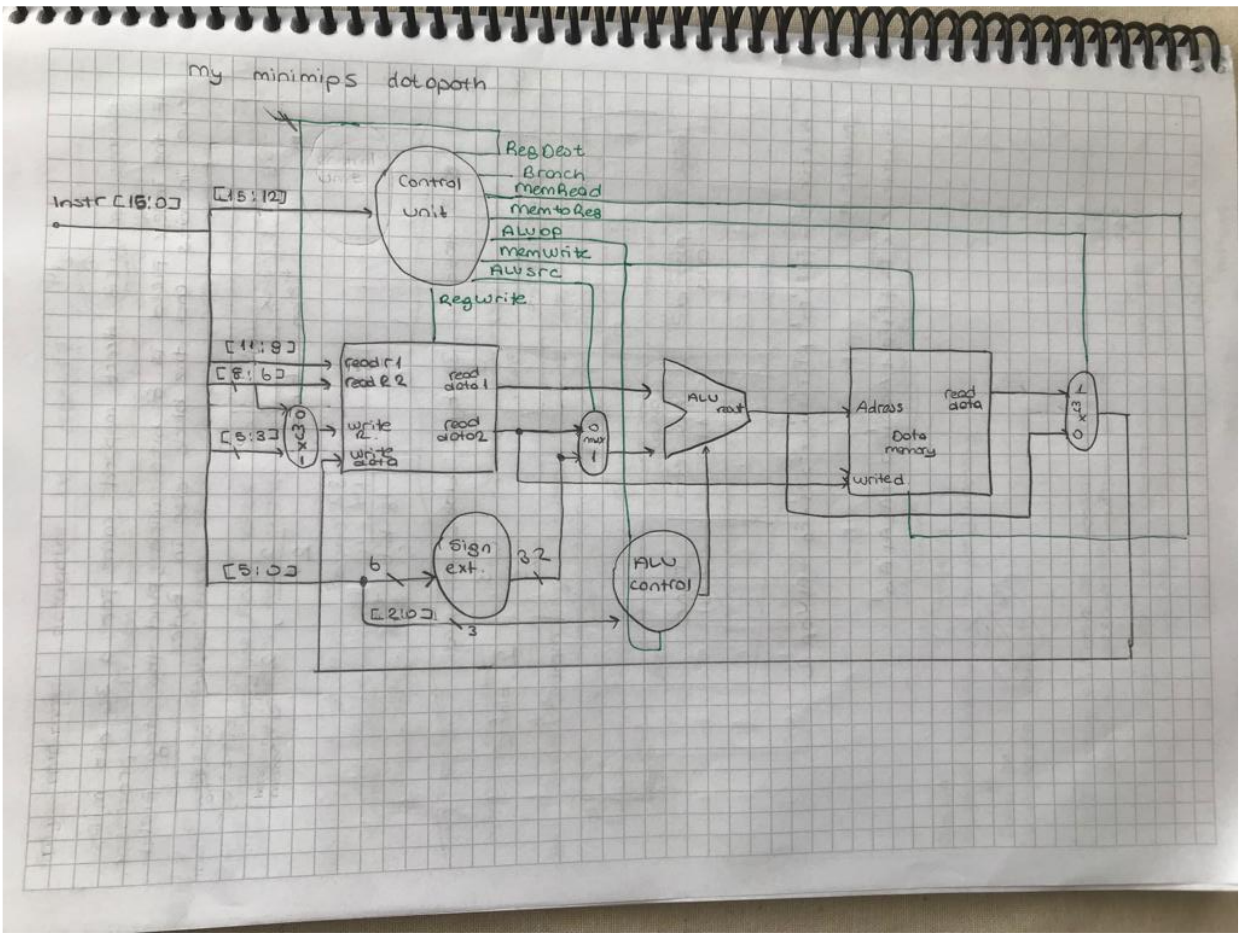
Cse 331 Computer Organization

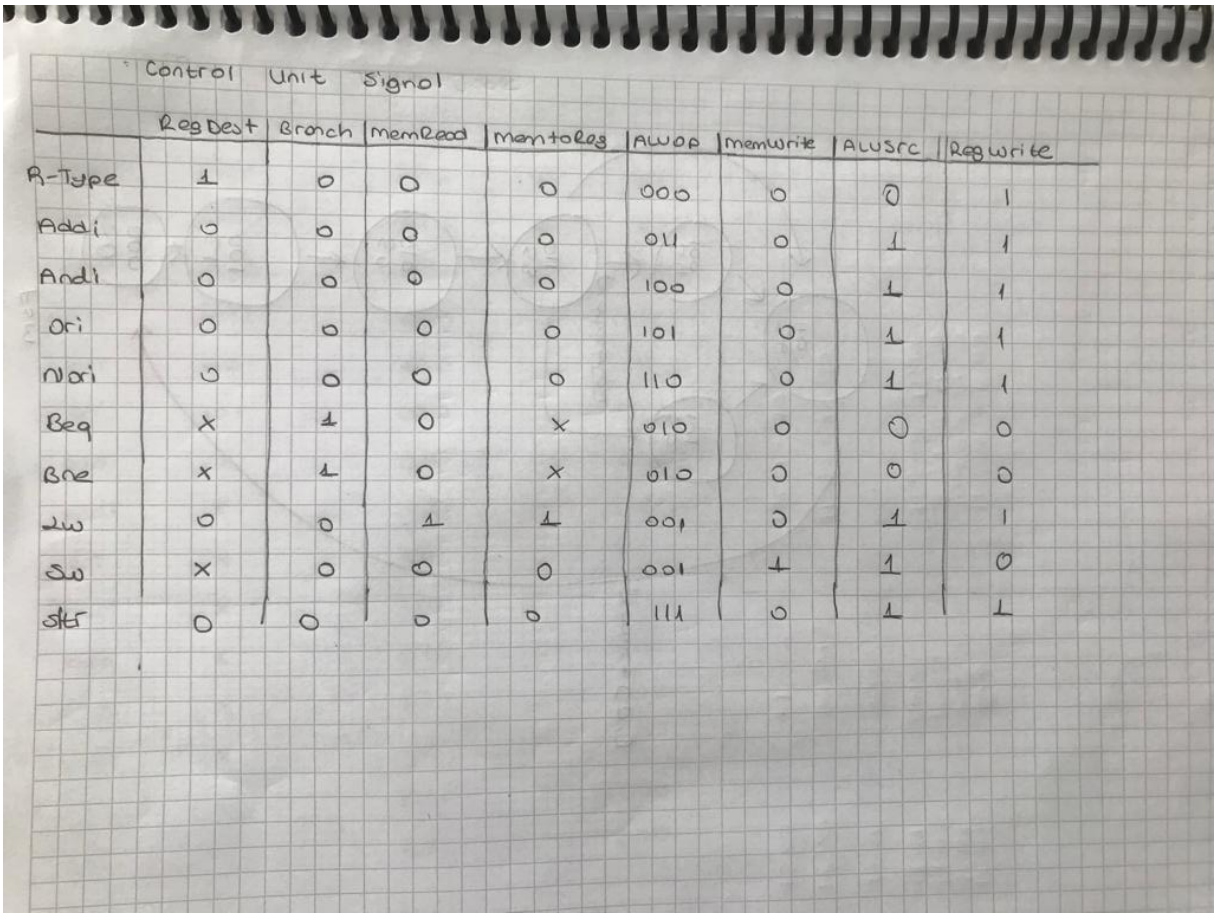
Tuba TOPRAK- 161044116

Final Project Report

First, I decided to create my own datapath and go through it step by step. (Like Mips)



Then the modules were made step by step. A signal table has been created for the control unit module. Signals work like mips. And from here, Aluop Code is generated to go to Alu Control.



Control Unit Signal

	RegDest	Branch	memRead	mem to reg	ALUop	memWrite	ALUsrc	Reg write
R-Type	1	0	0	0	000	0	0	1
Addi	0	0	0	0	011	0	1	1
Andi	0	0	0	0	100	0	1	1
Ori	0	0	0	0	101	0	1	1
Nori	0	0	0	0	110	0	1	1
Beq	x	1	0	x	010	0	0	0
Bne	x	1	0	x	010	0	0	0
lw	0	0	1	1	001	0	1	1
Sw	x	0	0	0	001	1	1	0
slt	0	0	0	0	111	0	1	1

```
module control_unit(
op_code,sign_reg_dest,
sign_branch,sign_mem_read,
sign_mem_to_reg,sign_mem_write,
sign_ALUsrc,sign_reg_write,sign_ALUop);

input [3:0] op_code;
output sign_reg_dest,sign_branch,sign_mem_read,sign_mem_to_reg,sign_mem_write,sign_reg_write;
output sign_ALUsrc;
output [2:0] sign_ALUop;
```

To make sure the signals are working correctly ,Testbench has been written.

Output:

```
# time= 0,opcode=0000,reg_dest=1,alu_src=0,mem_to_reg=0,reg_wrt=1,mem_read=0,mem_wrt=0,branch=0,alu_op=000
# time=20,opcode=0001,reg_dest=0,alu_src=1,mem_to_reg=0,reg_wrt=1,mem_read=0,mem_wrt=0,branch=0,alu_op=011
# time=40,opcode=0010,reg_dest=0,alu_src=1,mem_to_reg=0,reg_wrt=1,mem_read=0,mem_wrt=0,branch=0,alu_op=100
# time=60,opcode=0011,reg_dest=0,alu_src=1,mem_to_reg=0,reg_wrt=1,mem_read=0,mem_wrt=0,branch=0,alu_op=101
# time=80,opcode=0100,reg_dest=0,alu_src=1,mem_to_reg=0,reg_wrt=1,mem_read=0,mem_wrt=0,branch=0,alu_op=110
# time=100,opcode=0101,reg_dest=0,alu_src=0,mem_to_reg=0,reg_wrt=0,mem_read=0,mem_wrt=0,branch=1,alu_op=010
# time=120,opcode=0110,reg_dest=0,alu_src=0,mem_to_reg=0,reg_wrt=0,mem_read=0,mem_wrt=0,branch=1,alu_op=010
# time=140,opcode=0111,reg_dest=0,alu_src=1,mem_to_reg=0,reg_wrt=1,mem_read=0,mem_wrt=0,branch=0,alu_op=111
# time=160,opcode=1000,reg_dest=0,alu_src=1,mem_to_reg=1,reg_wrt=1,mem_read=1,mem_wrt=0,branch=0,alu_op=001
# time=180,opcode=1001,reg_dest=0,alu_src=1,mem_to_reg=0,reg_wrt=0,mem_read=0,mem_wrt=1,branch=0,alu_op=001
```

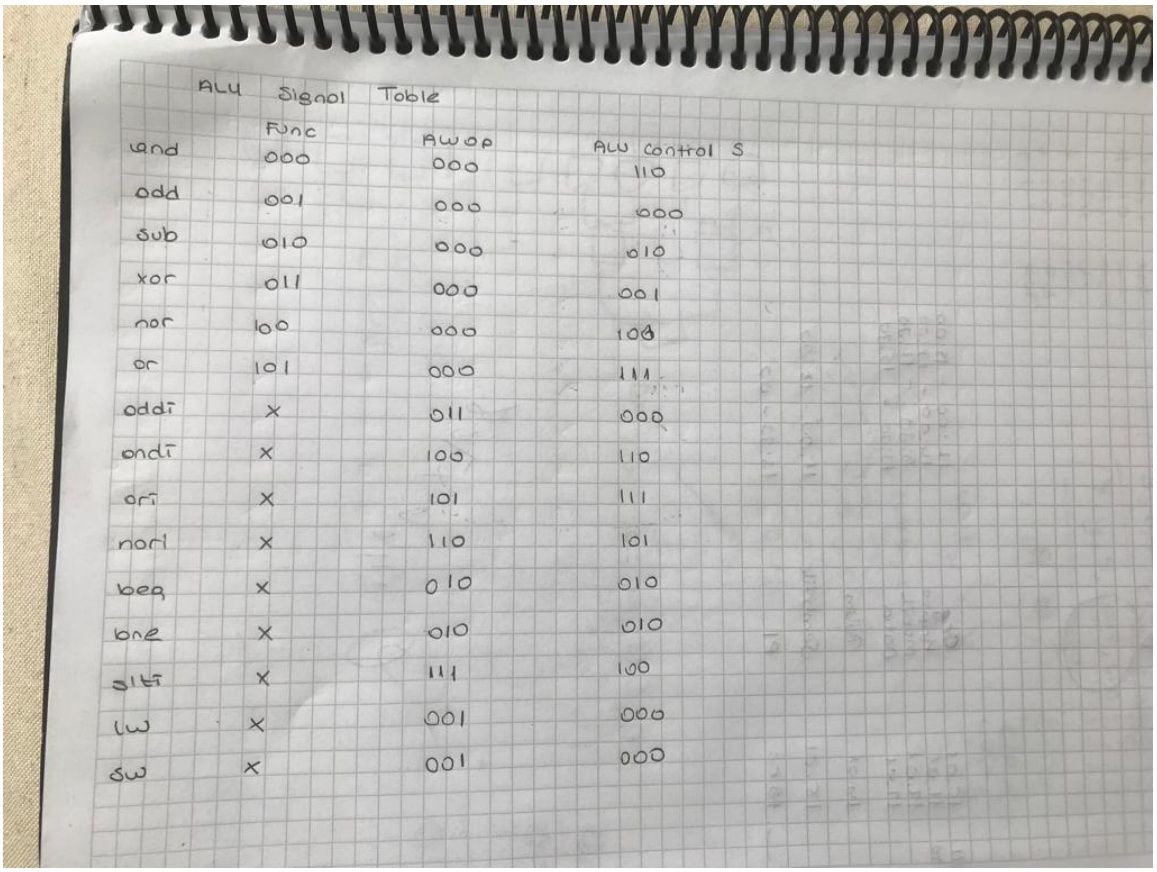
Confirmed that AluOP and Other signals are working correctly. Alu Control Module passed.

The Alu 32 we designed in the previous assignment was added to the project and the new instructions produced codes that the alu 32 could understand. that is, it produced 000 for addi so that it can use the add function of alu32. Because when 000 code is entered to alu32, it adds.

Lw-Sw inst add Operation, Beq -Bne inst. Sub Operation,Slti instr. Slt Opreration .

Instr	Opcode	Func
AND	0000	000
ADD	0000	001
SUB	0000	010
XOR	0000	011
NOR	0000	100
OR	0000	101
ADDI	0001	XXX
ANDI	0010	XXX
ORI	0011	XXX
NORI	0100	XXX
BEQ	0101	XXX
BNE	0110	XXX
SLTI	0111	XXX
LW	1000	XXX
SW	1001	XXX

ALUop	Operation
000	ADD
001	XOR
010	SUB
011	MULT
100	SLT
101	NOR
110	AND
111	OR



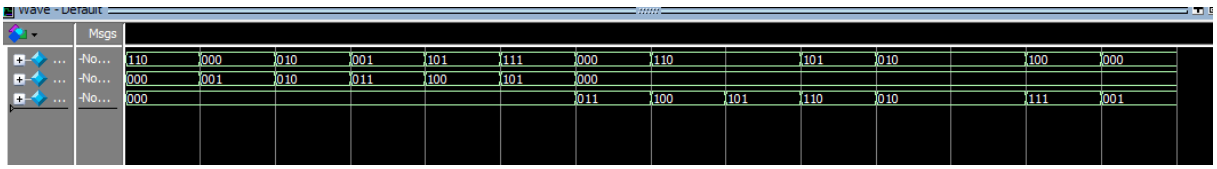

```
module alu_control(alu_control,opcode,funct);

input [2:0]opcode;
input[2:0] funct;
output[2:0] alu_control;
```

Here, the Opcode produced in the control unit is used as the opcode.

Testbench was written to make sure that Alu Control works correctly.

Output:

```
# time =  0, opcode =000, func_code=000, alu_control =110
# time = 10, opcode =000, func_code=001, alu_control =000
# time = 20, opcode =000, func_code=010, alu_control =010
# time = 30, opcode =000, func_code=011, alu_control =001
# time = 40, opcode =000, func_code=100, alu_control =101
# time = 50, opcode =000, func_code=101, alu_control =111
# time = 60, opcode =011, func_code=000, alu_control =000
# time = 70, opcode =100, func_code=000, alu_control =110
# time = 80, opcode =101, func_code=000, alu_control =110
# time = 90, opcode =110, func_code=000, alu_control =101
# time = 100, opcode =010, func_code=000, alu_control =010
# time = 120, opcode =111, func_code=000, alu_control =100
# time = 130, opcode =001, func_code=000, alu_control =000
```

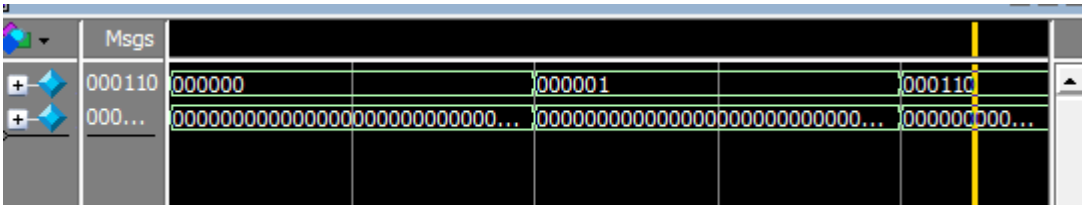


Verified Alu Control output with Aluop in Control unit signal table.

Sign Extend Module is started.And Testbench was written

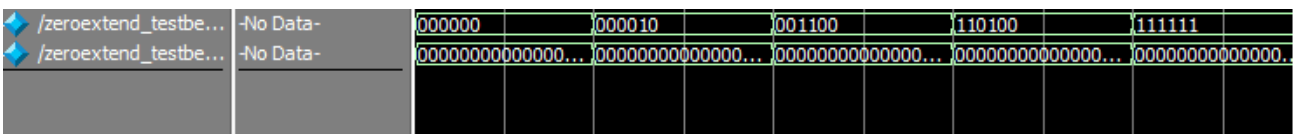
Output:

```
# time =  0, immediate =000000, Result=00000000000000000000000000000000
# time = 10, immediate =000010, Result=00000000000000000000000000000010
# time = 20, immediate =001100, Result=00000000000000000000000000001100
# time = 30, immediate =110100, Result=11111111111111111111111111110100
# time = 40, immediate =111111, Result=11111111111111111111111111111111
```



Zero Extend has been made.

```
# time =  0, immediate =000000, Result=00000000000000000000000000000000
# time = 10, immediate =000010, Result=00000000000000000000000000000010
# time = 20, immediate =001100, Result=00000000000000000000000000001100
# time = 30, immediate =110100, Result=00000000000000000000000000110100
# time = 40, immediate =111111, Result=00000000000000000000000000111111
```



I implement 3x1 mux. To Select write register rt or rd according to reg dest signal this mux has been added.

```
mux2x1_5 mux3(write_reg,rt,rd,sign_reg_dest);
```

Output:

```
# time =  0, a =010,b =100, s=0 , Result=010
# time = 10, a =101,b =001, s=1 , Result=001
```

I implemented a module that splits the instruction .

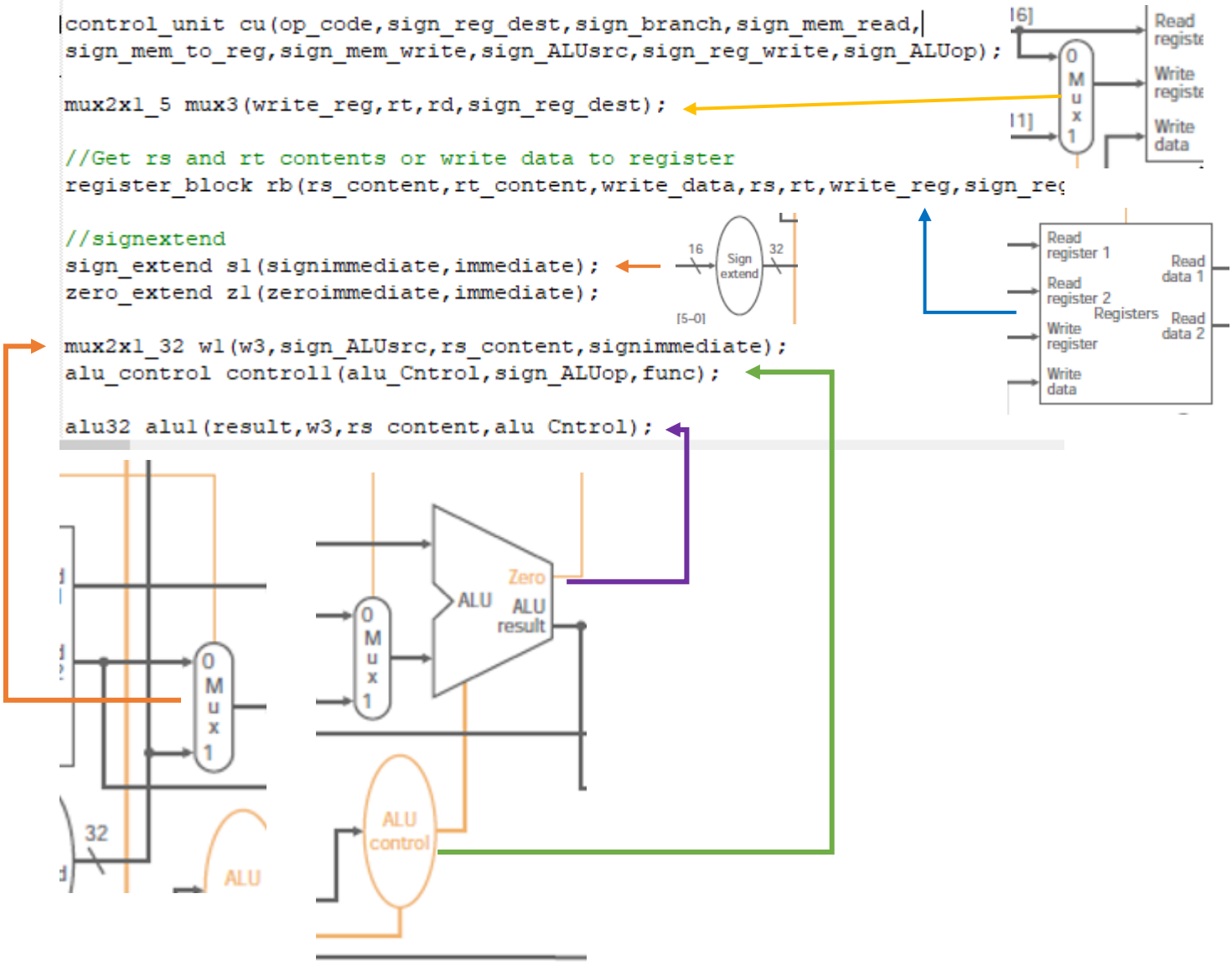
Output:

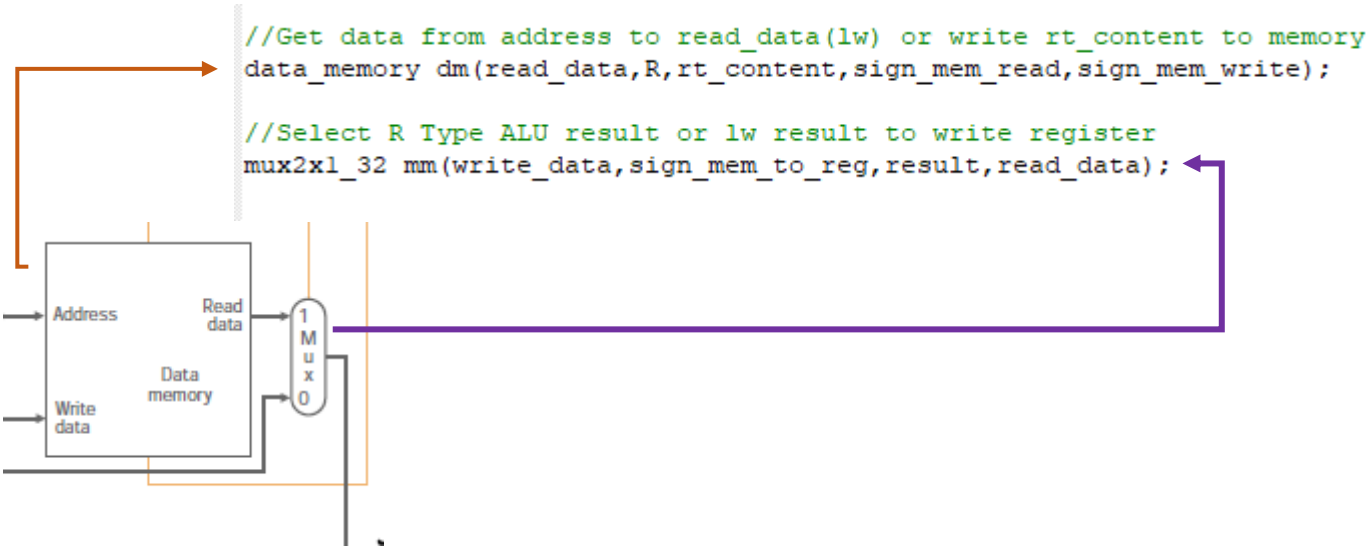
```
# time= 0, instruction=0101011001100101,  opcode=0101,  rs=011,  rt=001,  rd=  100,  function=101,  immediate=100101
# time=20, instruction=0101000001100101,  opcode=0101,  rs=000,  rt=001,  rd=  100,  function=101,  immediate=100101
# time=40, instruction=0001011001100101,  opcode=0001,  rs=011,  rt=001,  rd=  100,  function=101,  immediate=100101

VSIM 7>
```

Then I started writing my main module. First, the instruction is split. Then the control unit opcode was sent and the Signals were received. rd or rs selected according to reg_dest signal. and entered the Register blog.

According to my datapath, immediate should go to the sign extend module.

```
buf_instruction buf1(instruction,op_code,rs,rt,rd,func,immediate);

control_unit cu(op_code,sign_reg_dest,sign_branch,sign_mem_read,
sign_mem_to_reg,sign_mem_write,sign_ALUsrc,sign_reg_write,sign_ALUop);

mux2x1_5 mux3(write_reg,rt,rd,sign_reg_dest);

//Get rs and rt contents or write data to register
register_block rb(rs_content,rt_content,write_data,rs,rt,write_reg,sign_reg

//signextend
sign_extend s1(signimmediate,immediate);
zero_extend z1(zeroimmediate,immediate);

mux2x1_32 w1(w3,sign_ALUsrc,rs_content,signimmediate);
alu_control control1(alu_Cntrol,sign_ALUop,func);

alu32 alu1(result,w3,rs content,alu Cntrol);
```





I didn't get the correct output even though I ran the whole datapath step by step.

Each module works correctly individually, but does not work when combined.