COL216 Assignment 2 Stage 3
Aryan Dua – 2020CS50475

The epwaves shown are the result of the testbench that I have submitted along with my code in this zip file. I have tested my design on the two test examples given to us.

Design Description:

In my main multi_processor.vhd file, I have 6 components, namely – ALU, decoder, Register File, Flag updater, Memory, FSM, Mux.

I have attached their synthesis reports below. All these components have been instantiated under the processor architecture. The data path signals flow through them. The control signals are given to the processor by the FSM.

I have simulated and synthesized on edaplayground.com. There are some components whose synthesis reports I have already submitted in the last stage submission, therefore I have only attached the reports of the new components. The 2 new components are mux, fsm and memory. FSM was not synthesizable on edaplayground.

I have used 10 states in the finite state machine. All control signals needed are returned by this component. The decoder decodes the current instruction into the small parts required. Then it is a similar gluecode like that of last time which performs the task.

The 5 intermediate registers used to store values are: IR, DR, A, B, RES, just like that in the lecture slides.

The design works up to the expectation and nothing has been hardcoded. I have attached the epwave outputs of as many signals as I could/are relevant.

Synthesis Reports:

1. MUX

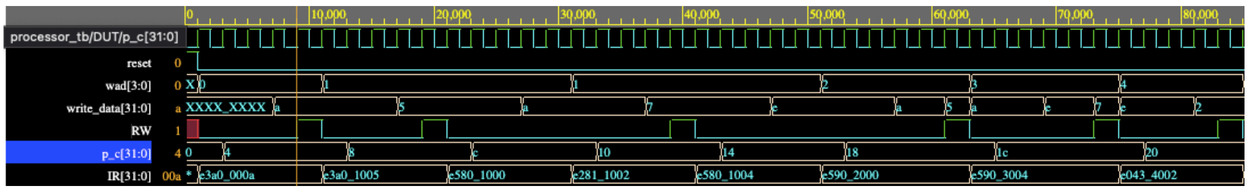
```
Device Utilization for 7A100TCSG324
# Info: ************************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: ------------------------------------------------------------
# Info: IOs                               162     210      77.14%
# Info: Global Buffers                    0       32        0.00%
# Info: LUTs                              32      63400     0.05%
# Info: CLB Slices                        8       15850     0.05%
# Info: Dffs or Latches                   0       126800    0.00%
# Info: Block RAMs                        0       135       0.00%
# Info: DSP48E1s                          0       240       0.00%
# Info: ------------------------------------------------------------
# Info: ****************************************************
# Info: Library: work    Cell: mux    View: behavior
# Info: ****************************************************
# Info:  Number of ports :                          162
# Info:  Number of nets :                           324
# Info:  Number of instances :                      194
# Info:  Number of references to this view :          0
# Info: Total accumulated area :
# Info:  Number of LUTs :                            32
# Info:  Number of Primitive LUTs :                  32
# Info:  Number of accumulated instances :          194
```

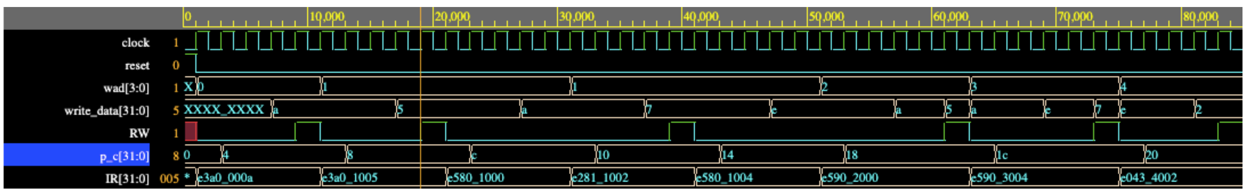
2. MUX

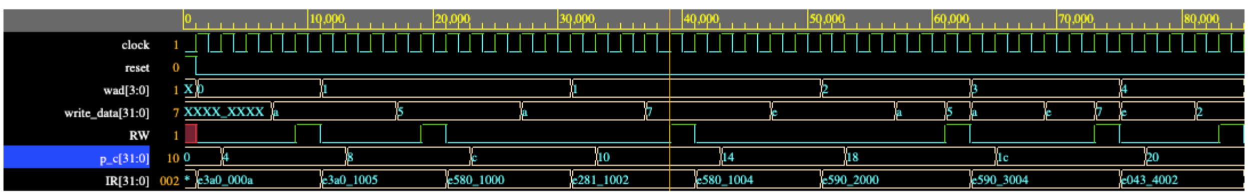
```
Device Utilization for 7A100TCSG324
# Info: ***************************************************************
# Info: Resource                          Used    Avail   Utilization
# Info: ---------------------------------------------------------------
# Info: IOs                               82      210      39.05%
# Info: Global Buffers                    1       32        3.12%
# Info: LUTs                              144     63400     0.23%
# Info: CLB Slices                        24      15850     0.15%
# Info: Dffs or Latches                   0       126800    0.00%
# Info: Block RAMs                        0       135       0.00%
# Info: Distributed RAMs
# Info:    RAM128X1D                      32
# Info: DSP48E1s                          0       240       0.00%
# Info: ---------------------------------------------------------------
# Info: ************************************************
# Info: Library: work    Cell: Mem    View: BEV
# Info: ************************************************
# Info:  Number of ports :                              82
# Info:  Number of nets :                               228
# Info:  Number of instances :                          115
# Info:  Number of references to this view :            0
# Info: Total accumulated area :
# Info:  Number of LUTs :                               144
# Info:  Number of Primitive LUTs :                     160
# Info:    Number of LUTs as Distributed RAM :          128
# Info:  Number of LUTs with LUTNM/HLUTNM :             32
# Info:  Number of accumulated instances :              148
```

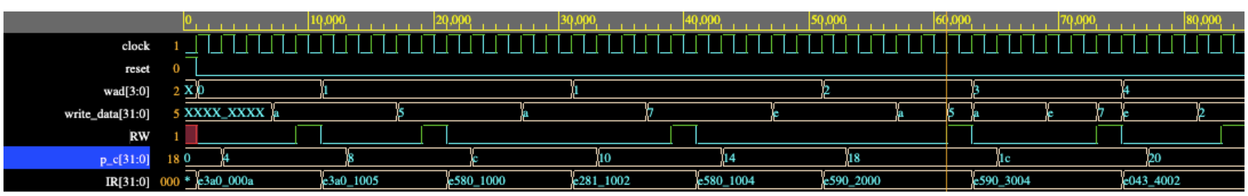
Test Cases and Output:

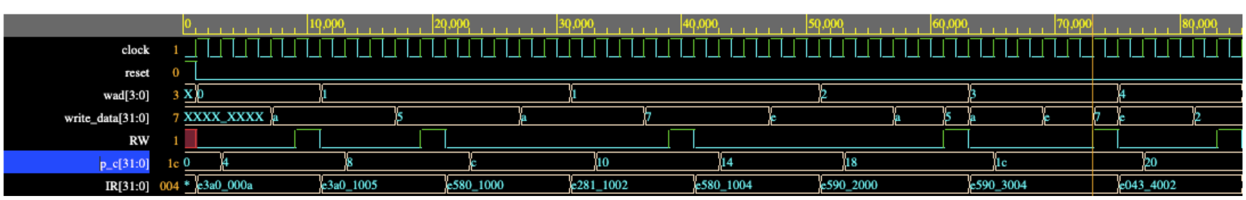
Here is the epwave output of the first example program. wad = write address and write_data is the data to be entered at that address.:

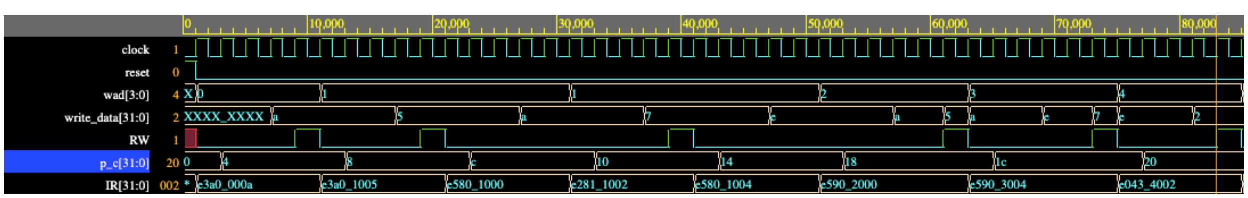










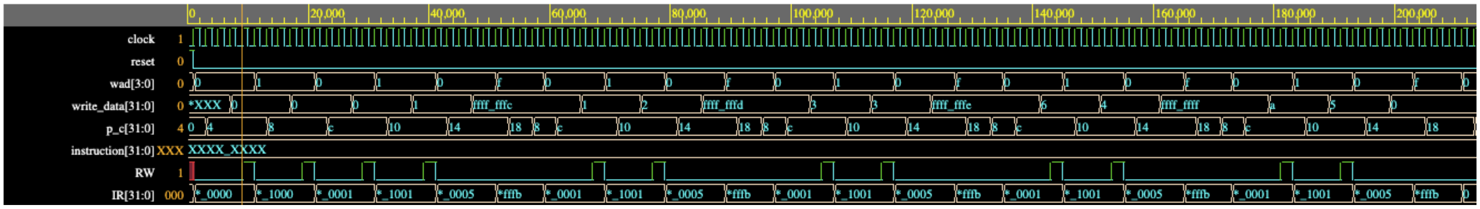


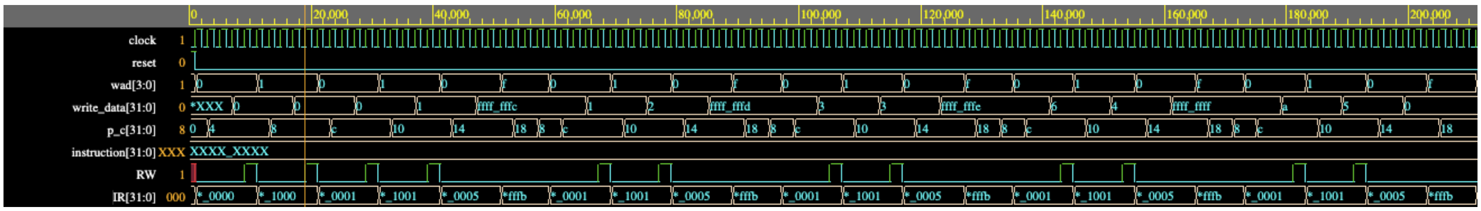
Step by step outputs of the parts of the program where the register is being written:

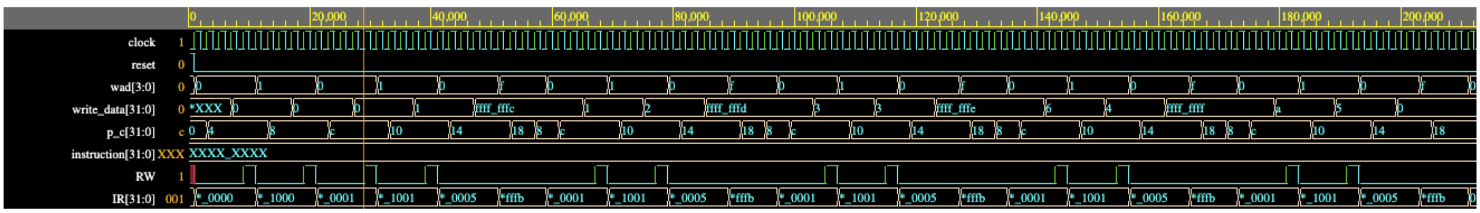
R0 = 10
R1 = 5
R1 = 7
R2 = 5
R3 = 7
R4 = 2
(The program should have r0 = 8 initially instead of 10.)

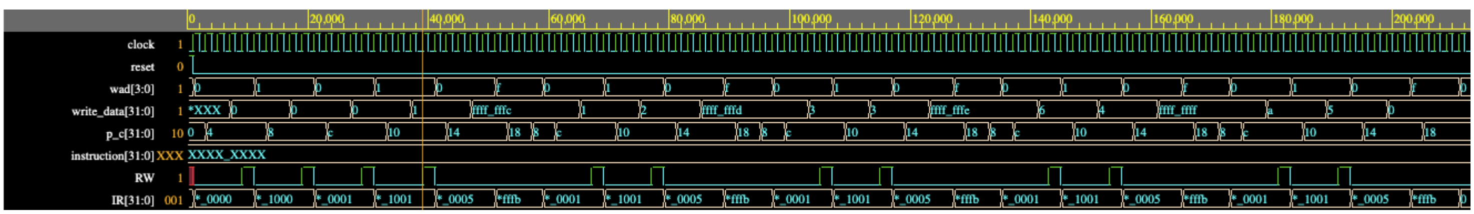
The epwave output of the second test program:

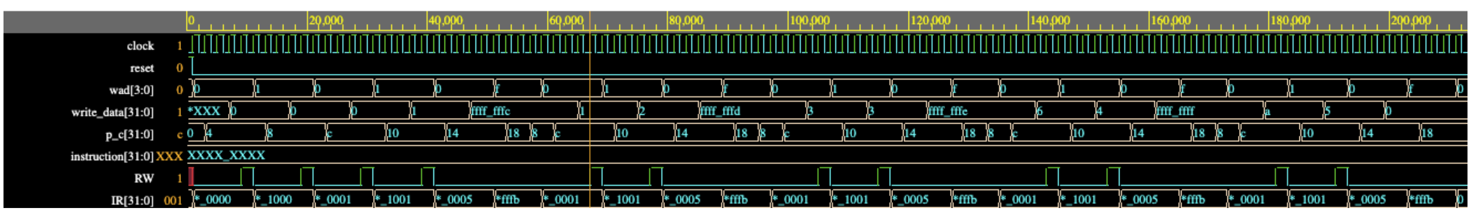
In this, the program will run 5 times until r1 reaches 0. (r1 = -4,-3,-2,-1,0). I have attached screenshots of each time the register file is modified(A register value changes). wad = write address and write_data is the data to be entered at that address.

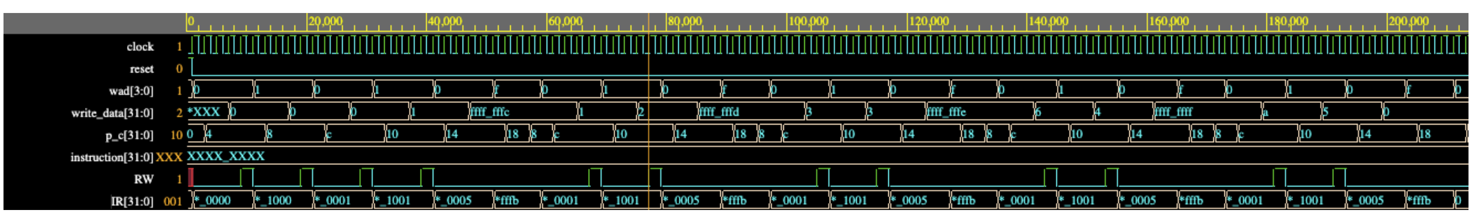

0
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
180,000
200,000
clock
reset
wad[3:0]
write_data[31:0]
p_c[31:0]
instruction[31:0]
RW
IR[31:0]

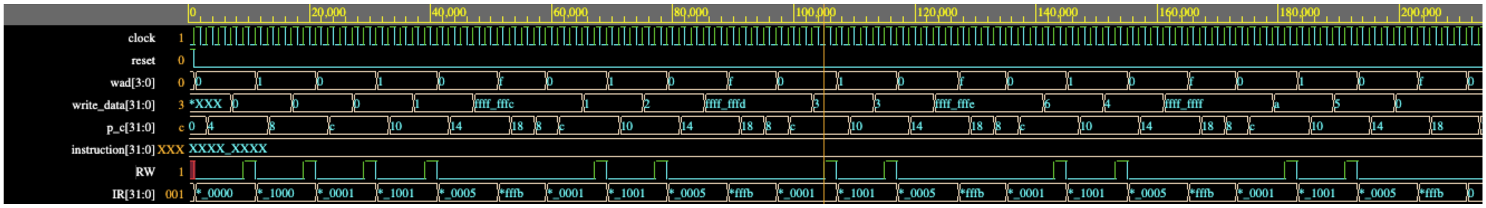

0
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
180,000
200,000
clock
reset
wad[3:0]
write_data[31:0]
p_c[31:0]
instruction[31:0]
RW
IR[31:0]

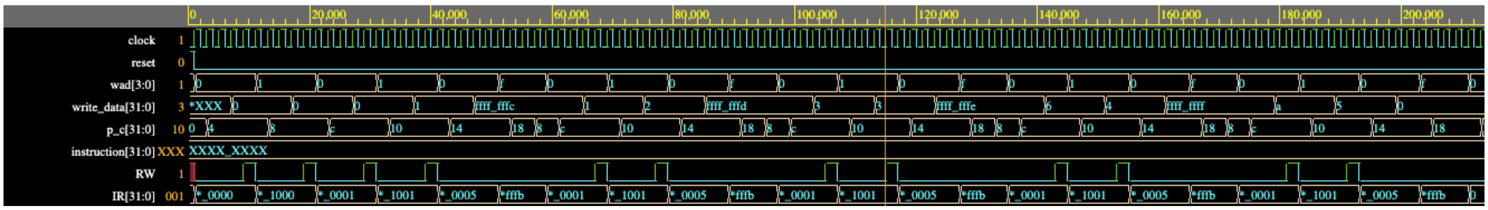

0
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
180,000
200,000
clock
reset
wad[3:0]
write_data[31:0]
p_c[31:0]
instruction[31:0]
RW
IR[31:0]

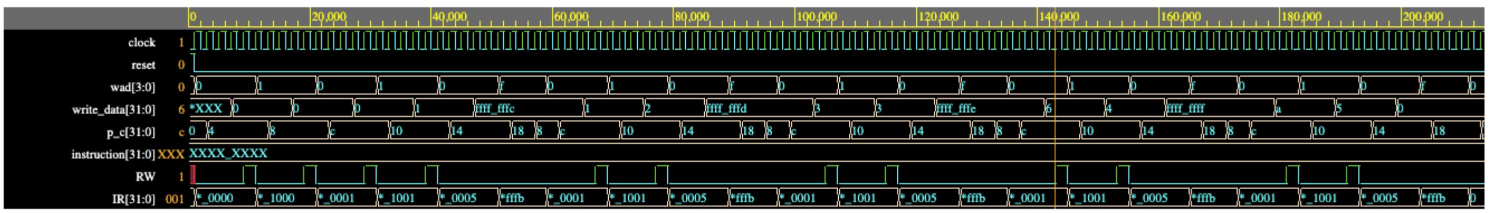

0
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
180,000
200,000
clock
reset
wad[3:0]
write_data[31:0]
p_c[31:0]
instruction[31:0]
RW
IR[31:0]

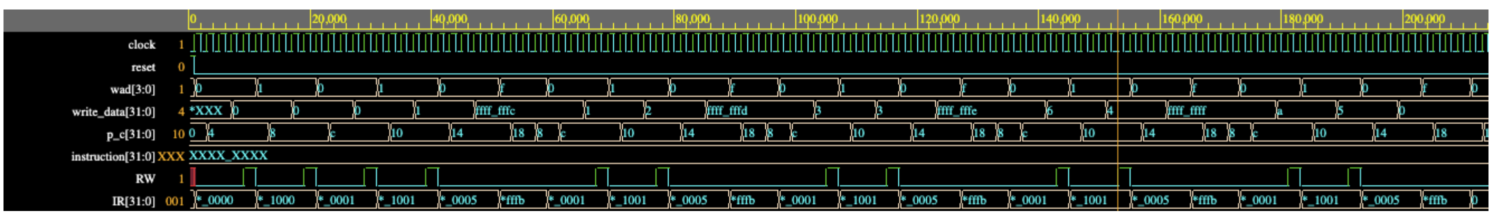

0
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
180,000
200,000
clock
reset
wad[3:0]
write_data[31:0]
p_c[31:0]
instruction[31:0]
RW
IR[31:0]

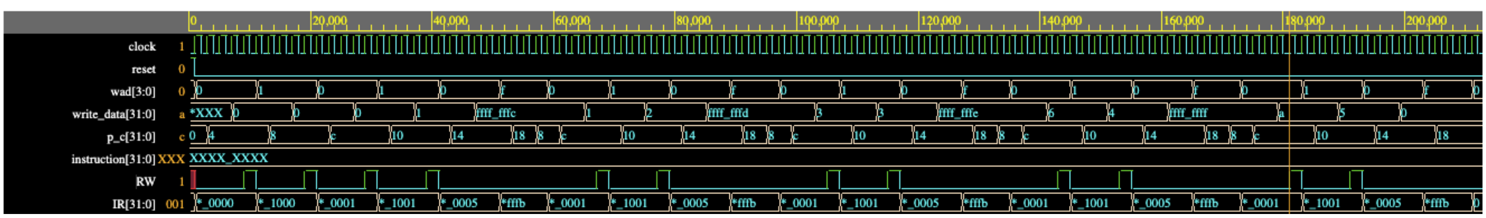

0
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
180,000
200,000
clock
reset
wad[3:0]
write_data[31:0]
p_c[31:0]
instruction[31:0]
RW
IR[31:0]

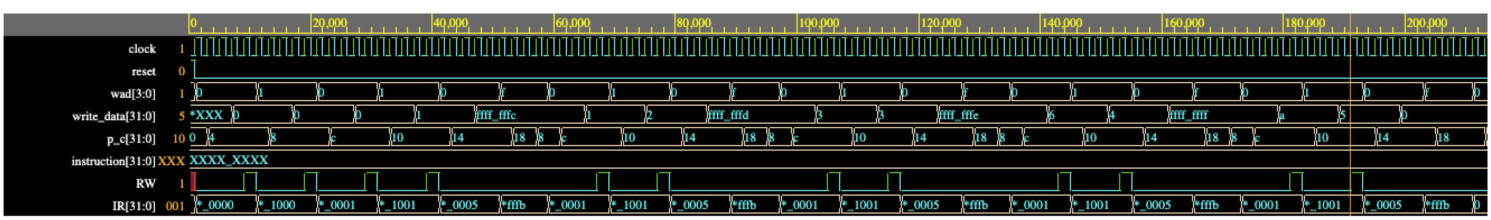






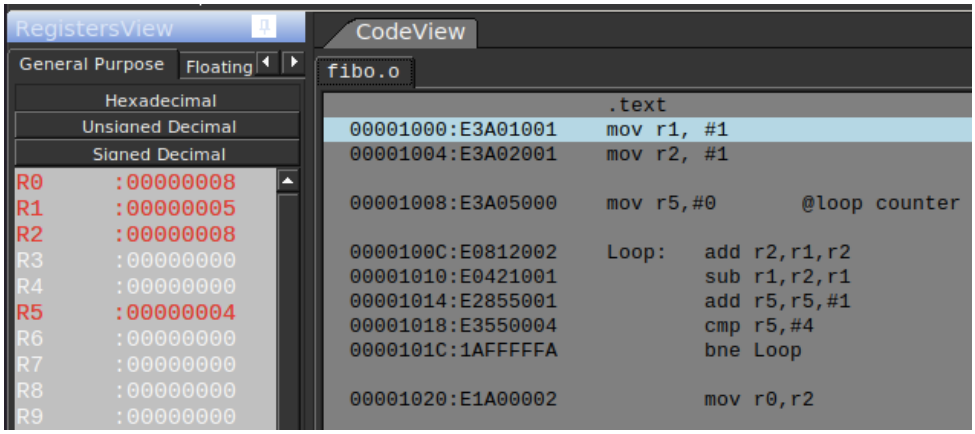




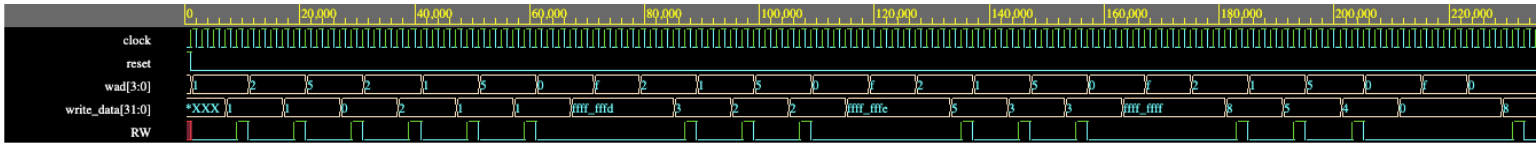


3. My own testcase:

Screenshot of my ARMSim program output:



Screenshot of my processor epwave output:



Here are the sequence of values being written to the register file:

R1 = 1
R2 = 1
R5 = 0

R2 = 2
R1 = 1
R5 = 1

R2 = 3
R1 = 2
R5 = 2

R2 = 5
R1 = 3
R5 = 3

R2 = 8
R1 = 5
R5 = 4
R0 = 8