The P5 Floating-Point Unit

Dror Avnon

Intel Corporation

Agenda

- Design Goals
- Micro-Architecture Overview
- Register-Stack Manipulation
- Transcendental Functions
- Compiler Optimization
- Summary
Design Goals

- Architectural Compatibility
  - Full Compatibility with Intel486™ CPU
  - IEEE Standard 754
- High Performance
  - 4-10 Times Intel486™ DX 33MHz CPU
    4-5 on SCALAR
    10-10 on VECTORIZABLE CODE

Micro-Architecture Overview

Floating Point Pipeline

- Three Dedicated Arithmetic Units
- Eight Stage Pipeline

Integer Pipe: \[ \text{PF} \ D1 \ D2 \ E \ WB \]

FP Pipe: \[ \text{PF} \ D1 \ D2 \ E \ X1 \ X2 \ \text{WFER} \]

- Three Execution Stages
Micro-Architecture Overview

Floating Point Pipeline Characteristics

- One Cycle Throughput
- Execution in U-Pipe
- U-Pipe and V-Pipe Used to Access Data Cache
- Concurrent Data Cache Access and FP Computation
- Tuned for Double Precision Memory-Register Operations

Micro-Architecture Overview

Safe Instruction Recognition

- Early Detection of Potential Exceptions

Example:

**FMULP Recognized as Safe**

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FMULP ST (2), ST</td>
</tr>
<tr>
<td>2</td>
<td>FADD QWORD PTR [EAX]</td>
</tr>
</tbody>
</table>

**FMULP Recognized as Unsafe**

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FMULP ST (2), ST</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>FADD QWORD PTR [EAX]</td>
</tr>
</tbody>
</table>
Micro-Architecture Overview

Arithmetic Units

- Multiplier
  - Full Extended Precision Multiply Array
  - Three Cycles Latency for All Precisions
  - Support for Integer Multiplication

- Adder
  - Execution of Majority of Basic Instructions
  - Three 71-Bit Adders
  - Two 69-Bit Shifters
  - Three Cycles Latency for All Precisions

- Divider
  - Divide, Remainder and Square-Root Operations
  - SRT Algorithm

Register Stack Manipulation

- Instruction Set Uses Top of Register Stack as Accumulator
- Parallel Execution of FXCH

Example:

<table>
<thead>
<tr>
<th>Cycle 1</th>
<th>FADD QWORD PTR [EAX]</th>
<th>FMUL QWORD PTR [EBX]</th>
<th>FXCH ST (2)</th>
<th>FXCH ST (3)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>st0</th>
<th>st1</th>
<th>st2</th>
<th>st3</th>
<th>st4</th>
<th>st5</th>
<th>st6</th>
<th>st7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>st0</td>
<td>C</td>
<td>D</td>
<td>st0</td>
<td>st1</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>D</td>
<td>st1</td>
<td>B</td>
<td>E</td>
<td>st1</td>
<td>st2</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>E</td>
<td>st2</td>
<td>A + [EAX]</td>
<td>F</td>
<td>st2</td>
<td>st3</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>F</td>
<td>st3</td>
<td>C x [EBX]</td>
<td>G</td>
<td>st3</td>
<td>st4</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
<td>G</td>
<td>st4</td>
<td>E</td>
<td>H</td>
<td>st4</td>
<td>st5</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>H</td>
<td>st5</td>
<td>F</td>
<td>st5</td>
<td>st6</td>
<td>st7</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
<td>E</td>
<td>st6</td>
<td>G</td>
<td>E</td>
<td>st6</td>
<td>st7</td>
</tr>
<tr>
<td>H</td>
<td>E</td>
<td>D</td>
<td>st7</td>
<td>F</td>
<td>F</td>
<td>st7</td>
<td>st7</td>
</tr>
</tbody>
</table>

Before Cycle 1 | After Cycle 1 | After Cycle 2

Hot Chips IV
Transcendental Functions

- Direct Microcode Support for All Architecturally Defined Transcendental Instructions:
  - Sine
  - Cosine
  - Sine-and-Cosine
  - Tangent
  - Arctangent
  - \(2^x - 1\)
  - \(x \log_2 x\)
  - \(x \log_2 (x + 1)\)

- Table Driven Algorithms Using Polynomial Approximation
- Performance and Error Bound Improvement over Intel486™ DX 33MHz CPU
- Comprehensive Validation Program

Compiler Optimization

- Instruction Scheduling
- Register Allocation
- Loop Unrolling
- Parallel FXCH
Summary

- Streamlined Pipeline Provides High Performance
  - Integration with Integer Pipeline
  - One Cycle Throughput
  - Tuning for Memory-Register Double Precision Operations
- Fast Arithmetic Units Using State of the Art Algorithms
  - Multiplier
  - Adder
  - Divider
- Improved Performance and Accuracy of Transcendental
- New Compiler Optimizations Co-Developed with Micro-Architecture