





Lab 8 – Design of Instruction and Data Memories

Objectives

In this lab, we will develop modules for instruction and data memories.

Section	
a) <u>Introduction</u>  A brief overview of this lab and some background about the instruction and data memories	05
b) <u>Implementation</u>  In this section, you will develop a module for instruction memory and verify its functionality.	60
<u>Exercise</u>  In this section, you will develop and verify the module of Data Memory.	60





a. Introduction

Instruction and Data memories are essential components of a processor. We will develop modules for the Instruction and Data memory separately. The instruction memory is used to store instructions. During the processor execution, it is required to only read the instruction from the instruction memory and not to write any data or instruction in it. Hence, we could say that the instruction memory is the read-only memory. However, the data memory is used to both read or write data.

Recall that the size of each instruction is 32-bits, and the data (to be written in memory or to obtain from memory) is 64-bits wide. Therefore, the size of each location in an instruction memory is 32-bits and that of data memory is 64-bits.

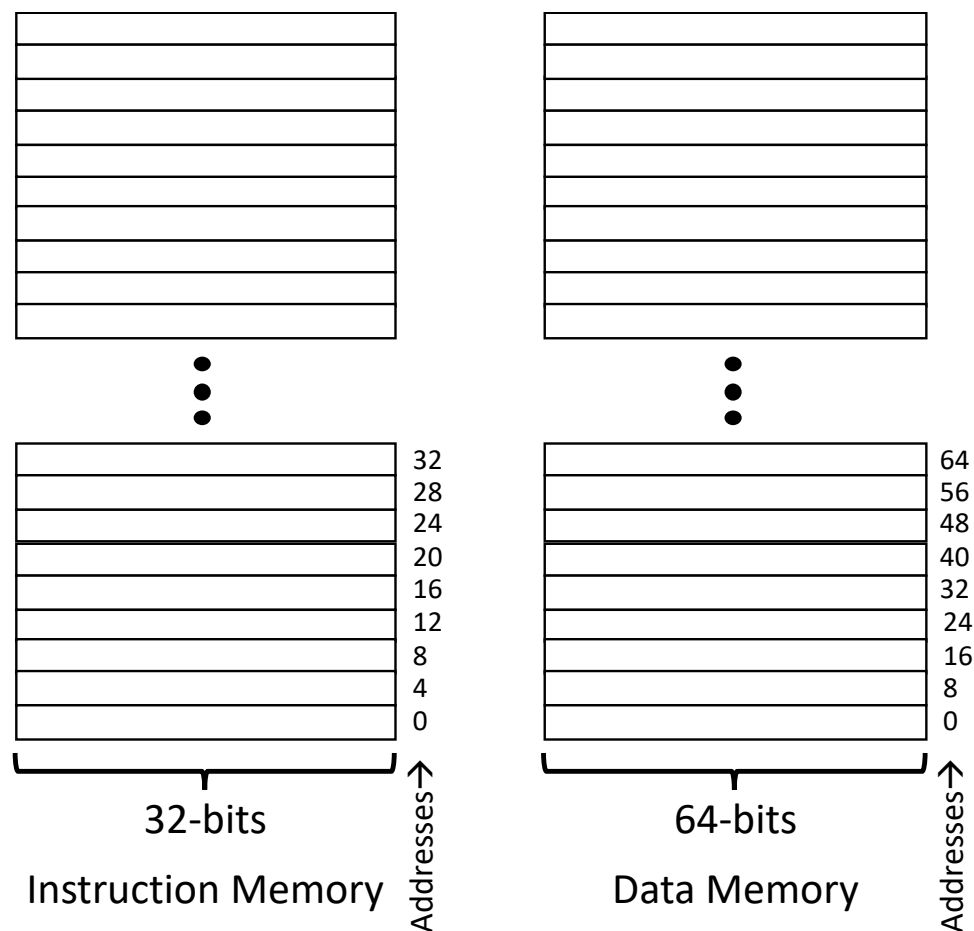


Fig. 8.1. Instruction and Data memories.

In lectures, we have been conceptually visualizing a memory as a stack of memory locations, as shown in Fig. 8.1. Each location in instruction memory is 32-bits (4 bytes) wide and that of data memory is 64-bits (8 bytes) wide.



b. Implementation

In this lab, we will design an instruction memory with each location 8-bits wide. Hence, four memory locations will be required to store a 32-bit instruction in an instruction memory, as shown in Fig. 8.2.

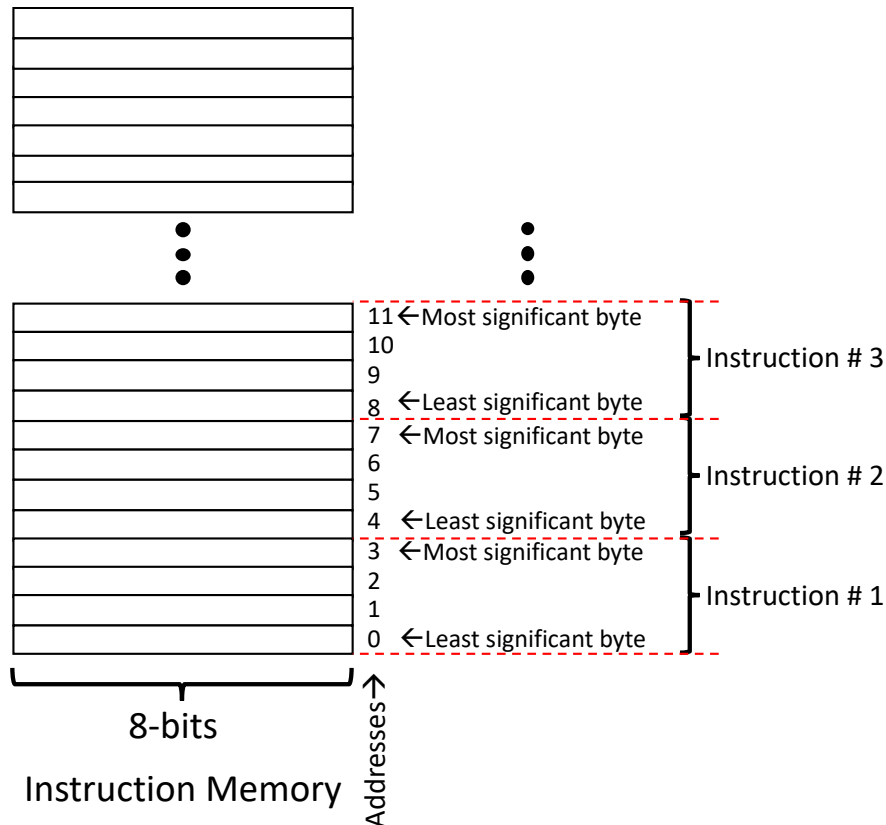


Fig. 8.2. 8xN bytes instruction memory, where N specify the number of n memory locations each of which are 8-bits (or 1-byte) wide.

i. Lab Task 01

Design a module named, `Instruction_Memory`, having a 64-bit input, `Inst_Address`, and a 32-bit output, `Instruction`. Now, declare a 1x16 bytes instruction memory internally in the module and initialized its contents as shown in Fig. 8.3.



Similar procedure is required to declare an instruction memory that you followed while constructing a register file in lab 04.

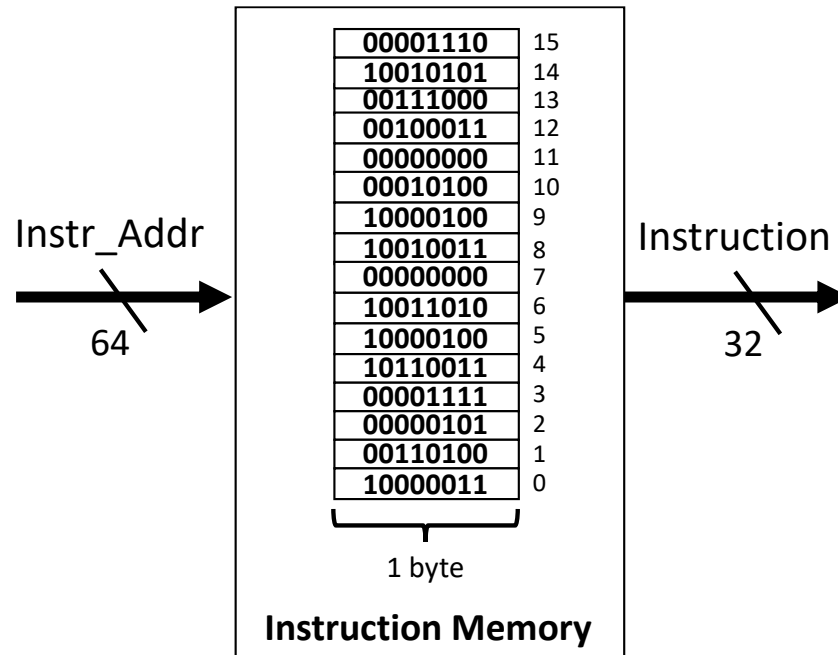


Fig. 8.3. 1x16 bytes (or 8x16 bits) instruction memory.



It should be noted that the 64-bits width of `Inst_Addr` input can access 2^{64} memory locations. Although, we need to have this input 4 bits wide only (to access 16 memory locations), but we are keeping its width to 64 bits just to make synchronization with other modules, which we will develop in upcoming labs.

The behavior of this module should be designed as such that whenever an `Inst_Address` field is changed, the 32-bit instruction corresponding to the `Inst_Address` should appear at the 32-bit output port, `Instruction`.

For example, if the `Inst_Address` is 0, the consecutive 4 bytes (byte no. 0 to 3) should appear at the 32-bit `Instruction` output. Similarly, when the value of `Inst_Address` is 8, the corresponding next four bytes (8 to 11) should be placed at the 32-bit output port, `Instruction`.



While placing four bytes on the 32-bit `Instruction` port, make sure that the bytes are in correct order. That is, the least-significant byte should be the right most byte and that the most-significant byte should be the left most byte in the 32-bit `Instruction` output.

Write a testbench and verify the functionality of this module.



Exercise

The top-level diagram of a Data Memory is shown in Fig. 6.4. Write a module named, `Data_Memory`, with inputs and outputs as shown in Fig. 8.4. The signals shown in blue color are the control signals. We will design the control unit in upcoming labs.

The data at the input port, `Write_Data`, should only be written at the positive edge of `clk` signal and when `MemWrite` signal is asserted (i.e. High). The data can be read from memory at any instant whenever the `Mem_Addr` value changes and when `MemRead` signal is asserted.

In contrast to instruction memory, here 8 consecutive bytes should be placed at the output. For example, if the value of `Mem_Addr` is 0, the data placed at memory locations 0 to 7 should be placed at the output port `Read_Data`. Similarly, if `Mem_Addr` is 16, the data of memory locations 16 to 23 should be placed at the output port `Read_Data`.

Initialized the memory with random data, write a testbench and verify its functionality.

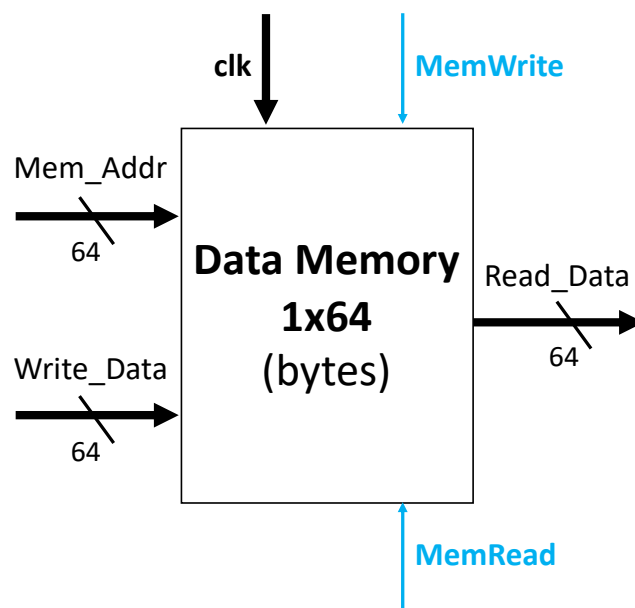


Fig. 8.4. IOs of Data Memory module. There are 64 memory locations, each of which is 1 byte wide.

Computer Architecture

Spring 20

Lab 08

Marks Distribution:

Task	LR2: Design/Code	LR5: Results
Task 1 (Instruction Memory)	20 Points	10 Points
Task 2 (Data Memory)	20 Points	10 Points
Total		

Marks Obtained:

Task	LR2: Design/Code	LR5: Results
Task 1 (Instruction Memory)		
Task 2 (Data Memory)		
Total		