CSE 331 Computer Organization
Homework 2
Hamza Yoğurtcuoğlu - 171044086
10 October , 2018

module four_to_one_mux(mux_output,a,b,c,d,selector_one,selector_two)
This mux is needed for Opcode0.1 that needs each 32 bits.There are four input and 2 selector inputs and 1 output.

module two_to_one_mux (mux_output,a,b,selector)
This module is used Opcode2 and rightshift and leftshift.There are 2 input and 1 selector input and 1 output.

module alu32 (R ,S, A , B ,zerocontrol,overflow,
zerocontrolSub,overflowSub,zerocontrolRight
,zerocontrolLeft,zerocontrolNor,
zerocontrolAnd,zerocontrolOr,zerocontrolXor)
All other models are called in here . There are 64 mux (4:1) for opcode1,0 and 32 mux (2:1) for opcode2 . Because we create a result according of selector opcodes.

module and_A_B (and_A_B, A , B,zerocontrolAnd)
32 bit A and 32 bit B are operating with and gates.

module or_A_B (or_A_B, A , B,zerocontrolOr)
32 bit A and 32 bit B are operating with or gates.

module full_adder(sum,cin_plus_one,cin,a,b)
This module is just for 1 bit add. Im using for sub ,add instractions .

module add_A_B(add_R,A,B,zerocontrol,overflow)
Each carry out bit and A and B bit are added by full_adder module. We can check overflow and Implementation of zero.

module xor_A_B(xor_R , A , B,zerocontrolXor)
32 bit A and 32 bit B are operating with xor gates.

module sub_A_B(sub_R,A,B,zerocontrolSub,overflowSub)
Firstly , We are getting two complement’s of B number . Then Each carry out bit and A and B bit are added by full_adder module. We can check overflow and Implementation of zero.

module rightAr_A_B(rightAr_R,A,B,zerocontrolRight)
At first just 1 bit shift to right with through 2:1 mux . But we need to put most significant bit digit of the number to empty left field .Then 2 bit ,4bit,8bit,16bit then finish checking.Because if there is a one bit B[5]-[31]. All result is 000.....000

module leftLo_A_B(leftLo_R,A,B,zerocontrolLeft)
Firstly , just 1 bit shift to left with through 2:1 mux . But we need to put ‘0’ to empty right field .Then 2 bit ,4bit,8bit,16bit then finish checking.Because if there is a one bit B[5]-[31]. All result is 000.....000

module nor_A_B(nor_R,A,B,zerocontrolNor)
32 bit A and 32 bit B are operating with nor gates.

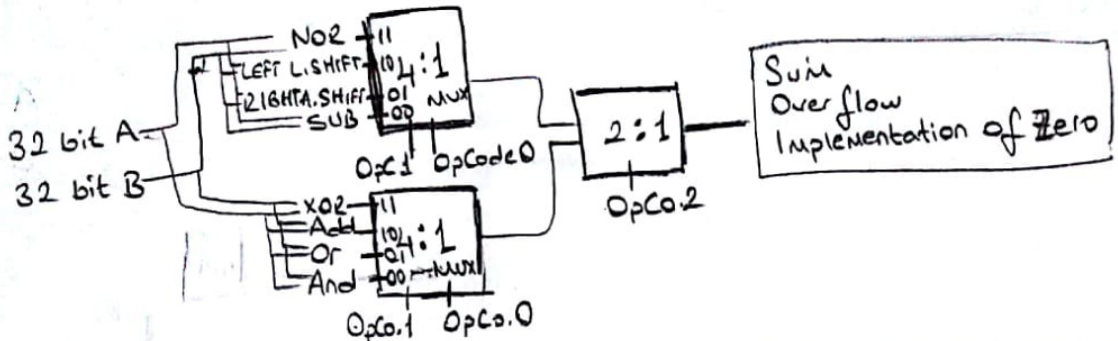
NOTE : ALL OVERFLOW AND IMPLEMENTATION OF ZERO IS CHECKED.

```
A = 11111100000000000100000010001001 B = 00000000000000000000000000001111 Result = 00000000000000000000000000001001 Select : 000
ZeroControl:0,Overflow:0,Carry_Out:0
A = 11111100000000000100000010001001 B = 00000000000000000000000000001111 Result = 11111100000000000100000010001111 Select : 001
ZeroControl:0,Overflow:0,Carry_Out:0
A = 11111100000000000100000010001001 B = 00000000000000000000000000001111 Result = 11111100000000000100000010011000 Select : 010
ZeroControl:0,Overflow:0,Carry_Out:0
A = 11111100000000000100000010001001 B = 00000000000000000000000000001111 Result = 11111100000000000100000010000110 Select : 011
ZeroControl:0,Overflow:0,Carry_Out:0
A = 11111100000000000100000010001001 B = 00000000000000000000000000001111 Result = 11111100000000000100000001111010 Select : 100
ZeroControl:0,Overflow:0,Carry_Out:1
A = 11111100000000000100000010001001 B = 00000000000000000000000000001111 Result = 11111111111111111111100000000000 Select : 101
ZeroControl:0,Overflow:0,Carry_Out:1
A = 11111100000000000100000010001001 B = 00000000000000000000000000001111 Result = 00100000010001001000000000000000 Select : 110
ZeroControl:0,Overflow:0,Carry_Out:1
A = 11111100000000000100000010001001 B = 00000000000000000000000000001111 Result = 00000011111111111011111101110000 Select : 111
ZeroControl:0,Overflow:0,Carry_Out:1

A = 11111100000000100000111111101001 B = 10000000100100001000111111010011 Result = 10000000000000000000111111000001 Select : 000
ZeroControl:0,Overflow:1,Carry_Out:1
A = 11111100000000100000111111101001 B = 10000000100100001000111111010011 Result = 11111100100100101000111111111011 Select : 001
ZeroControl:0,Overflow:1,Carry_Out:1
A = 11111100000000100000111111101001 B = 10000000100100001000111111010011 Result = 01111100100100101001111110111100 Select : 010
ZeroControl:0,Overflow:1,Carry_Out:1
A = 11111100000000100000111111101001 B = 10000000100100001000111111010011 Result = 01111100100100101000000000111010 Select : 011
ZeroControl:0,Overflow:1,Carry_Out:1
A = 11111100000000100000111111101001 B = 10000000100100001000111111010011 Result = 01111101101110001100000000010110 Select : 100
ZeroControl:0,Overflow:0,Carry_Out:1
A = 11111100000000100000111111101001 B = 10000000100100001000111111010011 Result = 00000000000000000000000000000000 Select : 101
ZeroControl:1,Overflow:0,Carry_Out:1
A = 11111100000000100000111111101001 B = 10000000100100001000111111010011 Result = 00000000000000000000000000000000 Select : 110
ZeroControl:1,Overflow:0,Carry_Out:1
A = 11111100000000100000111111101001 B = 10000000100100001000111111010011 Result = 00000011011011010111000000000100 Select : 111
ZeroControl:0,Overflow:0,Carry_Out:1

A = 00000100001111111111111111111111 B = 00000100001111111111111111111111 Result = 00000100001111111111111111111111 Select : 000
ZeroControl:0,Overflow:0,Carry_Out:0
A = 00000100001111111111111111111111 B = 00000100001111111111111111111111 Result = 00000100001111111111111111111111 Select : 001
ZeroControl:0,Overflow:0,Carry_Out:0
A = 00000100001111111111111111111111 B = 00000100001111111111111111111111 Result = 00000100001111111111111111111110 Select : 010
ZeroControl:0,Overflow:0,Carry_Out:0
A = 00000100001111111111111111111111 B = 00000100001111111111111111111111 Result = 00000000000000000000000000000000 Select : 011
ZeroControl:1,Overflow:0,Carry_Out:0
A = 00000100001111111111111111111111 B = 00000100001111111111111111111111 Result = 00000000000000000000000000000000 Select : 100
ZeroControl:1,Overflow:0,Carry_Out:1
A = 00000100001111111111111111111111 B = 00000100001111111111111111111111 Result = 00000000000000000000000000000000 Select : 101
ZeroControl:1,Overflow:0,Carry_Out:1
A = 00000100001111111111111111111111 B = 00000100001111111111111111111111 Result = 00000000000000000000000000000000 Select : 110
ZeroControl:1,Overflow:0,Carry_Out:1
A = 00000100001111111111111111111111 B = 00000100001111111111111111111111 Result = 11111011110000000000000000000000 Select : 111
ZeroControl:0,Overflow:0,Carry_Out:1
```

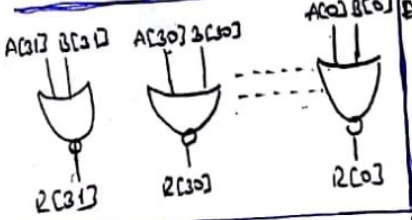
ALU-32 BIT

Hamza Yoğurtcuoğlu
171044086

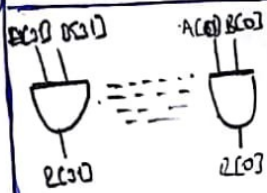
32 Bit ALU MODULE (ALU32)



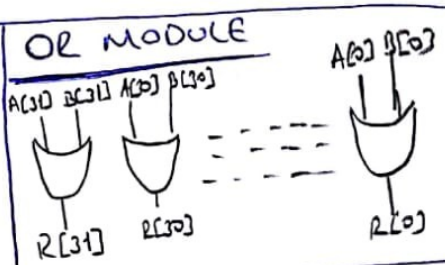
NOR OPERATION (module)



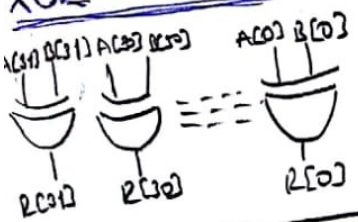
AND MODULE



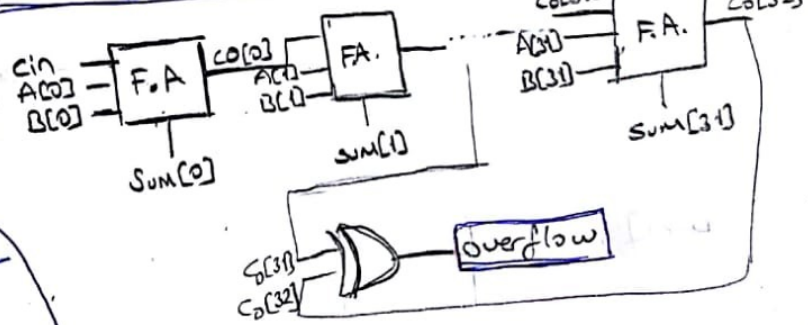
OR MODULE



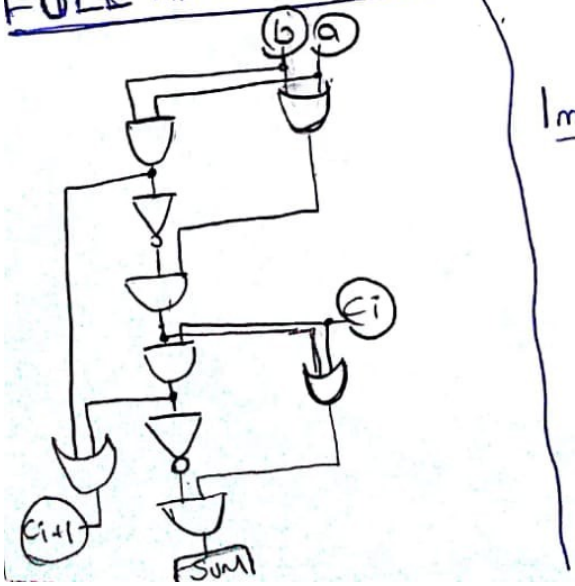
XOR MODULE



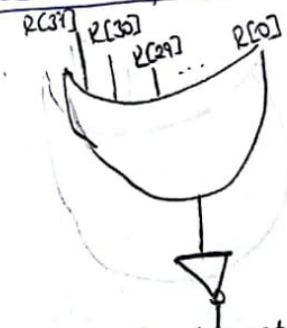
ADD NODULE



FULL ADDER NODULE



Implementation of Zero



-> OR GATE

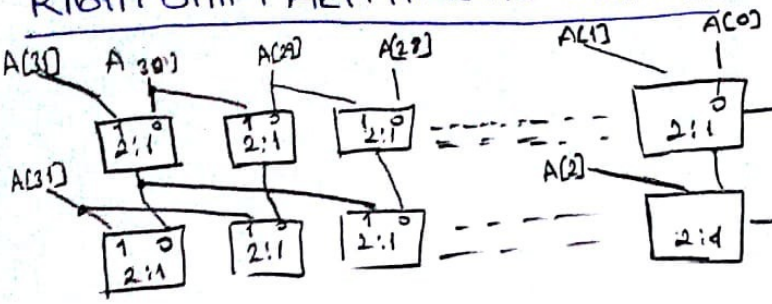
-> NOT GATE

Implementation of Zero

NOT: ALL MODULE HAVE Imp. of Zero Control

RIGHT SHIFT ARITHMETIC MODULE

HAMZA YAĞUZTÜRKOĞLU
171044086



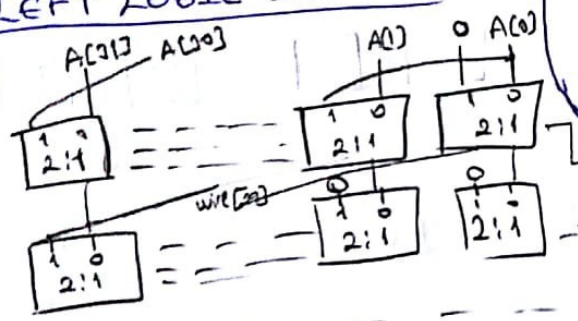
B[2] → 4 bit right shift

B[3] → 8 bit right shift

B[4] → 16 bit right shift

Not: we put most significant digit of number to empty field.

LEFT LOGIC SHIFT



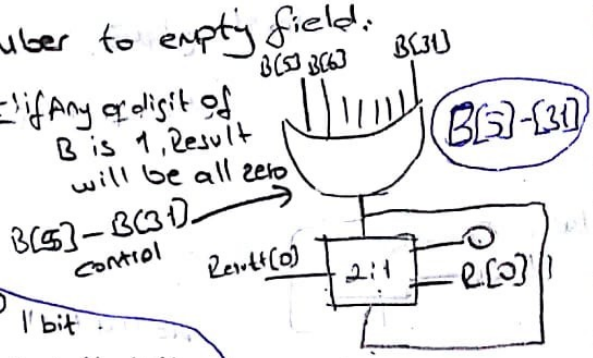
→ 2 bit left shift

→ 4 bit left shift

→ 8 bit left shift

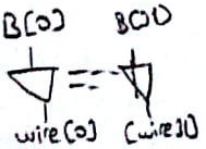
→ 16 bit left shift

Not: if Any of digit of B is 1, Result will be all zero



Not: we put 0 bit to empty field and we control B[5] to B[31] with above OR and MUX design.

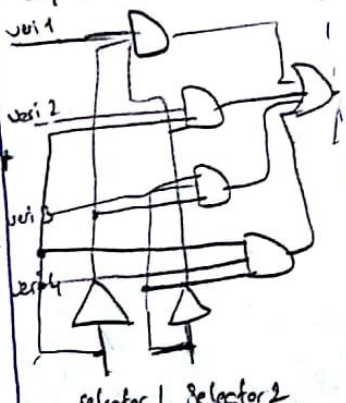
SUB-MODULE



Not: Then we need to 1 for two's complement and

Other side add module same

4:1 MUX



2:1 MUX