**Laboratory #8 Complete Processor**

**EE 310 Fundamentals of Computer Engineering**

College of Engineering, Informatics and Applied Sciences

Northern Arizona University

# Objective

In this lab, the student will fully integrate, test, and demonstrate all parts of the microprocessor. The student will complete the design of the control unit for the fetch and execution of all instructions. Testing will be by simulation on ModelSim and by board demonstration.

# Important Concepts

1. For full details about the timing and sequencing of the various instructions, please consult the “Lab 8 Tutorial.ppt” file. There are several key points:
   1. Two of the instructions (NOP and CLR) are only byte-wide, therefore fetch needs only one memory read cycle.
   2. Every memory read operation requires two clock cycles. The first clock cycle does MAR 🡨 PC. The second clock cycle does destination 🡨 MDR.
   3. Pay close attention to when control signals need to assert and when next state decisions must be made.
2. The control unit must expand to accommodate all states necessary for implementing the full instruction set. It will be up to you to define the names and behaviors of these additional states for executing the various instructions.

**Activity #1 Define the Full Control Unit**

In the following table, the Start state and the instruction fetch states have already been listed for you.

**Q1:** Complete the table by defining and naming additional states as needed to control the execution of every instruction in the instruction set. If you need more space, attach additional papers. Each line should define exactly what happens in that state and list which state must be next.

**Q2:** On a separate page, draw a state diagram to illustrate the flow of the control unit.

|  |  |  |  |
| --- | --- | --- | --- |
| **State**  **Name** | **State**  **Code** | **Description / Action** | **Next**  **State** |
| **Start** | 0 | Immediately on RESET  No action | PrepU |
| **PrepU** | 1 | Prepare for upper byte instruction fetch  MAR 🡨 PC | FetchU |
| **FetchU** | 2 | Fetch upper byte of instruction  IRU 🡨 MDR, PC 🡨 PC+1 | PrepL? |
| **PrepL** | 3 | Prepare for lower byte instruction fetch  MAR 🡨 PC | FetchL |
| **FetchL** | 4 | Fetch lower byte of instruction  IRL 🡨 MDR, PC 🡨 PC+1 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Activity #2 Write System Verilog and Simulate Processor in Full Operation**

Write the System Verilog module named up3, with outputs that expose the internal signals needed later for use in the testbench (all except Z and the flags, NFLG and ZFLG). This module must fully implement the control unit as specified in the table and state diagram from Activity #1. Create a **.mif** file to initialize the first fifteen memory locations with the following hex data: 02, E7, 03, 10, 09, 0B, 10, 0A, 0E, 03, 0F, 93, 04, 01, 10. This is a short machine language program we will execute to test our design.

**Q3:** Using the given memory data, figure out the data stored in each register while each instruction is executed. Fill your results in the table below. Assume the Reset occurs and returns to zero before the 1st Rising edge of the CLK.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Reset | CLK rising edge | PC | MAR | MDR | Opcode | Value /addr | AC | State name | State No. |
| 1->0 | 1st | 00 | 00 | 02 | 00 | 00 | XX | Start | 0 |
| 0 | 2nd | 00 | 00 | 02 | 00 | 00 | XX | PrepU | 1 |
| 0 | 3rd | 01 | 00 | 02 | 02 | 00 | XX | FetchU | 2 |
| 0 | 4th | 01 | 01 | E7 | 02 | 00 | XX | PrepL | 3 |
| 0 | 5th | 02 | 01 | E7 | 02 | E7 | XX | FetchL | 4 |
| 0 | 6th | 02 | E7 | XX | 02 | E7 | E7 | Exec | 5 |
| 0 | 7th | 02 | 02 | 03 | 02 | E7 | E7 | PrepU | 1 |
| 0 | 8th | 03 | 03 | 03 | 03 | E7 | E7 | FetchU | 2 |
| 0 | 9th | 03 | 03 | 10 | 03 | E7 | E7 | PrepL | 3 |
| 0 | 10th | 04 | 04 | 10 | 03 | 10 | E7 | FetchL | 4 |
| 0 | 11th | 04 | 10 | E7 | 03 | 10 | E7 | StoreMem | 5 |
| 0 | 12th | 04 | 04 | 09 | 03 | 10 | E7 | PrepU | 1 |
| 0 | 13th | 05 | 05 | 0B | 09 | 10 | E7 | FetchU | 2 |
| 0 | 14th | 06 | 05 | 0B | 09 | 10 | E7 | PrepL |  |
| 0 | 15th | 06 | 05 | 0B | 09 | 0B | E7 | FetchL |  |
| 0 | 16th | 06 | 0B | 93 | 09 | 0B | E7 | ReadMem |  |
| 0 | 17th | 06 | 0B | 93 | 09 | 0B | 6D | Exec |  |
| 0 | 18th | 07 | 06 | 10 | 09 | 0B | 6D | PrepU |  |
| 0 | 19th | 07 | 06 | 10 | 10 | 0B | 6D | FetchU |  |
| 0 | 20th | 08 | 07 | 0A | 10 | 0B | 6D | PrepL |  |
| 0 | 21th | 08 | 07 | 0A | 10 | 0A | 6D | FetchL |  |
| 0 | 22th | 0A | 0A | 0F | 10 | 0A | 6D | Jump |  |
| 0 | 23th | 0A | 0A | 0F | 10 | 0A | 6D | PrepU |  |
| 0 | 24th | 0B | 0B | 0F | 0F | 0A | 6D | FetchU |  |
| 0 | 25th | 0B | 0B | 93 | 0F | 0A | 6D | PrepL |  |
| 0 | 26th | 0C | 0C | 93 | 0F | 93 | 6D | FetchL |  |
| 0 | 27th | 0C | 0C | 04 | 0F | 93 | 0D | Exec |  |
| 0 | 28th | 0C | 0C | 04 | 0F | 93 | 0D | PrepU |  |
| 0 | 29th | 0D | 0C | 04 | 04 | 93 | 0D | FetchU |  |
| 0 | 30th | 0D | 93 | XX | 04 | 93 | 00 | Exec |  |
| 0 | 31st | 0E | 0D | 01 | 04 | 93 | 00 | PrepU |  |
| 0 | 32nd | 0E | 0E | 01 | 01 | 93 | 00 | FetchU |  |
| 0 | 33rd | 0E | 0E | 10 | 01 | 93 | 00 | PrepL |  |
| 0 | 34th | 0F | 0F | 10 | 01 | 10 | 00 | FetchL |  |
| 0 | 35th | 0F | 10 | 00 | 01 | 10 | 00 | ReadMem |  |
| 0 | 36th | 10 | 0F | 00 | 01 | 10 | E7 | FetchL |  |

Create a simulation force file to test your processor using this machine language program. Make your clock period 100 ns. This simulation should produce the same result as shown in the table above. Start the simulation by asserting RESET for one clock cycle, then release it well before a CLK rising edge. The simulation should show the clock-by-clock operation of the full processor.

Your waveforms should include **CLK, RESET, address, MDR, AC, opcode, addr/value, PC, ZFLG, and NFLG**. Also show control lines: **LOAD\_AC, LOAD\_IRU, LOAD\_IRL, LOAD\_PC, INCR\_PC, STORE\_MEM, and FETCH**. This simulation should fully show the fetch and execution of each instruction given in the memory.

# Lab report (including this page): on BBLearn this time, must submit in pdf format

**Activity #3 Board Demonstration of Processor in Full Operation (50% of the lab grade)**

Create a testbench named up3\_tb using the hardware configuration shown in the tutorial ppt file. Download your design to the FPGA board, test the functions and then ask the instructor to check.

When the instructor come to check your work, you will be first asked to demonstrate the result using the default instruction set saved in the memory. Then you will be asked to modify the memory data to run a random picked instruction.

Instructor Verification: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Date:\_\_\_\_\_\_\_\_\_\_