CSE-331 Homework 4 Report

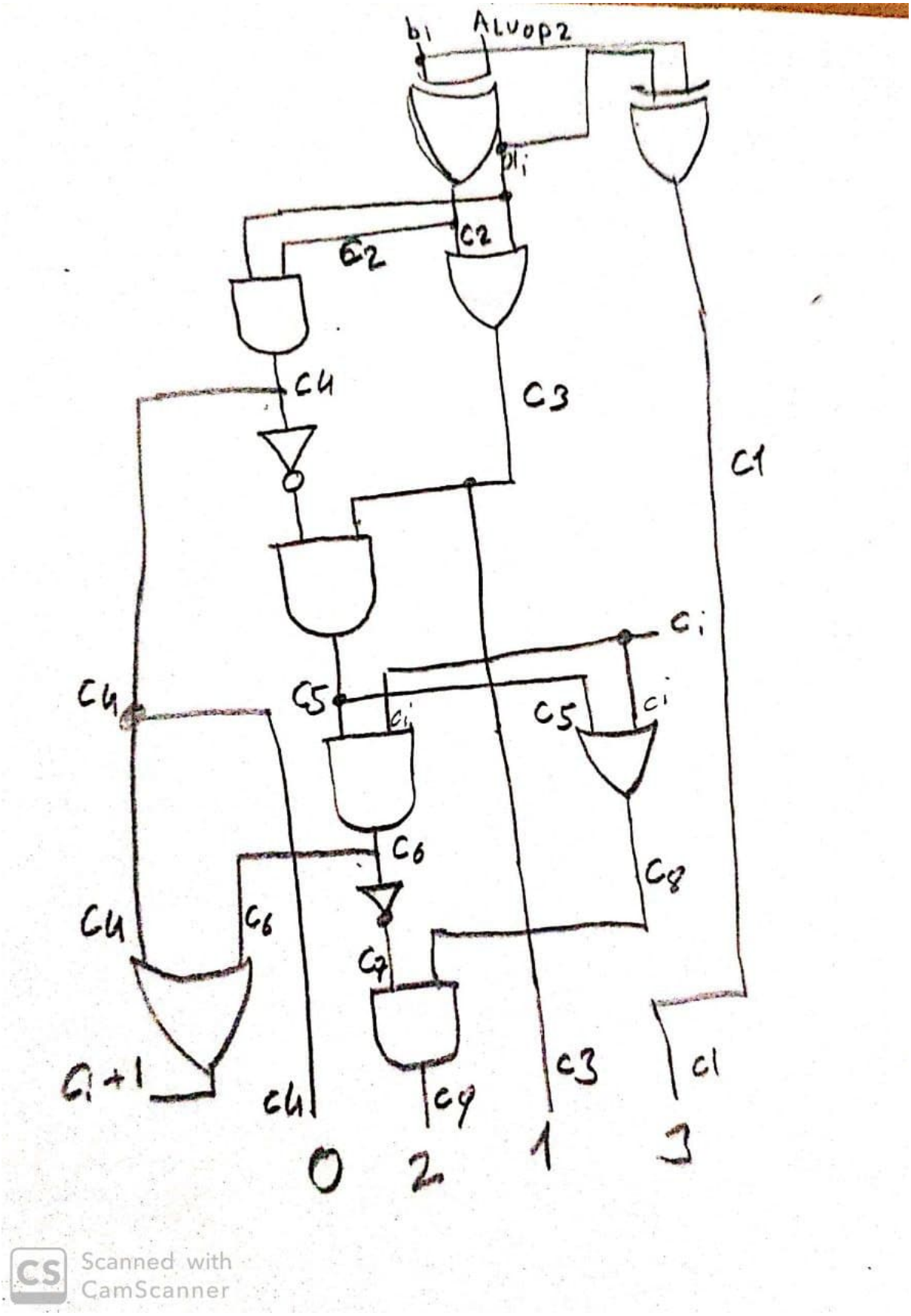
Computer Organization	HomeWork Lecture
Created	@Jan 20, 2021
Number	
Property	 Baran Hasan Bozduman

I have the project module by module but mips circuit does not work because I spend too much time to learn verilog and trying to solve errors on quartus. I made the design but I have not time to combine all modules.

- Instruction Memory
- Registers
- Alu Control
- Mux2s_1bit
- Shift left2
- Alu 1bit
- Mux4s_1bit
- Data Memory
- Control Unit
- Zero extender
- Mux2s_32bit
- Sign Extender
- Alu 32bit

Alu 1bit

I used xor gate instead of less.



Alu 1bit Test

```
sim:/testb_alulbit/result
VSIM 5> step -current
# (000And-001Or-010Add-110subs-111Xor) AluOp:000 a_i:0 b_i:0  result:0  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:000 a_i:0 b_i:1  result:0  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:000 a_i:1 b_i:0  result:0  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:000 a_i:1 b_i:1  result:1  c_o:1  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:001 a_i:0 b_i:0  result:0  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:001 a_i:0 b_i:1  result:1  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:001 a_i:1 b_i:0  result:1  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:001 a_i:1 b_i:1  result:1  c_o:1  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:111 a_i:0 b_i:0  result:0  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:111 a_i:0 b_i:1  result:1  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:111 a_i:1 b_i:0  result:1  c_o:1  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:111 a_i:1 b_i:1  result:0  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:010 a_i:0 b_i:0  result:0  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:010 a_i:0 b_i:1  result:1  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:010 a_i:1 b_i:0  result:1  c_o:0  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:010 a_i:1 b_i:1  result:0  c_o:1  c_i:0
# (000And-001Or-010Add-110subs-111Xor) AluOp:110 a_i:0 b_i:0  result:0  c_o:1  c_i:1
# (000And-001Or-010Add-110subs-111Xor) AluOp:110 a_i:0 b_i:1  result:1  c_o:0  c_i:1
# (000And-001Or-010Add-110subs-111Xor) AluOp:110 a_i:1 b_i:0  result:1  c_o:1  c_i:1
# (000And-001Or-010Add-110subs-111Xor) AluOp:110 a_i:1 b_i:1  result:0  c_o:1  c_i:1

VSIM 6>
```

Alu 32bit Test

```
VSIM 5> step -current
# a_i:00000000000000001111111111111111/ 65535  b_i:00000000111111110000000011111111/16711935  Result:00000000000000000000000011111111/   255  AluOp:000 Carry Out:0 Z:0 Less:0
# a_i:00000000000000001111111111111111/ 65535  b_i:00000000111111110000000011111111/16711935  Result:00000000111111111111111111111111/16777215  AluOp:001 Carry Out:0 Z:0 Less:0
# a_i:00000000000000000000000000000000/     0  b_i:00000000000000000000000000000000/     0  Result:00000000000000000000000000000000/     0  AluOp:010 Carry Out:0 Z:1 Less:0
# a_i:00000000000000000000000000000000/     0  b_i:00000000000000000000000000001101/    13  Result:00000000000000000000000000001101/    13  AluOp:010 Carry Out:0 Z:0 Less:0
# a_i:00000000000000000000000000001100/    12  b_i:00000000000000000000000000000000/     0  Result:00000000000000000000000000001100/    12  AluOp:010 Carry Out:0 Z:0 Less:0
# a_i:00000000000000000000000000001100/    12  b_i:00000000000000000000000000001101/    13  Result:00000000000000000000000000011001/    25  AluOp:010 Carry Out:0 Z:0 Less:0
# a_i:00000000000000000000000000000000/     0  b_i:00000000000000000000000000000000/     0  Result:00000000000000000000000000000000/     0  AluOp:110 Carry Out:1 Z:1 Less:0
# a_i:00000000000000000000000000000000/     0  b_i:00000000000000000000000000001101/    13  Result:11111111111111111111111111110011/4294967283  AluOp:110 Carry Out:0 Z:0 Less:1
# a_i:00000000000000000000000000001100/    12  b_i:00000000000000000000000000000000/     0  Result:00000000000000000000000000001100/    12  AluOp:110 Carry Out:1 Z:0 Less:0
# a_i:00000000000000000000000000001100/    12  b_i:00000000000000000000000000001101/    13  Result:11111111111111111111111111111111/4294967295  AluOp:110 Carry Out:0 Z:0 Less:1
# a_i:00000000000000001111111111111111/ 65535  b_i:00000000111111110000000011111111/16711935  Result:00000000111111111111111100000000/16776960  AluOp:111 Carry Out:0 Z:0 Less:0
```

Alu Control

```
VSIM 5> step -current
# FunctionField:000000  AluOp:00  AluSrc:010
# FunctionField:000000  AluOp:01  AluSrc:110
# FunctionField:100000  AluOp:10  AluSrc:010
# FunctionField:100010  AluOp:10  AluSrc:110
# FunctionField:100110  AluOp:10  AluSrc:111
# FunctionField:100100  AluOp:10  AluSrc:000
# FunctionField:100101  AluOp:10  AluSrc:001
# FunctionField:000000  AluOp:11  AluSrc:001
```

000 -> And
001 -> Or
010 -> Add
110 -> sub
111 -> xor

100100
100101
100000
100010
100111

Instruct	ALUop	operation	funct	Desired Alu	ALU control
Lw	000		✓	add	010
sw	000		✓	add	010
beq	001		✓	subs	110
bneq	011		✓	subs	110
addn	010		✓	add	010
subn	110		✓	sub	110
xorn	100		✓	xor	100
andn	110			and	000
orn	010			or	001
orl	111			or	001

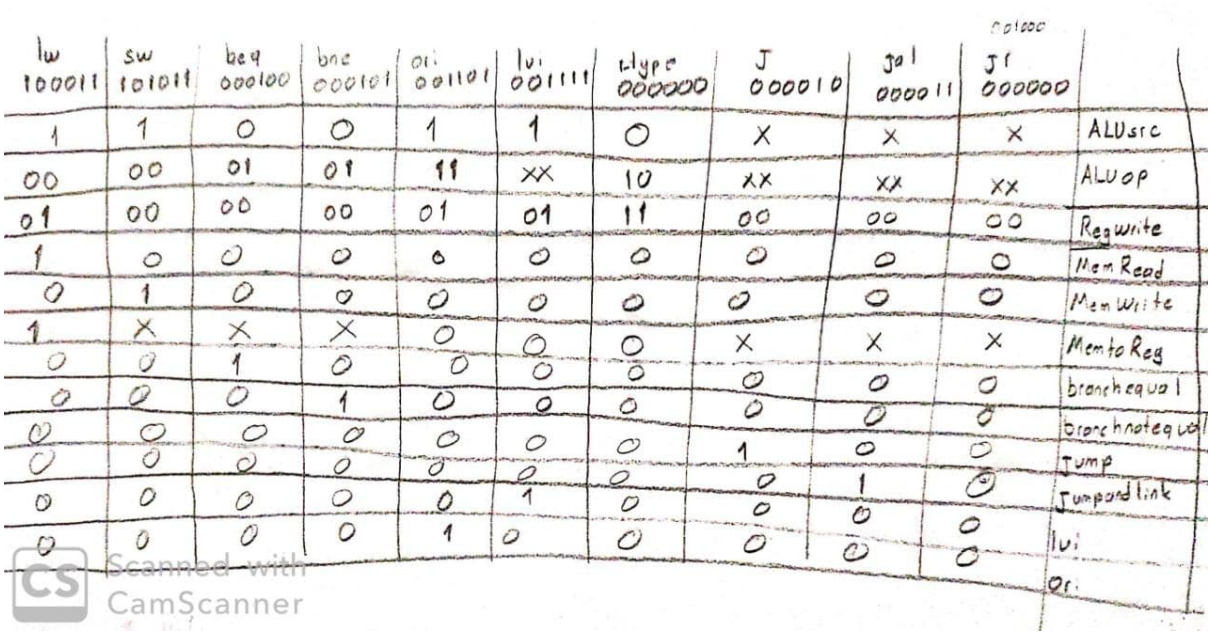
$$A2 = (A0 + (AP1 \& f1)) + !(AP1 \& AP0)$$

$$A1 = (!AP1 + !f2 + f1) + !(AP1 \& AP0)$$

$$A0 = (AP1 \& (f3 + f0)) + (f1 \& f2) + (AP1 \& AP0)$$

Control Unit

```
VSIM 5> step -current
# Opcode:100011 AluSrc:1, MemRead:1, MemWrite:0, MemtoReg:1, beq:0, bne:0, j:0, jal:0, lui:0,  ori:0 AluOp:00, RegWrite:01
# Opcode:101011 AluSrc:1, MemRead:0, MemWrite:1, MemtoReg:0, beq:0, bne:0, j:0, jal:0, lui:0,  ori:0 AluOp:00, RegWrite:00
# Opcode:000100 AluSrc:0, MemRead:0, MemWrite:0, MemtoReg:0, beq:1, bne:0, j:0, jal:0, lui:0,  ori:0 AluOp:01, RegWrite:00
# Opcode:000101 AluSrc:0, MemRead:0, MemWrite:0, MemtoReg:0, beq:0, bne:1, j:0, jal:0, lui:0,  ori:0 AluOp:01, RegWrite:00
# Opcode:001101 AluSrc:1, MemRead:0, MemWrite:0, MemtoReg:0, beq:0, bne:0, j:0, jal:0, lui:0,  ori:1 AluOp:11, RegWrite:01
# Opcode:001111 AluSrc:1, MemRead:0, MemWrite:0, MemtoReg:0, beq:0, bne:0, j:0, jal:0, lui:1,  ori:0 AluOp:00, RegWrite:01
# Opcode:000000 AluSrc:0, MemRead:0, MemWrite:0, MemtoReg:0, beq:0, bne:0, j:0, jal:0, lui:0,  ori:0 AluOp:10, RegWrite:11
# Opcode:000010 AluSrc:0, MemRead:0, MemWrite:0, MemtoReg:0, beq:0, bne:0, j:1, jal:0, lui:0,  ori:0 AluOp:00, RegWrite:00
# Opcode:000011 AluSrc:0, MemRead:0, MemWrite:0, MemtoReg:0, beq:0, bne:0, j:0, jal:1, lui:0,  ori:0 AluOp:00, RegWrite:00
```



Alu src: to determine i type or r type instruction

Alu Op: to determine which type alu operation will be

Reg Write: Writing register msb bit for the register2 other one for the register1

MemRead: to enable memory read

MemWrite: writing data memory

MemtoReg: determining memory or alu result

Mux2s 1bit

```
# m:0  i1:0  i2:1  out:0
# m:1  i1:0  i2:1  out:1
# m:0  i1:1  i2:0  out:1
# m:1  i1:1  i2:0  out:0
```

mux4s 1bit

```
VSIM 6> step -current
# input: 0001  mux: 00  output: 1
# input: 0010  mux: 01  output: 1
# input: 0100  mux: 10  output: 1
# input: 1000  mux: 11  output: 1
```

mux2s 32bit

```
sim:/testb_mux2s_32bit/m
VSIM 5> step -current
# m:0 i1:11111111111111111111111111111111  i2:00000000000000000000000000000000 out:11111111111111111111111111111111
# m:1 i1:11111111111111111111111111111111  i2:00000000000000000000000000000000 out:00000000000000000000000000000000
# m:0 i1:11111111111111111111111111111111  i2:00000000000000000000000000000000 out:11111111111111111111111111111111
```

Shift left2

```
VSIM 5> step -current
# Inputt::11111111111111111111111111111111 Outputt::11111111111111111111111111111100
# Inputt::11111111111111111111111111110011 Outputt::11111111111111111111111111001100
# Inputt::00111111111111111111111111111111 Outputt::11111111111111111111111111111100
# Inputt::00011111111111111111111111110011 Outputt::01111111111111111111111111001100
```

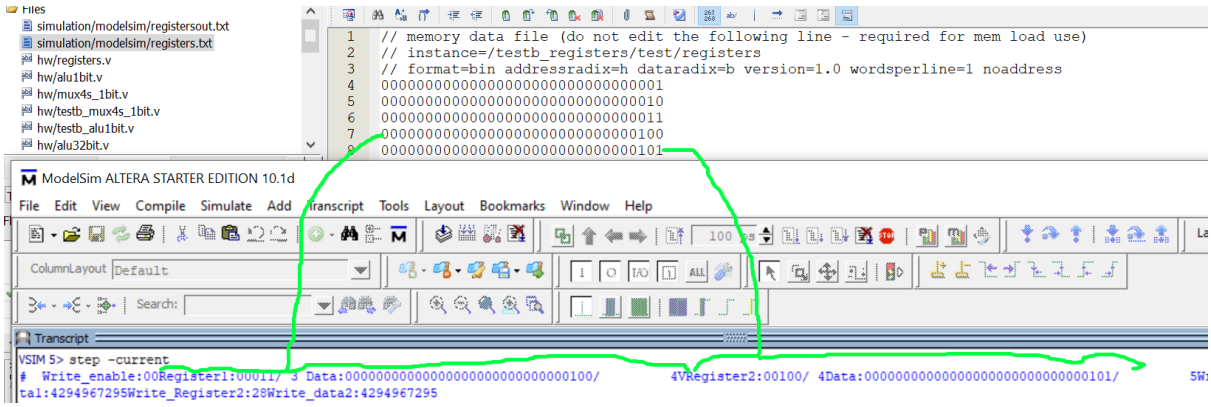
sign Extended

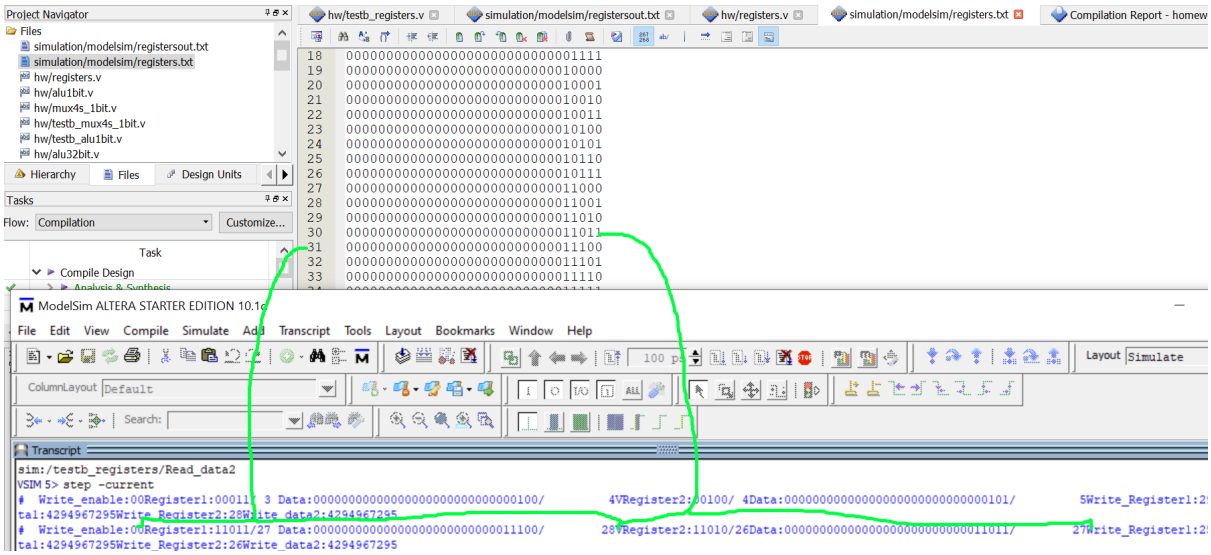
```
VSIM 4> step -current
# input:1111111111111111  output:11111111111111111111111111111111
# input:0111111111111111  output:00000000000000001111111111111111
# input:0000000000000001  output:00000000000000000000000000000001
# input:1000000000000001  output:11111111111111111000000000000001
```

Zero Extend

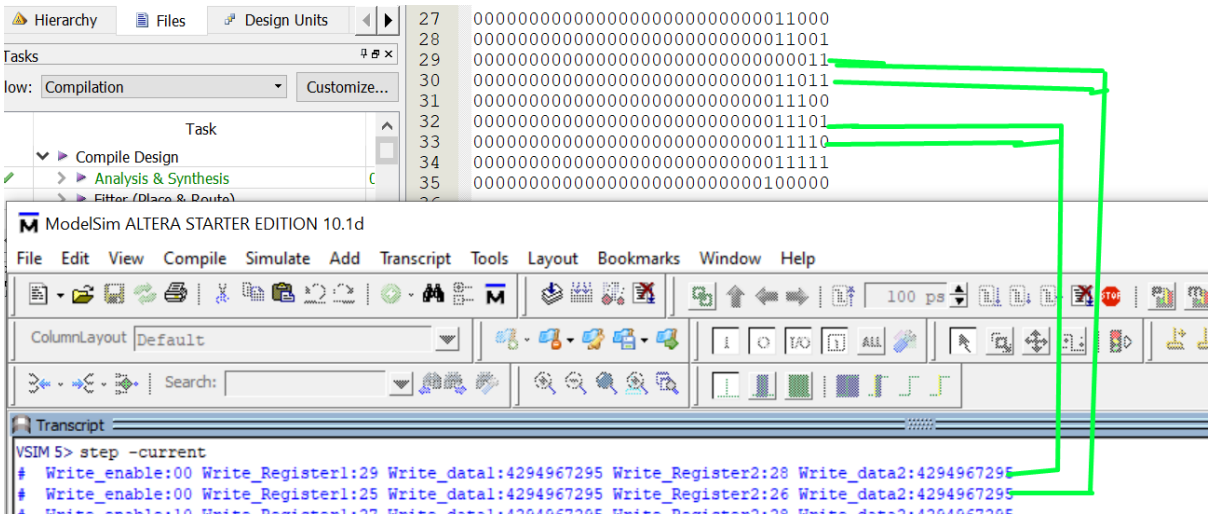
```
sim:/testb_zeroExt/o_value
VSIM 5> step -current
# input::1111111111111111  output::11111111111111110000000000000000
# input::0111111111111111  output::01111111111111110000000000000000
```

Register Read



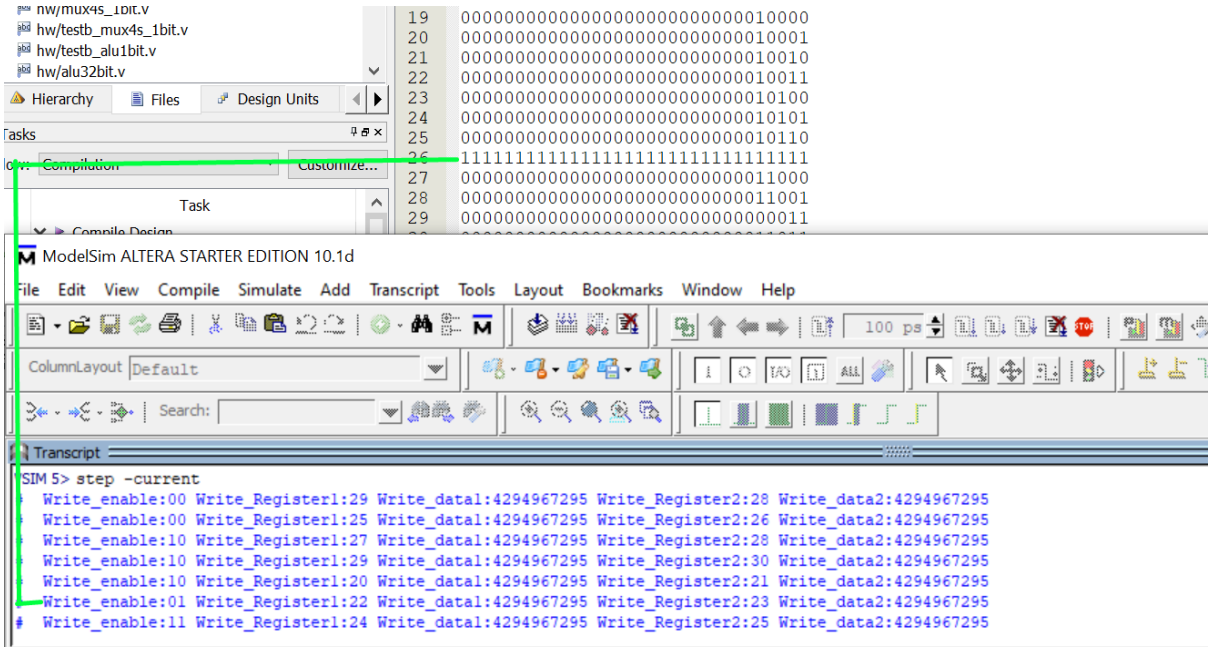


Register not write

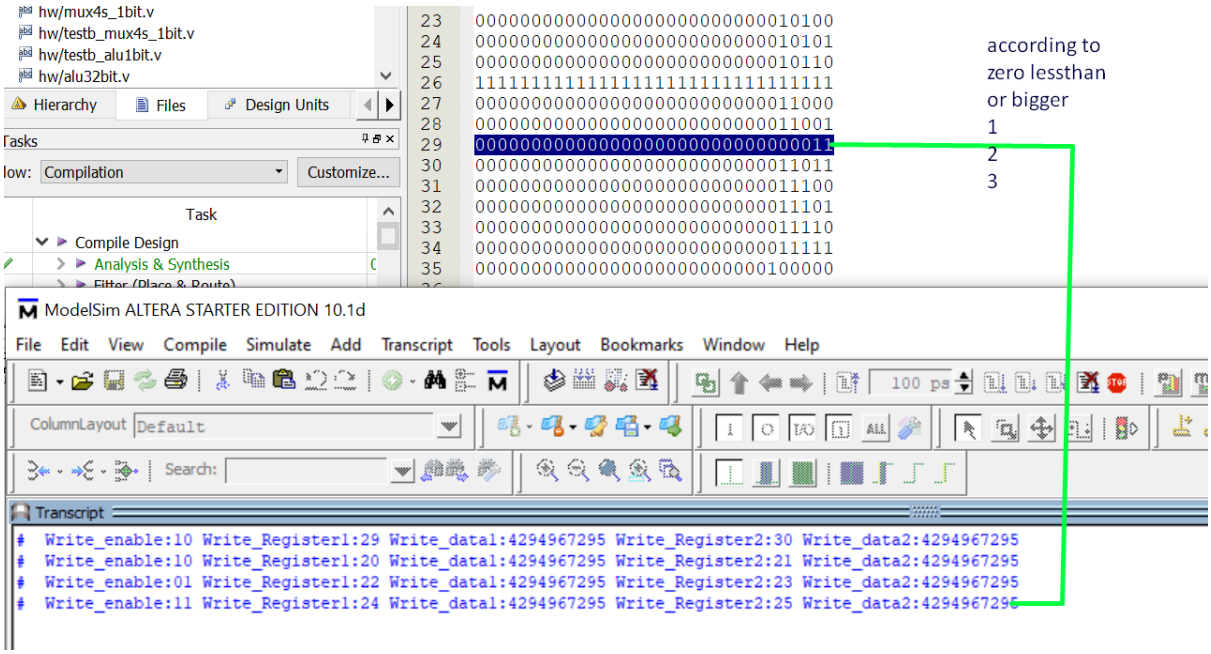


Register write

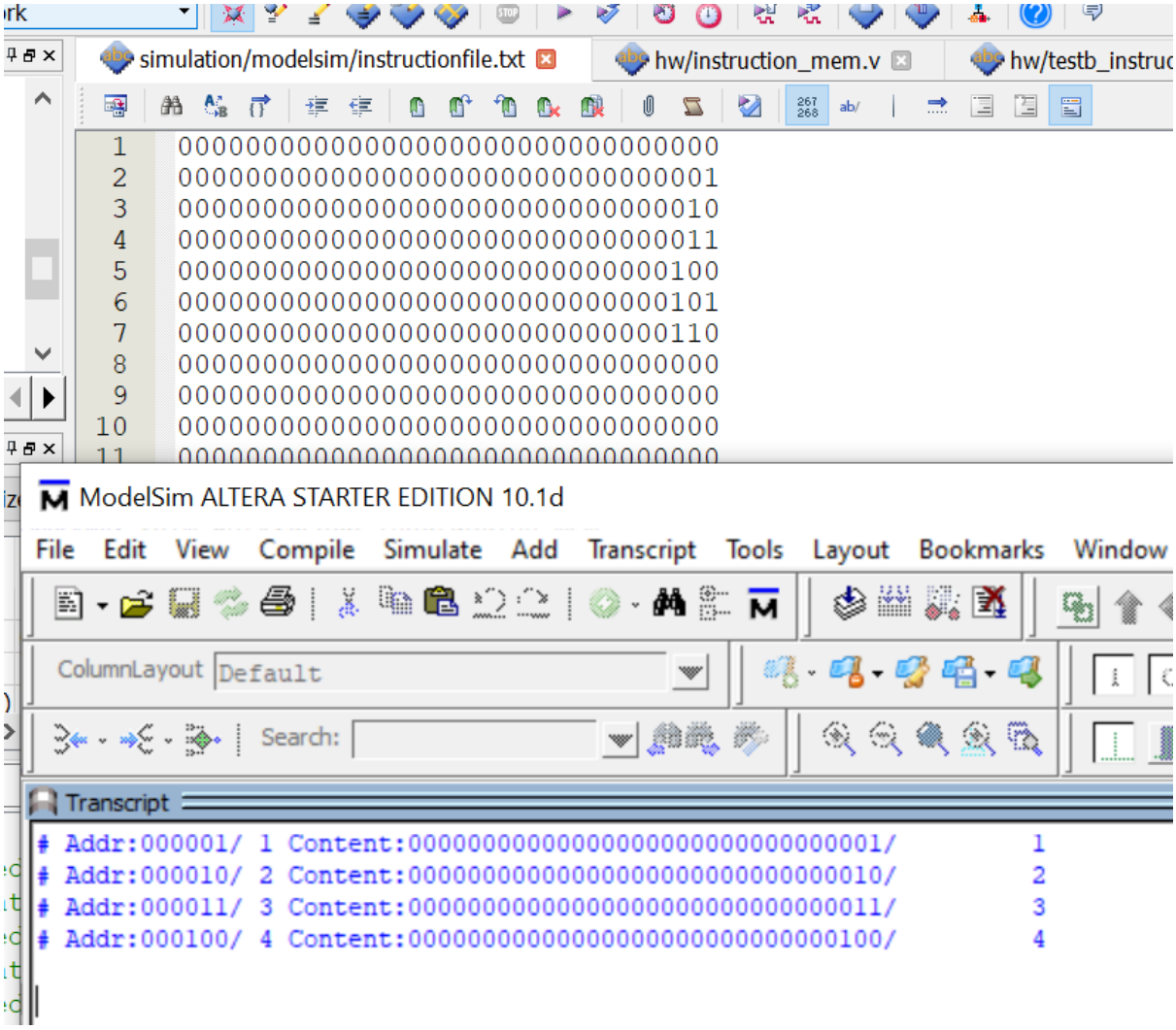
You can check my register.v file i also check the zero and less situation to make my job easy so i write to register according to that signals



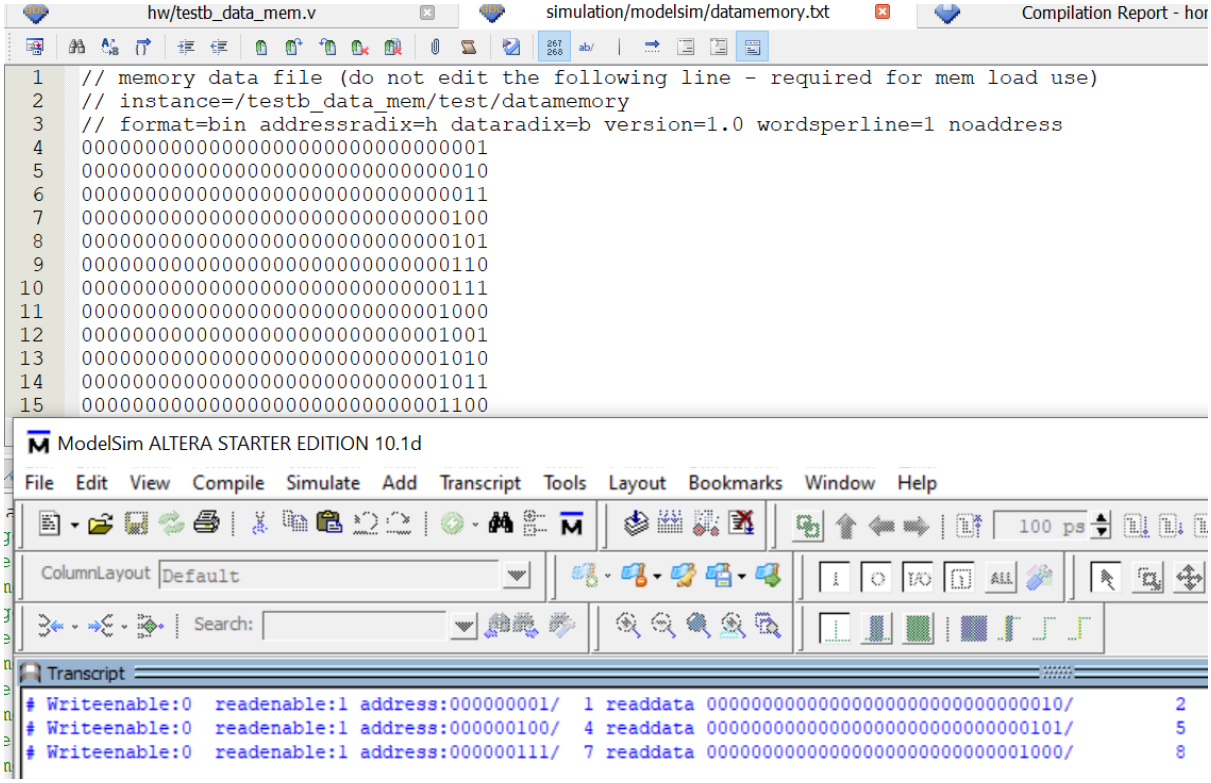
Register Write



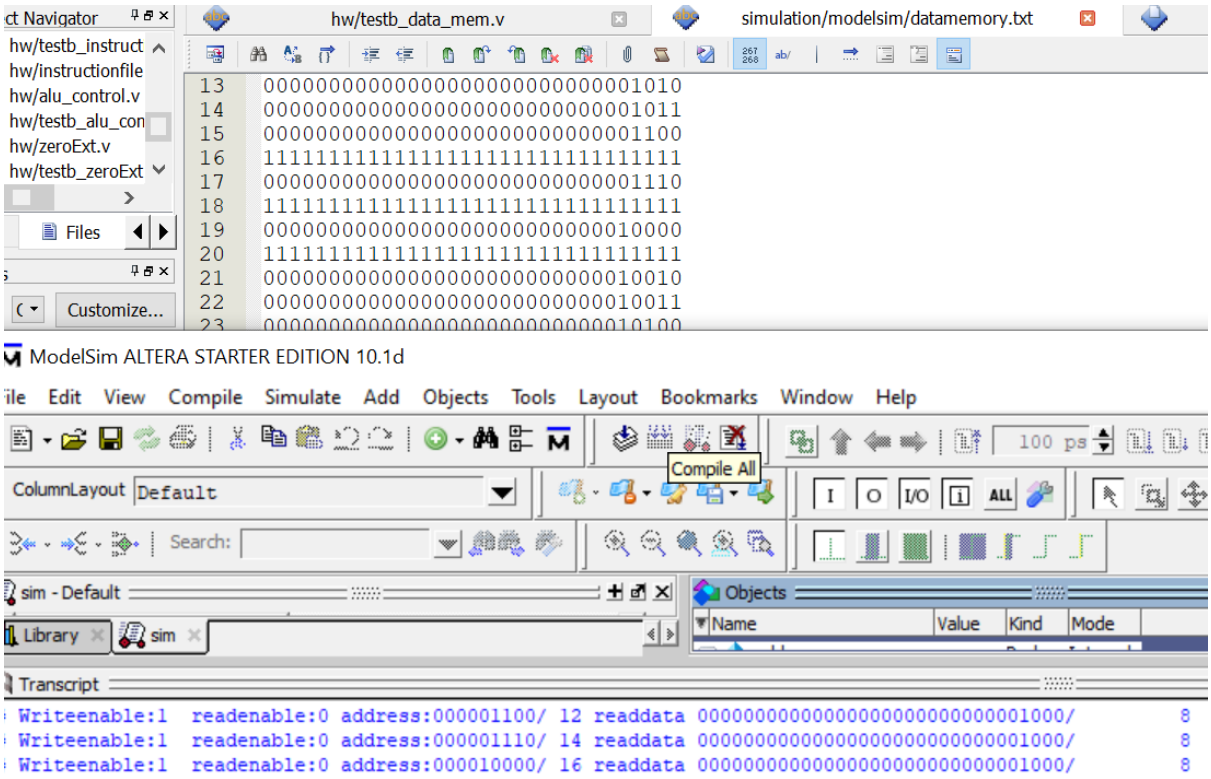
Instruction Memory



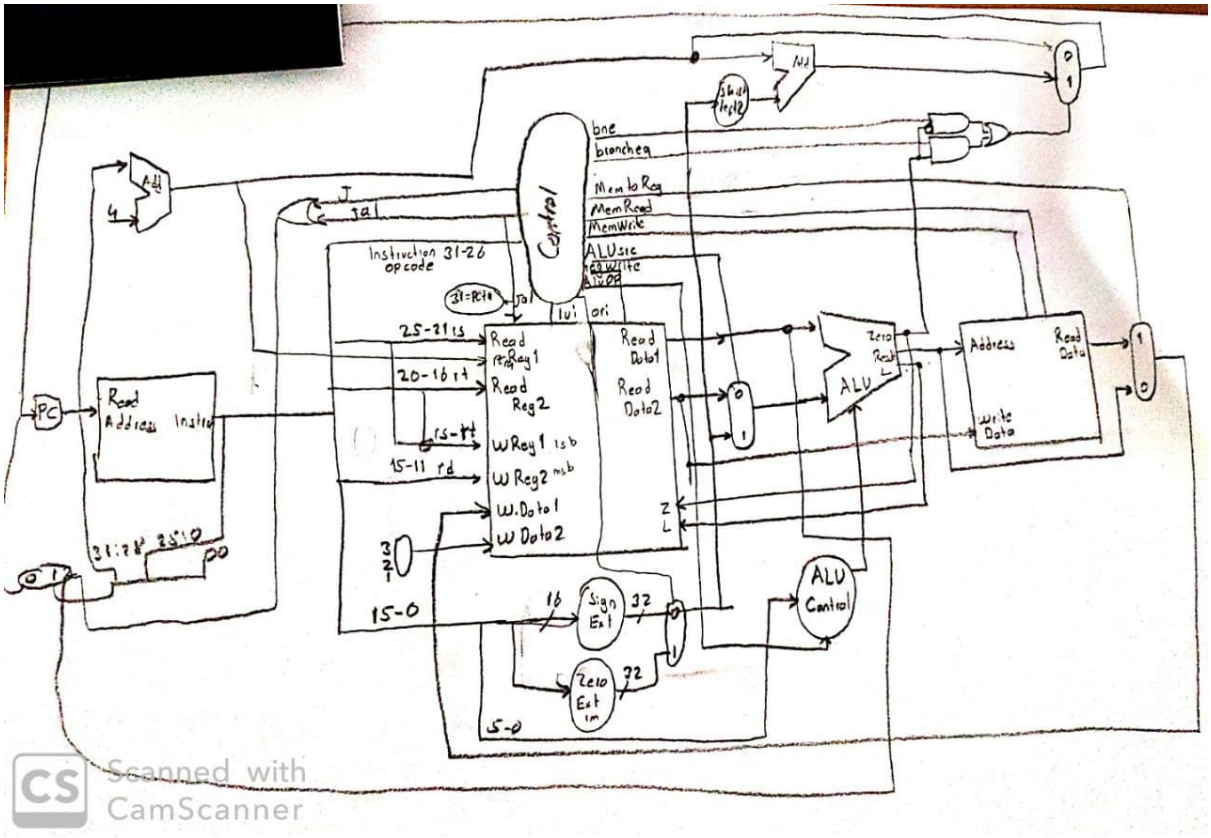
Read Data Memory



Write Data memory



General Design


Add
4
J
jal
Instruction 31-26
opcode
Control
bne
branchneq
MemtoReg
MemRead
MemWrite
ALUsrc
RegWrite
ALUop
31=R-t
Jal
lui
ori
PC
Read
Address
Instruct
31:28
25:0
00
25-21 rs
20-16 rt
15-11 rd
rs-rt
Read
reg Reg 1
Read
Reg 2
W Reg 1 lsb
W Reg 2 msb
W. Data 1
W Data 2
Read
Data 1
Read
Data 2
Z
L
3
2
1
15-0
16
Sign
Ext
32
Zero
Ext
im
32
5-0
ALU
Zero
Result
ALU
Address
Read
Data
Write
Data
ALU
Control
Scanned with
CamScanner