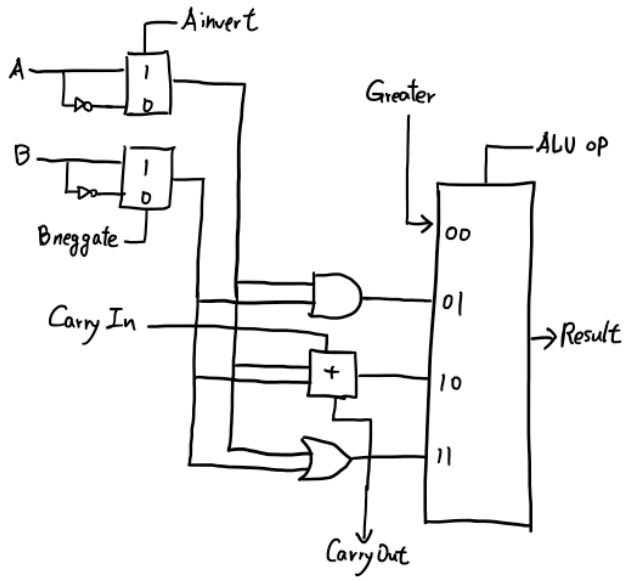
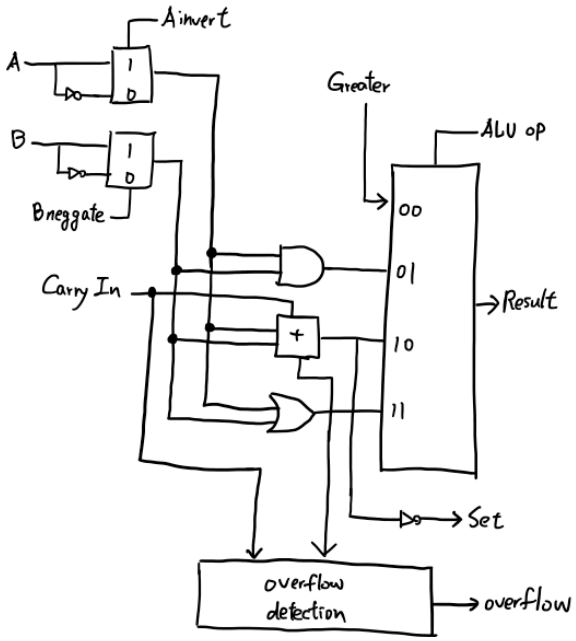
108062213 顏浩昀
Computer Architecture HW3
2022.04.14

1. In class we have described a 64-bit ALU whose control input ALUop is composed of 1-bit Ainvert, 1-bit Bnegate, and 2-bit Operation from left to right. Re-design the ALU to meet the following new specification. You need to draw the circuit diagrams of each 1-bit ALU and the 64-bit ALU, similar to those shown in the lecture notes.

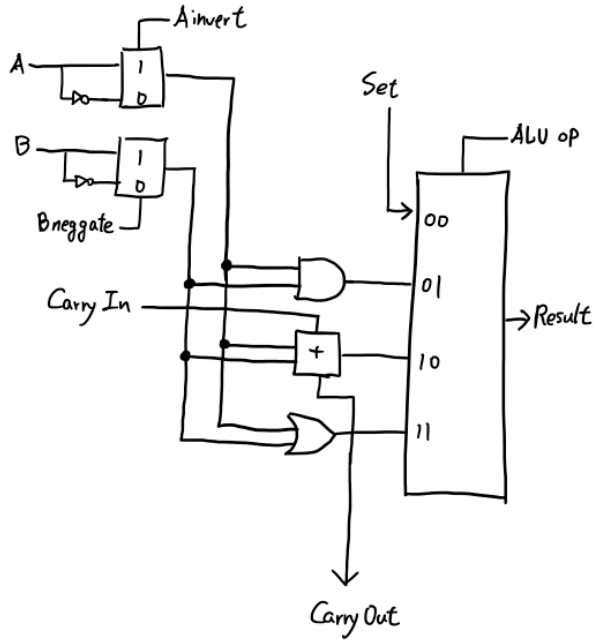
(1-bit ALU for bit1 to bit62)



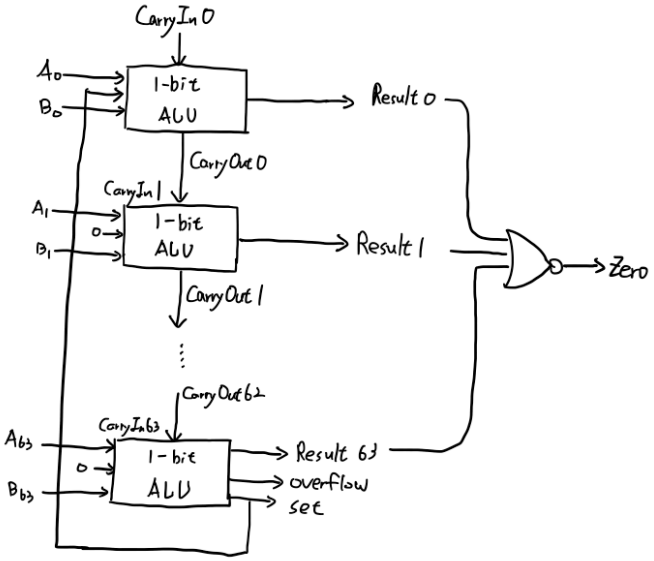
(1-bit ALU for bit63)



(1-bit ALU for bit0)



(64-bit ALU)



2. Consider two unsigned binary numbers: M = 1010 and N = 0101.

a). Write down each step of M × N according to version 1 of the multiply algorithm.

Step	Multiplicand(M)	Multiplier(N)	Product
0	0000 1010	0101	0000 0000
1	0001 0100	0010	0000 1010
2	0010 1000	0001	0000 1010
3	0101 0000	0000	0011 0010
Finish			0011 0010

b). Write down each step of M × N according to version 2 of the multiply algorithm.

Step	Multiplicand(M)	Multiplier(N)	Product
0	1010	0101	0000 0101
1	1010	010	0101 0010
2	1010	01	0010 1001
3	1010	0	0110 0100
Finish			0011 0010

c). Write down each step of M ÷ N according to version 1 of the divide algorithm.

Step	Dividend(M)	Divisor(N)	Remainder	Quotient
0	1010	0101 0000	0000 1010	0000
1	1010	0010 1000	0000 1010	0000
2	1010	0001 0100	0000 1010	0000
3	1010	0000 1010	0000 1010	0000
4	1010	0000 0101	0000 0000	0001
Finish			0000 0000	0010

d). Write down each step of M ÷ N according to version 2 of the divide algorithm.

Step	Divisor(N)	Remainder and Quotient
0	0101	0001 0100
1	0101	0010 1000
2	0101	0101 0000
3	0101	0000 0001
4	0101	0000 0010
Finish		0000 0010

3. Consider two decimal numbers: X = 785.3125 and Y = −13.125.
a). Write down X and Y in the IEEE 754 single precision format.

IEEE 754 standard

Sign bit	Exponent(8)	Fraction(23)

X = 785.3125 = 785+0.25+0.0625 = 1100010001.0101_2 = $1.1000100010101*2^9$
Exponent = 9+127=136 = 10001000_2 , Fraction = 100010010101

0	1000 1000	1000 1000 1010 1000 0000 000

Y = -13.125 = -1*(13+0.125) = $-1*1101.001_2$ = $-1*1.101001*2^3$
Exponent = 3+127 = 130 = 10000010, Fraction = 101001

1	1000 0010	1010 0100 0000 0000 0000 000

b). Assuming X and Y are given in the IEEE 754 single precision format, show all the steps to perform X + Y and write the result in the IEEE 754 single precision format.
X+Y = $1.1000100010101*2^9$ + $-1.101001*2^3$ = $1.1000100010101*2^9$+$-0.000001101001*2^9$ = $1.10000010000110*2^9$

0	1000 1000	1000 0010 0001 1000 0000 000

c). Assuming X and Y are given in the IEEE 754 single precision format, show all the steps to perform X × Y and write the result in the IEEE 754 single precision format.
X*Y = $1.1000100010101*2^9$ * $-1*1.101001*2^3$ = $-10.100001000011001110 1*2^{12}$ = $-1.010000100001100111 01*2^{13}$ Exponent = 13+127 = 140 = 10001100_2

1	1000 1100	0100 0010 0001 1001 1101 000

4. Let W be the hexadecimal pattern 0xFF8E0E13.

a). What decimal number does W represent if it is a two's complement integer?
0xFF8E0E13 = 1111 1111 1000 1110 0000 1110 0001 0011
=>(do 2's complement) 0000 0000 0111 0001 1111 0001 1110 1100 + 1 = 0000 0000 0111 0001 1111 0001 1110 1101 = 7467501 => W = -7467501

b). What decimal number does W represent if it is an IEEE 754 floating-point number?
0xFF8E0E13 = 1 1111 1111 000 1110 0000 1110 0001 0011

1	1111 1111	000 1110 0000 1110 0001 0011

Since Exponent part is 1111 1111 (255) and Fraction part not equal to 0, W is NaN.

c). Is it possible for W to be a RISC-V instruction? If yes, what is the corresponding assembly instruction? If not, why?
W is 32 bit. It may be a RISC-V instruction.
First check opcode=0010011 -> I-type.
func3 is 000 -> the instruction is addi, rs=11100=28, rs1=11100=28, imm[11:0]=111111111000= -8.
The instruction is addi, x28, x28, -8

5. Consider a new floating-point number representation that is only 16 bits wide. The leftmost bit is still the sign bit, the exponent is 5 bits wide and has a bias of 15, and the faction is 10 bits long. A hidden 1 to the left of the binary point is assumed. In this representation, any 16-bit binary pattern having 00000 in the exponent field and a non-zero fraction indicates a denormalized number: $(-1)^S \times (0 + \text{Fraction}) \times 2^{-14}$.

Sign bit	Exponent(5)	Fraction(10)

a).What is the smallest positive "normalized" number, denoted as a0?

0	00001	0

Real exponent = 1-15 = -14. Fraction = 0. The smallest number a0 is $1.0*2^{-14}$

b).What is the largest positive "denormalized" number, denoted as a1? What is the second largest positive "denormalized" number, denoted as a2 ?

0	00000	1111 1111 11
0	00000	1111 1111 10

The largest denormalized positive number a1 = $1*0.1111111111*2^{-14}$.

The second largest denormalized positive number a2 = $1*0.1111111110*2^{-14}$

c). Find the differences between a0 and a1, and between a1 and a2. Also describe what you observe and any implication from them.
a0-a1 = $0.0000000001*2^{-14} = 1.0*2^{-24}$ **,** a1-a2 = $0.0000000001*2^{-14} = 1.0*2^{-24}$

I find 2 point by above equation. Firstly, the smallest positive normalized number still larger than the largest denormalized number (a0 > a1). Secondly, the gap between 2 adjacent denormalized number might be same.

d). What decimal number does the binary pattern 1011110110100111 represent ?

1	01111	0110 1001 11

Exponent = 01111 = 15, real exponent = 0

Number = $-1 * 1.0110100111*2^0 = -1.0110100111 = -1.4130859375$

e). Let U be the nearest representation of the decimal number 1.24; that is, U has the smallest approximation error. What is U? What is the actual decimal number represented by U ?

In our design, there are only 10 bit to place fraction part, so
$1.24 = 1+0.24 \approx 1.0011110101 = 1.2392578125$ and also $1.24 \approx 1.0011110110 = 1.240234375$, the later one has slightly smaller difference between 1.24. I can get U = 1.240234375 = 1.0011110110
-> U express in our design standard is

0	01111	0011 1101 10

(exponent=0+15 = 01111)