Computer Systems Organization

CSE 521/560M
Lecture 3
Prof. Patrick Crowley

Plan for Today

- Newsgroup
- VHDL preparation
- Questions
- Instruction set architecture discussion

---

ISA: The Programmer’s Interface

- Designer’s Question: *what does the programmer need?*
- Instruction Set
  - Available operations (RISC vs. CISC)
  - Branching behavior
  - Valid instruction sequences
- Registers
  - Sizes
  - Data types
  - Organization (flat, windows, hierarchical)
- Addressing Modes
  - Register, Immediate, Indirect
  - Offset, relative

---

Interface vs. Implementation

- ISA: the programmer’s interface
- Implementation: the underlying resources and organization
- Ideally:
  - Programmer needs only ISA to write good programs
- In reality:
  - Programmers look into the implementation to improve performance
  - Consequences?

---

ALU-centric ISA Taxonomy

Simple vs. Complex Operations

- RISC: reduced instruction set computer (+ load/store arch)
  - Small number of instructions
  - Easy to encode
  - Poor code density
  - Compiler and implementation familiar
- CISC: complex instruction set computer
  - Many instructions
  - Good for special, hand-coded cases
  - Good code density
  - Complicated compilation and implementation
### Operation Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>Arithmetic, logic</td>
</tr>
<tr>
<td>Memory</td>
<td>Loads and stores</td>
</tr>
<tr>
<td>Control</td>
<td>Branch, jump, calls returns, traps</td>
</tr>
<tr>
<td>System</td>
<td>OS call, privileged system management</td>
</tr>
<tr>
<td>F.P.</td>
<td>Floating point arithmetic</td>
</tr>
<tr>
<td>Decimal</td>
<td>Decimal ops</td>
</tr>
<tr>
<td>String</td>
<td>Copy, compare, search</td>
</tr>
<tr>
<td>Multimedia</td>
<td>Pixel and vertex ops</td>
</tr>
</tbody>
</table>

### Control Flow Operations

- Conditional branches (usually PC-relative)
  - Condition codes
  - Condition register
  - Compare and branch
- Jumps (e.g., unconditional branches)
- Procedure calls and returns
  - Who saves which (return address, registers, params, etc.),
    where (stack, register) and where (caller, callee)
  - Combination of HW and SW conventions

### Addressing for Control Flow

- Register indirect (for dynamic targets)
- PC-relative displacement

### Unconventional Operations

- Decimal operations (IEEE-PA, F-P is used in x86)
- String operations (x86)
- Lack of byte instructions (early Alpha)
- Synchronization (atomic swap, "fence")
- Predicated execution (conditional moves)
- Cache hints (prefetch, flush)
- TLB instructions (TLB miss handled by software in MIPS)
- Multimedia (Sparc, MMX)
- Bit manipulation (Intel 64)
- Thread management (SMT)
**Registers**

- Types of registers
  - Integer (often 32 in number)
  - Floating-point (often 32)
- Special GPRs
  - Stack pointer, frame pointer, the PC (VAX)
- Special purpose registers
  - Control registers, segment registers (x86)
- Organization
  - Flat
  - Windows (Spec)
  - Hiernarch (Cray)

**Addressing and Byte Ordering**

- How are bytes ordered in multi-byte data objects?
- Example: 32-bit integer at address 0x100
  - $[x_{23} \rightarrow x_{0}]$ when broken into bytes,
  - $[x_{22} \rightarrow x_{16}]$, $[x_{15} \rightarrow x_{8}]$, $[x_{7} \rightarrow x_{0}]$

<table>
<thead>
<tr>
<th>Big-endian</th>
<th>Little-endian</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100</td>
<td>0x101</td>
</tr>
<tr>
<td>0x102</td>
<td>0x103</td>
</tr>
</tbody>
</table>

- $[x_{31} \rightarrow x_{24}]$, $[x_{23} \rightarrow x_{16}]$, $[x_{15} \rightarrow x_{8}]$, $[x_{7} \rightarrow x_{0}]$

**Taxonomy of Addressing Modes**

- General purpose
  - Register, immediate, absolute and displacement
- Often useful
  - Indexed, scaled-index
- Special purpose or obsolete
  - Memory indirect, auto-increment and decrement

**Addressing Modes**

- Memory indexed
- Scale
- Load
- Store
- Register indirect
- Immediate
- Displacement

**Displacement Distance**

- Percentage of displacement
  - Integer average
  - Floating-point average

**Frequency of Immediates**

- Loads
- ALU operations
- All instructions
Size of Immediates

Encoding Instructions

- **Flexibility**
  - Many registers
  - Multiple addressing options
- **Compactness**
  - Small instructions
- **Ease of implementation**
  - Instructions represent *arml*.

Examples

New Architectures

- **Memory addressing**
  - Modes:
    - Immediate of 4, 8, 16 bits
    - displacement (arithmetic/register index) of 12, 16 bits
- **Instructions**
  - Simple: load, store, add, subtract, move register-register and shift
  - Operands types:
    - Integer: 8, 16, 32, and 64 bit
    - Floating points: 32 and 64 bit
  - Control flow: PC-relative branch displacement of 8 bits, register indirect for procedure calls and returns

Additional Topics

- VLIW – very long instruction word architectures
- Multithreading
- Exception handling

Assignment

- **Readings**
  - For Wednesday
    - Paper: Retrospective on RISC Computers, submit commentary to [[email: instructor@ course.com]]
    - Read, Turner’s VHDL tutorial
  - For Monday
    - Ham/P: Chapter 2, sections 2.1, 2.3, 2.18 (2.13 is optional)
    - Ham/P: Appendix A, sections A.2, A.11