Instruction Types

Computation:
- arithmetic (e.g., add)
- logical (e.g., xor)
- compare (e.g., set if not equal)

Data transfer:
- load
- store

Control
- branch
- jump

MIPS Computation Instructions

Opcode rd, rs, rt

Opcode rt, rs, immed

- rd: destination register (modify)
- rs: source register (read-only)
- rt: source/destination register (read-only/modify)
- immed: 16-bit signed value (constant)
MIPS Computation Instructions

Some examples:

```
add $t0, $t1, $t2  # $t0 = $t1+$t2
addi $t0, $t1, 20  # $t0 = $t1+20
addu $t0, $t1, $t2  # $t0 = $t1+$t2
sub $t5, $0, $t5  # $t5 = -$t5
and $t0, $t1, $t2  # $t0 = $t1 & $t2
slt $t0, $t1, $t2  # if $t1 < $t2, $t0 = 1, else $t0 = 0
slti $t0, $t1, -6  # if $t1 < -6, $t0 = 1, else $t0 = 0
```

The GPRs are used to store the result of a condition. Alternative architecture: condition codes

- special 1-bit registers that store the result of specific conditions
  - whether the result is zero
  - whether the result is negative

The machine does not know if a value is signed or unsigned (the bag of bits) --- you have to specify this by using the appropriate instruction

Instruction Encoding

ISA defines the formats for instructions

- what fields they contain
- the size of the fields
- the field values & what the values signify

Knowing the instruction formats, shows us how the CPU processes instructions

- bridge between architecture & implementation
R-type Format

For arithmetic, logical, comparative instructions with register operands

<table>
<thead>
<tr>
<th>opcode</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>func</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>26</td>
<td>20</td>
<td>16</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

- **opcode, func** = operation
  - **opcode** = a computational instruction with register operands
  - **func** = which computation
- **rs, rt** = source operands
- **rd** = destination operand
- **shamt** = shift distance in bits

```
add $t0, $t1, $t2
0 9 10 8 unused 32
```

```
xor $t3, $t4, $t5
0 12 13 11 unused 38
```

```
sll $t2, $s0, 4
0 unused 16 10 4 0
```

I-type Format

For arithmetic, logical, comparative instructions with one register operand & one constant operand

<table>
<thead>
<tr>
<th>opcode</th>
<th>rs</th>
<th>rt</th>
<th>immed</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>26</td>
<td>20</td>
<td>16</td>
</tr>
</tbody>
</table>

- **opcode** = computational instruction
- **rs** = source operand
- **rt** = destination operand
- **immed** = constant, $\pm 2^{15}$
  - sign-extended when used (replicate msb)

Using an immediate value is faster than loading the constant from memory & saves using a register

```
ori $t0$, $t1$, -256
13 9 8 -256
```
Instruction Encoding

Being a RISC, MIPS has few (3) instruction formats

- all instructions are the same length, 32 bits
- most formats have similar fields
  - for example: an opcode, at least one source register
- fields that are common to more than one format have the same location in the instruction
  - for example: the opcode is always first
- fields that are common to more than one format are the same size
  - for example: the opcode is always 6 bits