The Impact of Integer Instructions in Floating Point Applications

HPC processors that run floating point applications need higher integer execution performance than it is believed.

- NAS Parallel Benchmarks[1] execute integer instructions as much as FP ones
- 7 of 9 workloads in NPB execute integer instructions more
- Integer is important in application perf.
- We benchmarked 150+ new and obsolete systems to understand the relation between microarch and perf.
- A large diversity is observed in integer performance of modern designs

Fig. 1: Instruction mix of dynamic exec. trace of NPB size=A in x86_64, GCC 4.5.3, -O3

Fig. 2: Dhrystone-per-MHz of old and new processors

Table 1: System configurations

<table>
<thead>
<tr>
<th>System</th>
<th>Processor</th>
<th>MHz</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Power 740 Express</td>
<td>POWER7</td>
<td>3550</td>
<td>GCC 4.5.2</td>
</tr>
<tr>
<td>NEC SX-9</td>
<td>SX-9</td>
<td>3200</td>
<td>NEC C++/SX Rev 068</td>
</tr>
<tr>
<td>Fujitsu FX10</td>
<td>SPARC64 1848</td>
<td>1848</td>
<td>Fujitsu C/C++ V1.2</td>
</tr>
<tr>
<td>Intel DX795i</td>
<td>Core i7-3960X</td>
<td>3300</td>
<td>GCC 4.5.3</td>
</tr>
<tr>
<td>IBM OpenPower 710</td>
<td>POWER5</td>
<td>1600</td>
<td>GCC 4.5.3</td>
</tr>
<tr>
<td>SX-6I</td>
<td>SX-6</td>
<td>500</td>
<td>NEC C++/SX V1.0</td>
</tr>
<tr>
<td>Fujitsu PRIMEPOWER 410</td>
<td>SPARC64 V</td>
<td>1100</td>
<td>GCC 4.5.2</td>
</tr>
<tr>
<td>AOpen i915Ga/EFRI</td>
<td>Pentium 4 3300</td>
<td>3000</td>
<td>GCC 4.5.2</td>
</tr>
</tbody>
</table>

Fig. 3: Per-socket NPB Performance

How can we improve integer performance?
- Per-clock Int. Insn. similar to (Fig. 4):
  - ARM Cortex-A8
  - SuperSPARC-II
  - PowerPC 604
- Known techniques to speed up integer can be applied, i.e.:
  - Branch prediction
  - Speculative execution

How can we improve integer performance?
- A better integer performance is required
- for both integer and floating point applications
- That is possible through applying known techniques
- Power and budget constraints must be evaluated in future

Fig. 4: Retrpspect of Dhrystone-per-MHz

Fig. 5: Per-clock Int. Insn. similar to (Fig. 4):

References