Chapter Five

The Processor: Datapath and Control
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, j
• Generic Implementation:
  – use the program counter (PC) to supply instruction address
  – get the instruction from memory
  – read registers
  – use the instruction to decide exactly what to do
• All instructions use the ALU after reading the registers
More Implementation Details

- Abstract / Simplified View:

**Two types of functional units:**
- elements that operate on data values (combinational)
- elements that contain state (sequential)
State Elements

- Unclocked vs. Clocked
- Clocks used in synchronous logic
  - when should an element that contains state be updated?

![Diagram showing cycle time, rising edge, and falling edge]
An unclocked state element

- The set-reset latch
  - output depends on present inputs and also on past inputs
Latches and Flip-flops

- Latches and flip-flops are the simplest memory elements.
- Output is equal to the stored value inside the element (don't need to ask for permission to look at the value)
- Change of state (value) is based on the clock
- Latches: whenever the inputs change, and the clock is asserted
- Flip-flop: state changes only on a clock edge (edge-triggered methodology)

"logically true", ?could mean electrically low

A clocking methodology defines when signals can be read and written. Wouldn't want to read a signal at the same time it was being written.
D-latch

- Two inputs:
  - the data value to be stored (D)
  - the clock signal (C) indicating when to read & store D
- Two outputs:
  - the value of the internal state (Q) and its complement
- When the latch is open (C asserted), the value of Q changes as D changes — transparent latch.
D flip-flop

- Flip-flops are not transparent
- Output changes only on the clock edge
- The first latch, called the master, is open and follows the input D when C is asserted. When the clock input falls, the first latch is closed, but the 2nd latch, called the slave, is open and gets its input from the output of the master latch.
Set-up time and Hold time

- **Set-up time**: the minimum time that the input must remain valid before the clock edge
- **Hold time**: the minimum time that the input must be valid after the clock edge (usually very small)
Our Implementation

- An edge triggered methodology
- Typical execution:
  - read contents of some state elements,
  - send values through some combinational logic
  - write results to one or more state elements
A register file consists of a set of registers that can be read and written by supplying a register number to be accessed.

Built using D flip-flops and decoders (specify register number)

Read part (left): supply a register number as input, and the output is the information stored in that register.

A register file with 2 read ports and 1 write ports. (right)
Register File

- Write part: need 3 inputs: a register number, the data to write, and a clock that controls the writing into the register.
- Note: we still use the real clock to determine when to write
Simple Implementation

- Basic components:
  - two state elements instruction memory and program counter are needed to store and access instructions.
  - An adder is needed to compute the next instruction address.
- Since the instruction memory is read-only, we can treat it as combinational logic.
Fetching instruction and incrementing PC

- A portion of the datapath used for fetching instructions and incrementing Program Counter

```
Instruction memory

PC  Read address

Instruction

Add

4
```
R-Format ALU operations

- R-format instruction has 3 register operands, 2 read and 1 write
Datapath for R-type Instruction

Instruction

Read register 1
Read register 2
Write register
Write data

Read data 1
Read data 2

ALU operation
Zero

ALU result
RegWrite

©1998 Morgan Kaufmann Publishers
Load and Store Instructions

- Load and store instructions compute a memory address by adding the base register, to a 16-bit signed offset field contained in the instruction.

![Diagram of data memory unit and sign-extension unit]

a. Data memory unit
b. Sign-extension unit
Datapath for load and store instructions
J-type Instruction

- Branch datapath
  - Needs to compute the branch target address
    - PC+4 is the address of the next instruction
      - Offset field is left-shifted two bits to make a world offset.
    - Needs to compare register contents
Branch Datapath

PC + 4 from instruction datapath

Add Sum Branch target

Shift left 2

ALU operation

ALU Zero To branch control logic

Instruction

Read register 1
Read register 2 Registers
Write register
Write data

Read data 1

Read data 2

RegWrite

Sign extend

16

32

3
Building the Datapath

- Use multiplexors to stitch them together