Processors

The performance of a Central Processing Unit (CPU), or microprocessor, can be quantified by the time it takes to execute a program. This time is fundamentally influenced by three key factors:

(1) Instruction Count (IC): This is the total number of instructions a program executes. It's determined primarily by the Instruction Set Architecture (ISA) - the set of operations the CPU can understand - and the compiler that translates high-level code into machine instructions. A more complex ISA or an inefficient compiler might result in a higher instruction count for the same task.

(2) Cycles Per Instruction (CPI): This is the average number of clock cycles required to execute a single instruction. Different instructions take different amounts of time (cycles) to complete.

(3) Cycle Time (τ): Also known as the clock period, this is the time duration of one clock cycle, usually measured in seconds. The inverse of the cycle time is the clock rate or frequency (measured in Hertz, Hz), which is often the figure used to market CPUs (e.g., 3.0 Hz).

The overall execution time (T) for a program can be mathematically expressed as:

$$T = \text{IC} \times \text{CPI} \times \tau \tag{1}$$

While the Instruction Count is mainly determined by the ISA and compiler, the CPI and Cycle Time are primarily determined by the CPU hardware implementation. Designers aim to minimize all three factors for better performance.

Simplified MIPS Instruction Subset To illustrate the concepts, we'll focus on a simple subset of MIPS instructions that demonstrates most aspects of CPU operation:

- **Memory Reference**:
  - `lw` (load word): Loads data from memory into a register.
  - `sw` (store word): Store data from a register into memory.
- **Arithmetic/Logical**:
  - `add` , `sub` : Addition and subtraction.
  - `and` , `or` : Bitwise logical operations.
  - `slt` (set on less than): Comparison operations.
- **Control Transfer**:
  - `beq` (branch if equal): Changes the program flow based on a condition.
  - `j` (jump): Unconditional change of program flow.

1.1 The Execution Cycle: Fundamental Steps

Every instruction, regardless of type, must pass through a sequence of fundamental steps to be executed by the CPU hardware:

(1) Instruction Fetch: The address stored in the Program Counter (PC) is sent to the Instruction Memory to fetch the instruction.

(2) Register Decode/Read: The instruction's register numbers are extracted and sent to the Register File to read the values of the source registers.

(3) Execution: The Arithmetic Logic Unit (ALU) is used to calculate the required result:

- Arithmetic Result: for `add` , `sub` , etc.
- Memory Address: for `lw` or `sw` , the ALU calculates the effective memory address (base address + offset).
- Branch Target Address: for `beq` , the ALu calculates the address to jump to if the branch condition is met.

(4) Memory Access (Load/Store Instructions Only): Data Memory is accessed only for `lw` (to read data) or `sw` (to write data).

(5) Write Back/PC Update: Write Back the result (e.g., from ALU or memory load) is updated for the next instruction. The Program Counter is updated for the next instruction:

- For sequential execution, PC $\leftarrow$ PC + 4 (since MIPS instructions are 4 bytes).
- For control transfer (`beq` , `j`), PC $\leftarrow$ Target Address.

Digital Information and Encoding. All information within a microprocessor is represented and processed using binary code. Low voltage typically represents a logic 0, while High voltage typically represents a logic 1. A single binary digit is called a bit. One wire is dedicated to carrying the signal for a single bit. Data composed of multiple bits (e.g., 32-bit or 64-bit words) is encoded on multi-wire buses, where each wire in the bus corresponds to one bit of the data word.

Combinational Elements. Combinational elements are the "calculators" of the CPU. Their core function is to operate on input data and produce an output that is purely a function of the current input values. They do not have memory or state. They transform and operate on data. Examples include basic logic gates (AND, OR, NOT) and complex structures like the Arithmetic Logic Unit (ALU). The output is always and immediately a function of the input. If the input changes, the output changes instantly (subject to small physical signal delays). These elements perform their transformations between clock edges - the time during which data is being processed before being captured by the next state element.

Sequential Elements. State (sequential) elements are the "memory" units of the CPU. Unlike combinational logic, they store information and are crucial for maintaining the "state" of the system over time. A fundamental state element is the register, a small, high-speed circuit that stores a data word:

- **Clock Signal**: Registers are governed by a clock signal (`Clk`), a periodic signal that dictates when the stored value can be updated. This synchronization is critical for orderly data movement.
- **Edge-Triggered Update**: Most registers are edge-triggered. This means the stored data is only updated at a specific moment - typically when the Clock signal changes from 0 to 1 (rising edge). This ensures data is captured synchronously across the entire CPU.

To allow the stored value to be retained across multiple clock cycles when an update isn't needed, registers often incorporate a write control input. The stored value only updates on the clock edge if and only if the write control input is simultaneously set to 1 (active). This control is essential because not all results need to be written back to a register on every cycle. It prevents unnecessary changes, ensuring the stored value is available for later use by other parts of the circuit.

Clock Cycles and Performance. The interaction between combinational and state elements defines the CPU's timing and maximum performance:

- **Data Flow**: Data moves from one state element (register) → through combinational logic (ALU) → to the next state element (register).
- **Timing Constraint**: The data transformation must be complete before the next clock edge arrives. Therefore, the longest delay through any combinational logic path in the entire circuit determines the minimum possible clock period (τ).
- **Clock Period**: The CPU's clock period (τ) must be set slightly longer than this longest delay to ensure all signals stabilize and valid data is ready for capture by the registers on the next clock edge.

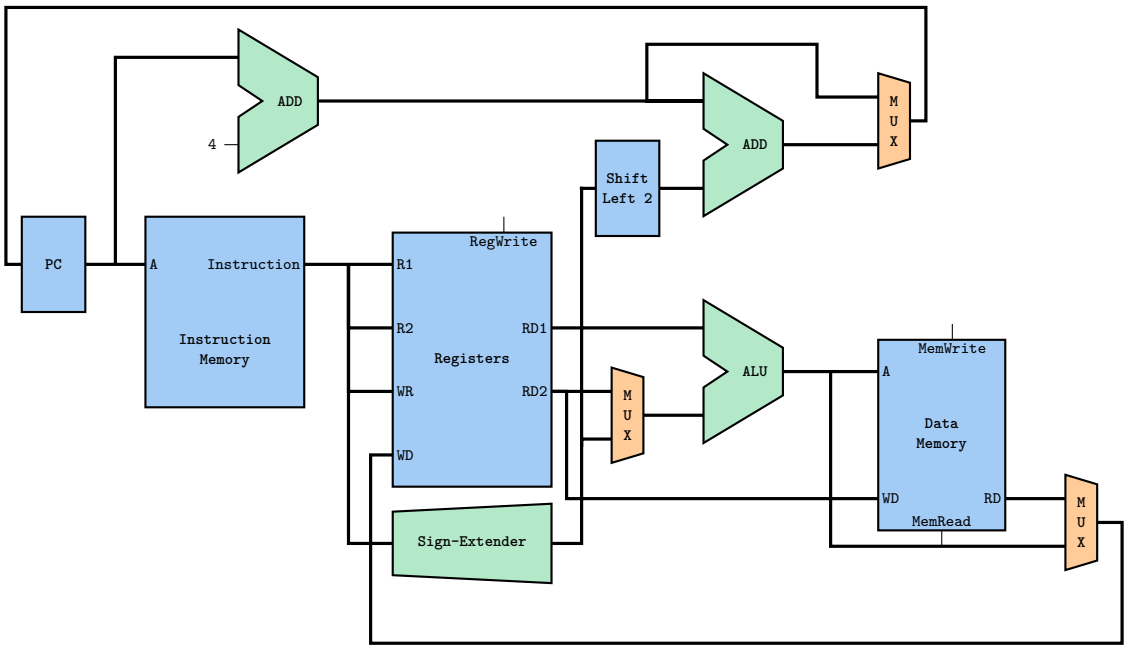

Figure 2 / Full Datapath.