CSE331 – COMPUTER ORGANIZATION

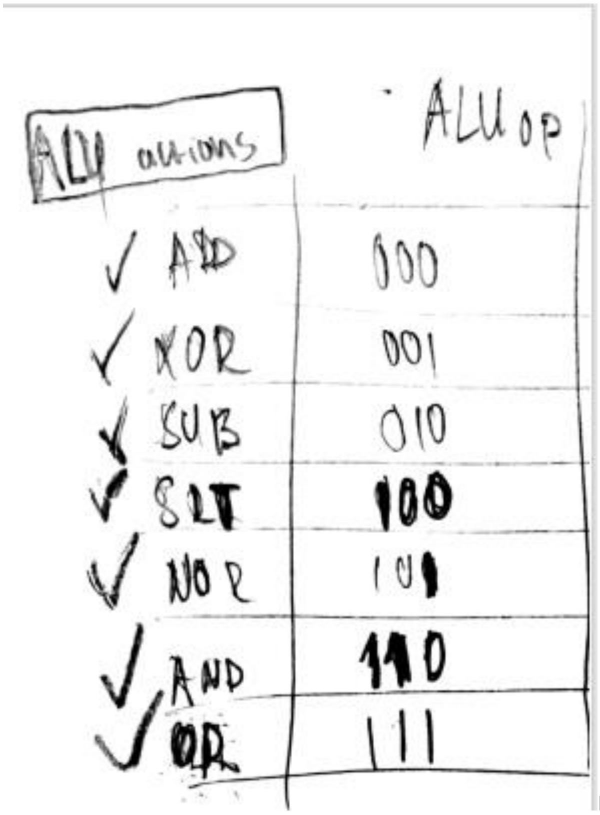
Homework 4

MIPS processor in structural Verilog

171044095 – Djuro Radusinovic

Homework was to implement Mini MIPS processor that will be able to execute some of the functionality normal MIPS is able to.

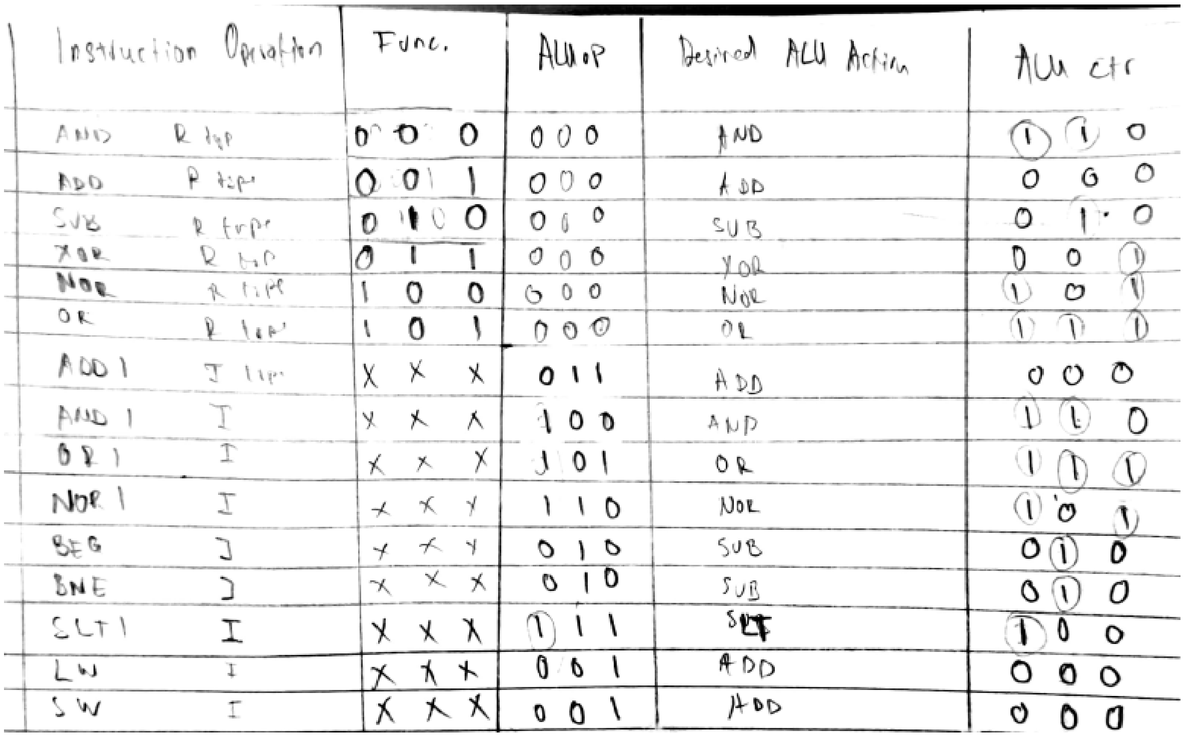
In order to make the OUR ALU-32bit, I just used the 32 bit ALU that I wrote in the last homework. Here are the Inputs and actions of this ALU



ALU actions	ALU op
✓ ADD	000
✓ XOR	001
✓ SUB	010
✓ SLT	100
✓ NOR	101
✓ AND	110
✓ OR	111

Note that this ALU-OP is actually ALU_CTR as I will later explain

For the ALU's Control Unit it is supposed to choose which desired action is to be performed according to FUNCT and ALU_OP input. The table I have writte for it is the following:



Instruction Operation		Func.	ALUoP	Desired ALU Action	ALU ctr
AND	R type	0 0 0	0 0 0	AND	1 1 0
ADD	R type	0 0 1	0 0 0	ADD	0 0 0
SUB	R type	0 1 0	0 0 0	SUB	0 1 0
XOR	R type	0 1 1	0 0 0	XOR	0 0 1
NOR	R type	1 0 0	0 0 0	NOR	1 0 1
OR	R type	1 0 1	0 0 0	OR	1 1 1
ADD I	I type	X X X	0 1 1	ADD	0 0 0
AND I	I	X X X	1 0 0	AND	1 1 0
OR I	I	X X X	1 0 1	OR	1 1 1
NOR I	I	X X X	1 1 0	NOR	1 0 1
BEG	J	X X X	0 1 0	SUB	0 1 0
BNE	J	X X X	0 1 0	SUB	0 1 0
SLT I	I	X X X	1 1 1	SLT	1 0 0
LW	I	X X X	0 0 1	ADD	0 0 0
SW	I	X X X	0 0 1	ADD	0 0 0

Here the instructions are divided into 3 different categories: I-type, J-type and R-type instructions.

In the table we can see desired actions depending on the instruction that is to be executed.

R-type instructions depend on the func 3-bit input and other will depend on the ALU-OP passed from the Main Control unit.

Main point of the ALU control unit is to produce ALU_CTR output which will control the instruction that is to be executed in the ALU. For example for SW and LW perform addition and BEQ and BNE will perform subtraction. Other instruction are more direct and for example SLTI executes subtraction and ADDI executes addition and similar.

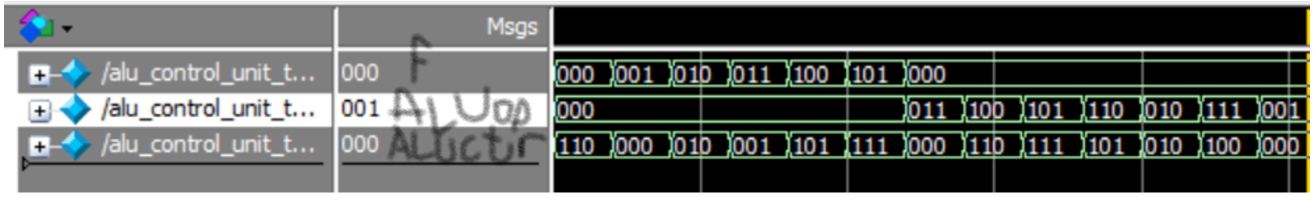
Combinational logic circuit written using boolean algebraic equations is as follows:

$$ALUctr\_2 = F_2'F_1'F_0' + F_2F_1'F_0' + F_2'F_1F_0 + ALOp2$$

$$ALUctr\_1 = F_2'F_1'F_0' + F_2'F_1F_0' + F_2F_1'F_0 + A_2A_1'A_0' + A_2A$$

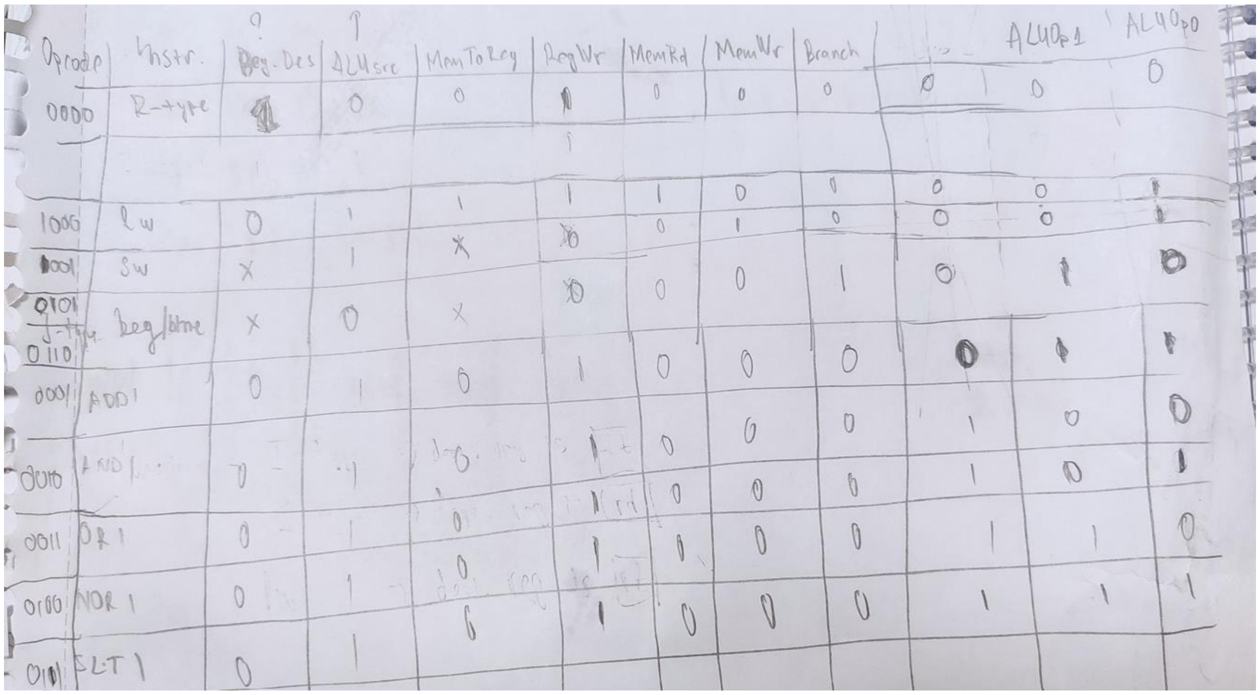
$$ALUctr\_0 = F_2 + F_1F_0 + A_2A_1'A_0 + A_2A_1A_0'$$

I implemented this component in verilog and tested it using the Altera's Modelsim. Here is the output of these signals in Modelsim.



```
                      Msgs
/alu_control_unit_t... 000  F       000 001 010 011 100 101 000
/alu_control_unit_t... 001  ALUop   000                     011 100 101 110 010 111 001
/alu_control_unit_t... 000  ALUctr  110 000 010 001 101 111 000 110 111 101 010 100 000
```

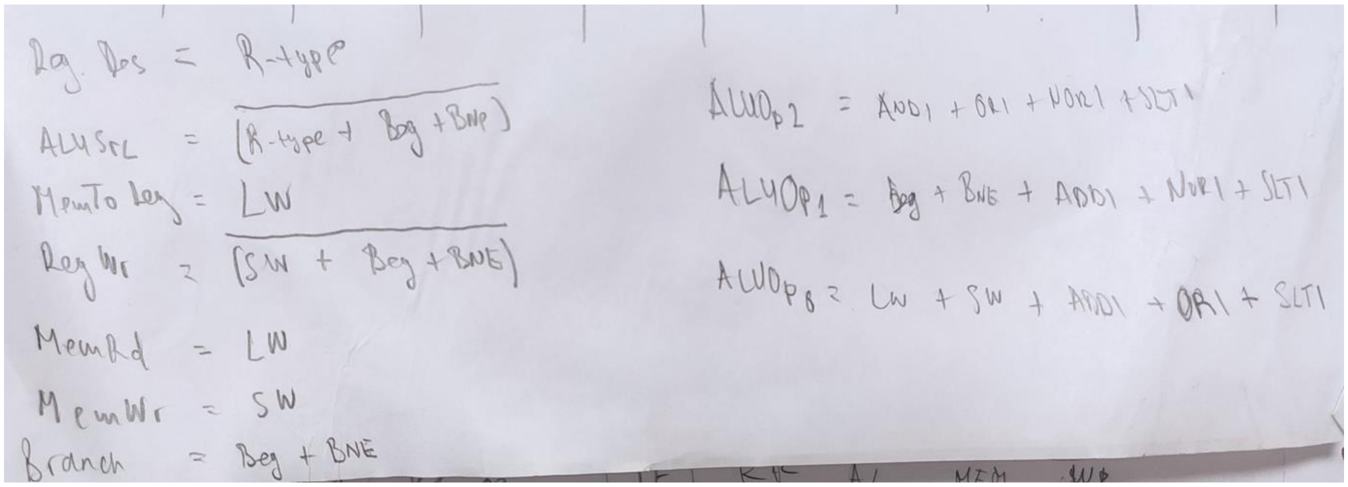
For the Main Control Unit I designed the following circuit:



Opcode	Instr.	Reg. Des	ALUsrc	MemToReg	RegWr	MemRd	MemWr	Branch		ALUOp1	ALUOp0
0000	R-type	1	0	0	1	0	0	0	0	0	0
1000	lw	0	1	1	1	1	0	0	0	0	1
1001	sw	X	1	X	~~X~~ 0	0	1	0	0	0	1
0101 J-type 0110	beq/bne	X	0	X	~~X~~ 0	0	0	1	0	1	0
0001	ADDI	0	1	0	1	0	0	0	0	1	1
0010	ANDI	0	1	0	1	0	0	0	1	0	0
0011	ORI	0	1	0	1	0	0	0	1	0	1
0100	NORI	0	1	0	1	0	0	0	1	1	0
0101	SLTI	0	1	0	1	0	0	0	1	1	1

Here I strictly followed the datapath that we were working on during our lectures.

Boolean mathematical equations for the signals I in this table are as follows:



Reg. Des = R-type

ALUSrc = $\overline{(\text{R-type} + \text{Beq} + \text{Bne})}$

MemToReg = LW

RegWr = $\overline{(\text{SW} + \text{Beq} + \text{BNE})}$

MemRd = LW

MemWr = SW

Branch = Beq + BNE

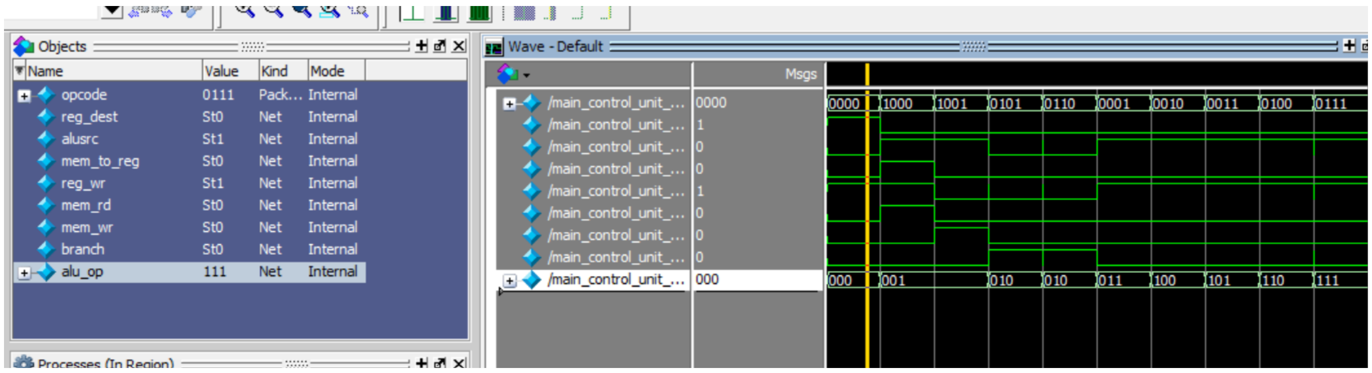
ALUOp2 = ANDI + ORI + NORI + SLTI

ALUOp1 = Beq + BNE + ADDI + NORI + SLTI

ALUOp0 = LW + SW + ADDI + ORI + SLTI

There are parts that I had to make for in order to implement the datapath from our slides.

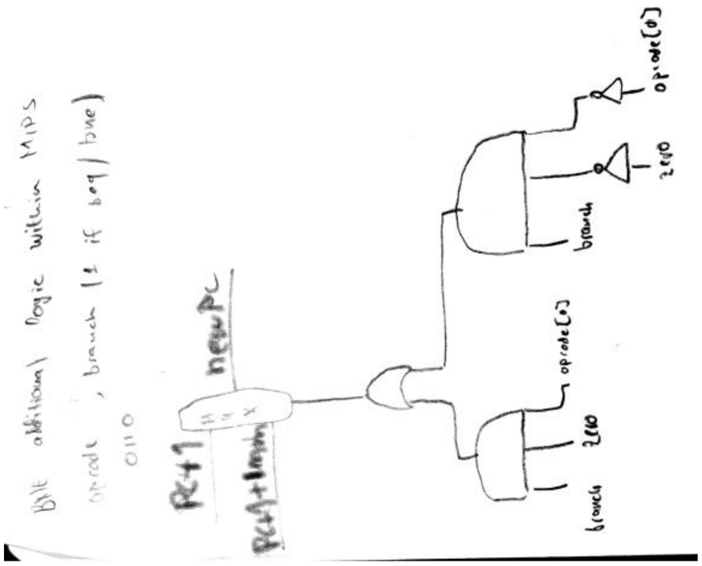
This was also of course implemented in verilog and tested in modelsim. Here is its test. It works and corresponds to the table above.



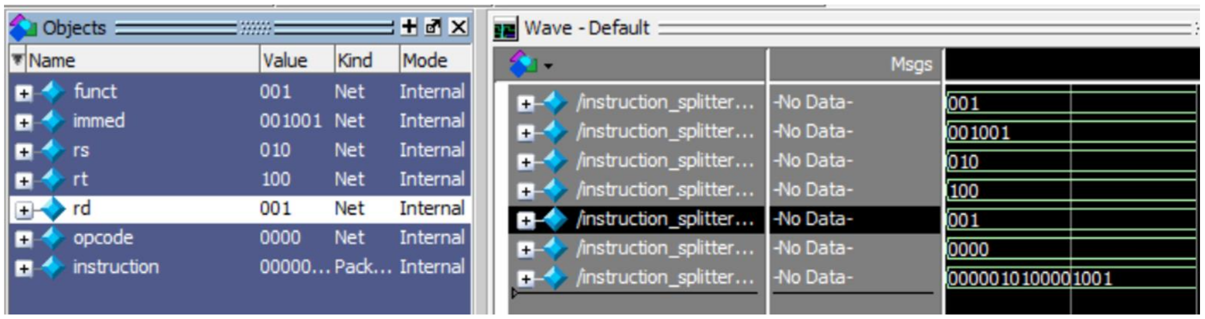
Name	Value	Kind	Mode
opcode	0111	Pack...	Internal
reg_dest	St0	Net	Internal
alusrc	St1	Net	Internal
mem_to_reg	St0	Net	Internal
reg_wr	St1	Net	Internal
mem_rd	St0	Net	Internal
mem_wr	St0	Net	Internal
branch	St0	Net	Internal
alu_op	111	Net	Internal

```
Wave - Default                  Msgs
/main_control_unit_...   0000   0000 1000 1001 0101 0110 0001 0010 0011 0100 0111
/main_control_unit_...   1
/main_control_unit_...   0
/main_control_unit_...   0
/main_control_unit_...   1
/main_control_unit_...   0
/main_control_unit_...   0
/main_control_unit_...   0
/main_control_unit_...   000    000  001       010  010  011  100  101  110  111
```

As you can see when producing a BRANCH signal, it does same for both Beq and Bne. In order to later on recognize which one will it be I decided to use a small cominatioral circuit within My MiniMIPS verilog file. For bne it branches if substraction of two registers is not zero and for beq if it is zero. The select bit for the Multiplexer that chooses if the Program couter will branch or not is the following:


BNE additional logic within MIPS
opcode, branch (1 if beq / bne)
0110
newPC
PC+1
PC+1+Imm
opcode[0]
zero
branch
opcode[0]
zero
branch

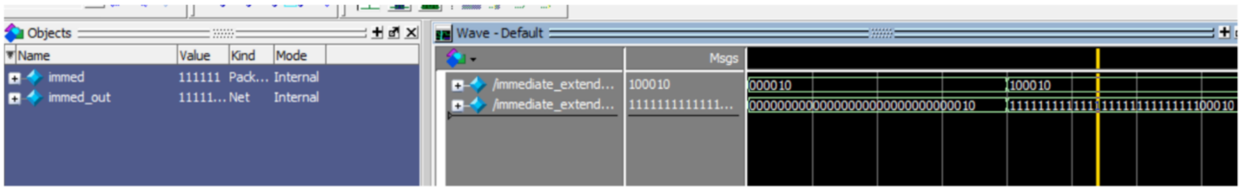
Also I implemented another separate Unit which is Instruction splitter. What it does is it takes the instruction and decodes it. It separates and returns rt, rs, rd, immed and func fields. Here is the test of this unit. In order to implemented I just used basic gates like and similar. This is just a big bus.



Name	Value	Kind	Mode
funct	001	Net	Internal
immed	001001	Net	Internal
rs	010	Net	Internal
rt	100	Net	Internal
rd	001	Net	Internal
opcode	0000	Net	Internal
instruction	00000...	Pack...	Internal

```
Wave - Default                Msgs
/instruction_splitter...   -No Data-   001
/instruction_splitter...   -No Data-   001001
/instruction_splitter...   -No Data-   010
/instruction_splitter...   -No Data-   100
/instruction_splitter...   -No Data-   001
/instruction_splitter...   -No Data-   0000
/instruction_splitter...   -No Data-   0000010100001001
```

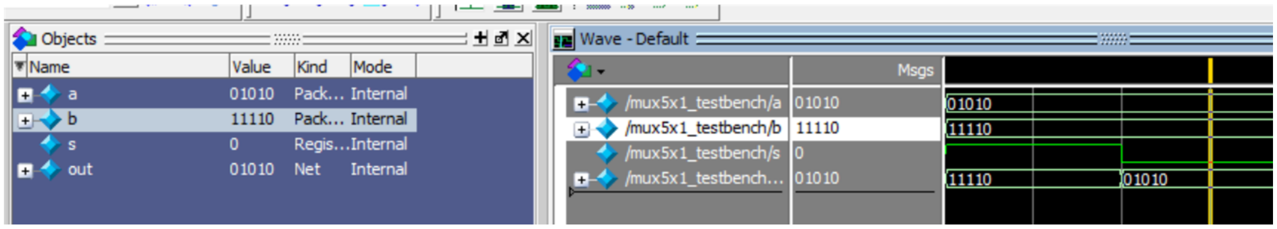
Another thing I needed was SignImmedExtented from the datapath. What it does is just extends its sign in order to make a 32-bit out of 6-bits gives as our IMMED value. Here are the tests:



Name	Value	Kind	Mode
immed	111111	Pack...	Internal
immed_out	11111...	Net	Internal

```
/immediate_extend...   100010            000010                              100010
/immediate_extend...   1111111111111...  00000000000000000000000000000010    11111111111111111111111111100010
```

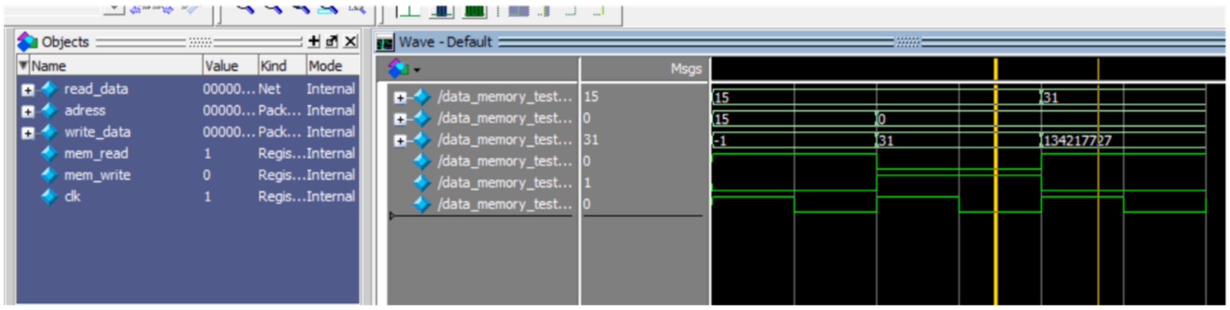
I also implemented a 5x1 mux and and here are the results for it



Name	Value	Kind	Mode
a	01010	Pack...	Internal
b	11110	Pack...	Internal
s	0	Regis...	Internal
out	01010	Net	Internal

```
/mux5x1_testbench/a   01010   01010
/mux5x1_testbench/b   11110   11110
/mux5x1_testbench/s   0
/mux5x1_testbench...  01010   11110   01010
```

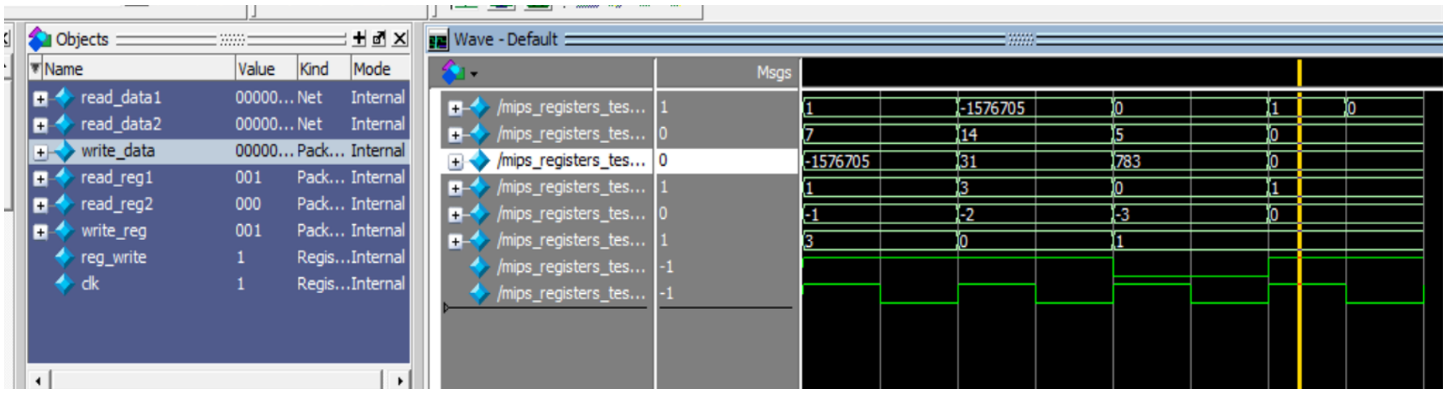
I also implemented a block unit for memory and tested it. I have an address I access and mem write and mem read bits in here.



Name	Value	Kind	Mode
read_data	00000...	Net	Internal
adress	00000...	Pack...	Internal
write_data	00000...	Pack...	Internal
mem_read	1	Regis...	Internal
mem_write	0	Regis...	Internal
clk	1	Regis...	Internal

```
/data_memory_test...  15   15          31
/data_memory_test...  0    15    0
/data_memory_test...  31   -1    31    134217727
/data_memory_test...  0
/data_memory_test...  1
/data_memory_test...  0
```

Also I implemented register data block where our register values are stored.

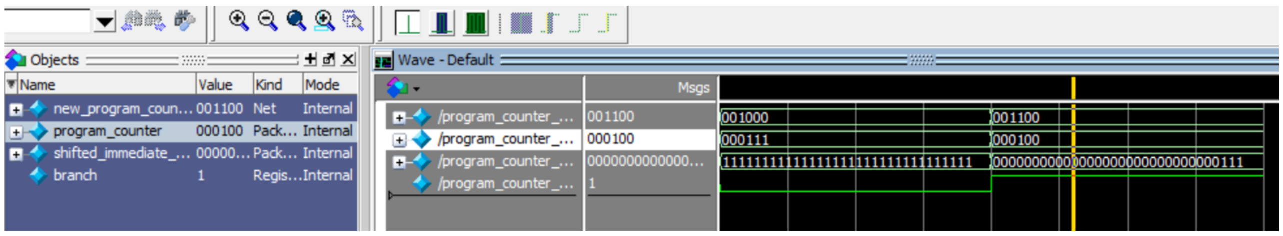


Name	Value	Kind	Mode
read_data1	00000...	Net	Internal
read_data2	00000...	Net	Internal
write_data	00000...	Pack...	Internal
read_reg1	001	Pack...	Internal
read_reg2	000	Pack...	Internal
write_reg	001	Pack...	Internal
reg_write	1	Regis...	Internal
clk	1	Regis...	Internal

```
/mips_registers_tes...  1    1         -1576705   0     1   0
/mips_registers_tes...  0    7         14         5     0
/mips_registers_tes...  0    -1576705  31         783   0
/mips_registers_tes...  1    1         3          0     1
/mips_registers_tes...  0    -1        -2         -3    0
/mips_registers_tes...  1    3         0          1
/mips_registers_tes...  -1
/mips_registers_tes...  -1
```

It can read 2 values at a time. It can write one value at a time. This is used to access or alter registers like Rs, Rt and Rd in Mips.

Also, I implemented a Program Counter which is being updated after each cycle. It test looks like this.

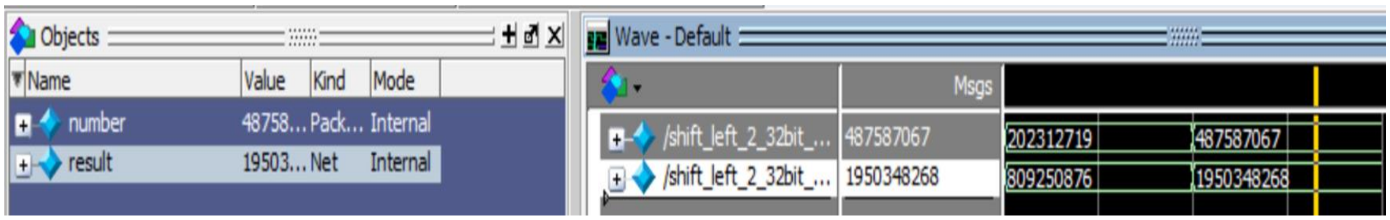


Name	Value	Kind	Mode
new_program_coun...	001100	Net	Internal
program_counter	000100	Pack...	Internal
shifted_immediate_...	00000...	Pack...	Internal
branch	1	Regis...	Internal

	Msgs		
/program_counter_...	001100	001000	001100
/program_counter_...	000100	000111	000100
/program_counter_...	0000000000000...	11111111111111111111111111111111	00000000000000000000000000000111
/program_counter_...	1		

We can see how on branch the value changes.

Also for the Program Counter I implemented a left shifted in order to get the value of the address multiplied by 4. Note that since we here PC is being incremented by 1 and not by 4 in each cycle since we are working with arrays I didn't use this shift left unit. Nevertheless, here is its test.



Name	Value	Kind	Mode
number	48758...	Pack...	Internal
result	19503...	Net	Internal

	Msgs		
/shift_left_2_32bit_...	487587067	202312719	487587067
/shift_left_2_32bit_...	1950348268	809250876	1950348268

For the other units you can see tests of my previous homework because I was using the same codes for things like adder, multiplexer, xor and similar.

Now for the MiniMIPS unit I implemented the datapath that was in the slides. The only deviation from it is the branch cominatioral gate structure I mentioned before.

In order to test this MIPS processor since In the draft we were given instruction as an input and result as an output I was strictly following that in order not to break the rules. Also, each instruction I have in my MIPS processor works correctly and each instruction is tested twice, meaning 30 instructions that were tested in total.

Also, there is register_output and data_output so that we can see the changes here.

Instructions executed and their tests:

Registers at the beginning:

00000000000000000000000000000000

00000000000000000000000000000001

00000000000000000000000000001000

00000000000000000000000000000011

00000000000000000000000000001111

00000000000000000000000000000101

00000000000000000000000000001110

00000000000000000000000000000111

instruction: b0000_010_100_001_001

add $1, $2, $4

00000000000000000000000000000000

00000000000000000000000000000001 ----> contents here changed to 10111

00000000000000000000000000001000

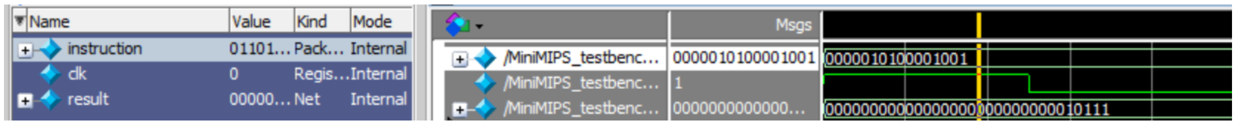
00000000000000000000000000000011

00000000000000000000000000001111

00000000000000000000000000000101

00000000000000000000000000001110

00000000000000000000000000000111



Name	Value	Kind	Mode
instruction	01101...	Pack...	Internal
clk	0	Regis...	Internal
result	00000...	Net	Internal

```
                        Msgs
/MiniMIPS_testbenc...   0000010100001001    0000010100001001
/MiniMIPS_testbenc...   1
/MiniMIPS_testbenc...   0000000000000...    00000000000000000000000000010111
```

Result value corresponds to the value of addition of the two registers given

b0000011001101001

add $5, $1, $3

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001000

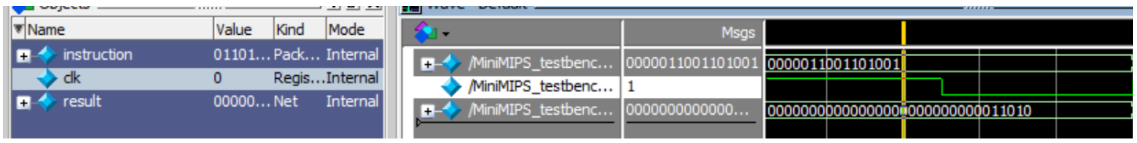
00000000000000000000000000000011

00000000000000000000000000001111

00000000000000000000000000000101 ----> contents here changed to 11010 ---> 10111 + 00011

00000000000000000000000000001110

00000000000000000000000000000111



```
Name         Value     Kind     Mode
instruction  01101...  Pack...  Internal
clk          0         Regis... Internal
result       00000...  Net      Internal

                      Msgs
/MiniMIPS_testbenc... 0000011001101001  0000011001101001
/MiniMIPS_testbenc... 1
/MiniMIPS_testbenc... 0000000000000...  0000000000000000 0000000000011010
```

Result value corresponds to the value of addition of the two registers given

b0000110111101010

sub $5, $7, $6

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001000

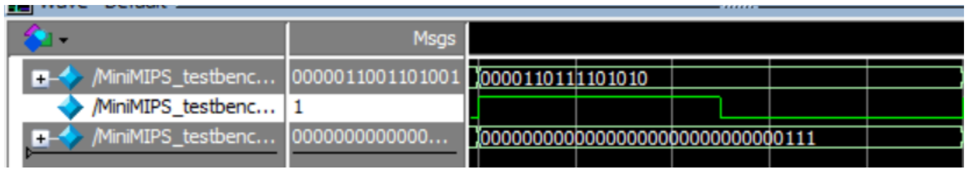
00000000000000000000000000000011

00000000000000000000000000001111

00000000000000000000000000011010 ----> contents here changed to 111 ---> 7 (14 – 7)

00000000000000000000000000001110

00000000000000000000000000000111



```
                      Msgs
/MiniMIPS_testbenc... 0000011001101001  0000110111101010
/MiniMIPS_testbenc... 1
/MiniMIPS_testbenc... 0000000000000...  00000000000000000000000000000111
```

Result value corresponds to the value of addition of the two registers given b0000101101101010

sub $5, $5, $5

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001000

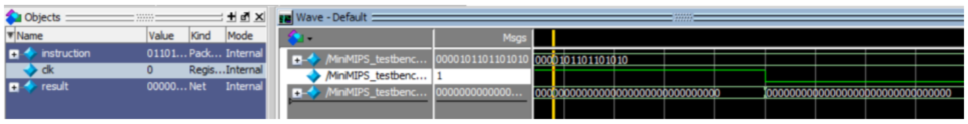
00000000000000000000000000000011

00000000000000000000000000001111

00000000000000000000000000000000 ----> contents here changed to 0

00000000000000000000000000001110

00000000000000000000000000000111



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal
```

```
Wave - Default
                          Msgs
/MiniMIPS_testbenc...     0000101101101010    0000101101101010
/MiniMIPS_testbenc...     1
/MiniMIPS_testbenc...     0000000000000...    00000000000000000000000000000000    00000000000000000000000000000000
```

Here we are also able to that by following the convention the register value's are updated on the negative edge when R-type is being run since RegWrite singal is ON!

b0000111110101000

and $5, $7, $6

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001000

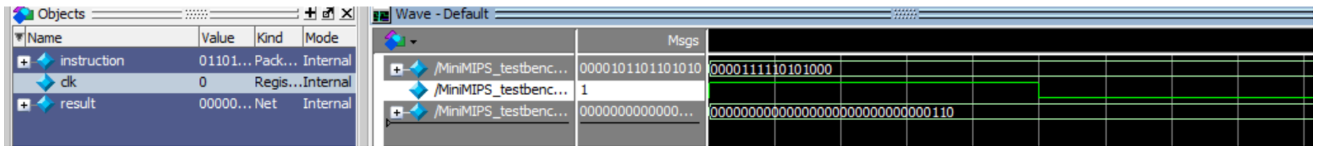
00000000000000000000000000000011

00000000000000000000000000001111

00000000000000000000000000000000 ----> contents here changed to 1110 & 0111 = 110 ---> 6

00000000000000000000000000001110

00000000000000000000000000000111



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal
```

```
Wave - Default
                          Msgs
/MiniMIPS_testbenc...     0000101101101010    0000111110101000
/MiniMIPS_testbenc...     1
/MiniMIPS_testbenc...     0000000000000...    00000000000000000000000000000110
```

b0000010100100000

and $4, $2, $4

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001000

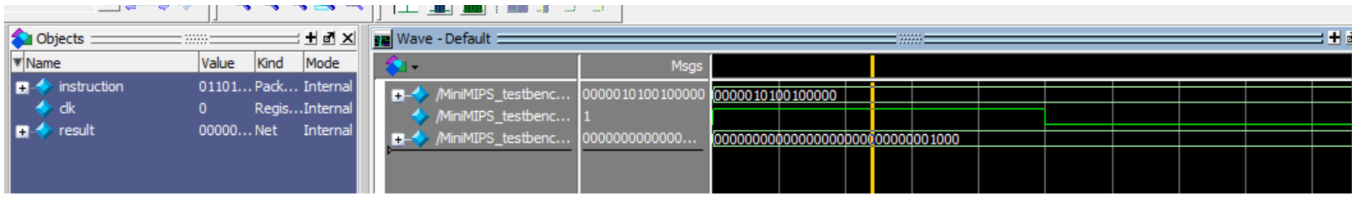
00000000000000000000000000000011

00000000000000000000000000001111 ----> contents here changed to 1000 ---> 8

00000000000000000000000000000110

00000000000000000000000000001110

00000000000000000000000000000111


Objects

Name	Value	Kind	Mode
instruction	01101...	Pack...	Internal
clk	0	Regis...	Internal
result	00000...	Net	Internal

Wave - Default

```
                        Msgs
/MiniMIPS_testbenc...   0000010100100000   0000010100100000
/MiniMIPS_testbenc...   1
/MiniMIPS_testbenc...   0000000000000...   00000000000000000000000000001000
```

b0000011100100011

xor $4, $3, $4

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001000

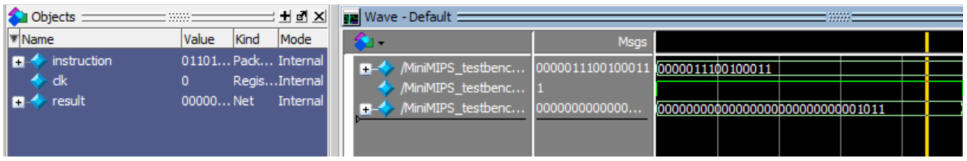
00000000000000000000000000000011

00000000000000000000000000001000 ----> contents here changed to 1011 ---> 11

00000000000000000000000000000110

00000000000000000000000000001110

00000000000000000000000000000111



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal
```

```
Wave - Default
                        Msgs
/MiniMIPS_testbenc...   0000011100100011   0000011100100011
/MiniMIPS_testbenc...   1
/MiniMIPS_testbenc...   0000000000000...   00000000000000000000000000001011
```

b0000_111_100_010_011

xor $2, $7, $4

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001000 ---> contents of this register should change to 1100

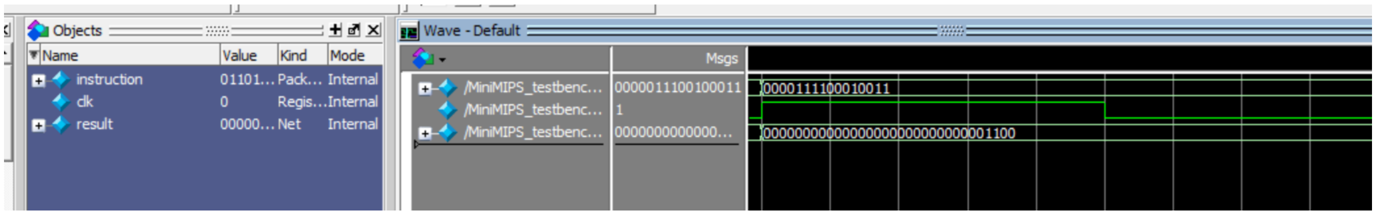
00000000000000000000000000000011

00000000000000000000000000001011

00000000000000000000000000000110

00000000000000000000000000001110

00000000000000000000000000000111



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal
```

```
Wave - Default
                        Msgs
/MiniMIPS_testbenc...   0000011100100011   0000011100100011
/MiniMIPS_testbenc...   1
/MiniMIPS_testbenc...   0000000000000...   00000000000000000000000000001100
```

b0000_001_010_110_100

nor $6, $1, $2

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001100

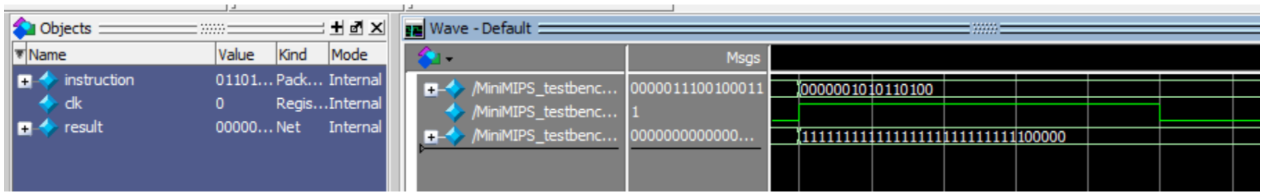
00000000000000000000000000000011

00000000000000000000000000001011

00000000000000000000000000000110

00000000000000000000000000001110 ---> contents of this register should change to 11111111111111111111111111100000

00000000000000000000000000000111



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal
```

```
Wave - Default
                        Msgs
/MiniMIPS_testbenc...   0000011100100011   0000001010110100
/MiniMIPS_testbenc...   1
/MiniMIPS_testbenc...   0000000000000...   11111111111111111111111111100000
```

b0000_110_100_011_100

nor $3, $6, $4

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001100

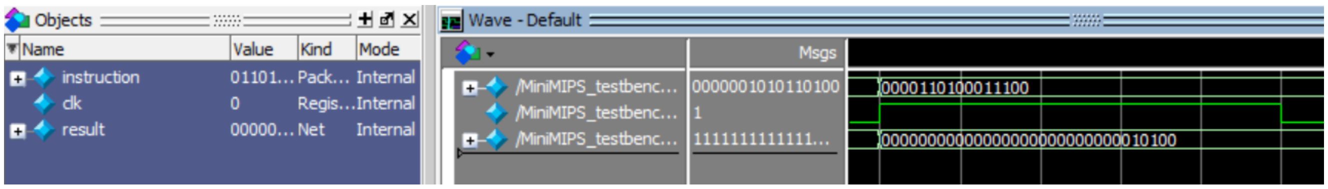
00000000000000000000000000000011 ---> contents of this register should change to 10100

00000000000000000000000000001011

00000000000000000000000000000110

11111111111111111111111111100000

00000000000000000000000000000111



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal
```

```
Wave - Default
                        Msgs
/MiniMIPS_testbenc...   0000001010110100   0000110100011100
/MiniMIPS_testbenc...   1
/MiniMIPS_testbenc...   1111111111111...   00000000000000000000000000010100
```

b0000_001_010_101_101

or $5, $1, $2

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001100

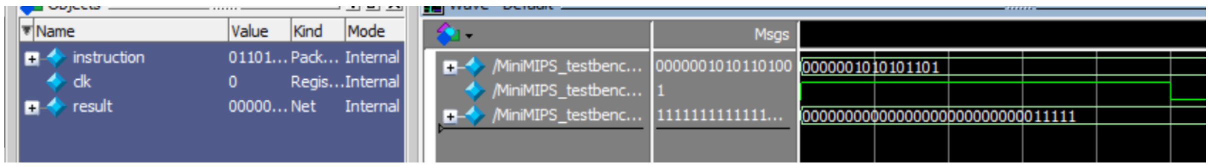
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000000110 ---> contents of this register should change to 11111

11111111111111111111111111100000

00000000000000000000000000000111



```
Name            Value     Kind      Mode
+ instruction   01101...  Pack...   Internal
  clk           0         Regis...  Internal
+ result        00000...  Net       Internal
```

```
                                Msgs
+ /MiniMIPS_testbenc...   0000001010110100   0000001010101101
  /MiniMIPS_testbenc...   1
+ /MiniMIPS_testbenc...   1111111111111...   00000000000000000000000000011111
```

b0000_110_101_111_101

or $7, $6, $5

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001100

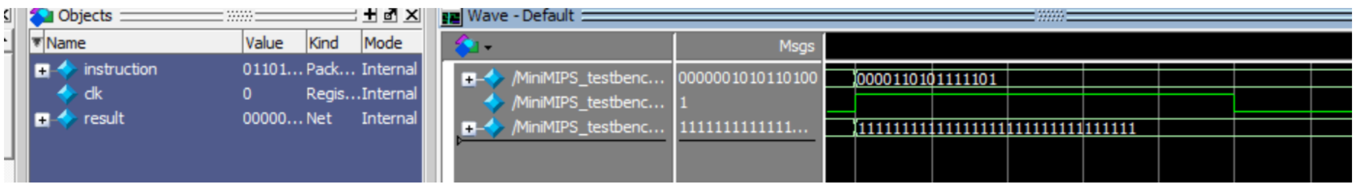
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

11111111111111111111111111100000

00000000000000000000000000000111 ---> contents of this register should change to
11111111111111111111111111111111



```
Objects
Name            Value     Kind      Mode
+ instruction   01101...  Pack...   Internal
  clk           0         Regis...  Internal
+ result        00000...  Net       Internal
```

```
Wave - Default
                                Msgs
+ /MiniMIPS_testbenc...   0000001010110100   0000110101111101
  /MiniMIPS_testbenc...   1
+ /MiniMIPS_testbenc...   1111111111111...   11111111111111111111111111111111
```

b0001_010_111_011_111

addi $7, $2, 011111

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001100

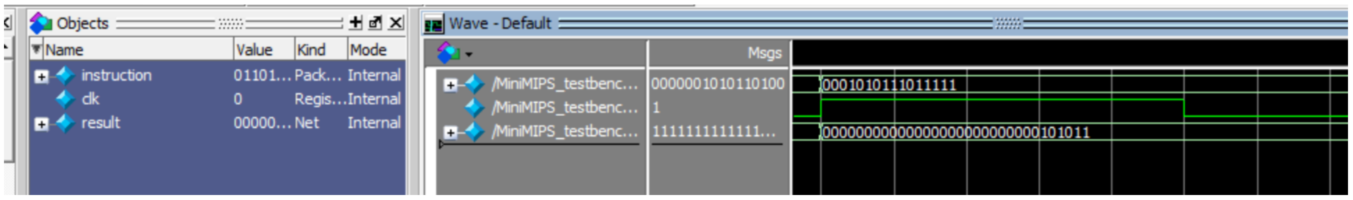
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

11111111111111111111111111100000 1100+11111

11111111111111111111111111111111 ----> contents of this register should change to 101011



```
Objects
Name          Value    Kind     Mode
+ instruction 01101... Pack...  Internal
  clk         0        Regis... Internal
+ result      00000... Net      Internal

Wave - Default
                          Msgs
+ /MiniMIPS_testbenc...   0000001010110100   0001010111011111
  /MiniMIPS_testbenc...   1
+ /MiniMIPS_testbenc...   1111111111111...   00000000000000000000000000101011
```

b0001_111_110_000_101

addi $6, $7, 000101

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001100

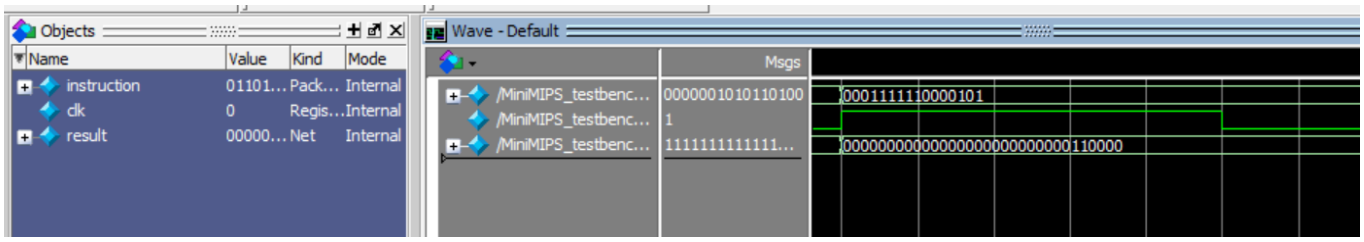
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

11111111111111111111111111100000 ----> contents of this register should change to 110000

00000000000000000000000000101011



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal

Wave - Default
                           Msgs
/MiniMIPS_testbenc...      0000001010110100   0001111110000101
/MiniMIPS_testbenc...      1
/MiniMIPS_testbenc...      1111111111111...   00000000000000000000000000110000
```

b0010_100_110_001_010

andi $6, $4, 1010

00000000000000000000000000000000

00000000000000000000000000010111

00000000000000000000000000001100

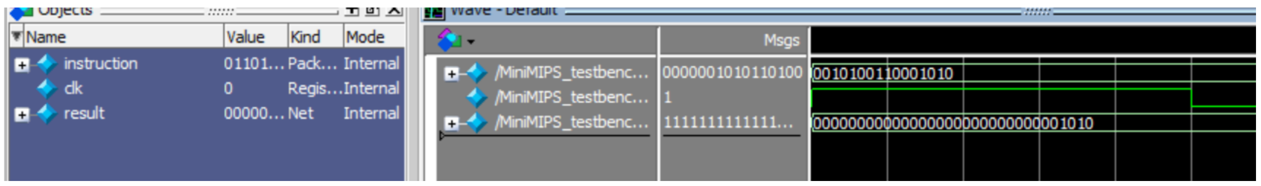
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000110000 ----> contents of this register should change to 1010

00000000000000000000000000101011



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal

Wave - Default
                           Msgs
/MiniMIPS_testbenc...      0000001010110100   0010100110001010
/MiniMIPS_testbenc...      1
/MiniMIPS_testbenc...      1111111111111...   00000000000000000000000000001010
```

b0010_001_001_000_001

andi $1, $2, 0001

00000000000000000000000000000000

00000000000000000000000000010111 ----> contents of this register should change to 000001

00000000000000000000000000001100

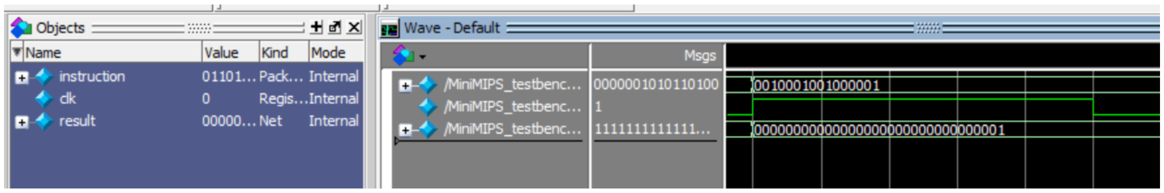
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000001010

00000000000000000000000000101011



```
Objects
Name         Value     Kind      Mode
instruction  01101...  Pack...   Internal
clk          0         Regis...  Internal
result       00000...  Net       Internal
```

```
Wave - Default
                        Msgs
/MiniMIPS_testbenc...   0000001010110100   0010001001000001
/MiniMIPS_testbenc...   1
/MiniMIPS_testbenc...   1111111111111...   00000000000000000000000000000001
```

b0011_100_110_001_111

ori $6, $4, 1111

00000000000000000000000000000000

00000000000000000000000000000001

00000000000000000000000000001100

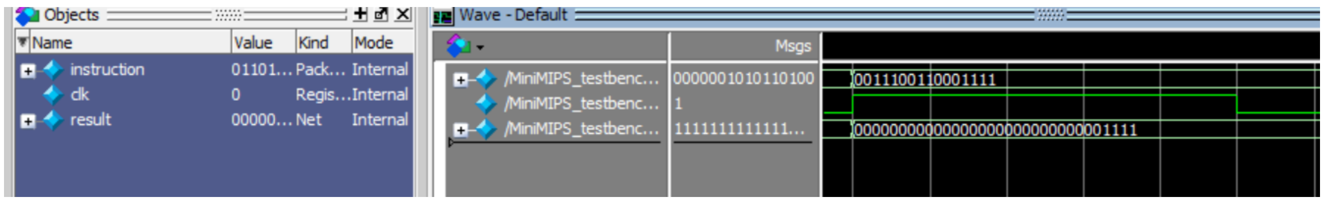
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000001010 ----> contents of this register should change to 1111

00000000000000000000000000101011



```
Objects
Name         Value     Kind      Mode
instruction  01101...  Pack...   Internal
clk          0         Regis...  Internal
result       00000...  Net       Internal
```

```
Wave - Default
                        Msgs
/MiniMIPS_testbenc...   0000001010110100   0011100110001111
/MiniMIPS_testbenc...   1
/MiniMIPS_testbenc...   1111111111111...   00000000000000000000000000001111
```

b0011_001_001_000_001

ori $1, $1, 0001

00000000000000000000000000000000

00000000000000000000000000000001 ----> contents of this register should stay 0001 since output is the same

00000000000000000000000000001100

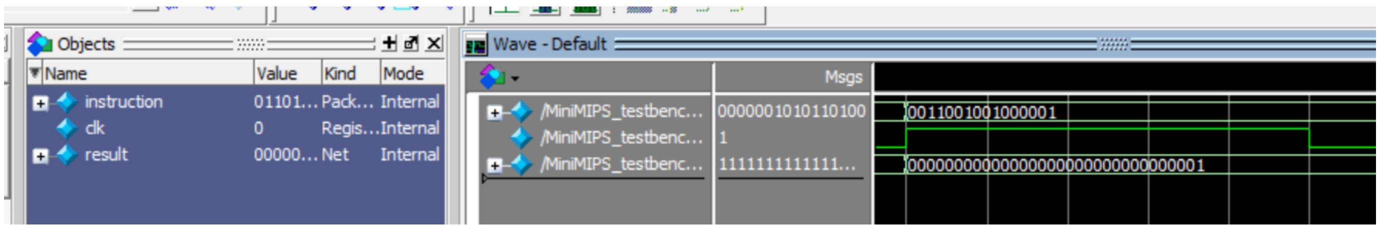
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000001111

00000000000000000000000000101011



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal
```

```
Wave - Default
                          Msgs
/MiniMIPS_testbenc...     000000101010110100    0011001001000001
/MiniMIPS_testbenc...     1
/MiniMIPS_testbenc...     1111111111111...      00000000000000000000000000000001
```

b0100_100_110_001_111

nori $6, $4, 1111

00000000000000000000000000000000

00000000000000000000000000000001

00000000000000000000000000001100

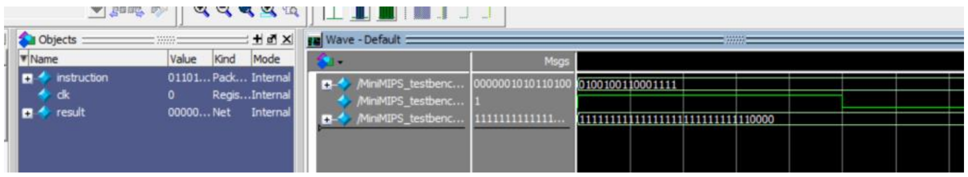
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000001111 ----> contents of this register change to 11111111111111111111111111110000

00000000000000000000000000101011



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal

Wave - Default
                          Msgs
/MiniMIPS_testbenc...  0000001010110100   0100100110001111
/MiniMIPS_testbenc...  1
/MiniMIPS_testbenc...  11111111111111...  11111111111111111111111111110000
```

b0100_001_001_000_001

nori $1, $1, 0001

00000000000000000000000000000000

00000000000000000000000000000001 ----> contents of this register should change to 11111111111111111111111111111110

00000000000000000000000000001100

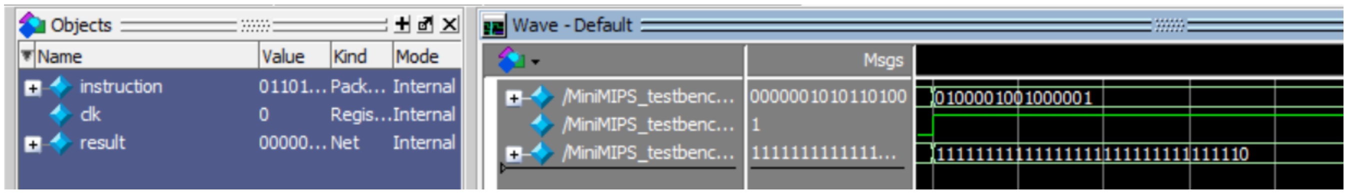
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

11111111111111111111111111110000

00000000000000000000000000101011



```
Objects
Name          Value     Kind      Mode
instruction   01101...  Pack...   Internal
clk           0         Regis...  Internal
result        00000...  Net       Internal

Wave - Default
                          Msgs
/MiniMIPS_testbenc...  0000001010110100   0100001001000001
/MiniMIPS_testbenc...  1
/MiniMIPS_testbenc...  1111111111111...   11111111111111111111111111111110
```

b0111_100_110_011_111

slti $6, $4, 11111

00000000000000000000000000000000

11111111111111111111111111111110

00000000000000000000000000001100

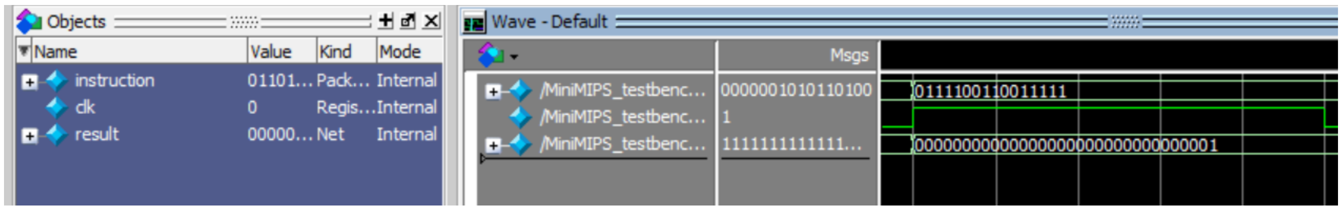
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

11111111111111111111111111110000 ----> contents of this register should change to 1 since 1011 < 11111

00000000000000000000000000101011



```
Objects
Name          Value     Kind      Mode
+ instruction 01101...  Pack...   Internal
  clk         0         Regis...  Internal
+ result      00000...  Net       Internal
```

```
Wave - Default
                                 Msgs
+ /MiniMIPS_testbenc...  0000001010110100   0111100110011111
  /MiniMIPS_testbenc...  1
+ /MiniMIPS_testbenc...  1111111111111...    00000000000000000000000000000001
```

b0111_001_001_000_000

slti $1, $1, 0000

00000000000000000000000000000000

11111111111111111111111111111110 ----> contents of this register should change to 1 since -2 < 0

00000000000000000000000000001100

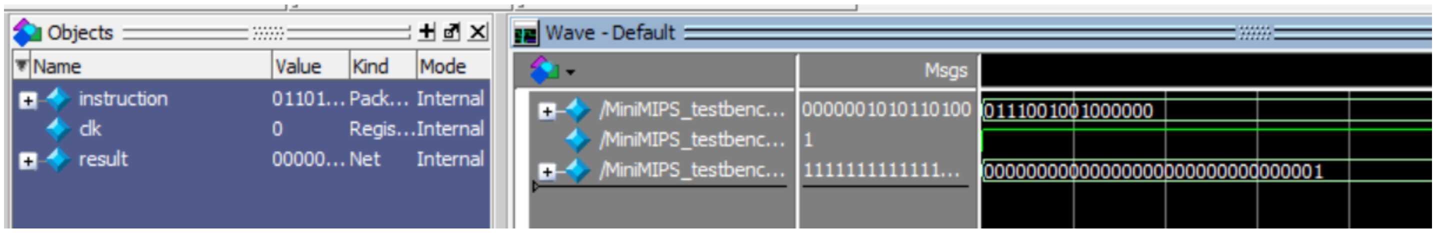
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000000001

00000000000000000000000000101011



```
Objects
Name          Value     Kind      Mode
+ instruction 01101...  Pack...   Internal
  clk         0         Regis...  Internal
+ result      00000...  Net       Internal
```

```
Wave - Default
                                 Msgs
+ /MiniMIPS_testbenc...  0000001010110100   0111001001000000
  /MiniMIPS_testbenc...  1
+ /MiniMIPS_testbenc...  1111111111111...    00000000000000000000000000000001
```

b1001_000_011_000_000

sw $3, 0($0) ----> should put 10100 in 00000 memory address location

Stays the same - regs

00000000000000000000000000000000

00000000000000000000000000000001

00000000000000000000000000001100

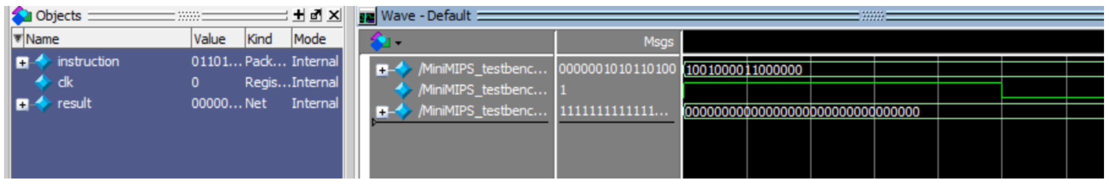
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000000001

00000000000000000000000000101011



Name	Value	Kind	Mode
instruction	01101...	Pack...	Internal
clk	0	Regis...	Internal
result	00000...	Net	Internal

	Msgs	
/MiniMIPS_testbenc...	0000001010110100	1001000011000000
/MiniMIPS_testbenc...	1	
/MiniMIPS_testbenc...	1111111111111...	00000000000000000000000000000000

Here we can only see that address of for the data to be altered is correctly calculated.

Data changes as predicted! - (will be shown a bit later)

b1001_000_111_000_011

sw $7, 3($0) ----> should put 101011 in 3rd memory address location

00000000000000000000000000000000

00000000000000000000000000000001

00000000000000000000000000001100

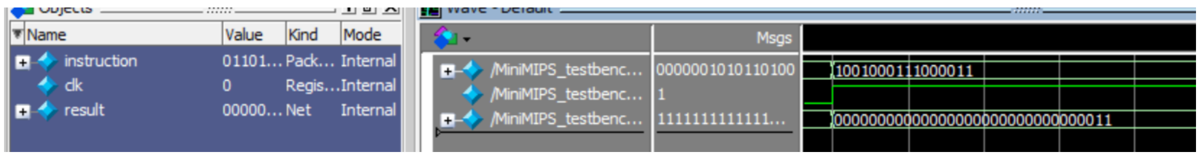
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000000001

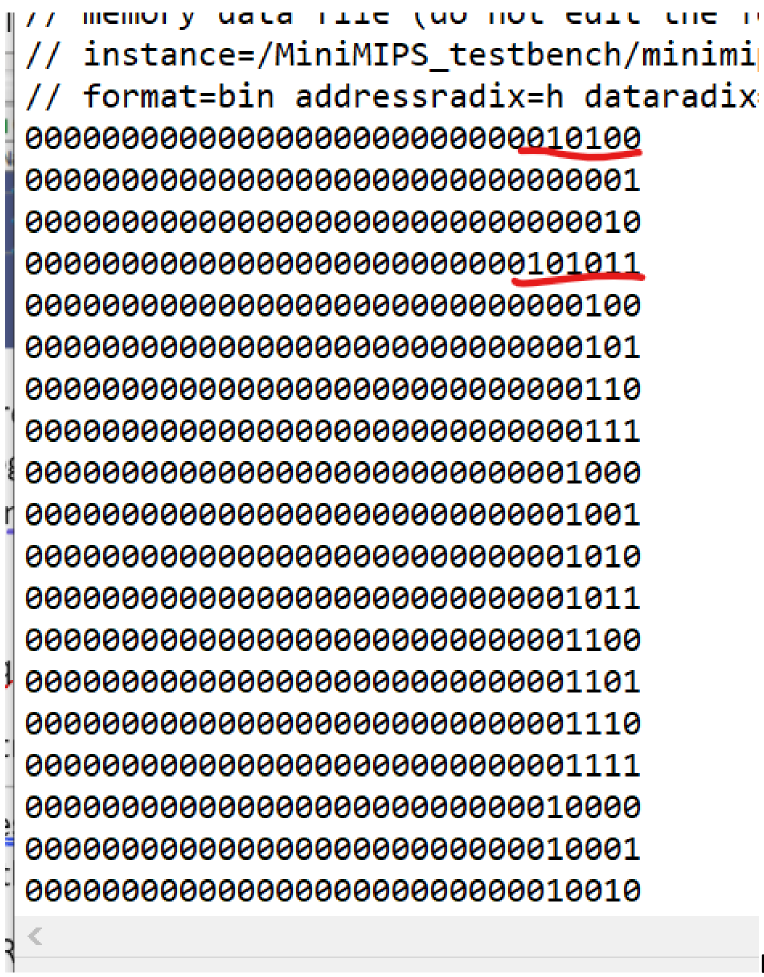
00000000000000000000000000101011



Name	Value	Kind	Mode
instruction	01101...	Pack...	Internal
clk	0	Regis...	Internal
result	00000...	Net	Internal

	Msgs	
/MiniMIPS_testbenc...	0000001010110100	1001000111000011
/MiniMIPS_testbenc...	1	
/MiniMIPS_testbenc...	1111111111111...	00000000000000000000000000000011

Same as previously, address for the data is calculated correctly.



```
// memory data file (do not edit the
// instance=/MiniMIPS_testbench/minimi
// format=bin addressradix=h dataradix
00000000000000000000000000010100
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000101011
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
```

Data correctly altered

b1000_000_001_000_000

lw $2, 0($0) ----> should put 10100 in $2

00000000000000000000000000000000

00000000000000000000000000000001

00000000000000000000000000001100 ----> contents of this register should change to 10100

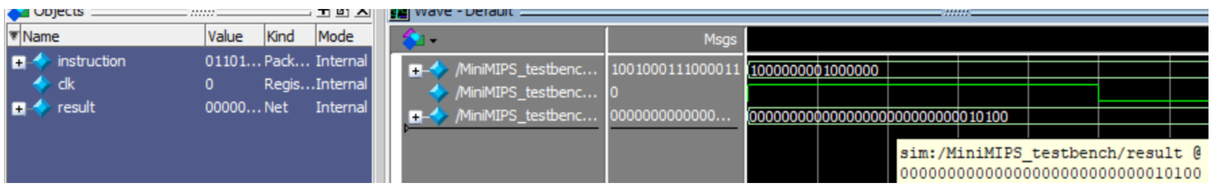
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000000001

00000000000000000000000000101011



Name	Value	Kind	Mode
instruction	01101...	Pack...	Internal
clk	0	Regis...	Internal
result	00000...	Net	Internal

```
                         Msgs
/MiniMIPS_testbenc...    1001000111000011    1000000001000000
/MiniMIPS_testbenc...    0
/MiniMIPS_testbenc...    0000000000000...    00000000000000000000000000010100

sim:/MiniMIPS_testbench/result @
00000000000000000000000000010100
```

We can see that data in here to be read from the memory is 10100 meaning the SW we preformed previously worked.

b1000_110_001_000_010

lw $1, 2($6) ----> should put 0 in $1

00000000000000000000000000000000

00000000000000000000000000000001 ----> contents of this register should change to 101011

00000000000000000000000000010100

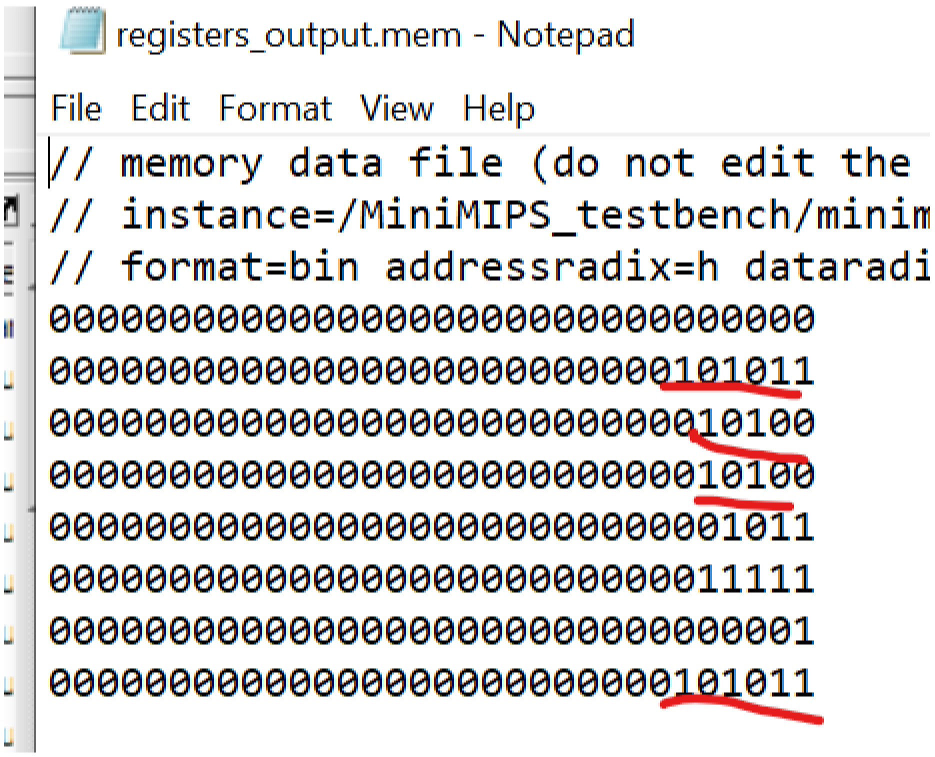
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000000001

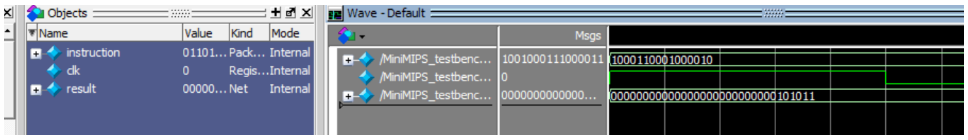
00000000000000000000000000101011


registers_output.mem - Notepad

File Edit Format View Help

```
// memory data file (do not edit the
// instance=/MiniMIPS_testbench/minim
// format=bin addressradix=h dataradi
00000000000000000000000000000000
00000000000000000000000000101011
00000000000000000000000000010100
00000000000000000000000000010100
00000000000000000000000000001011
00000000000000000000000000011111
00000000000000000000000000000001
00000000000000000000000000101011
```

Stored data correctly read!


Objects

Name	Value	Kind	Mode
instruction	01101...	Pack...	Internal
clk	0	Regis...	Internal
result	00000...	Net	Internal

Wave - Default

```
                          Msgs
/MiniMIPS_testbenc...   1001000111000011   10001100 01000010
/MiniMIPS_testbenc...   0
/MiniMIPS_testbenc...   0000000000000...   00000000000000000000000000101011
```

We can see that data in here to be read from the memory is 101011 meaning the SW we preformed previously worked.

b0101_111_001_000_011

beq $7, $1, 010 ----> should get the result of 010 + PC at this point ---> subtraction should give 0!

NOTHING CHANGES

00000000000000000000000000000000

00000000000000000000000000101011

00000000000000000000000000010100

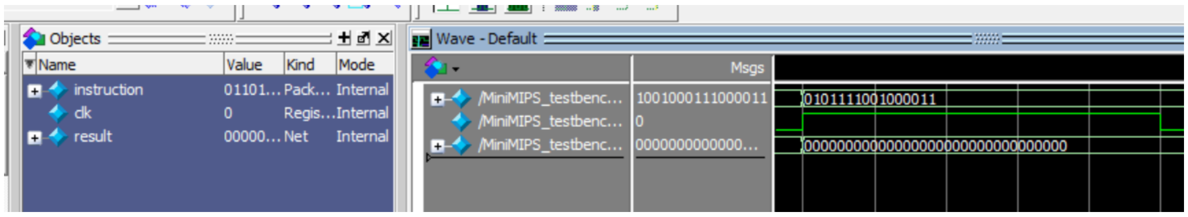
00000000000000000000000000010100

00000000000000000000000000001011

00000000000000000000000000011111

00000000000000000000000000000001

00000000000000000000000000101011



Name	Value	Kind	Mode
instruction	01101...	Pack...	Internal
clk	0	Regis...	Internal
result	00000...	Net	Internal

```
/MiniMIPS_testbenc...  1001000111000011  0101111001000011
/MiniMIPS_testbenc...  0
/MiniMIPS_testbenc...  0000000000000...  00000000000000000000000000000000
```

Here result is of course 0, meaning proper subtraction was performed and we can deduce from the program counter test that this indeed works fine. Please note that I wanted to show PC as output but didn't do it since I didn't want to in any way deviate from the draft provided!

beq $7, $2, 010

instruction = 16'b0101_111_010_000_011;

These 2 register contain different values and we can see that value returned is not 0 of course meaning all the correct singals were provided.

IN REGISTER DATA NOTHING CHANGES

For bne we perform that same and see similar results

//bne $7, $1, 010

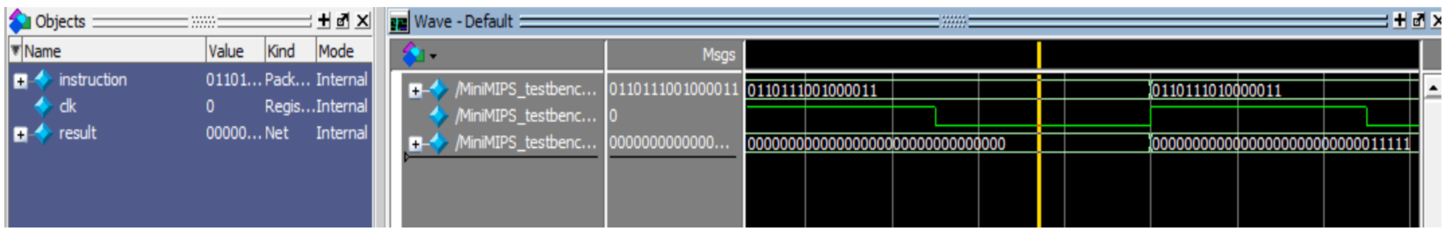
instruction = 16'b0110_111_001_000_011;

#`DELAY

//bne $7, $2, 010

instruction = 16'b0110_111_010_000_011;

#`DELAY



```
Objects
Name          Value    Kind     Mode
instruction   01101... Pack...  Internal
clk           0        Regis... Internal
result        00000... Net      Internal
```

```
Wave - Default
                        Msgs
/MiniMIPS_testbenc...   0110111001000011    0110111001000011                    0110111010000011
/MiniMIPS_testbenc...   0
/MiniMIPS_testbenc...   0000000000000...    00000000000000000000000000000000    00000000000000000000000000011111
```

Final note:

In this homework each requirement is implemented and tested.

Please be free to contact me if there are any ambiguities or if it would be easier for you to do a demo of this project.

I was following the draft for everything here and didn't change it a bit except of couple of mistakes that were present in it like changing reg [7:0] register [31:0]   to [31:0] register [7:0] and similar

Thank you for your attention.