CIS 314
Instruction Set Architecture

Prof. Michel A. Kinsky
The Full View System

Applications

Operating System

Compiler

Firmware

ISA

Processor

Memory organization

I/O system

Datapath & Control

Digital Design

Circuit Design

Layout
Another System View

Software

Instruction Set

Hardware
• Our first, very minimalist, view of computer organization
Instruction Set Architecture (ISA)

- Instructions are the language the computer understand
- Instruction Set is the vocabulary of that language
- It serves as the hardware/software interface
  - Defines data types
    - byte, int, float, double, string, vector…
  - Defines set of programmer visible state
    - Known as the programmer’s model of the machine
  - Defines instruction semantics (operations, sequencing)
    - operand location: register, immediate, indirect, …
    - add, sub, mul, move, compare, …
Instruction Set Architecture (ISA)

- Instructions are the language the computer understand
- Instruction Set is the vocabulary of that language
- It serves as the hardware/software interface
  - Defines instruction format (bit encoding)
    - Number of explicit operands per instruction
    - Operand location
    - Number of bits per instruction
    - Instruction length: fixed, short, long, or variable...
  - Examples: MIPS, Alpha, x86, IBM 360, VAX, ARM, JVM
Instruction Set Architecture (ISA)

• Many possible implementations of the same ISA
  ‣ 360 implementations: model 30 (c. 1964), z900 (c. 2001)
  ‣ x86 implementations: 8086 (c. 1978), 80186, 286, 386, 486, Pentium, Pentium Pro, Pentium-4, Core i7, AMD Athlon, AMD Opteron, Transmeta Crusoe, SoftPC
  ‣ MIPS implementations: R2000, R4000, R10000, ...
  ‣ JVM: HotSpot, PicoJava, ARM Jazelle, ...
Instruction Set Architecture (ISA)

- Many possible implementations of the same ISA
- ISA classes: Stack, Accumulator, and General-purpose register
- Most current systems use general-purpose register (GPR) based ISA
Instruction Set Architecture (ISA)

• Four principles are used in designing instruction set architecture:

1. Simplicity favors regularity
   • Total number of instructions in the instruction set

2. Smaller is faster
   • Number of addressable registers
   • Large number of registers increases access time

3. Good design demands good compromise
   • Computer designer must balance the programmer’s desire for more registers with the need to minimize access time

4. Make the common case fast
Instructions can be divided into 3 classes

1. Data movement instructions
   - Move data from a memory location or register to another memory location or register without changing its form
   - Load—source is memory and destination is register
   - Store—source is register and destination is memory
   - `lw $a0, 0($t0)`

2. Arithmetic and logic (ALU) instructions
   - Change the form of one or more operands to produce a result stored in another location
Instruction Set Architecture (ISA)

• Instructions can be divided into 3 classes

1. Data movement instructions

2. Arithmetic and logic (ALU) instructions
   • Change the form of one or more operands to produce a result stored in another location
   • Add, Sub, Shift, etc.
   • add $t3, $t4, $t5

3. Branch instructions (control flow instructions)
• Instructions can be divided into 3 classes

1. Data movement instructions
2. Arithmetic and logic (ALU) instructions
3. Branch instructions (control flow instructions)
   • Alter the normal flow of control from executing the next instruction in sequence
     • bgez $t8, else
The instruction format

- Size and meaning of fields within the instruction
- Operation to perform: `add r0, r1, r3`
- Op code: add, load, branch, etc.
- Where to find the operands: `rs`, `rt`, `rd`
- Place to store result: `rd`
- Common Instruction Formats
Instruction Set Architecture (ISA)

- The instruction format
  - Common Instruction Formats
    - OPCODE + 0 addresses
    - OPCODE + 1 (usually a memory address)
    - OPCODE + 2 (registers, or register + memory address)
    - OPCODE + 3 (registers, or combinations of registers and memory)
## Types of ISA

### Accumulator:

1-address

```
add A

Acc \leftarrow Acc + Mem[A]
```

### Stack:

0-address

```
add

ToS \leftarrow ToS + Next
```

### Memory-Memory:

2-address

```
add A, B

Mem[A] \leftarrow Mem[A] + Mem[B]
```

3-address

```
add A, B, C

Mem[A] \leftarrow Mem[B] + Mem[C]
```

### Register-Memory:

2-address

```
add R1, A

R1 \leftarrow R1 + Mem[A]
```

```
load R1, A

R1 \leftarrow Mem[A]
```

### Register-Register (Load/Store):

3-address

```
add R1, R2, R3

R1 \leftarrow R2 + R3
```

```
load R1, R2

R1 \leftarrow Mem[R2]
```

```
store R1, R2

Mem[R1] \leftarrow R2
```
ISA Complexity

• Less operands leads to shorter decode time and longer programs
• More operands implies complex operations that require longer decode time
• Complex operations raises complexity of ISA but shorter programs
• Metrics for measuring the ISA’s effectiveness:
  ‣ Main memory space occupied by a program
  ‣ Instruction length (in bits) and complexity
  ‣ Total number of instructions in the instruction set
ISA and Performance

- Instructions per program depends on source code, compiler technology and ISA
- Cycles per instructions (CPI) depends upon the ISA and the microarchitecture
- Time per cycle depends upon the microarchitecture and the base technology

\[
\text{Time} = \frac{\text{Instructions}}{\text{Cycles}} \times \text{Cycle}
\]
# Instruction Distribution (MIPS)

<table>
<thead>
<tr>
<th></th>
<th>SPEC2000 Int</th>
<th>SPEC2000 FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>26%</td>
<td>15%</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>2%</td>
</tr>
<tr>
<td>Add</td>
<td>19%</td>
<td>23%</td>
</tr>
<tr>
<td>Compare</td>
<td>5%</td>
<td>2%</td>
</tr>
<tr>
<td>Cond br</td>
<td>12%</td>
<td>4%</td>
</tr>
<tr>
<td>Cond mv</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td>Jump</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>LOGIC</td>
<td>18%</td>
<td>4%</td>
</tr>
<tr>
<td>FP load</td>
<td></td>
<td>15%</td>
</tr>
<tr>
<td>FP store</td>
<td></td>
<td>7%</td>
</tr>
<tr>
<td>FP others</td>
<td></td>
<td>19%</td>
</tr>
</tbody>
</table>
Next Class

• RISC vs. CISC ISAs