ASSIGNMENT 2 STAGE 6 REPORT

Sreemanti Dey

January 2022

1 Objective

The features that have been added in this stage include byte and half word transfers (signed and unsigned), auto increment/decrement with option of pre/post indexing

2 Assumptions

VHDL
GHDL
GTKWave for the waveforms

3 Implementation details

I have added support for byte and half word transfers by making the following changes:

1. I made a new entity-architecture pair called PMconnect.vhd that modifies the data to be transferred between processor and memory.
2. I have modified the signals to include the provision of pre/post indexing.

I have written the following assembly program files in my program to test if pre and post indexing and load-store works correctly.

1. ldrh and strh - p1.s
2. ldrsh - p2.s
3. ldrsb - p3.s
4. ldr and str - p4.s
5. ldrb and strb - p5.s
6. post-indexing - p6.s
7. pre-indexing with write-back - p7.s
8. pre-indexing with no write-back, also register offset - p8.s

4 p1.s - ldrh and strh

4.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

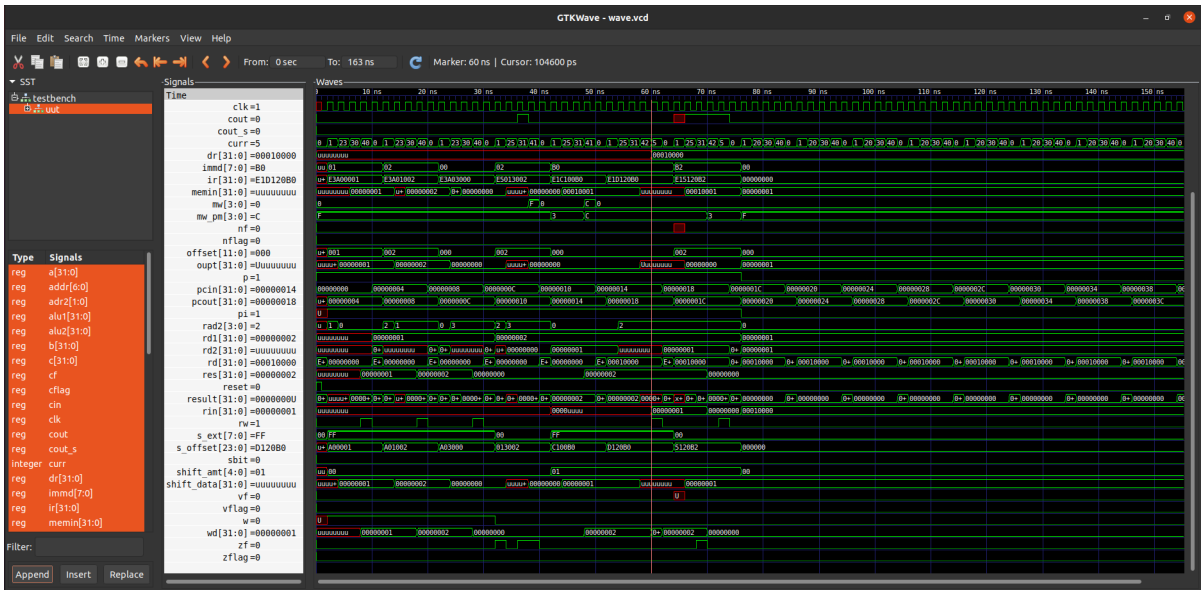

Figure 1: Program 1

Here, I have written half word 1 into the leftmost 16 bits, thus when we have rw =1 for second last time, we see wd is 1 (which is correct) and when rw = 1 for the last time, we see wd = 0 which is correct, thus verifying my code.

5 p2.s - ldrsh

5.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

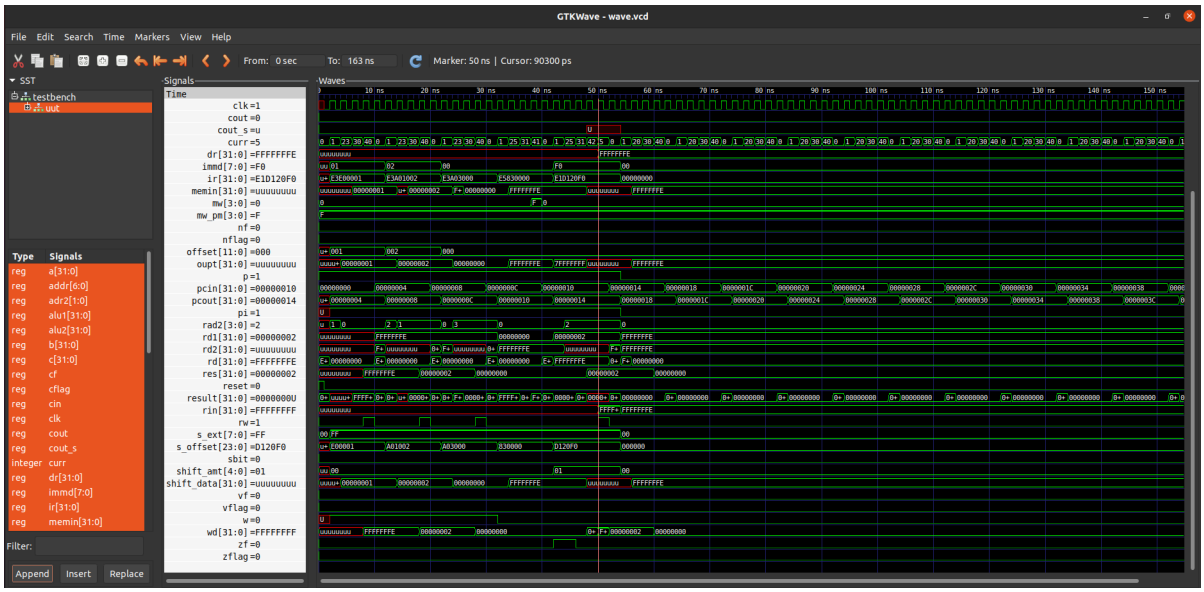

Figure 2: Program 2

Here, I have stored -2, then I have loaded the leftmost half word, which when extended by sign, becomes -1, and this can be seen in the image, hence verified.

6 p3.s - ldrsb

6.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

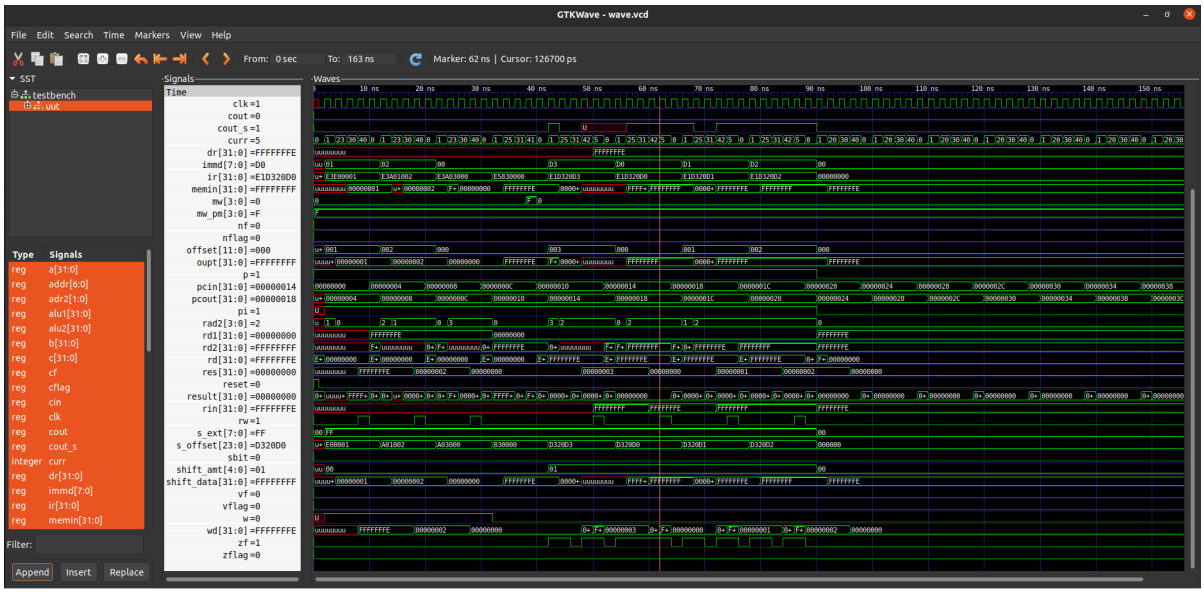

Figure 3: Program 3

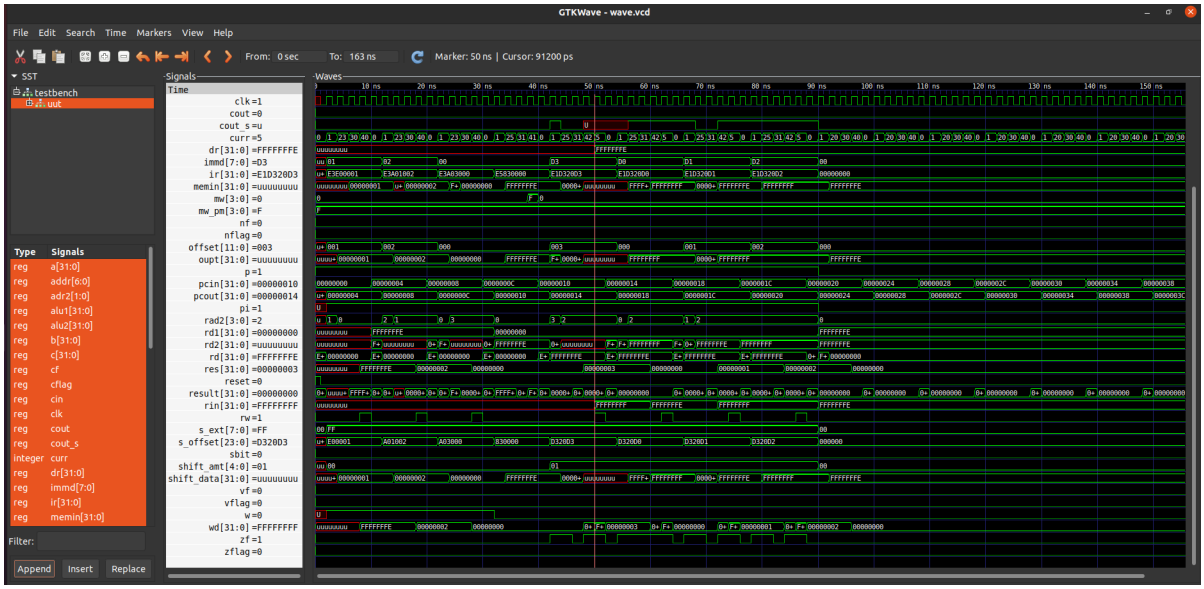

Figure 4: Program 3

Here, I have loaded -2 byte by byte using signed byte instruction, so the second instruction that loads the rightmost 8 bits, is -2, rest all are -1, hence

verifying my code is correct.

7 p4.s - ldr and str

7.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

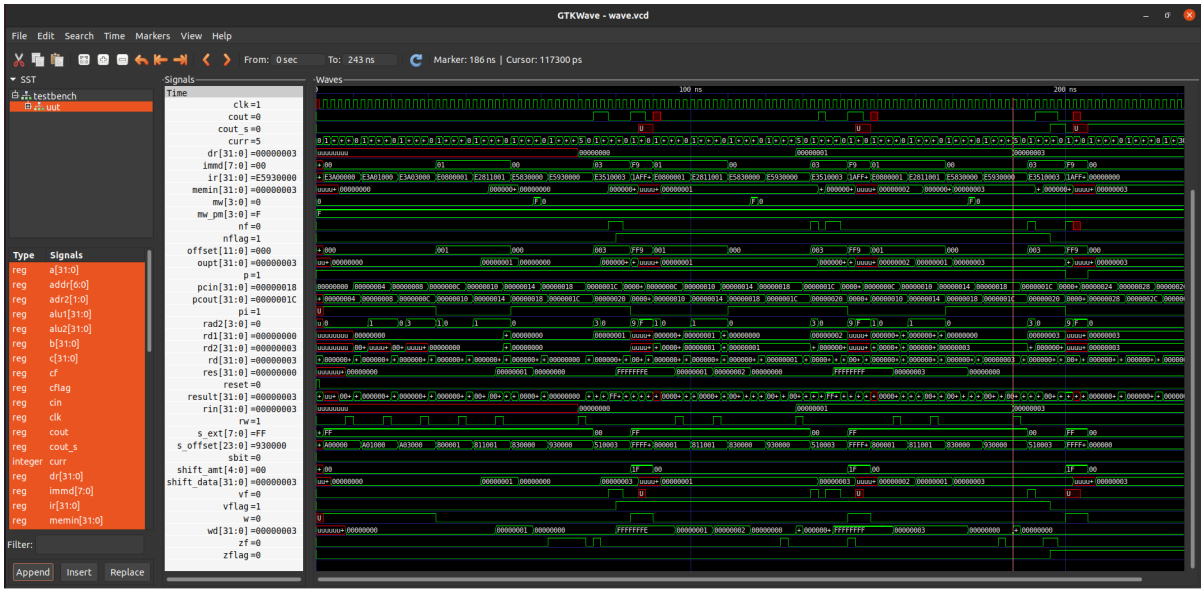

Figure 5: Program 4

Here, I have added branch instruction with ldr and str to check the overall construction. We can see that the times when we loaded r0 into a register, we have the correct value, and final value is 3, which is the correct result(1+2=3), hence verifying my code.

8 p5.s - ldrb and strb

8.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave

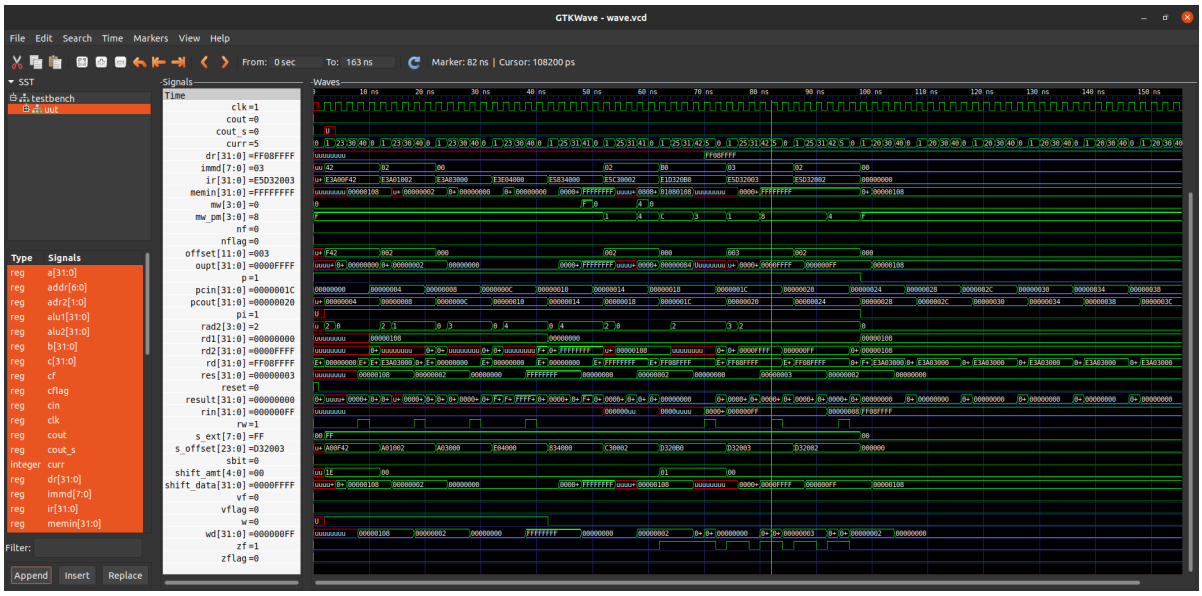

Figure 6: Program 5a

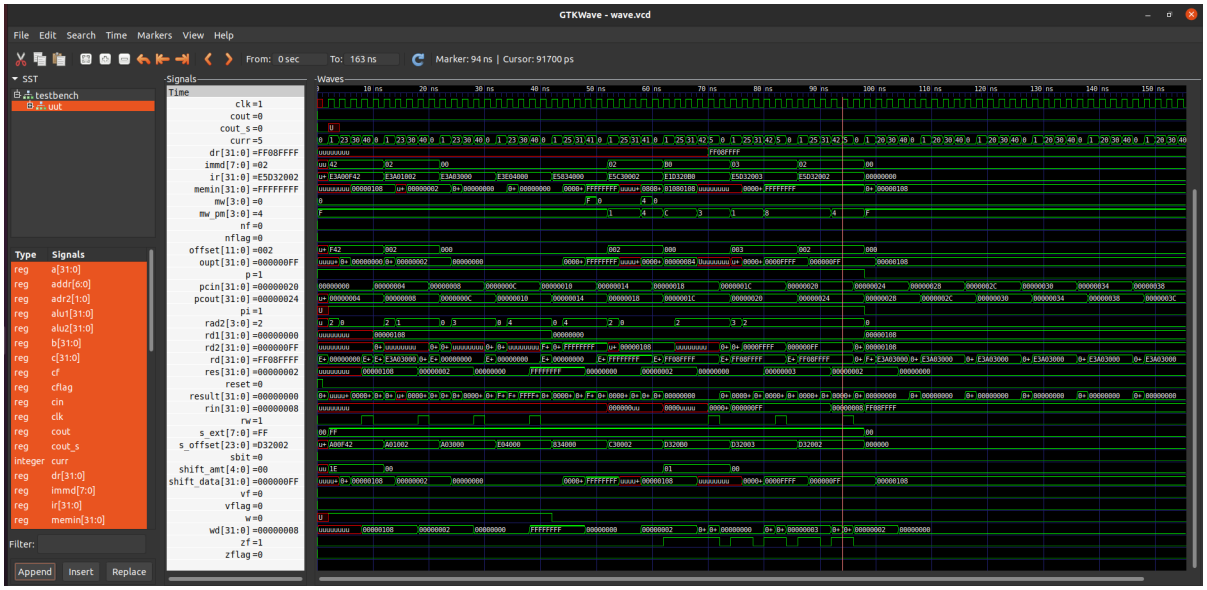

Figure 7: Program 5b

Here, we can see that the last time rw is 1, we have wd as 8, which indicates it is correct since I had specifically stored bits 15 downto 8 in the 2nd position

from the left, using strb, and we check the value using ldrb, which verifies the correctness of my code. also, second last time, rw is 1, we have wd as FF which is also correct as per my program.

9 p6.s - post-indexing

9.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave.

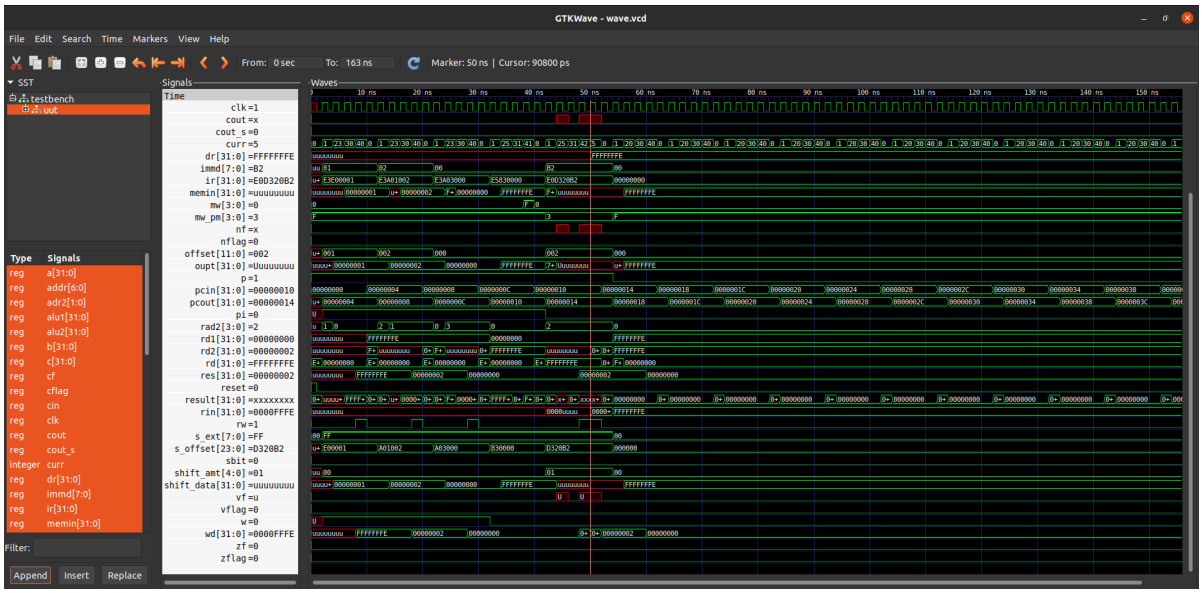

Figure 8: Program 6

Here, we can see that the half word loaded is the last 4 bits(hence we see wd as 0000FFFE) and we also have the register value updated(to 2), as we see rw = 1 for 2 cycles,thus verifying my code is correct.

10 p7.s - pre-indexing with write-back

10.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave.

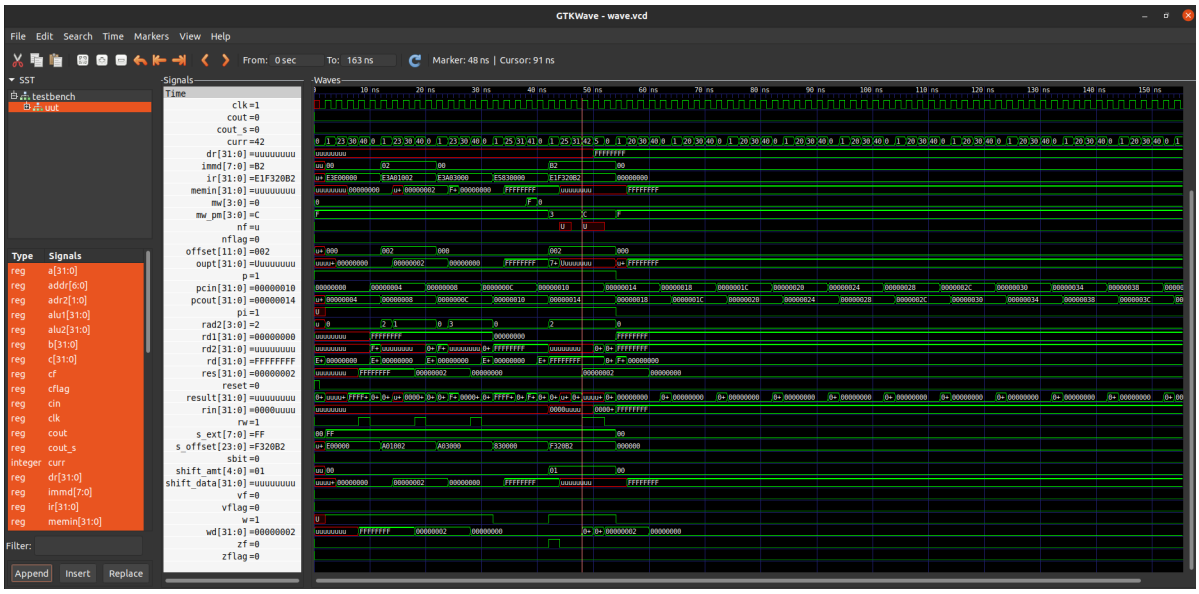

Figure 9: Program 7

We can see that wd is 2 and rw is 1, thus verifying that write back does occur correctly, and finally we see that wd is 0000FFFF which is the correct value to be written in register when rw is 1 for the last time, thus verifying my code.

11 p8.s - pre-indexing with no write-back and also register offset

11.1 Simulation results

Here is a picture of the simulation results I have achieved by EPWave.

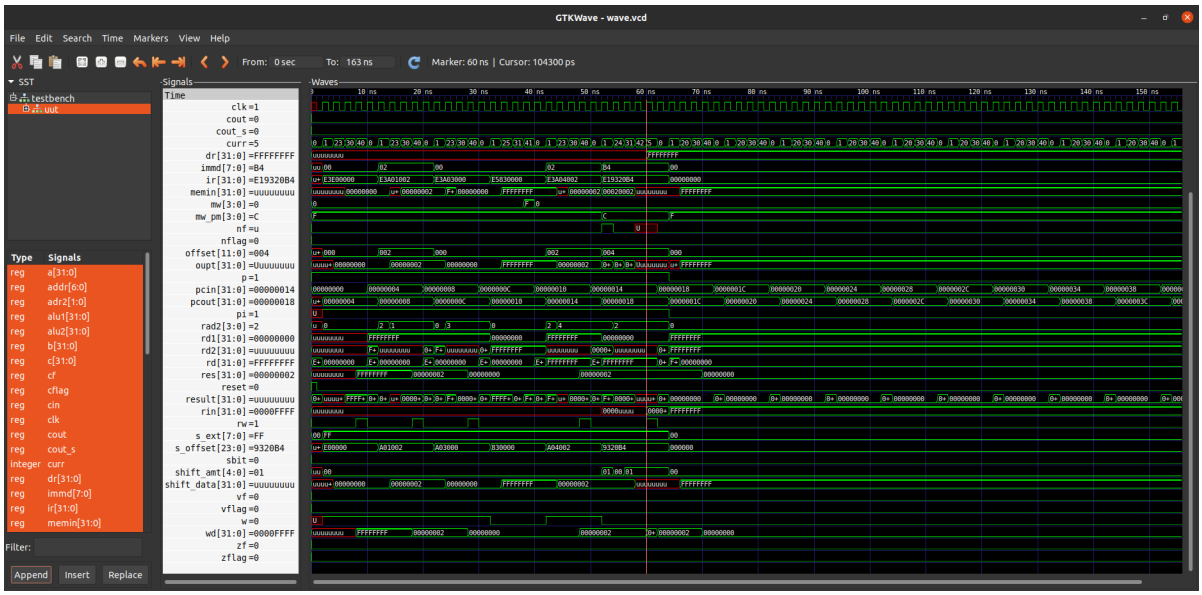

Figure 10: Program 8

This program checks register offset and pre-indexing without write-back, we see that the half word with appropriate zero extension has been written into the register, as we can see in wd value the last time rw becomes 1. hence verified.