ECE 251: Computer Architecture

Week 09 Notes - The Processor: Datapath & Control

Prof Rob Marano

Spring 2026

1. Anatomy of the MIPS Processor

The processor is built from two primary sub-systems acting in concert.

- **The Datapath (The Muscles)**: Contains the hardware elements that physically hold or operate on data. Key components include the Program Counter (PC), Instruction Memory, Registers, Arithmetic Logic Unit (ALU), and Data Memory.
- **The Control Unit (The Brain)**: Decodes the binary instruction formats (e.g., the 6-bit opcode) and acts as the explicit conductor, broadcasting specific electrical signals to direct the Datapath MUXes on where to route information.

2. The Core MIPS Subset & Performance

In Chapter 4, we define a processor mathematically tailored to support a core computational subset of the MIPS ISA:

- **Memory-Reference**: `lw`, `sw`
- **Arithmetic-Logical**: `add`, `sub`, `and`, `or`, `slt`
- **Control Flow**: `beq`, `j`

The Bottleneck Equation:
CPU Time = Instruction Count × CPI × Clock Cycle Time.
Because this is a Single-Cycle architecture, CPI is forcefully locked at 1.0. Therefore, the global Clock Cycle Time is constrained by the slowest possible component traversal.

3. Logic Design Conventions & Clocking

Before wiring data buses, we must establish circuit limitations.

- **Combinational Logic (ALU, Adders, MUXes)**: Elements with no memory. Their outputs cascade to new math targets immediately following raw logic gate decay delays.
- **Sequential State Elements (Registers, PC, Memory)**: Elements that lock in memory. They refuse to overwrite their internal payload unless commanded by a distinct synchronization square-wave: **The Clock**.

Edge-Triggered Constraints: State elements only safely capture data on the *rising edge* transition. Input data must successfully satisfy strict **Setup Time** (stabilizing early before the hit) and **Hold Time** (stabilizing after the hit) rules.

THE COOPER UNION

4. Building the Datapath: Fetch & R-Type Math

Hardware engineers wire sequential sub-blocks tailored directly to the specific instruction arrays.

1. **Phase 1 (Fetch)**: The PC targets Instruction Memory to pull the raw 32-bit Opcode array. Simultaneously, the PC address hits the `PC+4` Combinational Adder to actively calculate the next sequential sequence clock target.
2. **Phase 2 (R-Type)**: The 5-bit `rs` and `rt` targets extract payload values exclusively from the Register File. The math computes in the ALU, and the 32-bit calculated combinational result routes backward to target the `rd` write port, properly landing within the Setup phase before the upcoming clock triggers.

5. Building the Datapath: Loads, Stores, Branches

Memory and Control Flow instructions mandate heavy Address Offsets.

1. **Phase 3 (Load/Store)**: The 16-bit physical immediate base offset is run through a 32-bit **Sign Extender**. Rather than computing strict math, the ALU mathematically *adds* the register payload directly to the generated 32-bit extension base, establishing a final absolute physical boundary loop to target RAM Data Memory.
2. **Phase 4 (Branches)**: The ALU executes a rapid combinational Subtraction to analyze integer identicality, throwing a `Zero` flag if successful. Conducively, the Sign-Extended immediate natively shifts left by 2 bits, crashing into the **Branch Target Adder** to compute the distant execution trajectory.

6. Multiplexing the Subsystems

We cannot physically have 3 competing wires writing to the exact same Register File node. To fuse the Datapath, we deploy MUX ‘traffic cops‘:

- **ALUSrc MUX**: Selects between throwing Register 2 to the ALU, *or* shifting over into the 32-bit Sign-Extended Immediate path.
- **MemtoReg MUX**: Decides whether the Register File receives the absolute ALU math result *or* the freshly discovered memory array pulled from Data Memory.
- **PCSrc MUX**: Orchestrates whether the `PC+4` value continues *or* is forcefully overwritten by the distant Branch Target calculated trace.

These elements are all exclusively directed by the explicit Control Unit wiring!

7. SystemVerilog Behavioral Emulation

We can physically model the datapath hardware using SystemVerilog.

- **State Elements**: Modeled exclusively using `always_ff @(posedge clk)` directly mimicking Edge-Triggered constraints.
- **Combinational Elements**: Modeled utilizing `always_comb` and standard asynchronous `assign` arrays.

```
always_comb begin
    case (alucontrol)
        3'b010: result = a + b; // ADD
        3'b110: result = a - b; // SUB
    endcase
end
```

8. The Limitations of Single-Cycle *(Why we Pipelined)*

- The global clock cycle MUST be locked to accommodate the **slowest possible instruction** in the entire ISA to guarantee the Setup and Hold architectural physical stability.
- Because the `lw` instruction dynamically utilizes the Fetch, Register Reading, ALU Address Calculation, Memory Fetching, and Terminal Register writing, it takes **950 ps**. The Clock is physically forced to be 950 ps.
- **The Penalty**: A fast `add` finishes math early at 700ps, but sits electrically idle, wasting 250ps of raw CPU performance waiting for the slow `lw` clock boundary edge constraint constraint.
- **Session 10 Solution**: Multicycle Breakdowns and Pipelining.