Introduction to Computer Organization and Architecture

Topic 6. The central processing unit

Han, Nguyen Dinh (han.nguyendinh@hust.edu.vn)

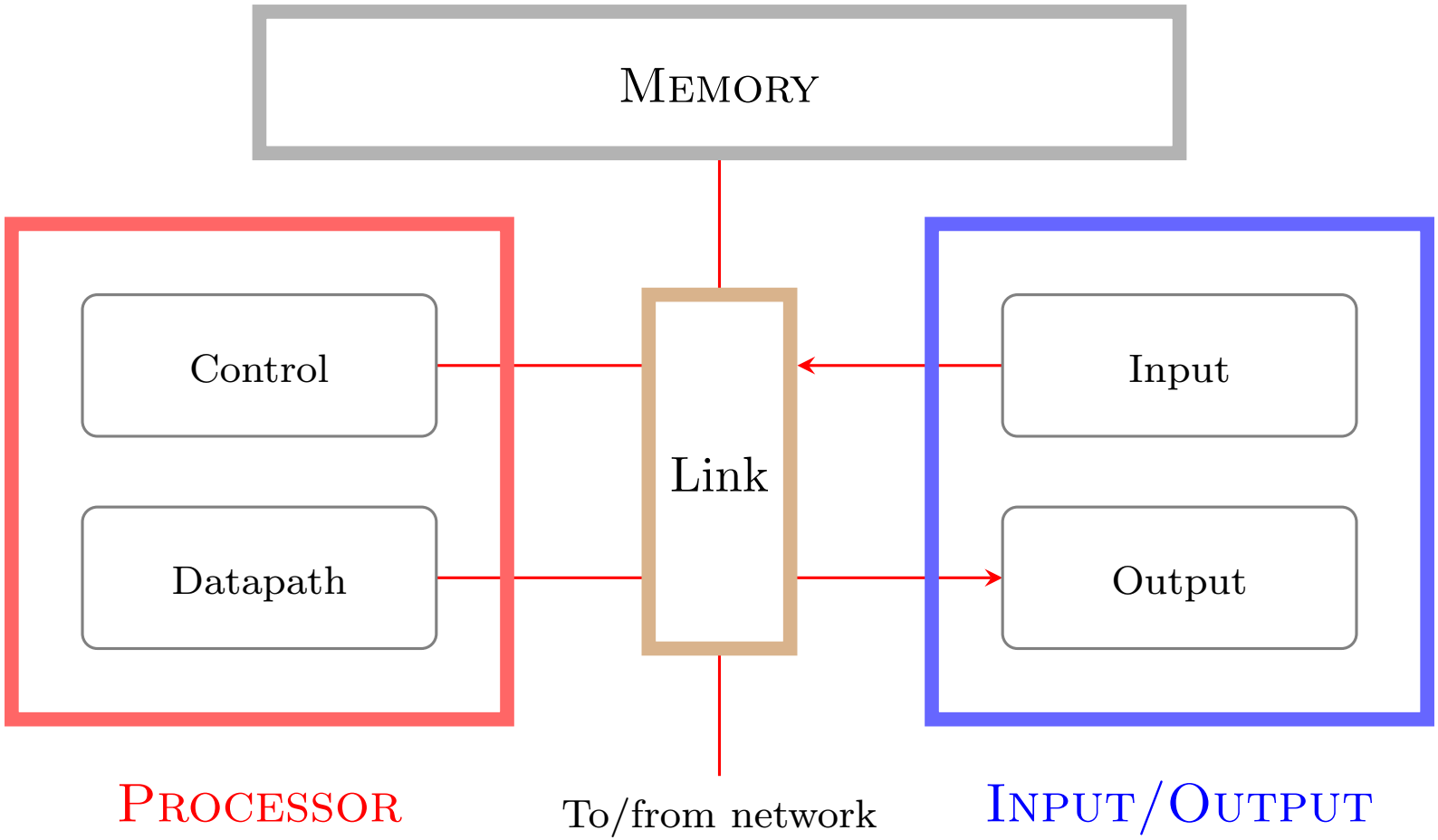
Faculty of Mathematics and Informatics
Hanoi University of Science and Technology

Outline

1

Introduction

The abstract overview of a digital computer



A basic MIPS implementation

We will be examining an implementation that includes a subset of the core MIPS instruction set:

- The memory-reference instructions `load word (lw)` and `store word (sw)`
- The arithmetic-logical instructions `add`, `sub`, `AND`, `OR`, and `slt`
- The instructions `branch equal (beq)` and `jump (j)`

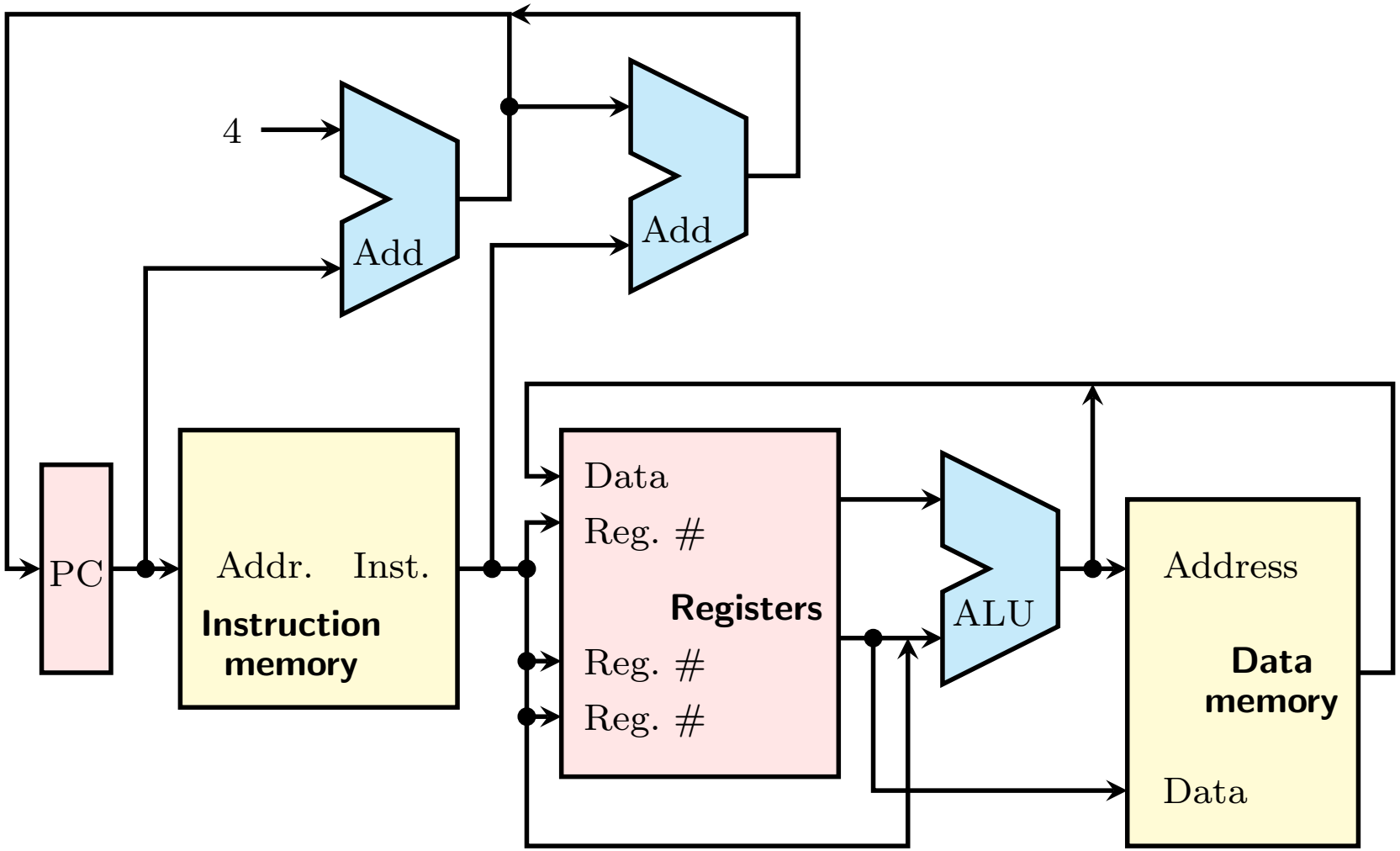
Note that, this subset does not include all the integer instructions. However, it illustrates the key principles used in creating a datapath and designing the control. The implementation of the remaining instructions is similar

An overview of the implementation

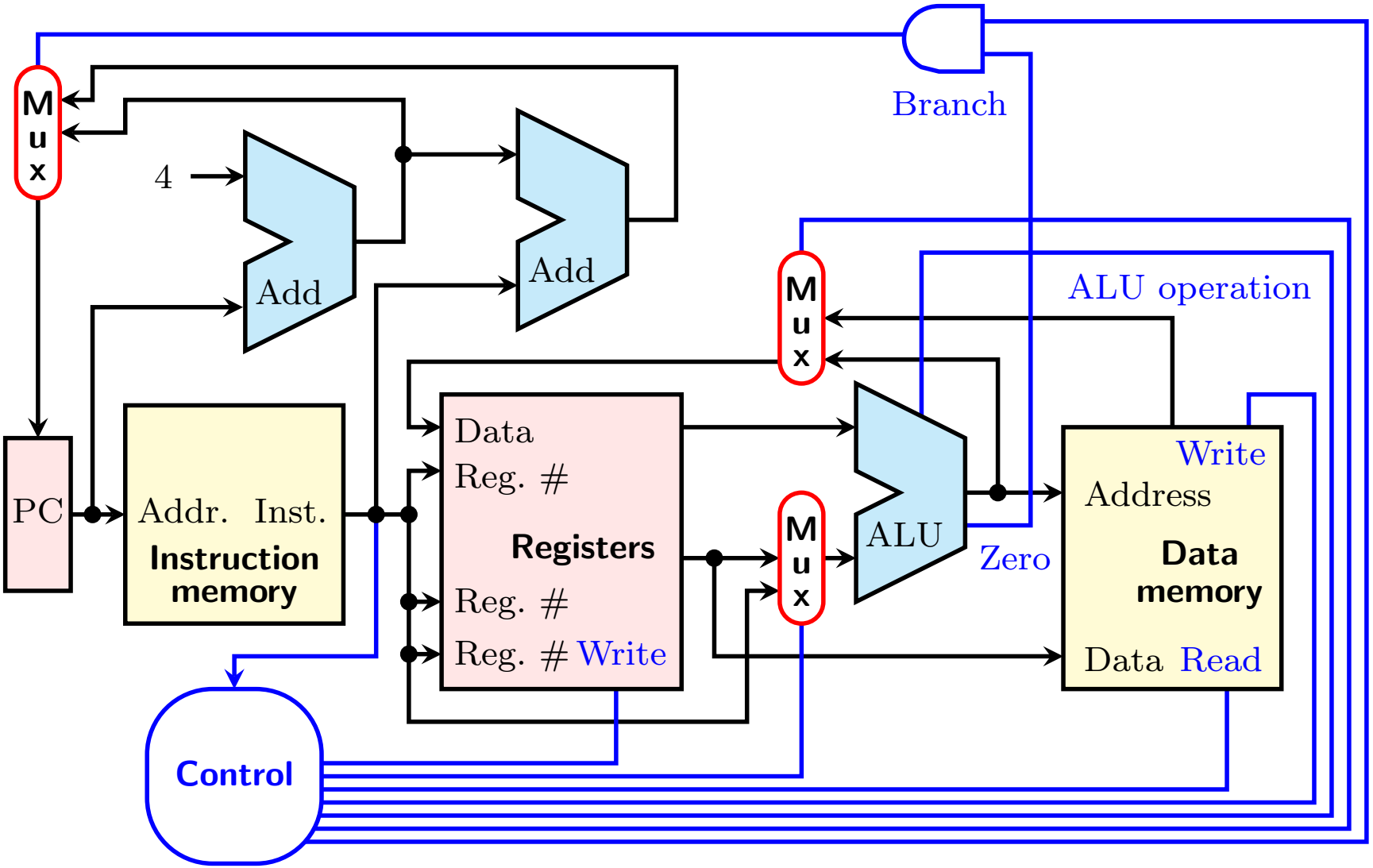
The simplicity and regularity of the MIPS instruction set simplifies the implementation by making the execution of many of the instruction classes (i.e. memory-reference, arithmetic-logical, and branches) similar. For every instruction, the first two steps are identical:

1. Send the *program counter* (PC) to the memory that contains the code and fetch the instruction from that memory
2. Read one or two registers, using fields of the instruction to select the registers to read. For the load word instruction, we need to read only one register, but most other instructions require reading two registers

The implementation of the MIPS subset



The basic implementation of the MIPS subset



2

Logic design conventions

3

Building a datapath

Logic design conventions

Let's run

pieces of Verilog codes given in

Sections B.4 and B.5, Appendix B of the textbook!

4

A simple implementation scheme

A simple implementation scheme

Let's share your experience!

Group 3 (next week):

Design and simulate a 8-bit CPU using Verilog

Group 9 (next week):

Design and simulate a pipeline CPU using Verilog

Group 6 (next two week?):

Design and simulate a computer keyboard using Verilog

5

Pipelined datapath and control

An overview of pipelining and hazards

Pipelining is an implementation technique in which multiple instructions are overlapped in execution. This helps to reduce the overall instruction execution time, hence enhancing the performance.

Today, *pipelining* is nearly universal. The division of an instruction into five stages means a five-stage pipeline, which in turn means that up to five instructions will be in execution during any single clock cycle.

If the stages are perfectly balanced, then the time between instructions on the pipelined processor (T) — assuming ideal conditions — is equal to

$$\mathrm{T} = \frac{\text{Time between instructions}_{\text{nonpipelined}}}{\text{Number of pipe stages}}$$

Your fourth assignment

Students work individually to carry out the following tasks:

1. Investigate insight-fully the basics of logic design through sections B.1 to B.5 of Appendix B in the textbook
2. Define basic gates (AND, OR, NOT) and the combinational logic circuits (decoder, multiplexor) using Verilog
3. Try to run Verilog codes, given in Section B.5, Appendix B of the textbook, that define the MIPS ALU
4. Collect and include all tested Verilog codes in a writing report (in Word or Latex)
5. Submit your results

NOTE $\rightsquigarrow$ Please convert your report to PDF when you submit it

Advanced topics

Let's share your experience!

Group 3 (next week - Week#11):

Design and simulate a 8-bit CPU using Verilog

Group 9 (Week#11):

Design and simulate a pipeline CPU using Verilog

Group 4 (Week#12):

Design and simulate a simple calculator using Verilog

Advanced topics (cont.)

Let's share your experience!

Group 6 (Week#12):

Design and simulate a computer keyboard using Verilog

Group 1 (Week#13):

Design and simulate a digital safe lock using Verilog

Group 2 (Week#13):

Design and simulate a parking system using Verilog

Advanced topics (cont.)

Let's share your experience!

Group 5 (Week#14):

Design and simulate a simple stop watch using Verilog

Group 7 (Week#14):

Robotic Car design and implementation using Verilog

Group 8 (Week#15):

Design and simulate a pong game device using Verilog

Group 10 (Week#15):

Design and simulate a traffic light controller using Verilog

The central processing unit

Thank you very much for your attention!