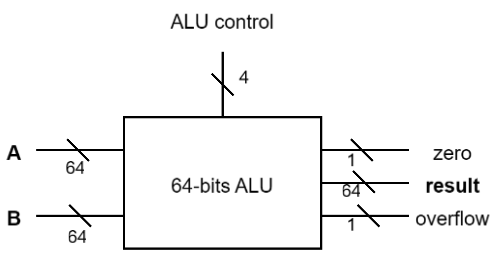
CS4100 Computer Architecture

Spring 2025, Homework 3

Due: 23:59, 4/12/2025

1. (18%) 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
			add-ext
			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.

2. (16%) Consider four unsigned binary numbers: M = 1100 , N = 1010, P = 0111 and Q = 0101.
   (a) (4%) Write down each step of M × N according to version 1 of the multiply algorithm.
   (b) (4%) Write down each step of M × N according to version 2 of the multiply algorithm.
   (c) (4%) Write down each step of P ÷ Q according to version 1 of the division algorithm.
   (d) (4%) Write down each step of P ÷ Q according to version 2 of the division algorithm.

3. (12%) Booth's Algorithm is a multiplication algorithm designed for signed binary numbers, particularly useful for numbers represented in two's complement form. It is widely used in computer arithmetic operations due to its efficiency in handling both positive and negative numbers. The following are the steps of Booth's Algorithm along with an example:
   I. Initialize two registers:
      - Place the n-bit multiplicand in the n-bit register Multiplicand.
      - Initialize the (2n+1)-bit register Product, where each bit in Product[2n:n+1] and Product [0] is set to 0, and the n-bit multiplier is placed in Product[n:1].
   II. Repeat the following for n times:
      - Product[1:0] = 10 → subtract Multiplicand from Product[2n:n+1];

Product[1:0] = 01 → add Multiplicand to Product[2n:n+1];

Product[1:0] = 00 or 11 → no operation is needed.

- Perform an arithmetic right shift on Product.

III. The final result is stored in Product[2n:1].

Example: $0010_2 \times 1101_2 = 1111\ 1010_2$

Iteration	Step	Multiplicand	Product
0	Initialize registers	0010	0000 1101 0
1	10 → Subtract Multiplicand from Product[8:5]	0010	1110 1101 0
	Arithmetically shift right on Product	0010	1111 0110 1
2	01 → Add Multiplicand to Product[8:5]	0010	0001 0110 1
	Arithmetically shift right on Product	0010	0000 1011 0
3	10 →Subtract Multiplicand from Product[8:5]	0010	1110 1011 0
	Arithmetically shift right on Product	0010	1111 0101 1
4	11 → No operation	0010	1111 0101 1
	Arithmetically shift right on Product	0010	$\underline{1111\ 1010}$ 1

(a) (6 %) Write down each step of $8_{10} \times 7_{10}$ using Booth's Algorithm.

(b) (6 %) Write down each step of $-5_{10} \times 6_{10}$ using Booth's Algorithm.

4. (12%) Answer the following questions in detail. You will receive 0 points if you only write down the answers.
   (a) (4%) What decimal number does the bit pattern $3A5F8C1B_{16}$ 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?
   (b) (4%) Answer question (a) with a different bit pattern $B7D4E2C9_{16}$.
   (c) (4%) What decimal numbers do $3A5F8C1B_{16}$ and $B7D4E2C9_{16}$ represent if they are IEEE 754 floating point numbers?

5. (12%) Consider two decimal numbers: X = 60.4375 and Y = −5.3125.
   (a) (6%) Write down X and Y in the IEEE 754 single precision format. You must detail how you get your answers, or you will receive 0 points.
   (b) (6%) 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.

6. (20%) 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%) What is the smallest positive "normalized" number, denoted as a0?

(b) (6%) What is the largest positive “denormalized” number, denoted as a1? What is the second largest positive “denormalized” number, denoted as a2?

(c) (4%) Find the differences between a0 and a1, and between a1 and a2.

(d) (3%) What binary number does the binary pattern 1011110110100111 represent?

(e) (4%) 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?

7. (10%) Which of the following statements are <u>incorrect</u>?

(a) In coding assembly, an integer multiply by a power of 2 can be replaced by a left shift, and an integer division by a power of 2 can be replaced by a right shift.

(b) Let x $= 1.2 \times 10^{40}$, y $= -1.2 \times 10^{40}$, and z = 1.0
In FP addition (x+y)+z gives the same result as x+(y+z).

(c) Considering the 32-bit IEEE 754 single precision floating point format,
the smallest positive denormalized number is: 1.0×2^{-149}

(d) If one wishes to increase the accuracy of the floating-point numbers that can be represented, then he/she should increase the size of exponent part in the floating-point format.

(e) IEEE 754 floating-point standard adopts bias scheme to increase maximum number in exponent part.