Computer Architecture HW 3

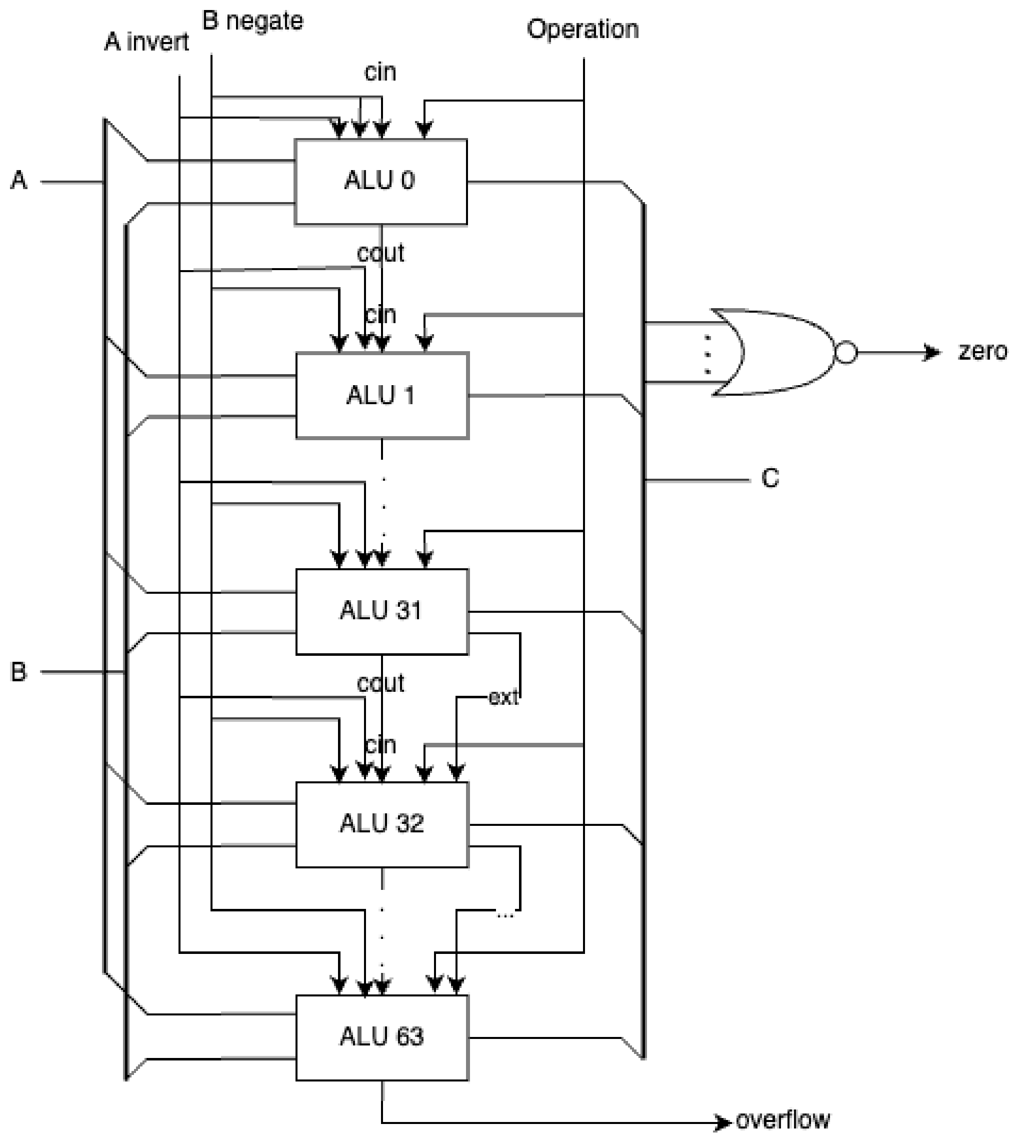
111062117, Hsiang-Sheng Huang

April 12, 2025

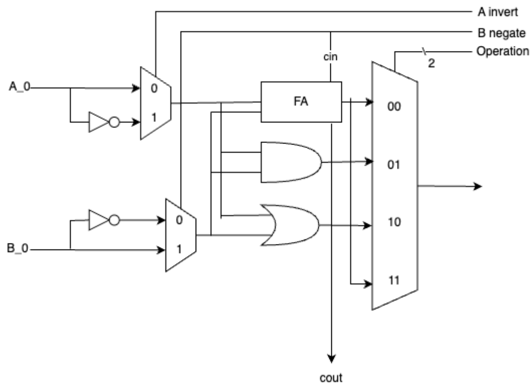
1

$A_{\textbf{invert}}$	$B_{\textbf{negate}}$	**Operation**	**Function**
0	1	11	add-ext
0	0	11	sub-ext

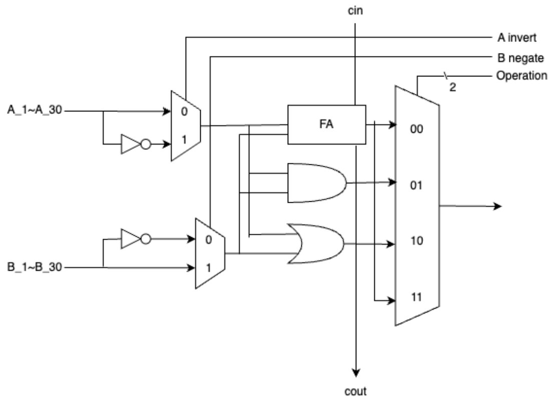
Complete 64-bit ALU Architecture



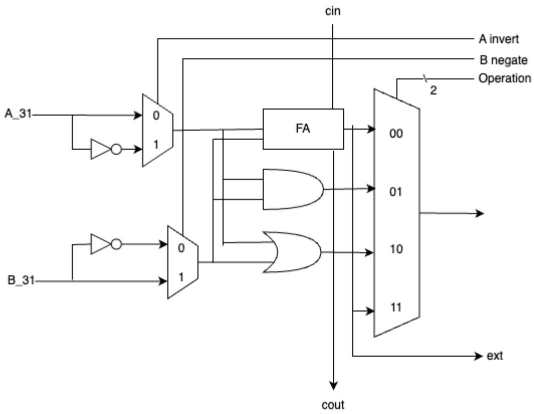
ALU0



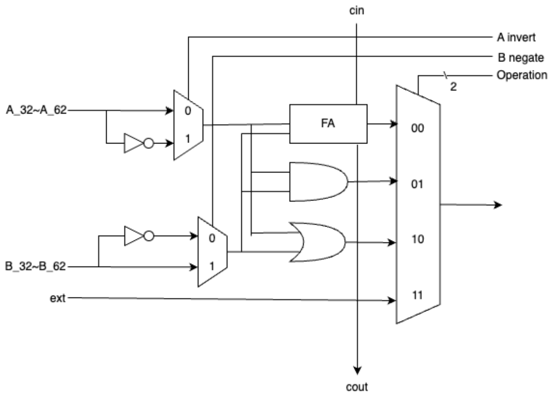
ALU1~30



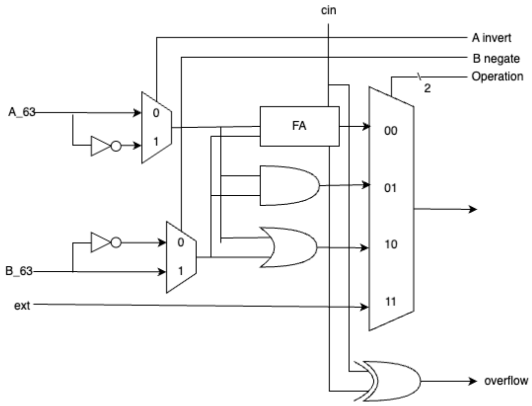
ALU31



ALU32~62



ALU63



2

(a)

Iteration	Step	Multiplier	Multiplicant	Product
0	Initial Values	1010	0000 1100	0000 0000
1	0 → `No operation`	1010	0000 1100	0000 0000
	Shift left Multiplicant	1010	0001 1000	0000 0000
	Shift right Multiplier	0101	0001 1000	0000 0000
2	1 → `Product += Multiplicant`	0101	0001 1000	0001 1000
	Shift left Multiplicant	0101	0011 0000	0001 1000
	Shift right Multiplier	0010	0011 0000	0001 1000
3	0 → `No operation`	0010	0011 0000	0001 1000
	Shift left Multiplicant	0010	0110 0000	0001 1000
	Shift right Multiplier	0001	0110 0000	0001 1000
4	1 → `Product += Multiplicant`	0001	0110 0000	0111 1000
	Shift left Multiplicant	0001	1100 0000	0111 1000
	Shift right Multiplier	0000	1100 0000	0111 1000

(b)

Iteration	Step	Multiplicand	Product
0	Initial Values	1100	0000 1010
1	0 → `No operation`	1100	0000 1010
	Shift right Product	1100	0000 0101
2	1 → `prod[left] += Multiplicand`	1100	1100 0101
	Shift right Product	1100	0110 0010
3	0 → `No operation`	1100	0110 0010
	Shift right Product	1100	0011 0001
4	1 → `prod[left] += Multiplicand`	1100	1111 0001
	Shift right Product	1100	0111 1000

(c)

Iteration	Step	Quotient	Divisor	Remainder
0	Initial Values	0000	0101 0000	0000 0111
1	Rem -= Div	0000	0101 0000	1011 0111
	Rem<0 $\rightarrow$ +Div, LSL Q, Q0=0	0000	0101 0000	0000 0111
	Shift Div right	0000	0010 1000	0000 0111
2	Rem -= Div	0000	0010 1000	1101 1111
	Rem<0 $\rightarrow$ +Div, LSL Q, Q0=0	0000	0010 1000	0000 0111
	Shift Div right	0000	0001 0100	0000 0111
3	Rem -= Div	0000	0001 0100	1111 0011
	Rem<0 $\rightarrow$ +Div, LSL Q, Q0=0	0000	0001 0100	0000 0111
	Shift Div right	0000	0000 1010	0000 0111
4	Rem -= Div	0000	0000 1010	1111 1101
	Rem<0 $\rightarrow$ +Div, LSL Q, Q0=0	0000	0000 1010	0000 0111
	Shift Div right	0000	0000 0101	0000 0111
5	Rem -= Div	0000	0000 0101	0000 0010
	Rem$\geq$0 $\rightarrow$ LSL Q, Q0=1	0001	0000 0101	0000 0010
	Shift Div right	0001	0000 0010	0000 0010

(d)

Iteration	Step	Remainder | (Quotient)	Divisor
0	Initial Values	0000 0111	0101
	Shift Remainder left	0000 1110	0101
1	Rem[left] -= Div	1011 1110	0101
	Rem[left]<0$\rightarrow$+Div, LSL Rem, R0=0	0001 1100	0101
2	Rem[left] -= Div	1100 1100	0101
	Rem[left]<0$\rightarrow$+Div, LSL Rem, R0=0	0011 1000	0101
3	Rem[left] -= Div	1110 1000	0101
	Rem[left]<0$\rightarrow$+Div, LSL Rem, R0=0	0111 0000	0101
4	Rem[left] -= Div	0010 0000	0101
	Rem[left]$\geq$0$\rightarrow$LSL Rem, R0=1	0100 0001	0101
5	Shift Rem[left] right	0010 0001	0101

3

(a)

Iteration	Step	Multiplicand	Product
0	Initial Values	1000	0000 0111 0
1	10 → `Subtract multiplicand from product[8:5]`	1000	1000 0111 0
	Arithmetic Shift Right Product	1000	1100 0011 1
2	11 → `No operation`	1000	1100 0011 1
	Arithmetic Shift Right Product	1000	1110 0001 1
3	11 → `No operation`	1000	1110 0001 1
	Arithmetic Shift Right Product	1000	1111 0000 1
4	01 → `Add multiplicand to product[8:5]`	1000	0111 0000 1
	Arithmetic Shift Right Product	1000	0011 1000 0

(b)

Iteration	Step	Multiplicand	Product
0	Initial Values	1011	0000 0110 0
1	00 → `No operation`	1011	0000 0110 0
	Arithmetic Shift Right Product	1011	0000 0011 0
2	10 → `Subtract multiplicand from product[8:5]`	1011	0101 0011 0
	Arithmetic Shift Right Product	1011	0010 1001 1
3	11 → `No operation`	1011	0010 1001 1
	Arithmetic Shift Right Product	1011	0001 0100 1
4	01 → `Add multiplicand to product[8:5]`	1011	1100 0100 1
	Arithmetic Shift Right Product	1011	1110 0010 0

4

(a)

$$\begin{aligned} 3A5F8C1B_{16} &= 0011\ 1010\ 0101\ 1111\ 1000\ 1100\ 0001\ 1011_2 \\ &= 979,340,315_{10} \end{aligned}$$

If it's an unsigned number, the result is the same as the two's complement. The reason is that the MSB is 0, which indicates a positive number in both representations.

(b)

$$\begin{aligned} B7D4E2C9_{16} &= 1011\ 0111\ 1101\ 0100\ 1110\ 0010\ 1100\ 1001_2 \\ &= -1,210,785,079_{10} \end{aligned}$$

If it's a unsigned number, the result is **NOT** same as the two's complement. The reason is that the MSB is 1, which indicates a negative number in two's complement representation.

(c)

(i)

$$3A5F8C1B_{16} = 0011\ 1010\ 0101\ 1111\ 1000\ 1100\ 0001\ 1011_2$$

Sign	Exponent (8 bits)	Fraction (23 bits)
0	01110100	10111111000110000011011

- Sign: 0 (positive)
- Exponent: $01110100_2 = 116_{10}$, which is $116 - 127 = -11$
- Significand: $1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-8} + 2^{-12} + 2^{-13} + 2^{-19} + 2^{-21} + 2^{-22} \approx 1.746094$

Representation: $1.746094 \times 2^{-11} \approx 8.53 \times 10^{-4}$

(ii)

$$B7D4E2C9_{16} = 1011\ 0111\ 1101\ 0100\ 1110\ 0010\ 1100\ 1001_2$$

Sign	Exponent (8 bits)	Fraction (23 bits)
1	01101111	10101001110001011001001

- Sign: 1 (negative)
- Exponent: $01101111_2 = 111_{10}$, which is $111 - 127 = -16$
- Significand: $1{+}2^{-1}{+}2^{-3}{+}2^{-5}{+}2^{-8}{+}2^{-9}{+}2^{-10}{+}2^{-11}{+}2^{-14}{+}2^{-15}{+}2^{-17}{+}2^{-20}{+}2^{-23} \approx 1.664$

Representation: $-1.664 \times 2^{-16} \approx -2.54 \times 10^{-5}$

5

(a)

(i)

$$X = 60.4375_{10} = 111100.0111_2 = 1.111000111_2 \times 2^5$$

In IEEE 754 single precision:

- Sign bit: 0 (positive)
- Exponent: $5 + 127 = 132_{10} = 10000100_2$
- Fraction: 111000111_2 followed by zeros

IEEE 754 representation: $X = 0\ 10000100\ 11100011100000000000000$

(ii)

$$Y = -5.3125_{10} = -101.0101_2 = -1.010101_2 \times 2^2$$

In IEEE 754 single precision:

- Sign bit: 1 (negative)
- Exponent: $2 + 127 = 129_{10} = 10000001_2$
- Fraction: 010101_2 followed by zeros

IEEE 754 representation: $Y = 1\ 10000001\ 01010100000000000000000$

(b)

Multiply $X \times Y$:

- Sign bit: $0 \oplus 1 = 1$ (result is negative)
- Exponents: $5 + 2 = 7$
- Significands: $1.111000111_2 \times 1.010101_2 = 10.100000100010011_2 = 1.0100000100010011_2 \times 2^1$

Final exponent: $7 + 1 = 8$
Biased exponent: $8 + 127 = 135 = 10000111_2$
IEEE 754 representation: 1 10000111 01000001000100110000000

6

(a)

- Sign bit: 0
- Exponent: $1 - 255 = -254$
- Fraction: 000000_2

So $a_0 = 0\ 000000001\ 000000_2 = 1 \times 1.000000_2 \times 2^{-254}$

(b)

(i)

- Sign bit: 0
- Exponent: -254
- Fraction: 111111_2

So $a_1 = 0\ 000000000\ 111111_2 = 0.111111_2 \times 2^{-254} = 1.11111_2 \times 2^{-255}$

(ii)

- Sign bit: 0
- Exponent: -254
- Fraction: 111110_2

So $a_2 = 0\ 000000000\ 111110_2 = 0.111110_2 \times 2^{-254} = 1.1111_2 \times 2^{-255}$

(c)

(i)

$$\begin{aligned} a_1 - a_0 &= 1.000000 \times 2^{-254} - 1.111111 \times 2^{-255} \\ &= 1.000000 \times 2^{-254} - 0.111111 \times 2^{-254} \\ &= 0.000001 \times 2^{-254} \\ &= 2^{-260} \end{aligned}$$

(ii)

$$\begin{aligned} a_1 - a_2 &= 1.11111 \times 2^{-255} - 1.11110 \times 2^{-255} \\ &= 0.00001 \times 2^{-255} \\ &= 2^{-260} \end{aligned}$$

(d)

- Sign bit: 1
- Exponent: $011110110_2 = 246_{10}$, so the exponent is $246 - 255 = -9$
- Fraction: 100111_2

So the binary number is $1\ 011110110\ 100111_2 = -1.100111_2 \times 2^{-9}$

(e)

To find the nearest representation of, we need to convert it to binary.
For integer part:

$$1 = 1.000000_2$$

For fractional part:

$$0.31 = 0.010011_2$$

So $U = 1.010011_2 \times 2^0$.
The actual decimal number represented by U is $1.010011_2 = 1.3125$.

7

Statements (b), (d), (e) are incorrect.

- (b) is incorrect because floating point addition is not associative. In this case, $(x+y)+z = 0+z = 1.0$, but $x+(y+z) \approx x+y \approx 0$ since z is negligible compared to y.
- (d) is incorrect because increasing the exponent size increases the range of representable numbers, not their accuracy. Accuracy (precision) is improved by increasing the fraction part.
- (e) is incorrect because the bias scheme in IEEE 754 is used to simplify comparison between floating point numbers, not to increase the maximum representable value.