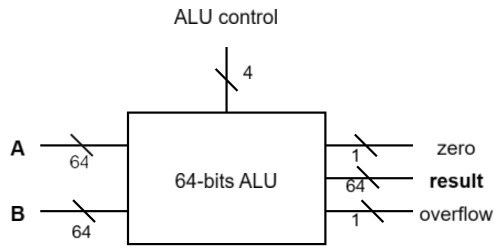
CS4100 Computer Architecture

Spring 2024, Homework 3

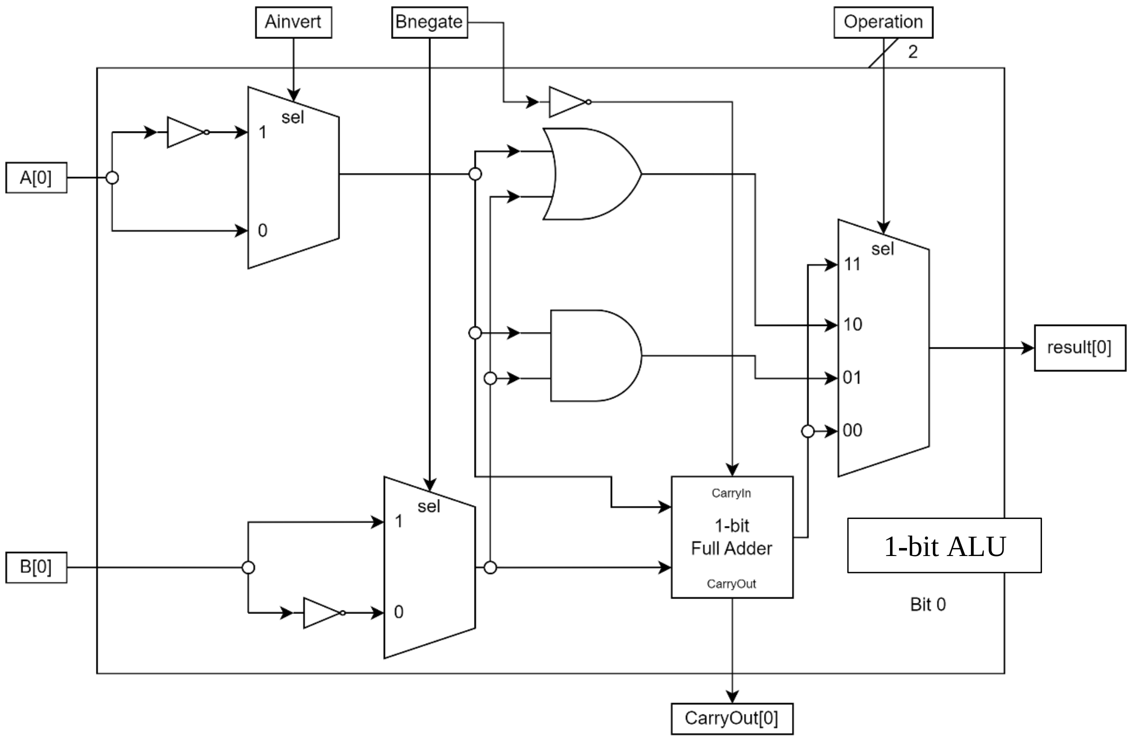
Due: 23:59, 4/21/2024

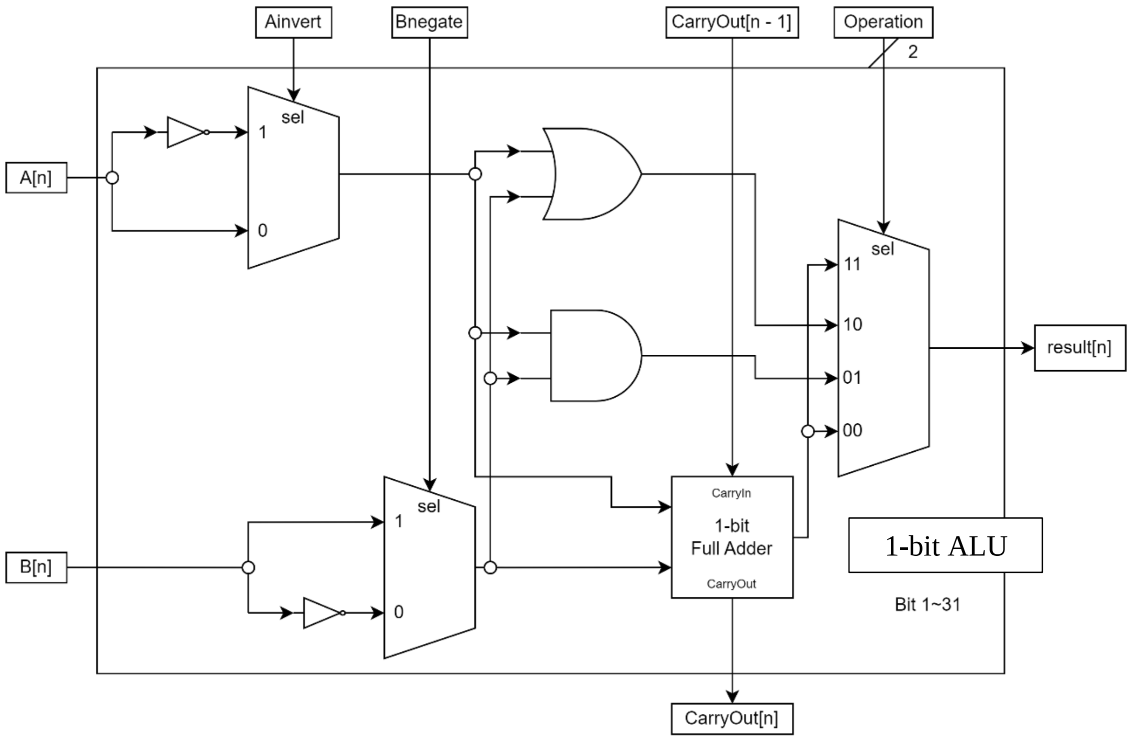
1. (18 points) Please modify the ALU design introduced in the class to satisfy the following requirements:

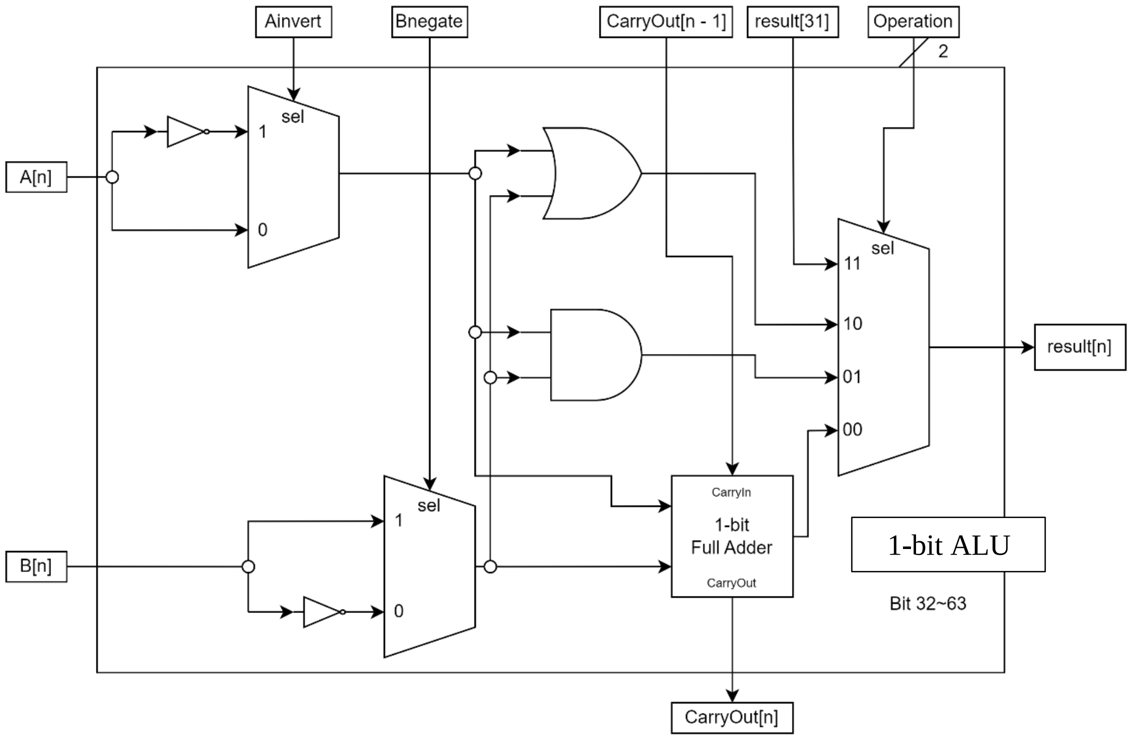
Ainvert	Bnegate	Operation	Function
0	1	01	AND
0	1	10	OR
0	1	00	add
0	0	00	sub
1	0	01	NOR
0	1	11	add-ext
0	0	11	sub-ext

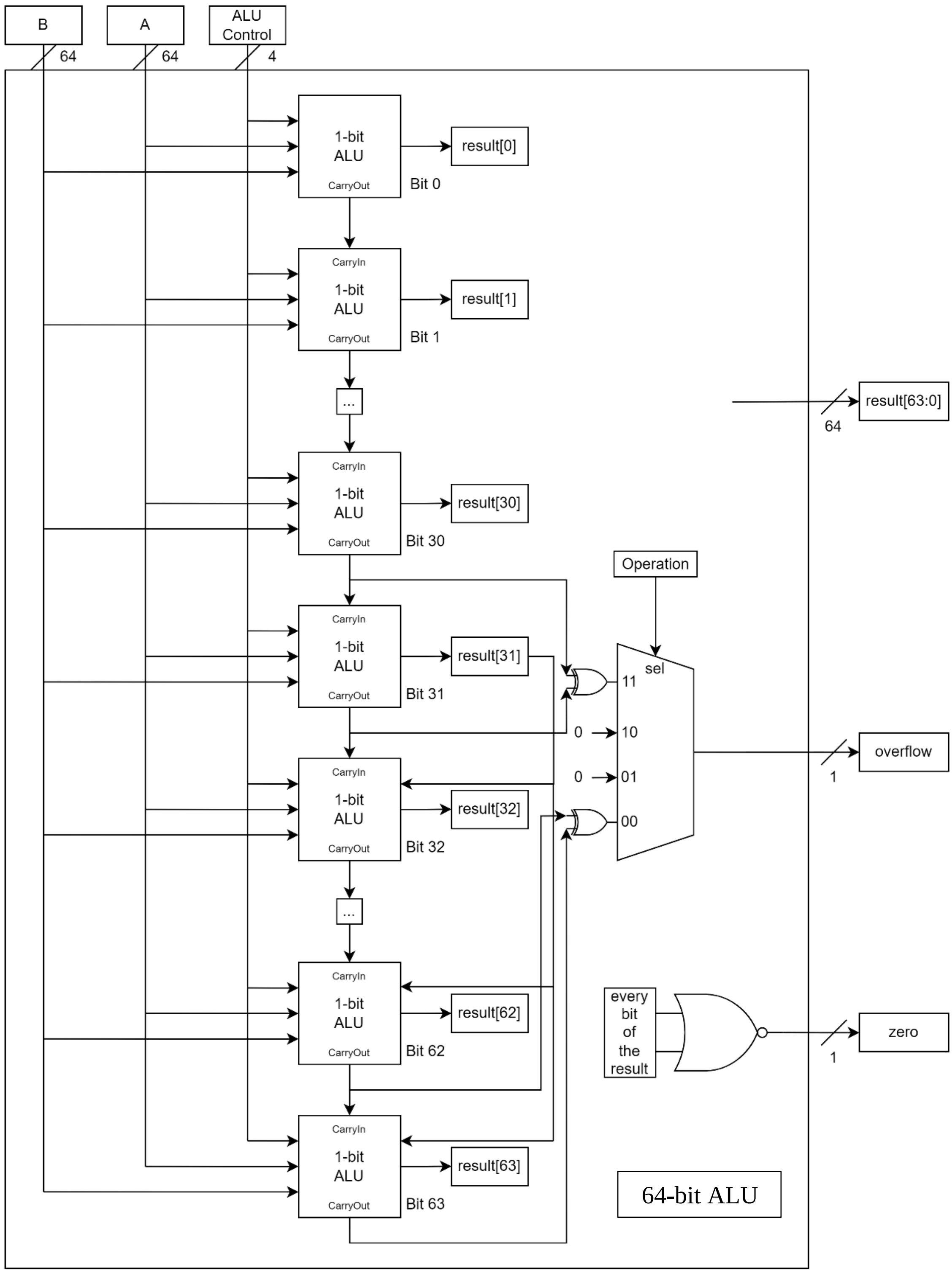


Here, "add-ext" and "sub-ext" refer to 32-bit addition and subtraction with sign-extension to 64 bits. You are required to draw the circuit diagrams for each 1-bit ALU and the 64-bit ALU. For each 1-bit ALU, use only one full adder to perform an addition or subtraction operation, similar to the method demonstrated in class. Additionally, show the ALU control signals for "add-ext" and "sub-ext" in your design.




Ainvert
Bnegate
CarryOut[n - 1]
Operation
2
A[n]
sel
1
0
B[n]
11
10
01
00
result[n]
CarryIn
1-bit
Full Adder
CarryOut
1-bit ALU
Bit 1~31
CarryOut[n]


Ainvert
Bnegate
CarryOut[n - 1]
result[31]
Operation
2
A[n]
sel
1
0
B[n]
11
10
01
00
result[n]
CarryIn
1-bit
Full Adder
CarryOut
1-bit ALU
Bit 32~63
CarryOut[n]


B
64
A
64
ALU Control
4
1-bit ALU
CarryIn
CarryOut
result[0]
Bit 0
result[1]
Bit 1
...
result[30]
Bit 30
result[31]
Bit 31
result[32]
Bit 32
result[62]
Bit 62
result[63]
Bit 63
Operation
sel
11
10
01
00
0
0
result[63:0]
64
overflow
1
every bit of the result
zero
1
64-bit ALU

2. (14 points) Consider two unsigned binary numbers: M = 1110 and N = 1001.

(a) (7 points) Write down each step of M × N according to version 1 of the multiply algorithm.

Iteration	**Step**	**Multiplier**	**Multiplicand**	**Product**
0	Initial values	1001	0000 1110	0000 0000
1	Prod = Prod + Mcand	1001	0000 1110	0000 1110
	Shift left multiplicand	1001	0001 1100	0000 1110
	Shift right multiplier	0100	0001 1100	0000 1110
2	Prod = Prod + 0	0100	0001 1100	0000 1110
	Shift left multiplicand	0100	0011 1000	0000 1110
	Shift right multiplier	0010	0011 1000	0000 1110
3	Prod = Prod + 0	0010	0011 1000	0000 1110
	Shift left multiplicand	0010	0111 0000	0000 1110
	Shift right multiplier	0001	0111 0000	0000 1110
4	Prod = Prod + Mcand	0001	0111 0000	0111 1110
	Shift left multiplicand	0001	1110 0000	0111 1110
	Shift right multiplier	0000	1110 0000	0111 1110

M * N = 0111 1110

(b) (7 points) Write down each step of M × N according to version 2 of the multiply algorithm.

Iteration	**Step**	**Multiplicand**	**Product**
0	Initial values	1110	0000 1001
1	LeftProd += Mcand	1110	1110 1001
	Shift right multiplier	1110	0111 0100
2	LeftProd += 0	1110	0111 0100
	Shift right multiplier	1110	0011 1010
3	LeftProd += 0	1110	0011 1010
	Shift right multiplier	1110	0001 1101
4	LeftProd += Mcand	1110	1111 1101
	Shift right multiplier	1110	0111 1110

M * N = 0111 1110

3. (14 points) Consider two unsigned binary numbers: M = 0111 and N = 0101.

(a) (7 points) Write down each step of M ÷ N according to version 1 of the division algorithm.

Iteration	**Step**	**Quotient**	**Divisor**	**Remainder**
0	Initial values	0000	0101 0000	0000 0111
1	Rem = Rem - Div	0000	0101 0000	1011 0111
	Rem < 0 ⇨ Rem += Div ⇨ Shift left quotient	0000	0101 0000	0000 0111
	Shift right divisor	0000	0010 1000	0000 0111
2	Rem = Rem - Div	0000	0010 1000	1101 1111
	Rem < 0 ⇨ Rem += Div ⇨ Shift left quotient	0000	0010 1000	0000 0111
	Shift right divisor	0000	0001 0100	0000 0111
3	Rem = Rem - Div	0000	0001 0100	1111 0011
	Rem < 0 ⇨ Rem += Div ⇨ Shift left quotient	0000	0001 0100	0000 0111
	Shift right divisor	0000	0000 1010	0000 0111
4	Rem = Rem - Div	0000	0000 1010	1111 1101
	Rem < 0 ⇨ Rem += Div ⇨ Shift left quotient	0000	0000 1010	0000 0111
	Shift right divisor	0000	0000 0101	0000 0111
5	Rem = Rem - Div	0000	0000 0101	0000 0010
	Rem >= 0 ⇨ Shift left quotient ⇨ quotient[0] = 1	0001	0000 0101	0000 0010
	Shift right divisor	0001	0000 0010	0000 0111

M / N = 0001…0010

(b) (7 points) Write down each step of M ÷ N according to version 2 of the division algorithm.

Iteration	Step	Divisor	Remainder / Quotient
0	Initial values	0101	0000 0111
1	Shift left remainder	0101	0000 1110
	LeftRem -= Dsivisor	0101	1011 1110
	Rem < 0 ⇨ LeftRem += Divisor ⇨ Shift left Rem	0101	0001 1100
2	LeftRem -= Dsivisor	0101	1100 1100
	Rem < 0 ⇨ LeftRem += Divisor ⇨ Shift left Rem	0101	0011 1000
3	LeftRem -= Dsivisor	0101	1110 1000
	Rem < 0 ⇨ LeftRem += Divisor ⇨ Shift left Rem	0101	0111 0000
4	LeftRem -= Dsivisor	0101	0010 0000
	Rem >= 0 ⇨ Shift left Rem ⇨ Rem[0] = 1	0101	0100 0001
5	Shift right leftRem	0101	0010 0001

M / N = 0001…0010

4. (12 points) Answer the following questions in detail. You will receive 0 point if you only write down the answers.

(a) (4 points) What decimal number does the bit pattern $05948DEC_{16}$ represent if it's a two's complement integer? If it's an unsigned number, is the result the same as the two's complement? If they are different, why?

$05948DEC_{16} = 12 + 14*16 + 13*16^2 + 8*16^3 + 4*16^4 + 9*16^5 + 5*16^6$

$= 93621740_{10}$

因為在 signed 的情況下 sign bit 是 0，因此 signed 與 unsigned 的結果會是一樣的。

(b) (4 points) Answer problem (a) with a different bit pattern $FA6B7214_{16}$.

signed number:

$FA6B7214_{16} = 1111\_1010\_0110\_1011\_0111\_0010\_0001\_0100_2$

$= -(0000\_0101\_1001\_0100\_1000\_1101\_1110\_1011_2 + 1)$

$= -(0000\_0101\_1001\_0100\_1000\_1101\_1110\_1100_2)$

$= -93621740_{10}$

unsigned number:

$FA6B7214_{16} = 1111\_1010\_0110\_1011\_0111\_0010\_0001\_0100_2$

$= 4201345556_{10}$

以 signed 的方式來看，sign bit 為 1 代表是負數，因此結果會與 unsigned 的結果不一樣。

(c) (4 points) What decimal numbers do $05948DEC_{16}$ and $FA6B7214_{16}$ represent if they are IEEE 754 floating point numbers.

$05948DEC_{16} = 0000\_0101\_1001\_0100\_1000\_1101\_1110\_1100_2$

⇨ 0_00001011_00101001000110111101100

⇨ sign bit: 0, exponent: $11_{10} - 127_{10}$ (bias) $= -116_{10}$,

fraction: $00101001000110111101100_2$

⇨ $1 + 2^{-3} + 2^{-5} + 2^{-8} + 2^{-12} + 2^{-13} + 2^{-15} + 2^{-16} + 2^{-17} + 2^{-18} + 2^{-20} + 2^{-21}$

$= 1 + 0.160581 = 1.160581$

⇨ Decimal number: $1.160581 * 2^{-116}$

$FA6B7214_{16} = 1111\_1010\_0110\_1011\_0111\_0010\_0001\_0100_2$

⇨ $1\_11110100\_11010110111001000010100_2$

⇨ sign bit: 1, exponent: $244_{10} - 127_{10} = 117_{10}$

fraction: $11010110111001000010100_2$

⇨ $1 + 2^{-1} + 2^{-2} + 2^{-4} + 2^{-6} + 2^{-7} + 2^{-9} + 2^{-10} + 2^{-11} + 2^{-14} + 2^{-19} + 2^{-21}$

$= 1 + 0.839419 = 1.839419$

⇨ Decimal number: $-1.839419 * 2^{117}$

5. (10 points) Consider two decimal numbers: X = 88.4375 and Y = −7.3125.

(a) (6 points) Write down X and Y in the IEEE 754 single precision format. You must detail how you get your answer, or you will receive 0 point.

$88.4375_{10} = 88_{10} + 0.25_{10} + 0.125_{10} + 0.0625_{10}$

$= 01011000.0111_2 = 1.0110000111_2 * 2_{10}{}^6$

X is positive => sign bit = 0

exponent = $6_{10} + 127_{10} = 133_{10} = 10000101_2$

fraction = $01100001110000000000000_2$

X = 01000010101100001110000000000000

$7.3125_{10} = 7_{10} + 0.25_{10} + 0.0625_{10}$

$= 00000111_2 + 0.0101_2 = 00000111.0101_2$

$= 1.110101_2 * 2_{10}{}^2$

Y is negative => sign bit = 1

exponent = $2_{10} + 127_{10} = 10000001_2$

fraction = $11010100000000000000000_2$

Y = 11000000111010100000000000000000

(b) (4 points) Assuming X and Y are given in the IEEE 754 single precision format. Show all the steps to perform X × Y and write the solution in the IEEE 754 single precision format.

Add exponent: 2 + 6 = 8，加上 bias 為 135

$1.01100001110000000000000_2 * 1.11010100000000000000000_2$

$= 10.100001101011001100000000000000000000000000000_2$

⇨ $10.1000011010110011000000_2 * 2^8 = 1.01000011010110011000000_2 * 2^9$

sign bit: 1, exponent: $9 + 127 = 136_{10} = 10001000_2$,

fraction: 01000011010110011000000

X * Y = 11000100001000011010110011000000

6. (20 points) Consider a new floating-point number representation that is only 16 bits wide. The leftmost bit is still the sign bit, the exponent is 9 bits wide and has a bias of 255, and the fraction is 6 bits long. A hidden 1 to the left of the binary point is assumed. In this representation, any 16-bit binary pattern having 000000000 in the exponent field and a non-zero fraction indicates a denormalized number: $(-1)^S \times (0 + \text{Fraction}) \times 2^{-254}$. Write the answers of (a), (b) and (c) in scientific notation, e.g., 1.0101×2^2.

(a) (3 points) What is the smallest positive "normalized" number, denoted as a0?
sign bit: 0, exponent: 000000001, fraction: 000000
exponent – bias = 1 – 255 = 254 => a0 = $1.000000_2 * 2^{-254}$.

(b) (6 points) What is the largest positive "denormalized" number, denoted as a1? What is the second largest positive "denormalized" number, denoted as a2?

sign bit: 0, exponent: 000000000, fraction: 111111

$0.111111_2 * 2^6 = 111111_2 = 63_{10}$

$a1 = 63 * 2^{-260} = 111111_2 * 2^{-260} = 1.111110_2 * 2^{-255}$

sign bit: 0, exponent: 000000000, fraction: 111110

$0.111110_2 * 2^6 = 111110_2 = 62_{10}$

$a2 = 62 * 2^{-260} = 111110_2 * 2^{-260} = 1.111100_2 * 2^{-255}$

(c) (4 points) Find the differences between a0 and a1, and between a1 and a2.
$a0 - a1 = 1.000000_2 * 2^{-254} - 1.111110_2 * 2^{-255} = 0.000001_2 * 2^{-254} = 1.000000_2 * 2^{-260}$
$a1 - a2 = 1.111110_2 * 2^{-255} - 1.111100_2 * 2^{-255} = 0.000010_2 * 2^{-255} = 1.000000_2 * 2^{-260}$

(d) (3 points) What binary number does the binary pattern 1011110110100111 represent?
sign bit: 1, exponent: 011110110, fraction: 100111
exponent - bias = $011110110_2 - 255_{10} = 246_{10} - 255_{10} = -9_{10}$
$1.100111_2 = 1100111_2 * 2_{10}^{-6} = 103_{10} * 2_{10}^{-6}$
This binary number is $-1100111_2 * 2_{10}^{-15}$.

(e) (4 points) Let U be the nearest representation of the decimal number 1.31; that is, U has the smallest approximation error. What is U? What is the actual decimal number represented by U?

0.31 * 2 => (0).62 * 2 => (1).24 * 2 => (0).48 * 2 => (0).96 * 2 => (1).92 * 2 => (1).84 * 2 => (1).68 (進位)。 => $1.31_{10} = 1.010100_2$。$255 + 1 = 256_{10}$

=> sign bit: 0, exponent: 100000000 (256_{10}), fraction: 010100

U is 0100000000010100. Its actual decimal number is 1.3125

7. (12 points) **X** is a 32-bit signed integer variable, **&** is the bitwise-AND operator, and ">>" is the sra (shift right arithmetic) operator. For the following options, determine whether they provide the correct result for (**X** / 4) and explain the reasons.

   (a) (**X** + 3) >> 2

   (b) ((**X** >= 0) ? **X** >> 2 : (**X** + 3) >> 2)

   (c) **X** >> 2

   (d) (**X** + ((**X** >> 31) **&** 3)) >> 2

Let A be a nonnegative 32-bit signed integer. Obviously, A >> 2 = A / 4.

(-A) / 4 = - (A / 4) = - (A >> 2) = ~ (A >> 2) + 1 = ((~A) >> 2) + 1 = (~A + 4) >> 2

= (~A + 1 + 3) >> 2 = (-A + 3) >> 2

Therefore, X / 4 = X >> 2 if X >= 0. Otherwise, X / 4 = (X + 3) >> 2.

Counter case:

- ⇨ Use X / 4 = (X + 3) >> 2: For X = 1 >= 0, (X + 3) >> 2 = 1 ≠ X / 4 = 0
- ⇨ Use X / 4 = X >> 2: For X = -1 < 0, X >> 2 = -1 ≠ X / 4 = 0

(a) 由上述可知，若 X >= 0 則 (X + 3) >> 2 不一定等於 X / 4

(b) 由上述可知，((X >= 0) ? X >> 2 : (X + 3) >> 2) = X / 4

(c) 由上述可知，若 X < 0 則 X >> 2 不一定等於 X / 4

(d) 若 X < 0 則 (X + ((X >> 31) & 3)) >> 2 = (X + 3) >> 2 = X / 4

若 X >= 0 則 (X + ((X >> 31) & 3)) >> 2 = (X + 0) >> 2 = X / 4

(X + ((X >> 31) & 3)) >> 2 = X / 4