Designing a Single Cycle Datapath

or

The Do-It-Yourself CPU Kit

The Big Picture: Where are We Now?

• The Five Classic Components of a Computer

• Today’s Topic: Datapath Design, then Control Design

The Big Picture: The Performance Perspective

• ET = Insts * CPI * Cycle Time

• Processor design (datapath and control) will determine:
  –
  –

• Starting today:
  – Single cycle processor:
    ▪ Advantage:
    ▪ Disadvantage:

The Processor: Datapath & Control

• We're ready to look at an implementation of the MIPS

• Simplified to contain only:
  – memory-reference instructions: lw, sw
  – arithmetic-logical instructions: add, sub, and, or, slt
  – control flow instructions: beq

• Generic Implementation:
  – use the program counter (PC) to supply instruction address
  –
  –
  – use the instruction to decide exactly what to do

• All instructions use the ALU after reading the registers
  memory-reference? arithmetic? control flow?
Review: The MIPS Instruction Formats

- All MIPS instructions are 32 bits long. The three instruction formats:

<table>
<thead>
<tr>
<th>R-type</th>
<th>I-type</th>
<th>J-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>rs</td>
<td>rt</td>
</tr>
<tr>
<td>target address</td>
<td>immediate</td>
<td>displacement</td>
</tr>
</tbody>
</table>

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

- LOAD and STORE
  - lw rt, rs, imm16
  - sw rt, rs, imm16

- BRANCH:
  - beq rs, rt, imm16

Where We're Going – The High-level View

Review: Two Types of Logic Components
Clocking Methodology

• All storage elements are clocked by the same clock edge

Storage Element: Register

• Register
  – Similar to the D Flip Flop except
    • N-bit input and output
    • Write Enable input
  – Write Enable:
    • 0: Data Out will not change
    • 1: Data Out will become Data In (on the clock edge)

Storage Element: Register File

• Register File consists of (32) registers:
  – Two 32-bit output buses:
  – One 32-bit input bus: busW
• Register is selected by:
  – RA selects the register to put on busA
  – RB selects the register to put on busB
  – RW selects the register to be written via busW when Write Enable is 1
• Clock input (CLK)

Storage Element: Memory

• Memory
  – One input bus: Data In
  – One output bus: Data Out
• Memory word is selected by:
  – Address selects the word to put on Data Out
  – Write Enable = 1: address selects the memory word to be written via the Data In bus
• Clock input (CLK)
  – During read operation, behaves as a logic block:
    • Address valid => Data Out valid after “access time.”
Register Transfer Language (RTL)

- is a mechanism for describing the flow of data between storage elements:

\[
\begin{align*}
PC & \leftarrow PC + 4 + R[5] \\
R[rd] & \leftarrow R[rs] + R[rt] \\
R[rt] & \leftarrow Mem[R[rs] + \text{immed}] \\
\end{align*}
\]

Overview of the Instruction Fetch Unit

- The common RTL operations
  - Fetch the Instruction: \(\text{inst} \leftarrow \text{mem}[\text{PC}]\)
  - Update the program counter:
    - Sequential Code: \(\text{PC} \leftarrow \text{PC} + 4\)
    - Branch and Jump: \(\text{PC} \leftarrow \text{"something else"}\)

Datapath for Register-Register Operations

- \(R[rd] \leftarrow R[rs] \text{ op } R[rt]\)  
  - Example: \(\text{add } rd, rs, rt\)
  - \(Ra, Rb, \text{ and } Rw \text{ comes from instruction's } rs, rt, \text{ and } rd \text{ fields}\)
  - \(\text{ALUctr and RegWr: control logic after decoding the instruction}\)

---

Program Counter Management

- Instruction Address
- Instruction Memory
- Instruction
- PC
- Adder

- Instruction memory
- Program counter
- Adder

- Instruction memory
- Instruction
- PC
- Adder

- Instruction
- Instruction memory
- PC
- Adder
Datapath for Load Operations

R[rt] <- Mem[R[rs] + SignExt[imm16]]  
Example: lw rt, rs, imm16

Datapath for Store Operations

Mem[R[rs] + SignExt[imm16]] <- R[rt]  
Example: sw rt, rs, imm16

Datapath for Branch Operations

beq rs, rt, imm16  
We need to compare Rs and Rt

Binary Arithmetic for the Next Address

• In theory, the PC is a 32-bit byte address into the instruction memory:
  – Sequential operation: PC<31:0> = PC<31:0> + 4
  – Branch operation: PC<31:0> = PC<31:0> + 4 + SignExt[Imm16] * 4
• The magic number “4” always comes up because:
  – The 32-bit PC is a address
  – And all our instructions are long
  – The 2 LSBs of the 32-bit PC are always zeros
  – There is no reason to have to keep the 2 LSBs
• In practice, we can simplify the hardware by using a 30-bit PC<31:2>:
  – Sequential operation: PC<31:2> = PC<31:2> + 1
  – Branch operation: PC<31:2> = PC<31:2> + 1 + SignExt[Imm16]
  – In either case: Instruction Memory Address = PC<31:2> concat “00”
Putting it All Together: A Single Cycle Datapath

- We have everything except control signals

The R-Format (e.g. \textit{add}) Datapath

The Load Datapath

The store Datapath
Key Points

- CPU is just a collection of state and combinational logic
- We just designed a very rich processor, at least in terms of functionality
- Performance = Insts * CPI * Cycle Time
  – where does the single-cycle machine fit in?