Review of Instruction Set Architecture Fundamentals

ISA Defined

- Patterson and Hennessy:
  - ”interface between the hardware and the lowest level software”
  - ”includes anything programmers need to know to make a binary machine language program work correctly, including instructions, I/O devices, and so on.”
  - ”enables many implementations of varying costs and performances to run identical software”
Some things that an ISA tells us (generally) about a processor design

- Instructions, instruction classes, and formats
- Data types and formats
- Number of operands per instruction
- Number and types of registers
- Addressing modes
- Ways of accessing memory

The ISA is the starting point for the processor design

"x-Address" Machines

<table>
<thead>
<tr>
<th>x</th>
<th>to do “a=b+c”</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>ADD a,b,c</td>
</tr>
<tr>
<td></td>
<td>“Natural” for most arithmetic ops</td>
</tr>
<tr>
<td>2</td>
<td>COPY a, c ADD a,b</td>
</tr>
<tr>
<td></td>
<td>Smaller instructions, but we need more</td>
</tr>
<tr>
<td>1</td>
<td>COPY1 c ADD b COPY2 a</td>
</tr>
<tr>
<td></td>
<td>Need still more instructions</td>
</tr>
<tr>
<td></td>
<td>Pretty much un-natural</td>
</tr>
<tr>
<td>0</td>
<td>PUSH b PUSH c ADD POP a</td>
</tr>
<tr>
<td></td>
<td>Still more instructions</td>
</tr>
<tr>
<td></td>
<td>Very un-natural</td>
</tr>
</tbody>
</table>
Some top-level options in ISA design

• Complex Instruction Set Computer (CISC) often has:
  • Many instructions doing compound operations
  • Many complex addressing modes and data types
  • Memory access by almost any instruction type
  • Many instruction classes, perhaps with varying numbers of operands
  • Many options, many different types and variants

• Reduced Instruction Set Computer (RISC) often has:
  • Fairly small set of instructions doing simple operations
  • Limited set of addressing modes and data types
  • Memory access restricted to certain instruction types
  • Few instruction classes, few differences in number of operands
  • Fewer options, types, variants

Some implications (CISC vs RISC)

• CISC
  • Instructions are complex, we can do a lot with each one
  • CPI high, IC low

• RISC
  • Instructions are simpler, it takes more of them to do a given operation
  • CPI low, IC high

Most current-generation high-performance microprocessor designs are RISC
Some things to think about when we’re designing (1)

“Simplicity favors regularity” (P & H Principle 1)
– The more variations in instructions (formats, …), the more logic it takes to identify which variation we have in a particular instruction.
– The more regular the instruction set, the less time we have to spend decoding the instruction type

Some things to think about when we’re designing (2)

“Smaller is faster” (P&H Principle 2)
– As we add more and more logic to a design, max speed tends to drop due to:
  • More “things” to select from (e.g., more registers)
    – More logic levels needed to decode identifiers
    – Wider instructions to specify more units
  • Longer path lengths needed
    – Signal propagation increases
Some things to think about when we’re designing (3)

“Make the common case fast” (P&H Principle 3)

– Based on Amdahl’s Law (Chapter 1)
– Speeding up things you do often gives greater payoff than speeding up things you do do infrequently.

Some things to think about when we’re designing (4)

• “Good design demands good compromises” (P&H Principle 4)
  – Much of the time, an improvement in one area compromises another (ex: Adding registers makes programmers happy, but may slow the processor down)
  – The best designs exhibit a balance of features
Some things to think about when we’re designing (5)

- Memory access is much slower than register access
  - Register access typically 1 clock cycle
  - RAM access may be dozens or hundreds of clock cycles