>T-9</td>
</tr>
<tr>
<td>T-10</td>
</tr>
</tbody>
</table>

A 5-stage pipeline, showing which unit is processing each instruction number in each clock cycle.
Instruction Caching

• Hardware provides area for multiple instructions in the CPU
  • Reduces number of memory accesses
  • Instructions are available for immediate execution
  • Might cause problems with decision, repetition, and procedure structures in programs
Multiprocessor Parallelism (shared memory)
Multicomputer Parallelism (distributed memory)
Comparisons

• Multiprocessor
  • Difficult to build
  • Relatively easy to program

• Multicomputer
  • Easy to build (given networking technology)
  • Extremely difficult to program

• Hybrid systems
  • Cloud computing
Interconnection Network

- Communication among processors
- Multiprocessor system
  - Communication through circuits/memory
- Multicomputer system
  - Communication through networking technologies
  - Packets (data, source/destination information, etc.)
  - Links, switches, interfaces, etc.
Software Parallelism

- Parallelizability of algorithms
  - Number of processors
  - Trade-offs and efficiency
  - Sequential/parallel parts

- Amdahl’s Law
  - \( n \) = number processors
  - \( f \) = fraction of code that is sequential
  - \( t \) = time to process entire algorithm sequentially (one processor)

\[
\text{speedup} = \frac{n}{1 + (n-1)f}
\]

- Note: total execution time is

\[
fT + \frac{(1-f)T}{n}
\]
Software Parallelism

(a) A program has a sequential part and a parallelizable part

(b) Effect of running the parallelizable part on a multi-processor architecture
Software Parallelism

Example:
An algorithm takes 10 seconds to execute on a single 2.4G processor. 40% of the algorithm is sequential. Assuming zero latency and perfect parallelism in the remaining code, how long should the algorithm take on a 16 X 2.4G processor parallel machine?

\[ speedup = \frac{n}{1+(n-1)f} = \frac{16}{1+.4\times15} = \frac{16}{7} \]

Therefore the expected time is
\[ \frac{T}{speedup} \]
10 / (16 / 7) = 4.375 seconds
Another way: (0.4 * 10) + (0.6 * 10) / 16
sequential + parallel
**Software Parallelism**

\[ \text{speedup} = \frac{n}{1 + (n-1)f} \]

- Assuming perfect scalability, what are the implication on Amdahl’s law when \( n \to \infty \)?

- \( \text{speedup} \to 1/f \) (assuming \( f \neq 0 \))

- Therefore, if \( f = 0.4 \), parallelism can never make the program run more than 2.5 times as fast
More Parallelism

• As a Computer Scientist, you will encounter parallel systems, parallel algorithms, parallel programming ... everywhere.

• It is important to understand the fundamentals of computer hardware in order to make the best uses of parallelism
Research & Innovation
Parallel Computing Performance Depends on Hardware/Software

• Hardware
  • CPU speed of individual processors
  • I/O speed of individual processors
  • Interconnection network
  • Scalability

• Software
  • Parallelizability of algorithms
  • Application programming languages
  • Operating systems
  • Parallel system libraries
Hardware Parallelism

• CPU and I/O speed:
  • Same factors as for single-processor machines ... plus:

• Interconnection network
  • **Latency** (wait time):
    • Distance
    • Collision / collision resolution
  • **Bandwidth** (bps)
    • Bus limitations
    • CPU and I/O limitations

• Scalability
  • Adding more processors affects latency and bandwidth
Software Parallelism

- Parallel system libraries
  - Precompiled functions designed for multiprocessing (e.g., matrix transformations)
  - Functions for control of communication (e.g., background printing)

- Application programming languages
  - Built-in functions for creating child processes, threads, parallel looping, etc.
  - Mostly imperative (e.g., C)

- Operating systems
Application of Parallelism

• Multi-user systems
  • Networks
  • Internet

• Speed up single processes
  • Chess example
  • Expert systems
  • Other AI applications
Research in Parallelism

• Parallelism is an extremely hot research area
  • Especially in parallel software systems, parallel algorithms, etc.
  • Knowledge of parallel architectures is useful.
Innovations

• ExtremeTech
  • Learn about the “bleeding edge”

• What’s going on with embedding computing?
  • Integration / Specialization
Be Confident...

Now you are able to...

• Access and interpret binary data stored in memory.

• Illustrate the Instruction Execution Cycle.

• Create and analyze well-modularized assembly language programs utilizing decision, repetition, and procedure structures.

• Utilize a debugger to identify and correct bugs in assembly language programs.

• Illustrate the system stack as it is used for procedure calls and parameter passing.

• Understand the primary components of a modern computer architecture, and explain their function.
Final Remarks...

• Thank you so much for your commitment to this course

• Future improvements?
  • MyOSU → Student Records →

• ULA position
  • Contact me! And apply through: https://jobs.oregonstate.edu/postings/103887
Final Remarks...

• Submit all your work by the deadline
  • Weekly Summary 10
  • Quiz 4
  • Final Project

• Grade disputation:
  • By 3/20 6pm