Chapter 5:
“The Processor: Datapath and Control”

Part One, The Single Cycle Processor
The Performance Big Picture

• **Execution Time** = \(\text{Insts} \times \text{CPI} \times \text{Cycle Time}\)
• **Processor design** (datapath and control) will determine:
  - Clock cycle time
  - Clock cycles per instruction

• **Starting today:**
  - **Single cycle processor:**
    • Advantage: CPI = 1
    • Disadvantage: long cycle time
What parts of MIPS?

- We’ll cover a subset of MIPS
  - Memory instructions
  - Arithmetic/Logical (and just a subset of these, but you should be able to figure out how to add many of them)
  - BEQ and J
  - Basic load/store architecture with these steps:
    - Read PC and Fetch Inst
    - Read Registers
    - Do Operation
    - Write memory/registers
    - Repeat

.. but you should be able to extend what we do to handle more of the instructions
### The MIPS core subset

#### R-type
- **add rd, rs, rt**
  1. Read registers rs and rt
  2. Feed them to ALU
  3. Update register file

- **sub, and, or, slt**

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>6 bits</td>
</tr>
</tbody>
</table>

#### LD/ST
- **lw rt, rs, imm**
  1. Read register rs (and rt for store)
  2. Feed rs and immed to ALU
  3. Move data between mem and reg

- **sw rt, rs, imm**

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

#### BRANCH:
- **beq rs, rt, imm**
  1. Read registers rs and rt
  2. Feed to ALU to compare
  3. Add PC to disp; update PC
Register Transfer Language (RTL)

- Is a mechanism for describing the movement of data between storage elements
  - Gives us a precise way to describe various actions of our instructions
  - May be more than 1 RTL statement per inst
  - $PC \leftarrow PC + 4$
  - $R[rd] \leftarrow R[rs] + R[rt]$
The basic design algorithm (after you have the ISA) - you’ll do this for your own ISA for 141L

- Build the datapath on the whiteboard
  - one by one,
    simulate each instruction
    on the current datapath “sketch”:
    - make sure it is workable
    - if not, modify datapath

- Design the control logic
  - one by one,
    simulate each instruction on the current datapath + control logic:
    - make sure it is workable
    - if not, modify control or datapath
The Instruction Execution Cycle

- **Instruction Fetch**
  - Obtain instruction from program storage

- **Instruction Decode**
  - Determine required actions and instruction size

- **Operand Fetch**
  - Locate and obtain operand data

- **Execute**
  - Compute result value or status

- **Result Store**
  - Deposit results in storage for later use

- **Next Instruction**
  - Determine successor instruction
The MIPS core subset

• **R-type**
  - *add rd, rs, rt*
  - *sub, and, or, slt*

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>6 bits</td>
</tr>
</tbody>
</table>

1. Read registers rs and rt
2. Feed them to ALU
3. Update register file

• **LD/ST**
  - *lw rt, rs, imm*
  - *sw rt, rs, imm*

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

1. Read register rs (and rt for store)
2. Feed rs and immed to ALU
3. Move data between mem and reg

• **BRANCH:**
  - *beq rs, rt, imm*

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

1. Read registers rs and rt
2. Feed to ALU to compare
3. Add PC to disp; update PC
R-Format/ Lw/ Sw/ BEQ

<table>
<thead>
<tr>
<th>ALUsrc</th>
<th>ALUop</th>
<th>MemRead</th>
<th>MemWrite</th>
<th>MemToReg</th>
<th>RegDst</th>
<th>RegWrite</th>
<th>PCsrc</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRR</td>
<td>1</td>
<td>tbd</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>lw</td>
<td>0</td>
<td>tbd</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1_9</td>
</tr>
<tr>
<td>bne</td>
<td>1</td>
<td>cmp</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
• (Derivation of Datapath and Control on Blackboard)