Processing an instruction

- Steps to process one instruction:
  - Instruction fetch
  - Decode
  - Register fetch
  - Branch (optional)
  - Execute (ALU operation)
  - Memory access (optional)
  - Write-back any result to destination register

Instruction Fetch

- Program counter (PC) used as an address to do a memory access
- Takes time to do the memory access
- Result placed in an instruction register (IR) or similar
- PC = PC + 4
  - gets PC ready to fetch the next instruction

Decode & Register Fetch

- Decoding expands the instruction into a more usable state, e.g.:
  - 32-bit sign extended immediate
  - is the instruction a branch/jump?
  - does the instruction do memory access (load/store)?
  - what arithmetic to do (e.g. add, sub, or, and, xor, etc.)
    - lots of instructions end up being add, e.g. address calculation for load or store instructions
  - what are the source registers and the destination register
    - easy to figure out on RISC-V
- Register fetch
  - look up the source registers

Branch

- Branch (optional, may be done with decode or execute)
  - PC = PC + signExtend(immediate)
  - return address may be stored in a register

Execute, Memory Access, Write-back

The following are usually performed as separate steps:

- Execute
  - performs an integer or logical operation using an arithmetic logic unit (ALU)
  - pretty easy, but this is what does all the work!
- Memory access (optional)
  - use the address calculated in Execute to either:
    - load data (8-bits, 16-bits or 32-bits)
    - store data (8-bits, 16-bits or 32-bits)
- Write-back
  - write any result to Rd, the destination register
### Example Instructions

- **Add immediate**: addi $x5$, $x6$, 8
- **Action**: $x5 = x6 + 8$
- **It is an I-type instruction**:

<table>
<thead>
<tr>
<th>12-bits</th>
<th>5-bits</th>
<th>3-bits</th>
<th>5-bits</th>
<th>7-bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>imm[11:0]</td>
<td>$x1$</td>
<td>funct3</td>
<td>$x0$</td>
<td>opcode</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>3'b001</td>
<td>5</td>
<td>7'b000011</td>
</tr>
</tbody>
</table>

- **Load word**: lw $x5$, 8($x6$)
- **Action**: $x5 = \text{mem}[x6 + 8]$
- **It is also an I-type instruction**:

<table>
<thead>
<tr>
<th>12-bits</th>
<th>5-bits</th>
<th>3-bits</th>
<th>5-bits</th>
<th>7-bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>imm[11:0]</td>
<td>$x1$</td>
<td>funct3</td>
<td>$x0$</td>
<td>opcode</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>3'b010</td>
<td>5</td>
<td>7'b0000011</td>
</tr>
</tbody>
</table>