Review

- Parallelism is one of the great ideas
- Request Level Parallelism
- Data Level Parallelism
  - At the disk/server level – scale out to solve bigger problems on more data
    - Map Reduce/Hadoop
  - At the memory level – today’s topic

New-School Machine Structures (It’s a bit more complicated!)

- Parallel Requests
  - Assigned to computer
    - e.g., Search “Katz”
- Parallel Threads
  - Assigned to core
    - e.g., Lookup, Ads
- Parallel Instructions
  - >1 instruction @ one time
    - e.g., 5 pipelined instructions
- Parallel Data
  - >1 data item @ one time
    - e.g., Add of 4 pairs of words
- Hardware description
  - All gates @ one time

Alternative Kinds of Parallelism: Hardware vs. Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Sequential</th>
<th>Concurrent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial</td>
<td>Matrix Code written in MATLAB running on an Intel Pentium 4</td>
<td>Windows Vista Operating System running on an Intel Pentium 4</td>
</tr>
<tr>
<td>Parallel</td>
<td>Matrix Code written in MATLAB running on an Intel Xeon e5345 (Clovertown)</td>
<td>Windows Vista Operating System running on an Intel Xeon e5345 (Clovertown)</td>
</tr>
</tbody>
</table>

- Concurrent software can also run on serial hardware
- Sequential software can also run on parallel hardware
- Our focus today is on sequential or concurrent software running on parallel hardware

Flynn Taxonomy

- In 2011, SIMD and MIMD most common parallel computers
- Most common parallel processing programming style: Single Program Multiple Data (“SPMD”)
  - Single program that runs on all processors of an MIMD
  - Cross-processor execution coordination through conditional expressions (thread parallelism - next lecture)
- SIMD (aka hw-level data parallelism): specialized function units, for handling lock-step calculations involving arrays
  - Scientific computing, signal processing, multimedia (audio/video)

Agenda

- Flynn Taxonomy
- DLP and SIMD
- Intel SSE (part 1)
Flynn Taxonomy: SISD
Single Instruction/Single Data Stream
- Sequential computer that exploits no parallelism in either the instruction or data streams.
- Examples of SISD architecture are traditional uniprocessor machines.

Flynn Taxonomy: MISD
Multiple Instruction/Single Data Stream
- Exploits multiple instruction streams against a single data stream for operations that can be naturally parallelized. E.g., certain kinds of array processors.
- No longer commonly encountered, mainly of historical interest only.

Flynn Taxonomy: SIMD
Single Instruction/Multiple Data Stream
- Computer that applies a single instruction stream to multiple data streams for operations that may be naturally parallelized.
- E.g., SIMD instruction extensions or Graphics Processing Unit (GPU).

SIMD Architectures
- Data parallelism: executing one operation on multiple data streams
- Example to provide context:
  - Multiplying a coefficient vector by a data vector (e.g., in filtering)
    \[ y[i] := c[i] \times x[i], \ 0 \leq i < n \]
- Sources of performance improvement:
  - One instruction is fetched & decoded for entire operation
  - Multiplications are known to be independent
  - Pipelining/concurrency in memory access as well

"Advanced Digital Media Boost"
- To improve performance, Intel’s SIMD instructions
  - Fetch one instruction, do the work of multiple instructions
  - MMX (MultiMedia eXtension, Pentium II processor family)
  - SSE (Streaming SIMD Extension, Pentium III and beyond)
Example: SIMD Array Processing

for each \( f \) in array
\[
 f = \sqrt{f}
\]
for each \( f \) in array
\{  
load \( f \) to the floating-point register  
calculate the square root  
write the result from the register to memory
\}
for each 4 members in array
\{  
load 4 members to the SSE register  
calculate 4 square roots in one operation  
write the result from the register to memory
\}

Fall 2011 - Lecture #19

SSE Instruction Categories for Multimedia Support

<table>
<thead>
<tr>
<th>Instruction category</th>
<th>Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsigned add/subtract</td>
<td>Eight 8-bit or Four 16-bit</td>
</tr>
<tr>
<td>Saturating add/subtract</td>
<td>Eight 8-bit or Four 16-bit</td>
</tr>
<tr>
<td>Max/min/minimum</td>
<td>Eight 8-bit or Four 16-bit</td>
</tr>
<tr>
<td>Average</td>
<td>Eight 8-bit or Four 16-bit</td>
</tr>
<tr>
<td>Shift right/left</td>
<td>Eight 8-bit or Four 16-bit</td>
</tr>
</tbody>
</table>

• SSE2+ supports wider data types to allow 16 x 8-bit and 8 x 16-bit operands

Example: Add Two Single Precision FP Vectors

Computation to be performed:
\[
\begin{align*}
\text{vec}_x.x &= v_1.x + v_2.x; \\
\text{vec}_x.y &= v_1.y + v_2.y; \\
\text{vec}_x.z &= v_1.z + v_2.z; \\
\text{vec}_x.w &= v_1.w + v_2.w; \\
\end{align*}
\]

Example: Intel Architecture SSE2+

128-Bit SIMD Data Types

<table>
<thead>
<tr>
<th>Fundamental 128-Bit Packed SIMD Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>127</td>
</tr>
<tr>
<td>127</td>
</tr>
<tr>
<td>127</td>
</tr>
</tbody>
</table>

• Note: In Intel Architecture (unlike MIPS) a word is 16 bits
  – Single precision FP: Double word (32 bits)
  – Double precision FP: Quad word (64 bits)

XMM Registers

• Architecture extended with eight 128-bit data registers: XMM registers
  – IA32 64-bit address architecture: available as 16 64-bit registers (XMM8 – XMM15)
  – E.g., 128-bit packed single-precision floating-point data type (doublewords), allows four single-precision operations to be performed simultaneously

Example: SIMD Array Processing

for each \( f \) in array
\[
 f = \sqrt{f}
\]
for each \( f \) in array
\{  
load \( f \) to the floating-point register  
calculate the square root  
write the result from the register to memory
\}
for each 4 members in array
\{  
load 4 members to the SSE register  
calculate 4 square roots in one operation  
write the result from the register to memory
\}

Fall 2011 - Lecture #19

Intel Architecture SSE2+

128-Bit SIMD Data Types

<table>
<thead>
<tr>
<th>Fundamental 128-Bit Packed SIMD Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>127</td>
</tr>
<tr>
<td>127</td>
</tr>
<tr>
<td>127</td>
</tr>
</tbody>
</table>

• Note: In Intel Architecture (unlike MIPS) a word is 16 bits
  – Single precision FP: Double word (32 bits)
  – Double precision FP: Quad word (64 bits)
Floating point numbers \((fn)\) and \((x(n))\) in `src1` and `src2`; \(p\) in `dest`;
C implementation for \(N = 4\) (128 bits):

\[
\text{for } (\text{int } i = 0; i < 4; i++) \\
\quad \text{\hspace{1em}} p = p + \text{src}\{1\} * \text{src}\{2\};
\]

- SSE2 instructions for the inner loop:
  \[
  \text{mulps } \%\text{xmm}1, \%\text{ xmm}2;
  \text{faddps } \%\text{ xmm}0, \%\text{ xmm}2;
  \]

- Number regular instructions executed: 2 SSE2 instructions vs. 8 x86

- SSE5 instruction accomplishes same in one instruction:
  \[
  \text{fmaddps } \%\text{ xmm}0, \%\text{ xmm}1, \%\text{ xmm}2;
  \]

- Number regular instructions executed: 1 SSE5 instruction vs. 8 x86