

Lecture 10

MIPS Processor

School of Computer Science and Engineering
Soongsil University

4. The Processor

- 4.1 Introduction**
- 4.2 Logic Design Conventions**
- 4.3 Building a Datapath**
- 4.4 A Simple Implementation Scheme**
- 4.5 An Overview of Pipelining**
- 4.6 Pipelined Datapath and Control**
- 4.7 Data Hazards: Forwarding versus Stalling**
- 4.8 Control Hazards**
- 4.9 Exceptions**
- 4.10 Parallelism via Instructions**
- 4.11 Real Stuff: The ARM Cortex-A8 and Intel Core i7 Pipelines**
- 4.12 Going Faster: Instruction-Level Parallelism and Matrix Multiply**

4.1 Introduction

A Basic MIPS Implementation

- **We will examine two MIPS implementations**
 - ❖ A simplified version
 - ❖ A more realistic pipelined version
- **Simple subset, shows most aspects**
 - ❖ Memory-reference instructions
 - ◆ lw, sw
 - ❖ Arithmetic-logical instructions
 - ◆ add, sub, and, or, slt
 - ❖ Branch instructions
 - ◆ beq, j

An Overview of the Implementation

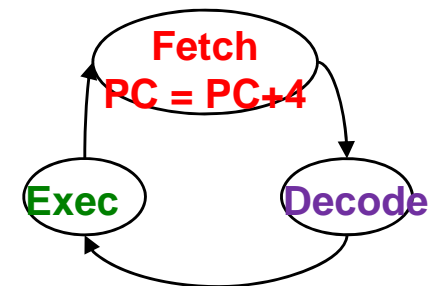
■ The First 2 Steps of Every Instruction

1. The first step

- ❖ Instruction fetch
 - Send PC to memory
 - Send READ signal to memory
- ❖ $PC = PC + 4$

2. The second step

- ❖ Opcode decoding
- ❖ Register prefetch



The Third Step

- **Memory-reference instructions**

- ❖ Use ALU for an effective address calculation

- **Arithmetic-logical instructions**

- ❖ Use ALU for the operation execution

- **Branch instructions**

- ❖ Use ALU for comparison

The Final Step

- **Memory-reference instructions**

- ❖ **Store**: access the memory to write data
- ❖ **Load**: access the memory to read data

- **Arithmetic-logical instructions**

- ❖ Write the data from the ALU back into a register

- **Branch instructions**

- ❖ Change PC based on the comparison

An Abstract View of the MIPS

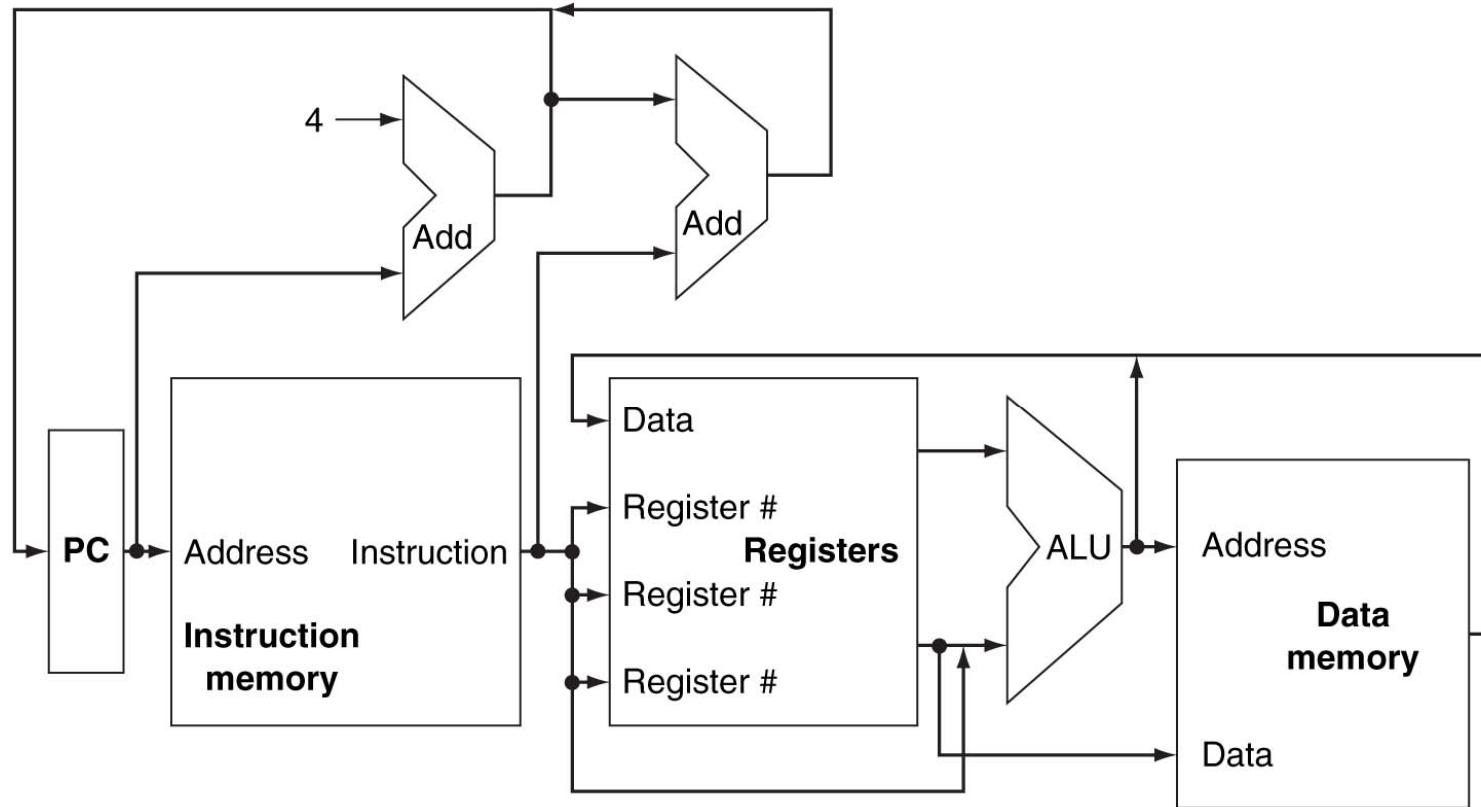
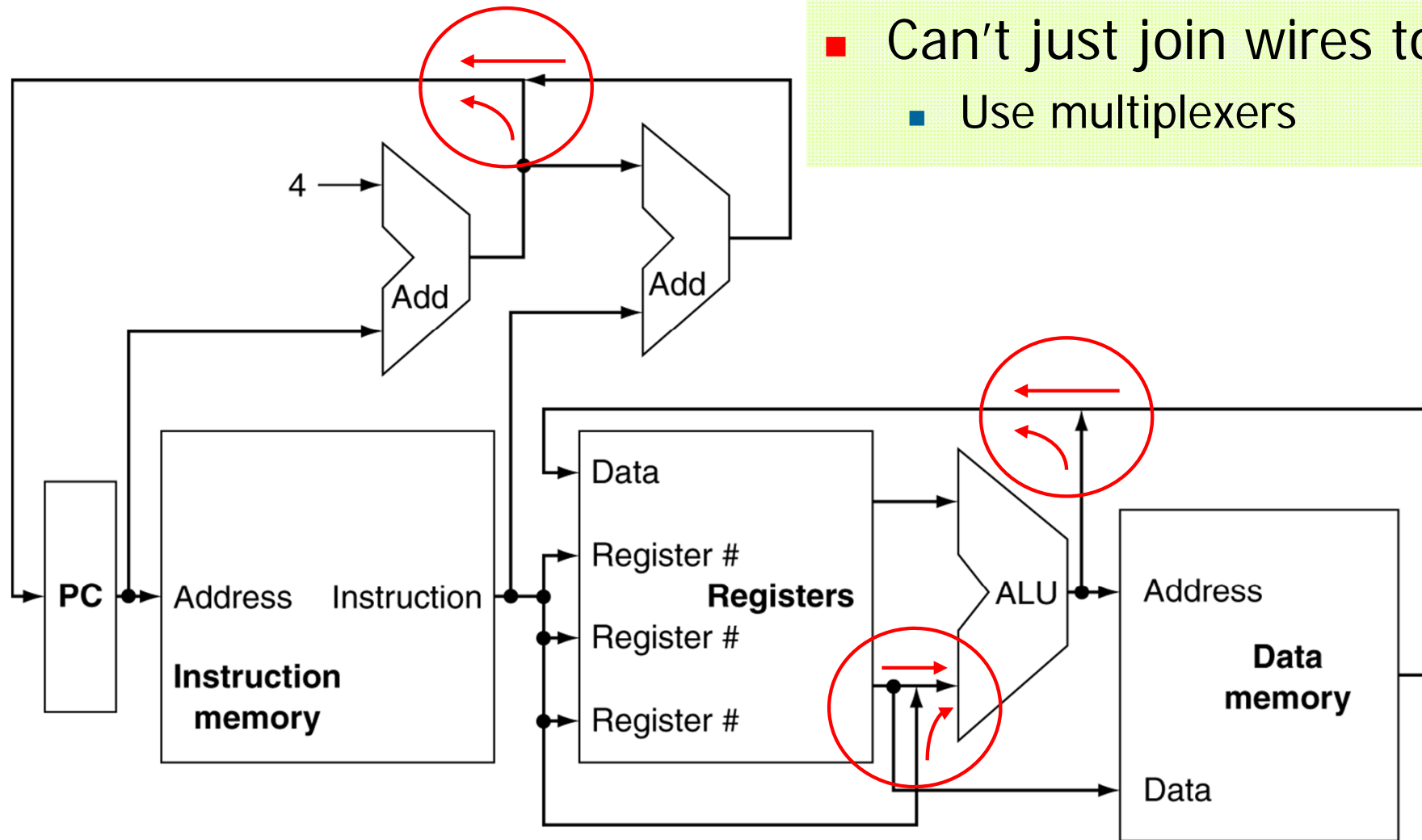


Figure 4.1

Multiplexers



- Can't just join wires together
 - Use multiplexers

Basic Implementation of the MIPS Subset

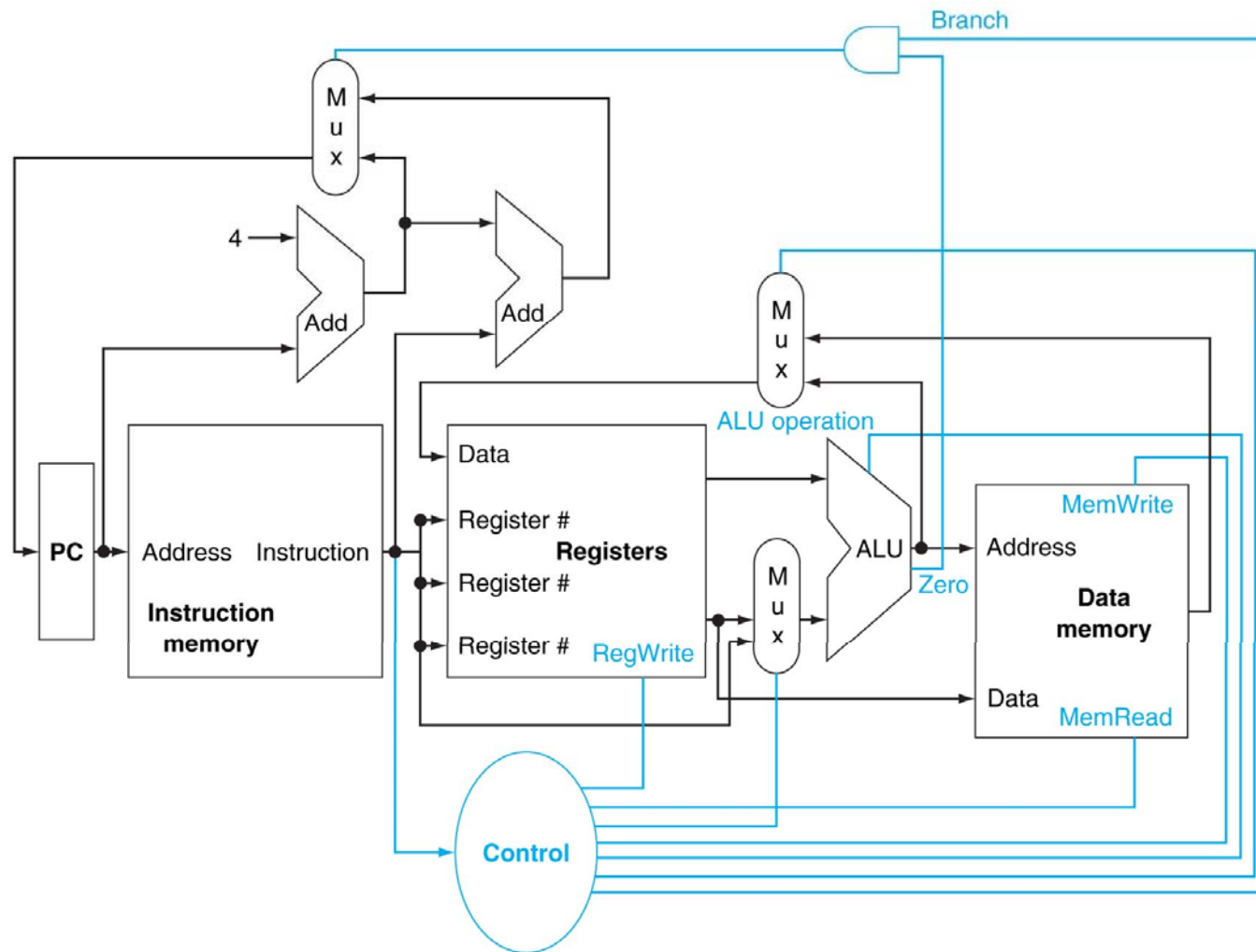


Figure 4.2

4.2 Logic Design Conventions

- **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

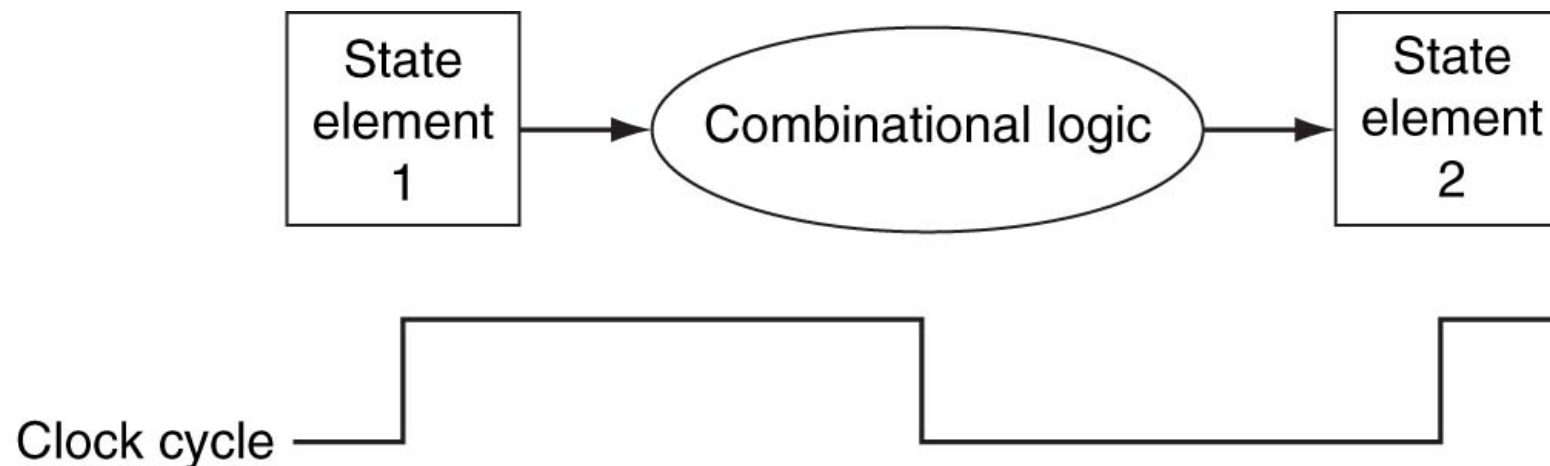
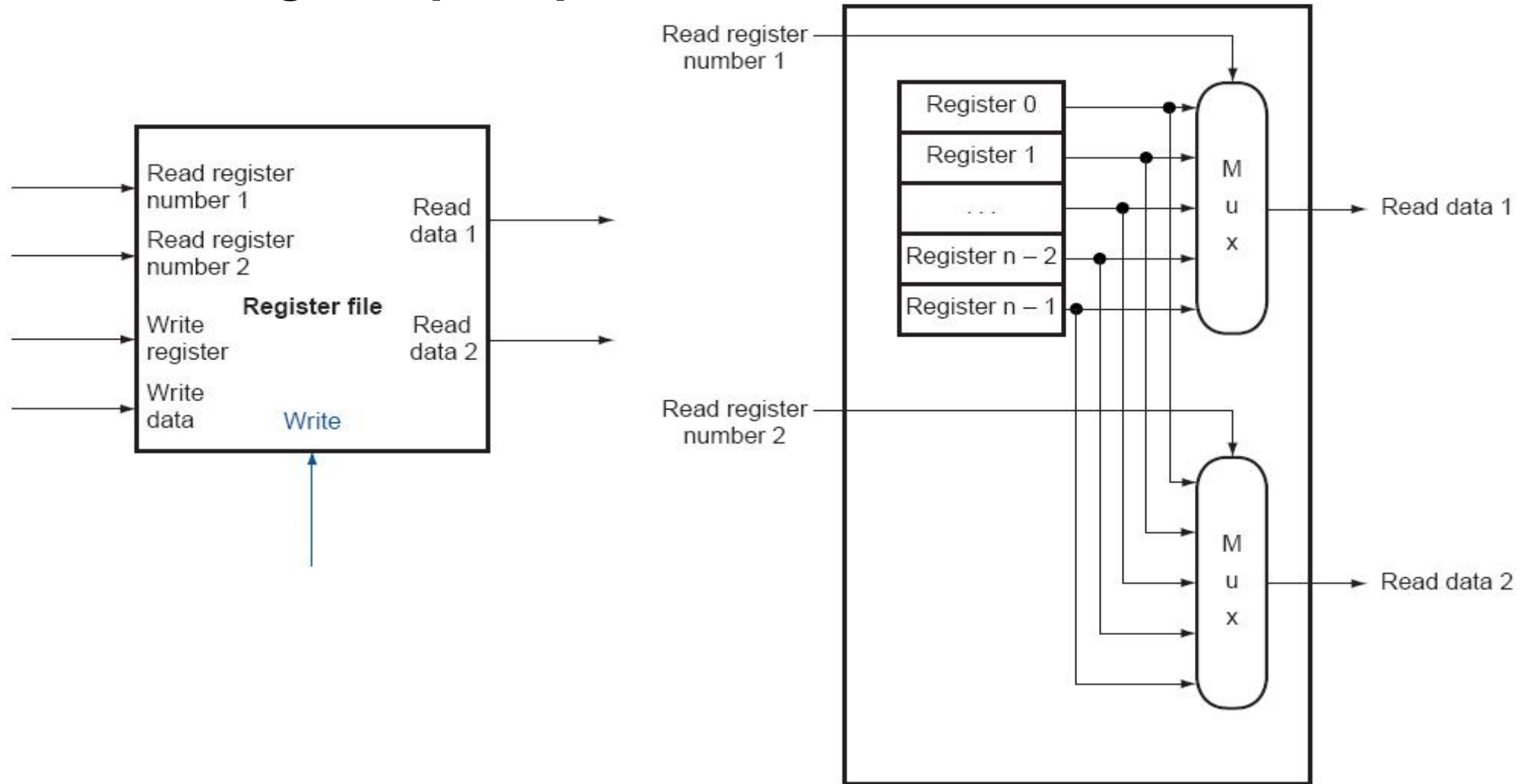


Figure 4.3

Register File - Output

- Built using D flip-flops



Register File - Input

