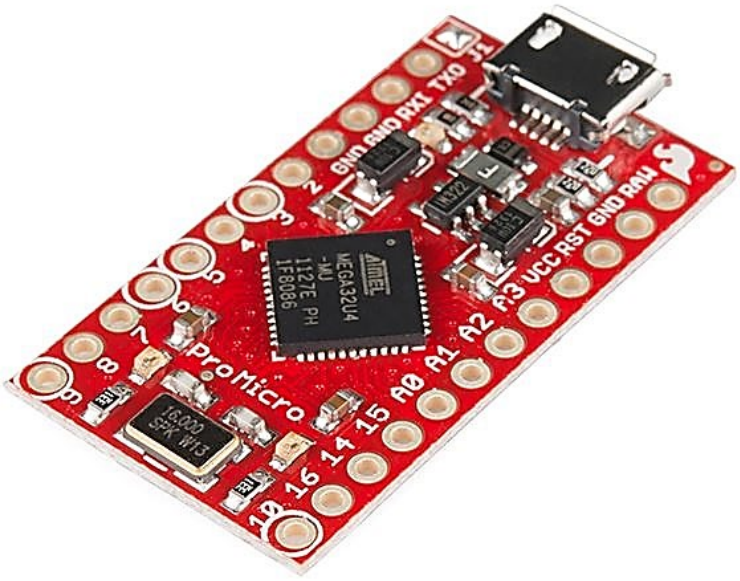
Electrical and Computer Engineering
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas
Dr. Tooraj Nikoubin



Project #1: (1) Data Memory and Instruction Memory (2) Opcode selection

EEDG/CE 6302
Microprocessors and Embedded Systems

Objectives:

- Designing and simulating synthesizable MCU memories
- Getting familiar with writing codes in hardware description languages (VHDL or Verilog)
- Writing testbench to test the written codes
- Assigning Opcodes to the given instruction set
- Specifying the action of each instruction related to its Opcode

1. Introduction:

In this project, you will write your first HDL code. This code will perform the task of two memories in your MCU: Program Memory and Data Memory.

2. Memory

2.1. Program Memory:

Also called Instruction Memory, it is usually made of non-volatile memory in real-world applications. Examples of non-volatile memories include: ROM, PROM, EPROM, EEPROM, Flash, etc. Once it is programmed, it will not change. Your program that you want to run on the microprocessor is stored in this memory. At address of the program memory, an instruction is saved. It must be fetched and run by your MCU. To access each location of the memory, you need an input address port, and to read the value at that location, you need an output data port. Thus, it needs to have following interfaces:

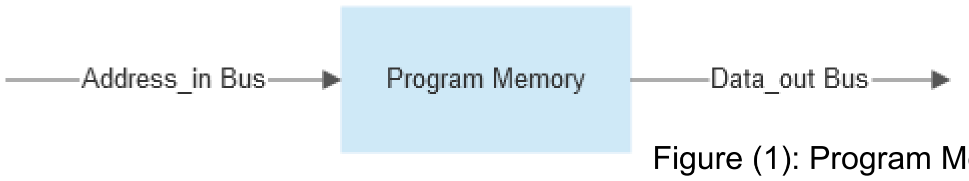


Figure (1): Program Memory

2.2. Data memory:

Also called Work Memory, it is usually made of volatile memory in real-world applications. DRAM and SRAM are examples of Data Memory. It is not directly programmed by the user; the instructions manipulate Data Memory to save or restore data. Two simplest instructions which work directly with Data memory are LOAD and STORE. Although there are many instructions in CISC microprocessors that have directly access to the Data Memory, it has been limited to LOAD and STORE in RISC microprocessors. To access each location of the memory, you need an input address port, to read the value at that location, you need an output data port, and to write the value to that location, you need an input data port. Moreover, an input control signal should be available to instruct the memory to perform either read or write operation. Thus, it needs to have following interfaces:

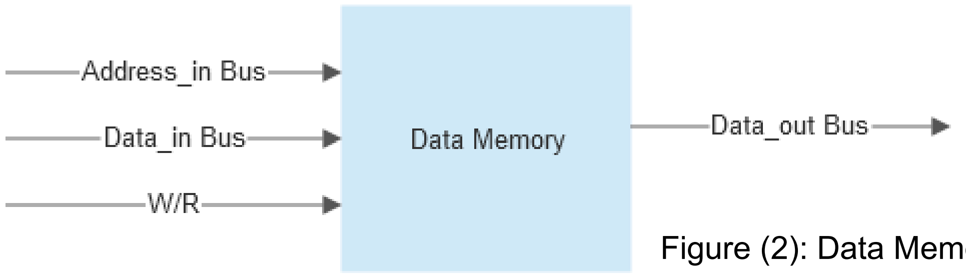


Figure (2): Data Memory

2.3. Implementation:

Sizes of Program Memory and Data Memory are not necessarily the same. Even the size of the address bus of each memory may be different. In this project, following sizes are specified:

	Data Memory	**Program Memory**
Address Bus	8	8
Data_in	8	N/A
Dara_out	8	17
W/R	1	N/A

3. Opcodes:

You are given a list of opcodes that your MCU must be capable to run. Since opcode is 5-bit long, the MCU can perform up to 32 different instructions. Among them, NOP (No OPeration or nothing to do), IN (get INput from input port), INK (get INput from Keyboard) and OUT (set OUTput to output port) are mandatories for all students. Remaining opcodes are randomly assigned to each of you. You need to create a table for the given opcodes with the following columns:

#	Operation	Symbolic Notation	Opcode	Action	ALU/FU requirement
1	No Operation	NOP	"00000"	None	No
2	Input	IN		R[DR] <- Input Port	No
3	AND	AND		R[DR] <- R[SA] ^ R[SB]	Yes
4	Move A	MOVA		R[DR] <- R[SA]	No

Except the NOP operation, which has a dedicated Opcode of “00000”, you can assign arbitrary Opcodes to your instruction set. For each instruction, you need to write the action of the instruction in the action column. other examples can be found in the reference text.

4. Project requirements:

- Write a code in an HDL for Data Memory and Program Memory separately.
- Write a testbench for each code. Verify and show both your coding works correctly using waveforms. Your testbench should include all conditions that may happen in normal operation of each module.
- Take a screenshot of the top-level schematic view of the written code. The schematic can be generated by Xilinx ISE or any other HDL synthesizers.
- Each student should submit a report no more than 4 pages (excluding cover page, TOC, appendix, etc).
- Do not include your codes in the report. Submit your HDL codes along with their testbenches separately.
- Complete Opcode table in your report
- Answer to the questions at the end of this manual in the report.

5. Questions:

1. What are the long formats of ROM, PROM, EPROM and EEPROM? Explain briefly (2 or 3 lines) their pros, cons and differences in a table. Also consider Flash memory in your comparisons.

2. Why is stored data in the Program Memory larger than Data Memory (in this project)?

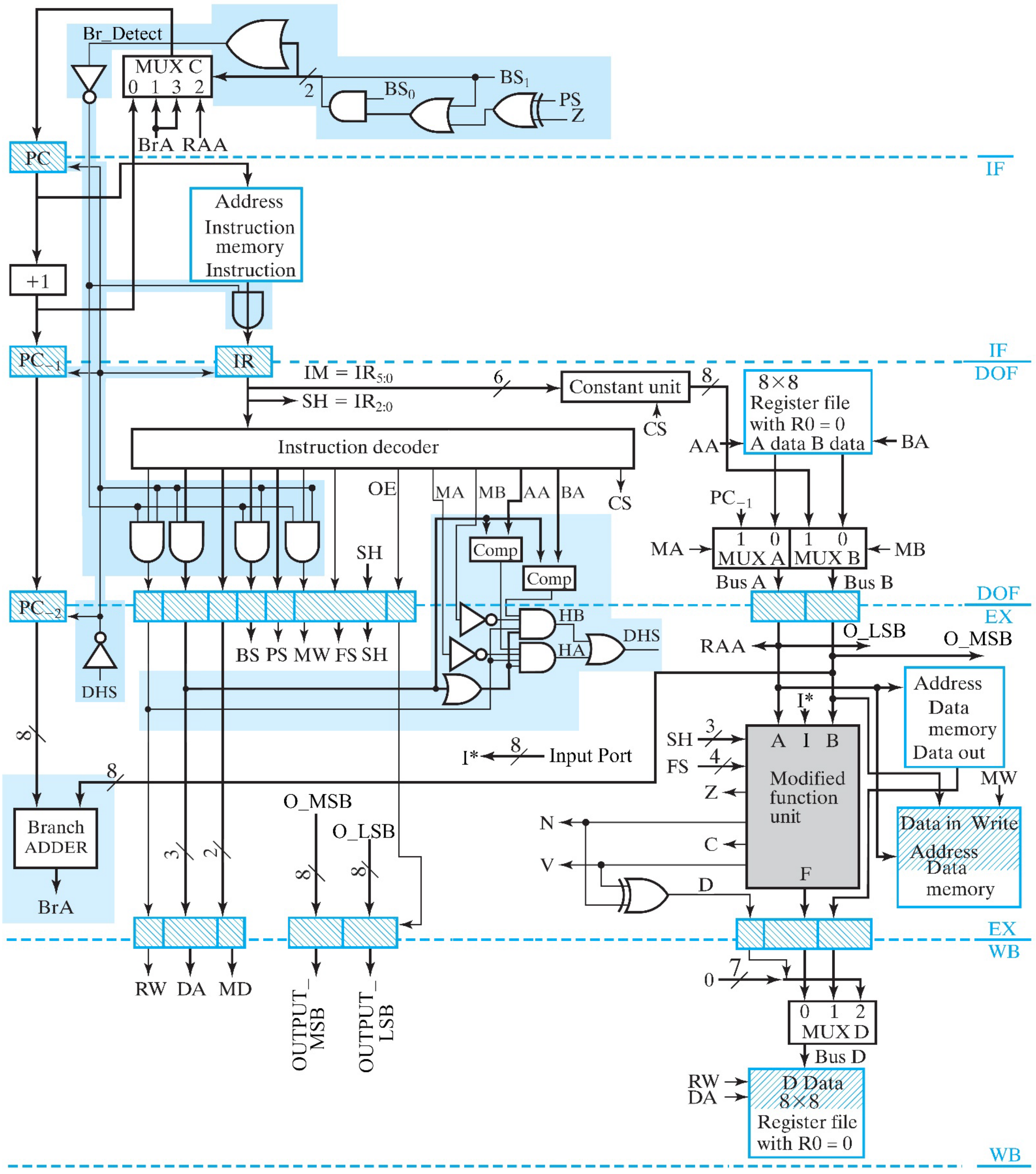


Figure (3): Block diagram of the MCU

Reference:

Mano, M. M. AND Kime, C. R. AND Martin M. “*Logic and Computer Design Fundamentals*“, 5th ed. Hoboken, NJ: Pearson Higher Education Inc., 2015, pp 585-615.