Lab 6 Tutorial

The Complete CPU (Overall Project)

OVERVIEW

In this lab we will implement the complete CPU design.

PREREQUISITES

Before creating a new project in Quartus II, create a new folder “Lab6” in your working directory and copy the following files into your new “Lab6” folder.

From Lab 4b:

1. UZE.vhd
2. Register32.vhd
3. RED.vhd
4. Mux4to1.vhd
5. Mux2to1.vhd
6. LZE.vhd
7. Fulladd.vhd
8. Data_mem.vhd
9. Alu.vhd
10. Adder32.vhd
11. Adder16.vhd
12. Adder4.vhd
13. Add.vhd
14. Pc.vhd
15. Data_Path.vhd

From Lab5:

16. Control.vhd

From the Lab6 folder on D2L:

17. Cpu1.vhd
18. Cpu_test_sim.vhd
19. System_memory.mif
20. System_memory.qip
21. System_memory.vhd

PROCEDURE

Part I – CPU Reset Circuitry

After successfully adding all the design files required for this lab to your "Lab6" folder, we are now ready to create a project in Quartus II.

1. Create a new project "Lab6" in your "Lab6" folder.

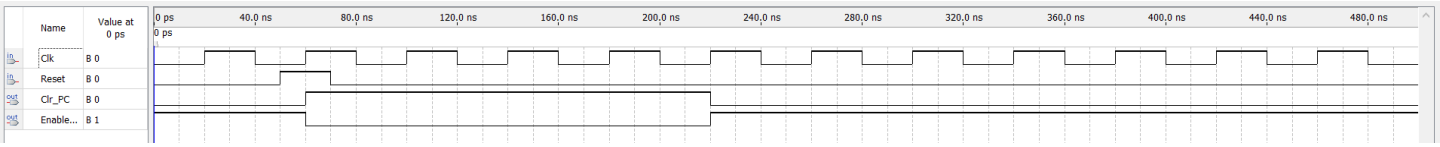
   DON'T ADD THE FILES IN THE PRE-REQUISITES SECTION TO THE PROJECT YET. WE NEED TO DESIGN AND TEST THE CPU RESET CIRCUITRY FIRST.

   MAKE SURE THAT YOU CHOOSE THE "EP4CE115F29C7" DEVICE IN THE PROJECT WIZARD.

2. Create a new VHDL file in your "Lab6" project and save it as "reset_circuit.vhd"
3. Write the following code in your "reset_circuit.vhd" file.

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3   USE ieee.std_logic_arith.ALL;
4   USE ieee.std_logic_unsigned.ALL;
5
6   ENTITY reset_circuit IS
7      PORT(
8          Reset : IN STD_LOGIC;
9          Clk : IN STD_LOGIC;
10         Enable_PD : OUT STD_LOGIC := '1';
11         Clr_PC : OUT STD_LOGIC
12     );
13  END reset_circuit;
14
15  ARCHITECTURE Behavior OF reset_circuit IS
16      TYPE clkNum IS (clk0, clk1, clk2, clk3);
17      SIGNAL present_clk: clkNum;
18  BEGIN
19     process(Clk)begin
20        if rising_edge(Clk) then
21           if Reset = '1' then
22              Clr_PC <= '1';
23              Enable_PD <='0';
24              present_clk <= clk0;
25           elsif present_clk <= clk0 then
26              present_clk <= clk1;
27           elsif present_clk <= clk1 then
28              present_clk <= clk2;
29           elsif present_clk <= clk2 then
30              present_clk <= clk3;
31           elsif present_clk <= clk3 then
32              Clr_PC <= '0';
33              Enable_PD <= '1';
34           end if;
35        end if;
36     end process;
37  END Behavior;
```

4. Set “reset_circuit.vhd” as the top-level entity, and compile. Once your design compiles without any errors, move to the next step.
5. Create a new University Program Vector Waveform File “reset_circuit.vwf” and simulate “reset_circuit.vhd.”
6. Once your simulation results match the results below, move to the next step.



Part 2 – The Complete CPU System

We are now ready to start assembling the CPU:

1. Please add the files that you copied into your “Lab6” folder to your lab6 project.

   You can do this by going to “Project” at the top of your Quartus II main window, and selecting “Add/Remove Files in Project.”
2. At this point, we need to do a few modifications to two files that are currently in the project.
3. The first file is “Control.vhd”:
   i. Open “Control.vhd” from your Project Navigator.
   ii. Change the line:

   ENTITY Control IS

   To:

   ENTITY Control_New IS

   iii. Change the line:

   END Control;

   To:

   END Control_New;

iv. Change the line:

ARCHITECTURE description OF Control IS

To:

ARCHITECTURE description OF Control_New IS

v. Save the file as “Control_New.vhd” in your “Lab6” folder.

4. The second file is “add.vhd”

i. Open “add.vhd” from your Project Navigator.

ii. Change the line:

B <= A + 4;

To:

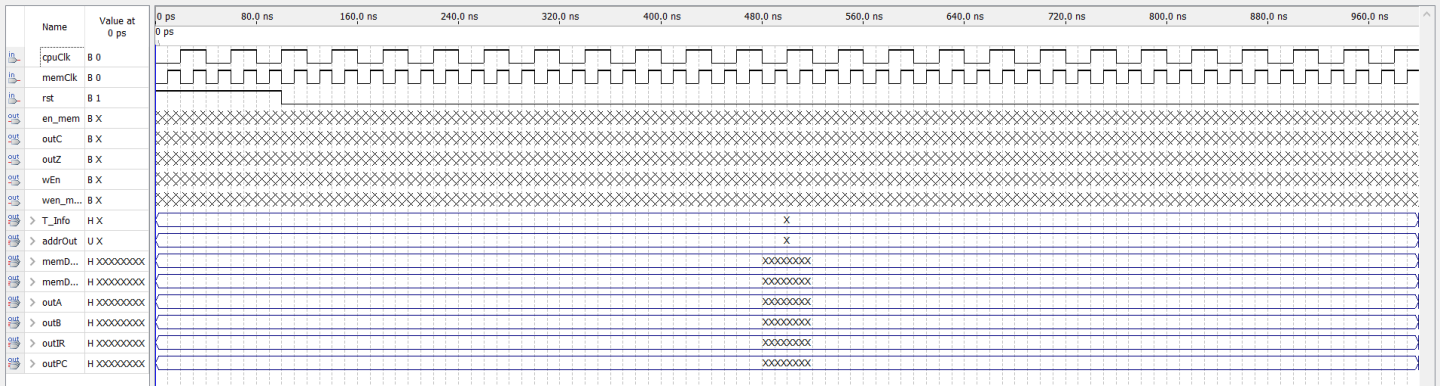
B <= A + 1;

iii. Save the file as “add.vhd” in your “Lab6” folder.

We are now ready to compile and test our design:

1. Set “cpu_test_sim.vhd” as the top-level entity.
2. Compile your design, once your design compiles without any issues, move to the next step.
3. Create a new University Program Vector Waveform File “cpu_test_sim.vwf” as shown below.

   Note that cpuClk and memClk time periods used in the following functional simulations are 40 ns and 20 ns respectively.



IMPORTANT

For the each of the tests shown after this point, you are a required to open “system_memory.mif” and set it as shown in the corresponding test and save. Then for each for the tests, you must recompile “cpu_test_sim.vhd” before running the simulator. Thereafter, run the simulator on “cpu_test_sim.vwf”, verify your results, and take screenshots of your simulations.

LDAI, STA, CLRA, LDA

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	20000001	75000000	90000001	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....

Figure 1: system_memory.mif

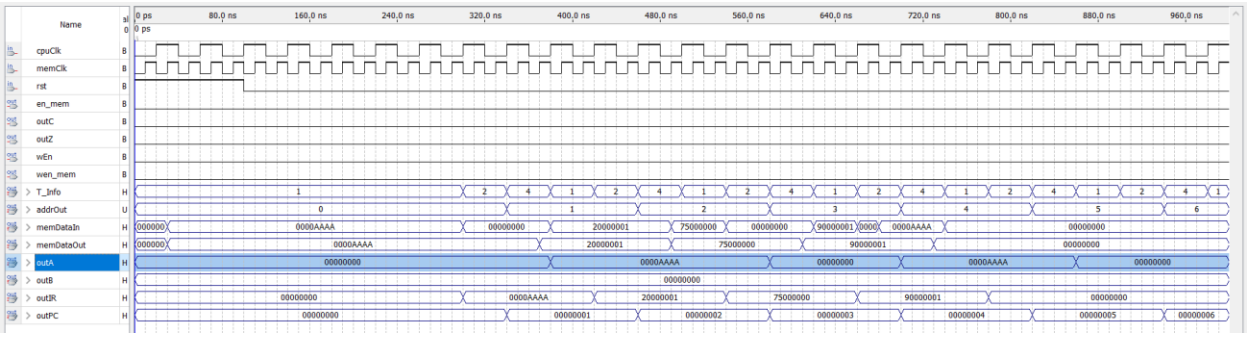
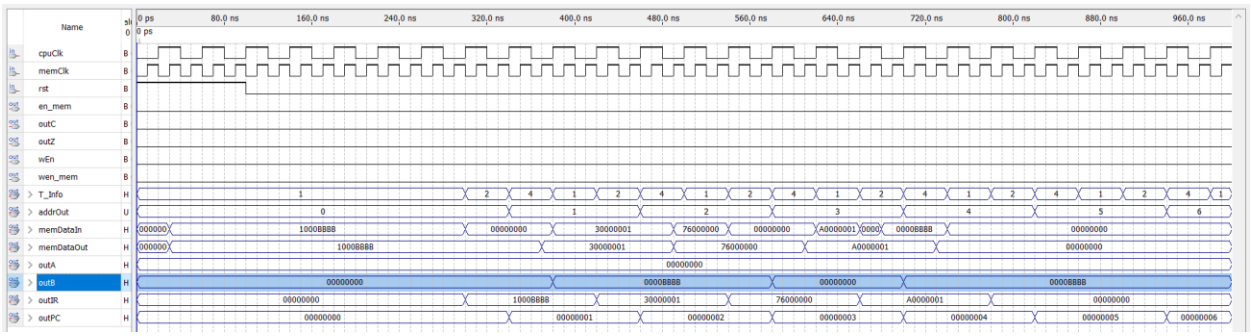


Figure 2: Functional Simulation

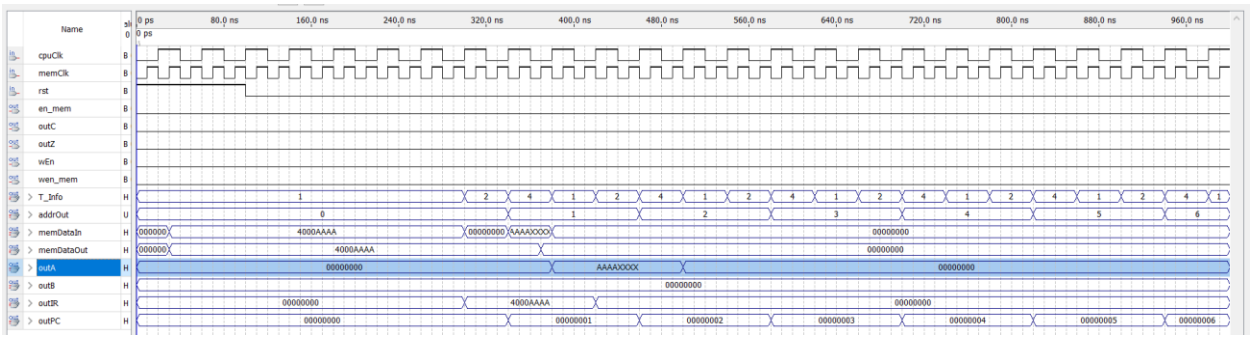
LDBI, STB, CLRB, LDB

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	1000BBBB	30000001	76000000	A0000001	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



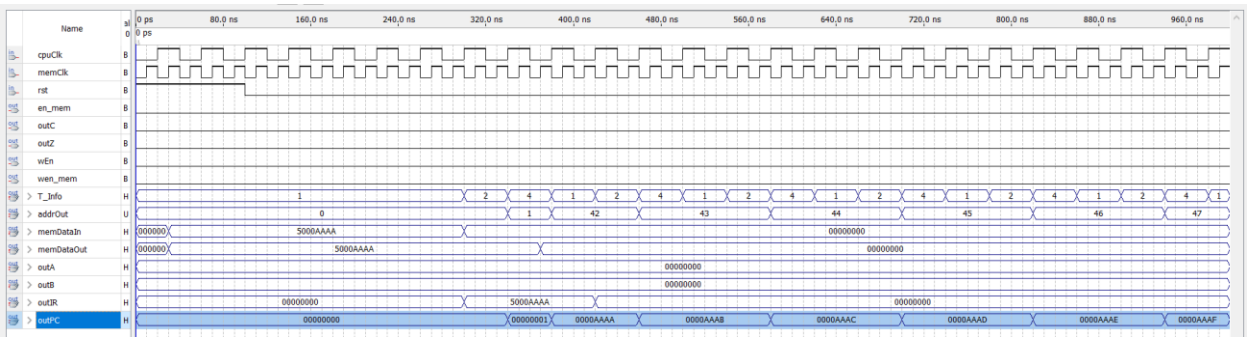
LUI

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	4000AAAA	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



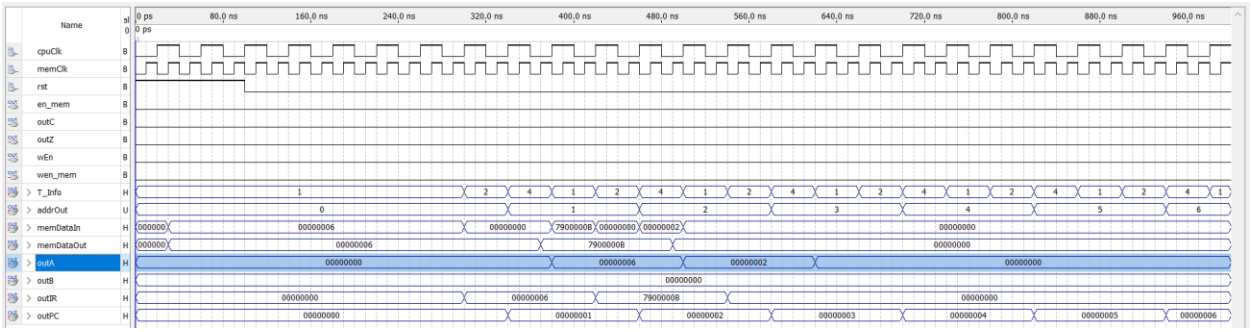
JMP

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	5000AAAA	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



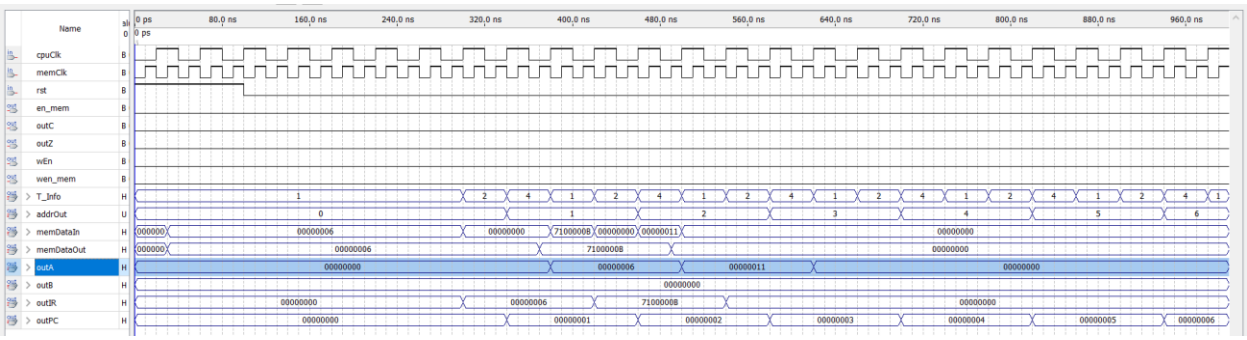
ANDI

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000006	7900000B	00000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....



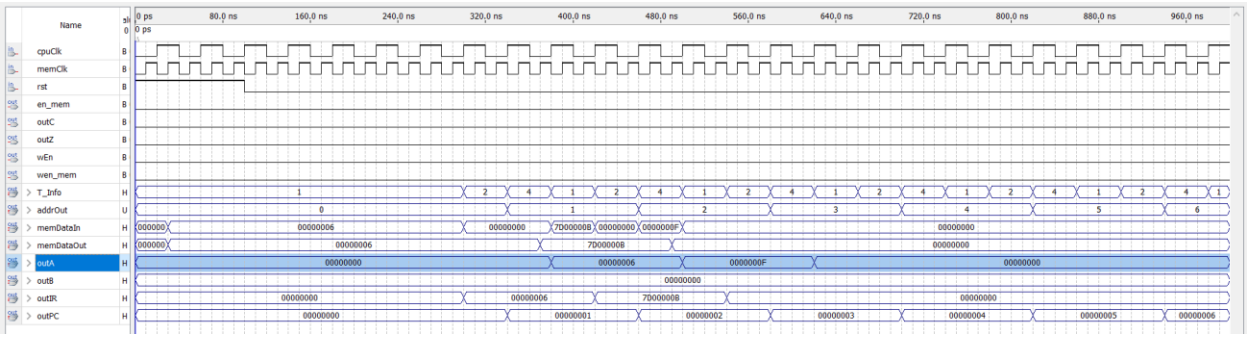
ADDI

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000006	7100000B	00000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



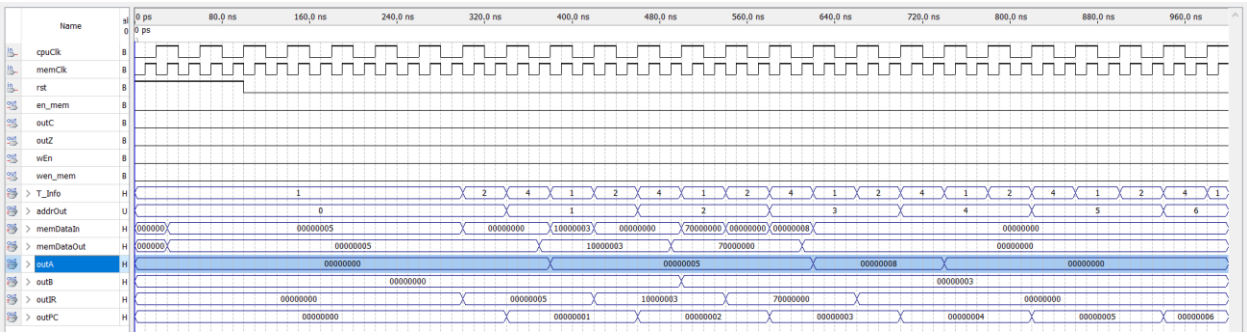
ORI

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000006	7D00000B	00000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



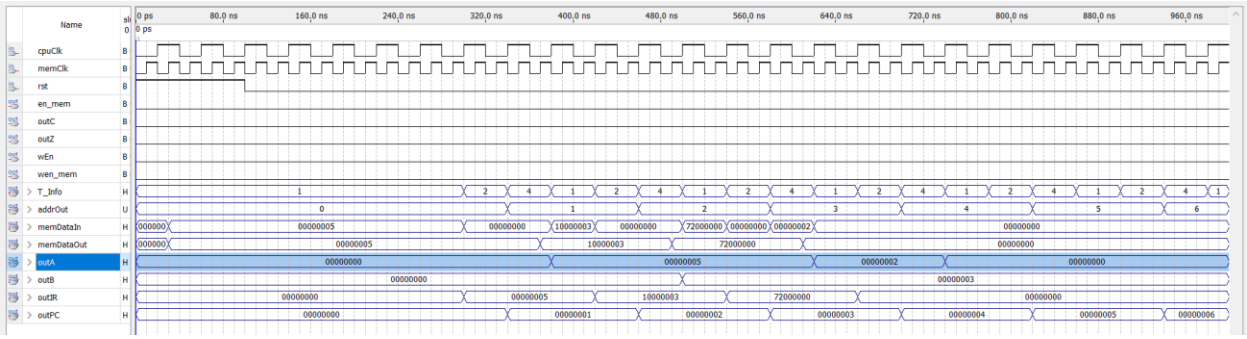
ADD

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000005	10000003	70000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



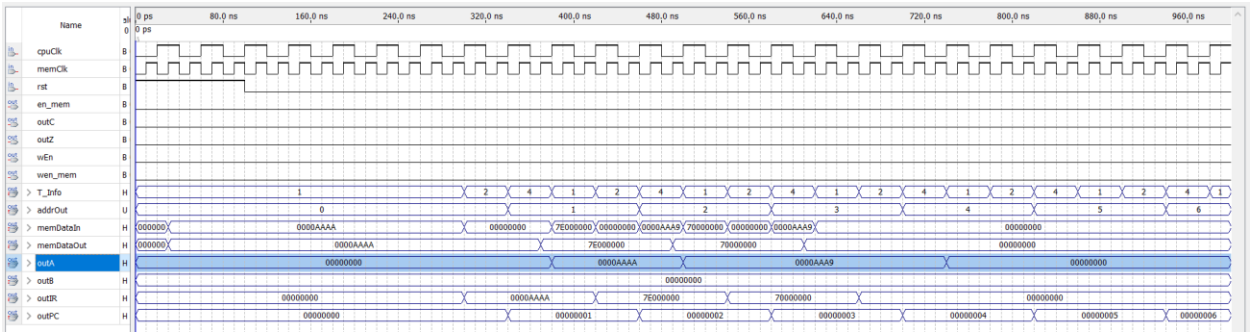
SUB

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000005	10000003	72000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



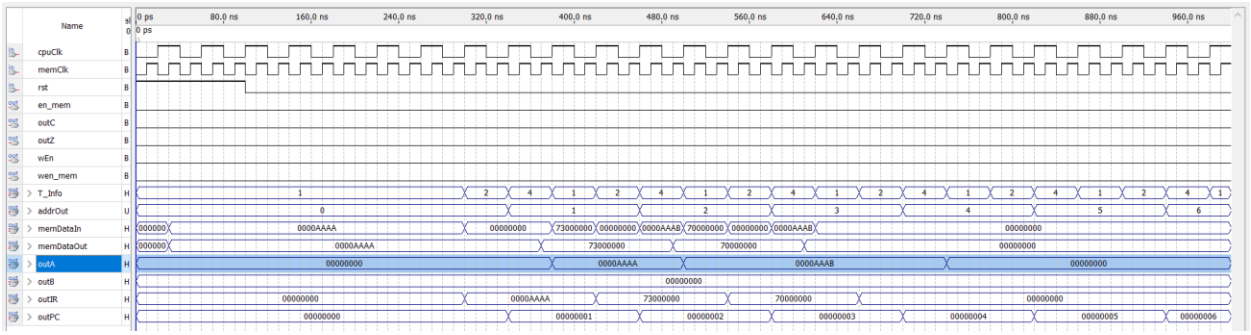
DECA

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	7E000000	70000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



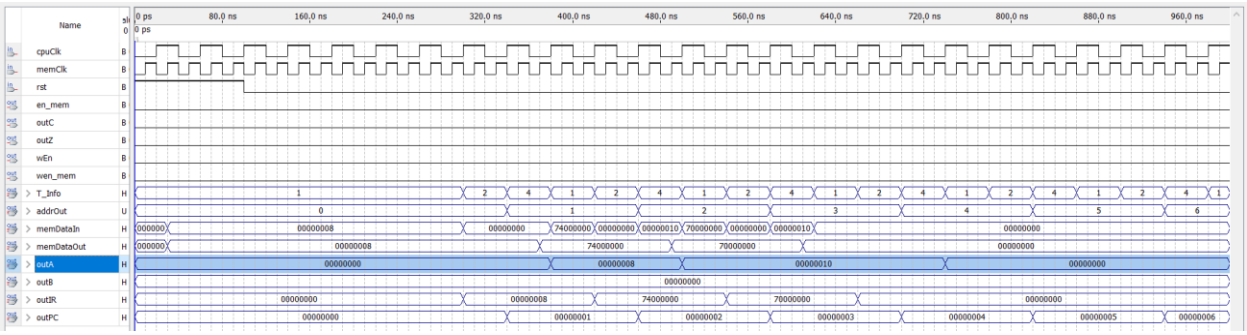
INCA

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	73000000	70000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



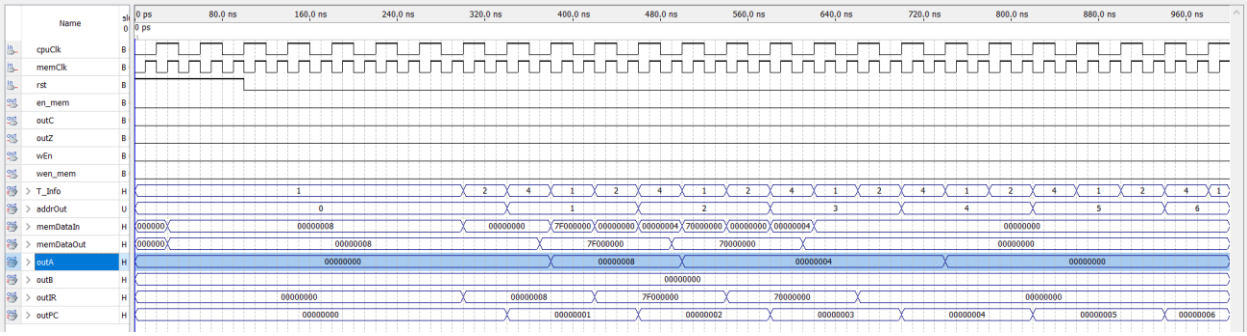
ROL

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000008	74000000	70000000	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



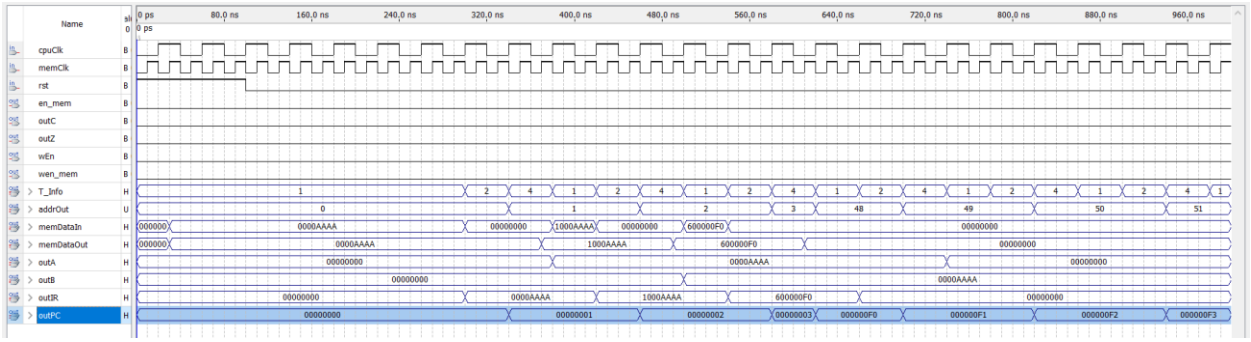
ROR

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	00000008	7F000000	70000000	00000000	00000000	00000000	00000000	00000000	....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	....



BEQ

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	1000AAAA	600000F0	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....



BNE

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	0000AAAA	1000BBBB	800000F0	00000000	00000000	00000000	00000000	00000000	.....
8	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
16	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
24	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
32	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
40	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
48	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
56	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....