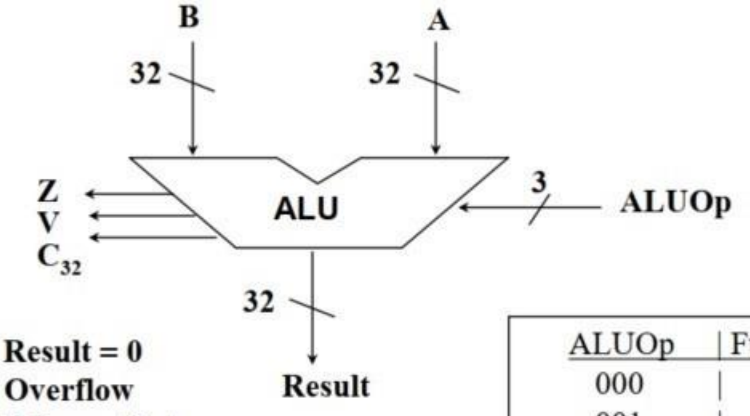
CSE 331
COMPUTER ORGANIZATION

HOMEWORK 4

ŞEYDA ÖZER
171044023

ALU

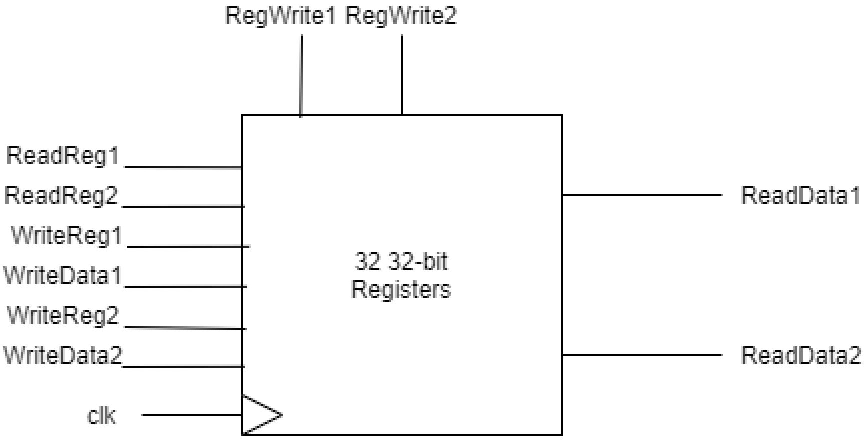


$Z = 1$, if Result $= 0$
$V = 1$, if Overflow
$C_{32} = 1$, if Carry-Out

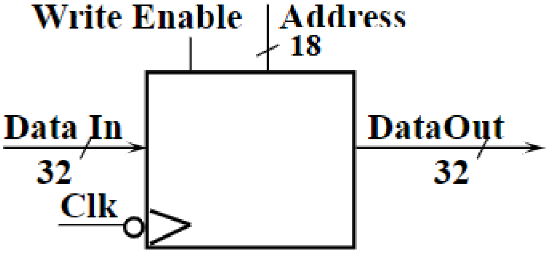
ALUOp	Function
000	AND
001	OR
010	ADD
110	SUBTRACT
111	XOR

I put the xor operation instead of set-less-than operator.

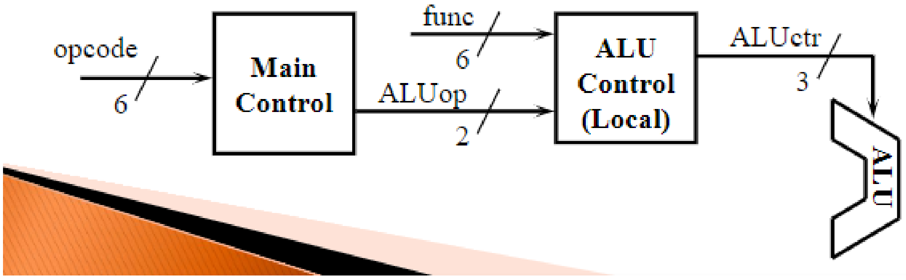
Register File



Memory



Control



Instruction	Reg Dst	Mem Read	Mem to Reg	ALU Op	Mem Write	ALU Src	Reg write 1	Reg write 2	Jr	J	Jal	Branch	Branch Not	Ori	Lui
R-type 000000	1	0	0	11	0	0	1	1	0	0	0	0	0	0	0
lw 100011	0	1	1	00	0	1	1	0	0	0	0	0	0	0	0
sw 101011	X	0	X	00	1	1	0	0	0	0	0	0	0	0	0
beq 000100	X	0	X	01	0	0	0	0	0	0	0	1	0	0	0
bne 000101	X	0	X	01	0	0	0	0	0	0	0	0	1	0	0
j 000010	X	0	X	X	0	X	0	0	0	1	0	0	0	0	0
jal 000010	X	0	X	X	0	X	1	0	0	0	1	0	0	0	0
jr 001000	X	0	X	X	0	X	0	0	1	0	0	0	0	0	0
ori 001101	0	0	0	10	0	1	1	0	0	0	0	0	0	1	0
lui 001111	0	0	X	X	0	X	1	0	0	0	0	0	0	0	1

Instruction	*R-type*	*lw*	*sw*	*beq*	*bne*	*j*	*jal*	*jr*	*ori*	*lui*
Opcode	000000	100011	101011	000100	000101	000010	000010	001000	001101	001111

ALUOp1 = R-type + ori

ALUOp0 = R-type + beq + bne

RegDst = R-type

MemRead = lw

MemtoReg = lw

MemWrite = sw

ALUSrc = lw + sw + ori

RegWrite1 = R-type + lw + jal + ori + lui

RegWrite2 = R-type

Jr = jr

J = j

Jal = jal

Branch = beq

BranchNot = bne

Ori = ori

Lui = lui

ALU Control

Instruction opcode	ALUOp	Instruction Operation	Function Field	Desired ALU action	ALU control
lw	00	Load word	XXXXXX	add	010
sw	00	Store word	XXXXXX	add	010
beq	01	Branch equal	XXXXXX	subtract	110
bne	01	Branch not equal	XXXXXX	subtract	110
ori	10	Or immediate	XXXXXX	or	001
R-type	11	addn	100000	add	010
R-type	11	subtractn	100010	subtract	110
R-type	11	ANDn	100100	and	000
R-type	11	ORn	100101	or	001
R-type	11	XORn	100110	xor	111

Truth Table for ALU Control

ALUop		**Function field**						**Operation**		
ALUop1	ALUop0	F5	F4	F3	F2	F1	F0	ALUctr2	ALUctr1	ALUctr0
0	0	X	X	X	X	X	X	0	1	0
0	1	X	X	X	X	X	X	1	1	0
1	0	X	X	X	X	X	X	0	0	1
1	1	1	0	0	0	0	0	0	1	0
1	1	1	0	0	0	1	0	1	1	0
1	1	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	1	0	0	1
1	1	1	0	0	1	1	0	1	1	1

line1 = F5.F4’.F3’.F2’.F1’.F0’ , line2 = F5.F4’.F3’.F2’.F1.F0’

line4 = F5.F4’.F3’.F2.F1’.F0 , line5 = F5.F4’.F3’F2.F1.F0’

ALUctr2 = ALUop1’.ALUop0 + ALUop1. ALUop0.line2 + ALUop1. ALUop0.line5

ALUctr1 = ALUop1’. ALUop0’ + ALUop1’. ALUop0 + ALUop1. ALUop0.line1 + ALUop1. ALUop0.line2 + ALUop1. ALUop0.line5

ALUctr0 = ALUop1. ALUop0’ + ALUop1. ALUop0.line4 + ALUop1. ALUop0.line5

About Homework:

- InstructionMemory
- Registers
- DataMemory
- ALU (oneBitALU, oneBitMsbALU)
- Control

These modules are working. But ALUControl module is not working correctly. ALUctr is produced incorrectly.

I could not run the Mips32 module. so I did not change the clock and increment the counter. I couldn't test the whole assignment, but the modules I mentioned above have testbenches.

Test Instructions:

1. Addn
   000000_10000_10001_01000_00000_100000
   rs=$16, rt=$17, rd=$8, function field = 20 (hex)
2. Addn
   000000_10001_10010_01001_00000_100000
   rs=$17, rt=$18, rd=$9, function field = 20 (hex)
3. subbn
   000000_10000_10001_01010_00000_100010
   rs=$16, rt=$17, rd=$10, function field = 22 (hex)
4. subbn
   000000_10001_10010_01011_00000_100010
   rs=$17, rt=$18, rd=$11, function field = 22 (hex)
5. andn
   000000_10000_10001_01100_00000_100100
   rs=$16, rt=$17, rd=$12, function field = 24 (hex)
6. andn
   000000_10001_10010_01101_00000_100100
   rs=$17, rt=$18, rd=$13, function field = 24 (hex)
7. orn
   000000_10000_10001_01110_00000_100101
   rs=$16, rt=$17, rd=$14, function field = 25 (hex)
8. orn
   000000_10001_10010_01111_00000_100101
   rs=$17, rt=$18, rd=$15, function field = 25 (hex)
9. xorn
   000000_10000_10001_11000_00000_100110
   rs=$16, rt=$17, rd=$24, function field = 26 (hex)

10. xorn
    000000_10001_10010_11001_00000_100110
    rs=$17, rt=$18, rd=$25, function field = 26 (hex)
11. Ori
    001101_10000_10111_0000000001011110
    rs=$16, rt=$23
12. Ori
    001101_10001_10110_0000000001011110
    rs=$17 , rt=$22
13. Lui
    001111_00000_10101_0000000011100110
    rs=X , rt=$21
14. Lui
    001111_00000_10100_0000000011100110
    rs=X , rt=$20