SuperScalar Architecture of the P5
Intel’s Next Generation Microprocessor

Donald Alpert
Intel Corporation

Outline
- Integer Pipeline
- Superscalar Execution
- Branch Prediction
- Dual-Access Data Cache
- Compiler Optimizations
Instruction Issue Algorithm

Decode Two Consecutive Instructions: I1 and I2
If the Following Are All True
   I1 is a "Simple" Instruction
   I2 is a "Simple" Instruction
   I1 is Not a JUMP Instruction
   Destination of I1 ≠ Source of I2
   Destination of I1 ≠ Destination of I2
Then Issue I1 to U-Pipe and I2 to V-Pipe
Else Issue I1 to U-Pipe

"Simple" Instructions Are Generally ALU or MOV Operations, Including Reg-Reg, Imm-Reg, Mem-Reg, and Reg-Mem Formats, and JUMPS

Example

<table>
<thead>
<tr>
<th>Proc2:</th>
<th>U-Pipe</th>
<th>V-Pipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>pushl</td>
<td>%ebx</td>
<td>movl</td>
</tr>
<tr>
<td>movl</td>
<td>(%ecx), %edx</td>
<td>movb</td>
</tr>
<tr>
<td>addl</td>
<td>$10, %edx</td>
<td>jne</td>
</tr>
<tr>
<td>cmpb</td>
<td>$65, %ah</td>
<td>movl</td>
</tr>
<tr>
<td>decl</td>
<td>%edx</td>
<td>movl</td>
</tr>
<tr>
<td>movl</td>
<td>%edx, %ebx</td>
<td>subl</td>
</tr>
<tr>
<td>movl</td>
<td>%edx, (%ecx)</td>
<td>movl</td>
</tr>
</tbody>
</table>

.B4_4:
| popl   | %ebx   |        |        |
| ret    |        |        |        |
Compiler Optimization

- Instruction Selection
  - Use Simple Formats for Efficient Decoding
- Instruction Scheduling
  - Minimize Address Generation Interlocks
  - Maximize Parallel Execution
- Register Allocation
  - Schedule and Allocate Together to Make Best Use of Small Register Set

Summary

- Superscalar Microarchitecture
  - Dual Integer Pipelines
  - Branch Target Buffer
  - Dual-Access Data Cache
- Fully Compatible with Intel486™ CPU