Binary to Octal, and Hexadecimal

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Signed Numbers: Binary, Octal, and Hexadecimal

For any binary, octal, or hexadecimal number, the most significant digit represents the sign of the number. If the MSD is less than half the base of the number, the sign of the number is positive. If the MSD is greater than or equal to the base of the number, the sign of the number is negative.

Example: 01010_2 is positive… 0 < (2/2) = 0 < 1
: 11010_2 is negative… 1 >= (2/2) = 1 >= 1

Example: 2025_8 is positive… 2 < (8/2) = 2 < 4
: 6547_8 is negative… 6 >= (8/2) = 6 >= 4

Example: $4A2E_{16}$ is positive… 4 < (16/2) = 4 < 8
: $954F_{16}$ is negative… 9 >= (16/2) = 9 >= 8

Padding

When padding a binary, octal, or hexadecimal number, there are only two options. If the number is positive, the number is padded with 0.
If the number is negative, the number is padded with the largest digit in the base. The largest number of the base is the base minus 1.

Example: 01010_2 pad to length 10 => 0000001010
: 11010_2 pad to length 8 => 11111010

Example: 2025_8 pad to length 7 => 0002025
: 6547_8 pad to length 8 => 77776547

Example: $4A2E_{16}$ pad to length 10 => 0000004A2E
: $954F_{16}$ pad to length 6 => FF954F

Trimming

When trimming a binary, octal, or hexadecimal number, there are only two options for removal. If the number is positive, the only digits that get removed are 0s. If the first digit that’s not 0 is less than half the base, the remaining 0 digit can be removed. If the number is negative, the only digits that get removed are the largest digit for the base. The largest number of the base is the base minus 1. If the first digit that’s not the largest number for the base is greater than or equal to half the base, the remaining largest digit can be removed.

Example: 000001010_2 => 01010
: 11111010_2 => 1010

Example: 000002025_8 => 2025
: 777776547_8 => 6547

Example: 000005025_8 => 05025
: 777772547_8 => 72547

Example: $0004A2E_{16}$ => 4A2E
: $FFFF954F_{16}$ => 954F

Example: $000BA2E_{16}$ => 0BA2E
: $FFFF154F_{16}$ => F154F

One’s Complement

The one’s complement of a binary, octal or hexadecimal number is the number with each of its digits subtracted from the largest number in the base. The largest number of the base is the base minus 1.

Example: 01010_2 => 10101… 1-0=1, 1-1=0,1-0=1, 1-1=0, 1-0=1
: 11010_2 => 00101… 1-1=0, 1-1=0,1-0=1, 1-1=0, 1-0=1

Example: 2025_8 => 5752… 7-2=5, 7-0=7, 7-2=5, 7-5=2
: 6547_8 => 1230… 7-6=1, 7-5=2, 7-4=3, 7-7=0

Example: $4A2E_{16}$ => B5D1… 15-4=B, 15-A=5, 15-2=D, 15-E=1
: $954F_{16}$ => 6AB0… 15-9=6, 15-5=A, 15-4=B, 15-F=0

Two’s Complement

The two’s complement of a binary, octal or hexadecimal number is the number with each of its digits subtracted from the largest number in the base. The largest number of the base is the base minus 1. The result is the one’s complement. Afterwards a positive value of 1 is added to it.

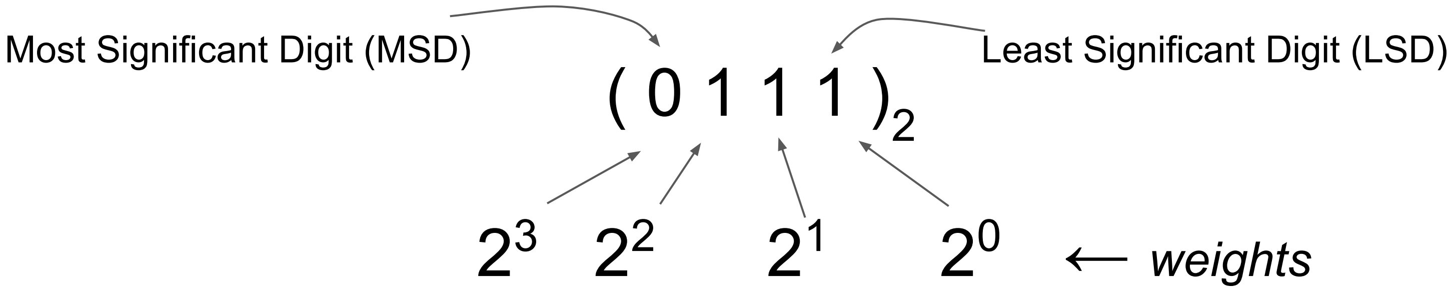
Example: 01010_2 => 10101 + 01 = 10110
: 11010_2 => 00101 + 01 = 00110

Example: 2025_8 => 5752 + 01 = 5753
: 6547_8 => 1230 + 01 = 1231

Example: $4A2E_{16}$ => B5D1 + 01 = B5D2
: $954F_{16}$ => 6AB0 + 01 = 6AB1

To Decimal

Example: Consider 0111, signed binary number (base-2 number)



Most Significant Digit (MSD)

Least Significant Digit (LSD)

$$(\ 0\ 1\ 1\ 1\ )_2$$

$$2^3 \quad 2^2 \quad 2^1 \quad 2^0 \quad \leftarrow \textit{weights}$$

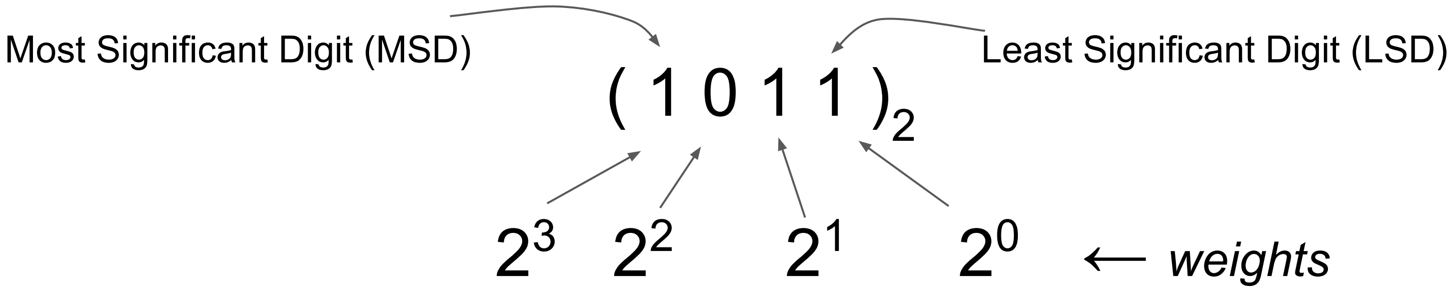
- starting with lowest weight of 2^0, moving from right-to-left
- the **weight** of each digit, increases by a factor of 2
- The decimal conversion is the sum of all the weights multiplied by the digits. If the most significant digit is greater than or equal to half the base, subtract the base raised to the power of the number of digits.

$$(\ 0\ 1\ 1\ 1\ )_2 = (0 \times (2^3)) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

$$(\ 0\ 1\ 1\ 1\ )_2 = 0 + 4 + 2 + 1 = 7$$

To Decimal

Example: Consider 1011, signed binary number (base-2 number)



Most Significant Digit (MSD)

$(\ 1\ 0\ 1\ 1\ )_2$

Least Significant Digit (LSD)

$2^3\quad 2^2\quad 2^1\quad 2^0$ ← *weights*

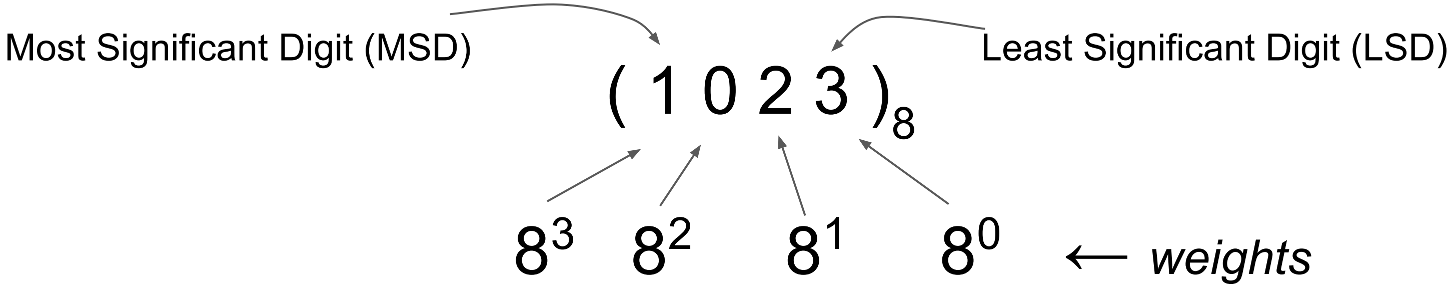
- starting with lowest weight of 2^0, moving from right-to-left
- the **weight** of each digit, increases by a factor of 2
- The decimal conversion is the sum of all the weights multiplied by the digits. If the most significant digit is greater than or equal to half the base, subtract the base raised to the power of the number of digits.

$$(\ 1\ 0\ 1\ 1\ )_2 = -2^4 + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

$$(\ 1\ 0\ 1\ 1\ )_2 = -16 + 8 + 0 + 2 + 1 = -5$$

To Decimal

<u>Example</u>: Consider 1023, signed hexadecimal number (base-8 number)



Most Significant Digit (MSD)

$(1\ 0\ 2\ 3)_8$

Least Significant Digit (LSD)

$8^3 \quad 8^2 \quad 8^1 \quad 8^0$ ← *weights*

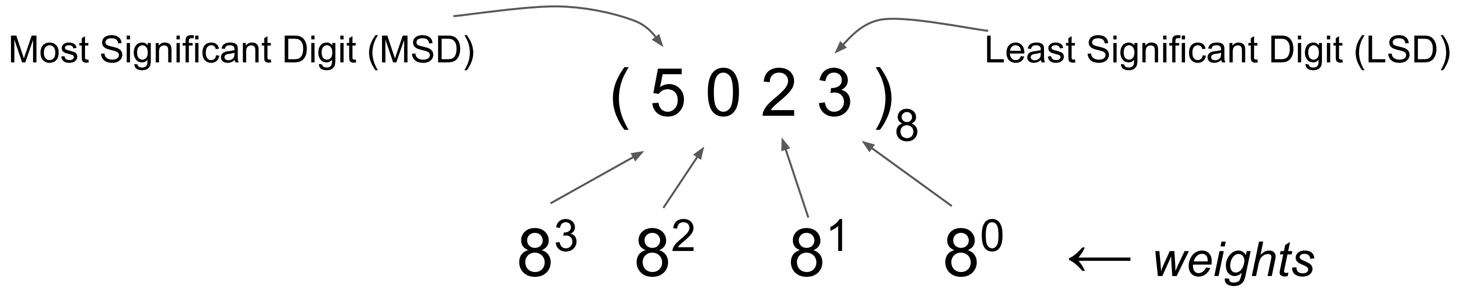
- starting with lowest weight of 8^0, moving from right-to-left
- the **weight** of each digit, increases by a factor of 8
- The decimal conversion is the sum of all the weights multiplied by the digits. If the most significant digit is greater than or equal to half the base, subtract the base raised to the power of the number of digits.

$$(1023)_8 = (1 \times (8^3)) + (0 \times 8^2) + (2 \times 8^1) + (3 \times 8^0)$$

$$(1023)_8 = 512 + 0 + 16 + 3 = 531$$

To Decimal

Example: Consider 5023, signed hexadecimal number (base-8 number)



Most Significant Digit (MSD)

Least Significant Digit (LSD)

$(\ 5\ 0\ 2\ 3\ )_8$

8^3 8^2 8^1 8^0 ← *weights*

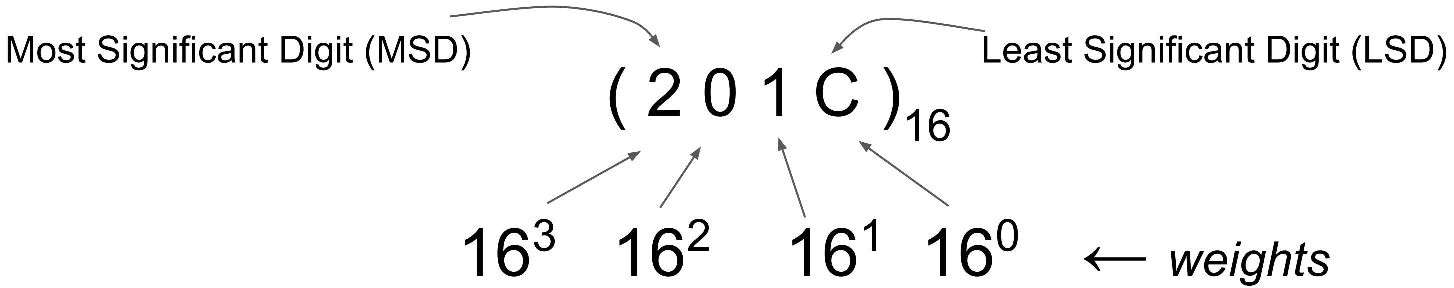
- starting with lowest weight of 8^0, moving from right-to-left
- the **weight** of each digit, increases by a factor of 8
- The decimal conversion is the sum of all the weights multiplied by the digits. If the most significant digit is greater than or equal to half the base, subtract the base raised to the power of the number of digits.

$$(5023)_8 = -8^4 + (5 \times 8^3) + (0 \times 8^2) + (2 \times 8^1) + (3 \times 8^0)$$

$$(5023)_8 = -4{,}096 + 2{,}560 + 0 + 16 + 3 = -2{,}541$$

To Decimal

<u>Example</u>: Consider 201C, signed hexadecimal number (base-16 number)



Most Significant Digit (MSD)

Least Significant Digit (LSD)

$$(2\ 0\ 1\ C)_{16}$$

$$16^{3} \quad 16^{2} \quad 16^{1} \quad 16^{0}$$

← *weights*

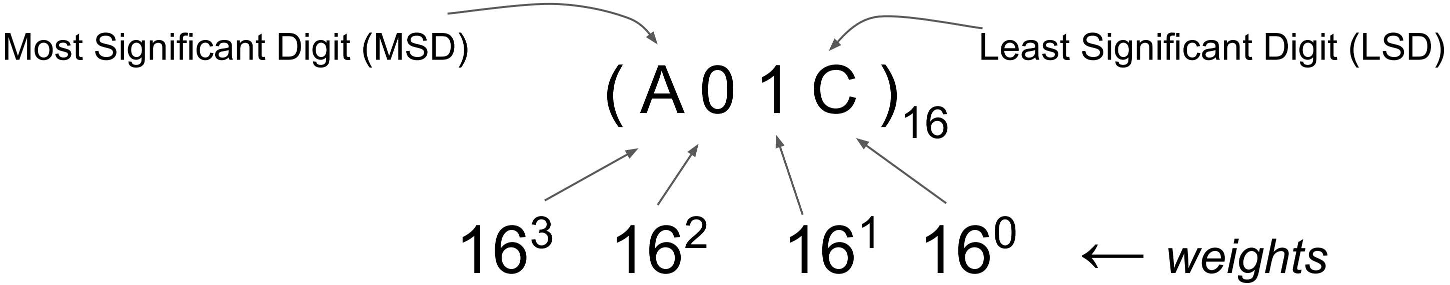
- starting with lowest weight of 16^{0}, moving from right-to-left
- the **weight** of each digit, increases by a factor of 16
- The decimal conversion is the sum of all the weights multiplied by the digits. If the most significant digit is greater than or equal to half the base, subtract the base raised to the power of the number of digits.

$$(201C)_{16} = (2 \times (16^{3})) + (0 \times 16^{2}) + (1 \times 16^{1}) + (C \times 16^{0})$$

$$(201C)_{16} = 8{,}192 + 0 + 16 + 12 = 8{,}220$$

To Decimal

Example: Consider A01C, signed hexadecimal number (base-16 number)



Most Significant Digit (MSD)

Least Significant Digit (LSD)

$(A\ 0\ 1\ C)_{16}$

16^3 16^2 16^1 16^0 ← *weights*

- starting with lowest weight of 16^0, moving from right-to-left
- the **weight** of each digit, increases by a factor of 16
- The decimal conversion is the sum of all the weights multiplied by the digits. If the most significant digit is greater than or equal to half the base, subtract the base raised to the power of the number of digits.

$(A01C)_{16} = -16^4 + (A \times (16^3)) + (0 \times 16^2) + (1 \times 16^1) + (C \times 16^0)$

$(A01C)_{16} = -65,656 + 40,960 + 0 + 16 + 12 = -24,548$

Decimal to Signed Binary $(25)_{10} \rightarrow (?)_2$

Repeated Division

Divide the given decimal number by base=2.

Write the remainder after each division until a quotient of zero is obtained.

Division	Quotient	Remainder	
25/2	12	1	**LSB**
12/2	6	0	
6/2	3	0	
3/2	1	1	
1/2	0	1	
-	-	0	**MSB**

We read the answer from the remainder column, bottom-up, adding 0 as the sign bit. If the decimal number is negative. Find the two’s complement to get the final answer.

011001_2

Decimal to Signed Binary $(-25)_{10} \rightarrow (?)_2$

Repeated Division

Divide the given decimal number by base=2.

Write the remainder after each division until a quotient of zero is obtained.

Division	Quotient	Remainder	
25/2	12	1	**LSB**
12/2	6	0	
6/2	3	0	
3/2	1	1	
1/2	0	1	
-	-	0	**MSB**

We read the answer from the remainder column, bottom-up, adding 0 as the sign bit. If the decimal number is negative. Find the two's complement to get the final answer.

$(011001)_2 = 100111_2$

answer = 100111_2

Decimal to Signed Octal $(2625)_{10} \to (?)_8$

Repeated Division

Divide the given decimal number by base=8.

Write the remainder after each division until a quotient of zero is obtained.

Division	Quotient	Remainder	
2625/8	328	1	**LSB**
328/8	41	0	
41/8	5	1	
5	0	5	
-	-	0	**MSB**

We read the answer from the remainder column, bottom-up. If the most significant digit is >= half the base, prepend a 0. If the decimal number is negative, find the two's complement to get the final answer.

05101_8

Decimal to Signed Octal $(-2625)_{10} \rightarrow (?)_8$

Repeated Division

Divide the given decimal number by base=8.

Write the remainder after each division until a quotient of zero is obtained.

Division	Quotient	Remainder	
2625/8	328	1	**LSB**
328/8	41	0	
41/8	5	1	
5	0	5	
-	-	0	**MSB**

We read the answer from the remainder column, bottom-up. If the most significant digit is >= half the base, prepend a 0. If the decimal number is negative, find the two's complement to get the final answer.

$(05101)_8 = 72677_8$

answer = 72677_{16}

Decimal to Signed Hexadecimal $(2625)_{10} \rightarrow (?)_{16}$

Repeated Division

Divide the given decimal number by base=16.

Write the remainder after each division until a quotient of zero is obtained.

Division	Quotient	Remainder	
2625/16	164	1	**LSB**
164/16	10	4	
10/16	0	A	
-	-	0	**MSB**

We read the answer from the remainder column, bottom-up. If the most significant digit is >= half the base, prepend a 0. If the decimal number is negative, find the two's complement to get the final answer.

$0A41_{16} = F5BF_{16}$

answer = $F5BF_{16}$

Decimal to Signed Hexadecimal $(-2625)_{10} \rightarrow (?)_{16}$

Repeated Division

Divide the given decimal number by base=16.

Write the remainder after each division until a quotient of zero is obtained.

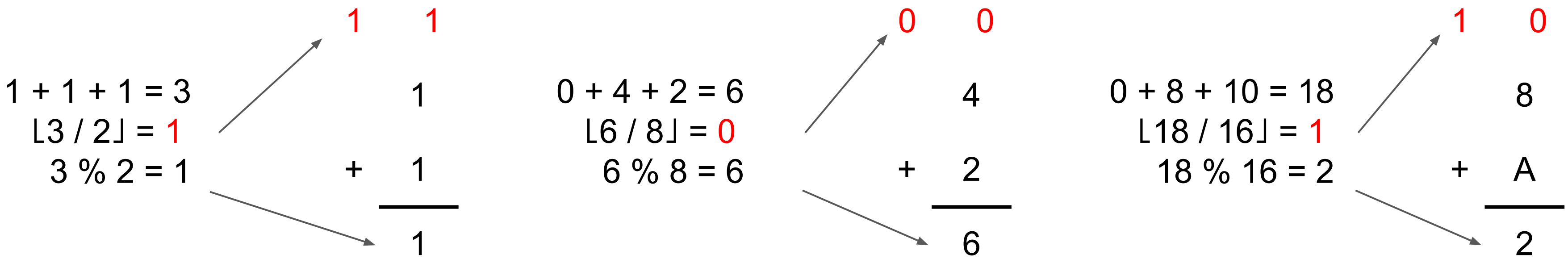
Division	Quotient	Remainder	
2625/16	164	1	**LSB**
164/16	10	4	
10/16	0	A	
-	-	0	**MSB**

We read the answer from the remainder column, bottom-up. If the most significant digit is >= half the base, prepend a 0. If the decimal number is negative, find the two's complement to get the final answer.

$0A41_{16}$

Signed Addition

When performing addition on two signed numbers, going from least significant digit (LSD) to most significant digit (MSD), we add the digit of each number at the current position plus any carry-over (0 or 1) from the previous addition. The remainder of this position sum becomes the digit for the current position of the larger sum. The carry-over for the next position's addition is the floor of the current position's sum divided by the base.



1 1

1 + 1 + 1 = 3
⌊3 / 2⌋ = 1
3 % 2 = 1

1
+ 1
1

0 0

0 + 4 + 2 = 6
⌊6 / 8⌋ = 0
6 % 8 = 6

4
+ 2
6

1 0

0 + 8 + 10 = 18
⌊18 / 16⌋ = 1
18 % 16 = 2

8
+ A
2

Signed Binary Addition

Overflow occurs when the result of an operation is too large to fit in allowed number of digits.

- **When adding two signed numbers of different signs, the sign of the sum is the correct sign.**
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Start at LSB

Example: 01010_2 + 11111110_2 (10 + (-2) = 8)

carry:	1	1	1	1	1	1	1	0	0
		0	0	0	0	1	0	1	0
+		1	1	1	1	1	1	1	0
		0	0	0	0	1	0	0	0

Signed Binary Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- **When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.**
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Example: 1111001_2 + 111110_2 (-7 + (-2)) = -9

Start at LSB

carry:	1	1	1	1	1	0	0	0	0
		1	1	1	1	1	0	0	1
+		1	1	1	1	1	1	1	0
		1	1	1	1	0	1	1	1

Signed Binary Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- **If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.**
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Start at LSB

Example: $10000001_2 + 10000001_2$ (-127 + (-127)) = -254)

carry:	1	0	0	0	0	0	0	1	0
		1	0	0	0	0	0	0	1
+		1	0	0	0	0	0	0	1
	1	0	0	0	0	0	0	1	0

Signed Binary Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- **When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.**
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Example: 00010001$_2$ + 00011$_2$ (17 + 3 = 20)

Start at LSB

carry:	0	0	0	0	0	0	1	1	0
		0	0	0	1	0	0	0	1
+		0	0	0	0	0	0	1	1
		0	0	0	1	0	1	0	0

Signed Binary Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- **If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.**

Start at LSB

Example: 01000001$_2$ + 01000001$_2$ (65 + 65 = 130)

carry:	0	1	0	0	0	0	0	1	0
		0	1	0	0	0	0	0	1
+		0	1	0	0	0	0	0	1
	0	1	0	0	0	0	0	1	0

Signed Octal Addition

Overflow occurs when the result of an operation is too large to fit in allowed number of digits.

- **When adding two signed numbers of different signs, the sign of the sum is the correct sign.**
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Start at LSB

Example: $23400751_8 + 73476230_8$

carry:	1	0	1	0	0	1	1	0	0
		2	3	4	0	0	7	5	1
+		7	3	4	7	6	2	3	0
		1	7	0	7	7	2	0	1

Signed Octal Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- **When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.**
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Start at LSB

Example: 77771125_8 + 57210364_8

carry:	1	1	1	1	0	0	1	1	0
		7	7	7	7	1	1	2	5
+		5	7	2	1	0	3	6	4
		5	7	2	0	1	5	1	1

Signed Octal Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- **If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.**
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Example: $50000001_8 + 60000001_8$

Start at LSB

carry:	1	0	0	0	0	0	0	0	0
		5	0	0	0	0	0	0	1
+		6	0	0	0	0	0	0	1
	7	3	0	0	0	0	0	0	2

Signed Octal Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- **When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.**
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Example: $27210406_8 + 06104413_8$

Start at LSB

carry:	0	1	0	0	0	1	0	1	0
		2	7	2	1	0	4	0	6
+		0	6	1	0	4	4	1	3
		3	5	3	1	5	0	2	1

Signed Octal Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- **If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.**

Start at LSB

Example: $31000001_8 + 11000001_8$

carry:	0	0	0	0	0	0	0	0	0
		3	1	0	0	0	0	0	1
+		1	1	0	0	0	0	0	1
	0	4	2	0	0	0	0	0	2

Signed Hexadecimal Addition

Overflow occurs when the result of an operation is too large to fit in allowed number of digits.

- **When adding two signed numbers of different signs, the sign of the sum is the correct sign.**
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Start at LSB

Example: $23A00F51_{16}$ + $FA478C30_{16}$

carry:	1	0	0	0	0	1	0	0	0
		2	3	A	0	0	F	5	1
+		F	A	4	7	8	C	3	0
		1	D	E	7	9	B	8	1

Signed Hexadecimal Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- **When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.**
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Start at LSB

Example: FFFFAA1A_{16} + 989ACD0B_{16}

carry:	1	1	1	1	1	1	0	1	0
		F	F	F	F	A	A	1	A
	+	9	8	9	A	C	D	0	B
		9	8	9	A	7	7	2	5

Signed Hexadecimal Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- **If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.**
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Start at LSB

Example: 90000001_{16} + $A0000001_{16}$

carry:	1	0	0	0	0	0	0	0	0
		9	0	0	0	0	0	0	1
+		A	0	0	0	0	0	0	1
	F	3	0	0	0	0	0	0	2

Signed Hexadecimal Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- **When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.**
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.

Start at LSB

Example: $37A10B01_{16} + 2CE08011_{16}$

carry:	0	1	1	0	0	0	0	0	0
		3	7	A	1	0	B	0	1
+		2	C	E	0	8	0	1	1
		6	4	8	1	8	B	1	2

Signed Hexadecimal Addition

Overflow occurs when the result of a binary operation is too large to fit in allowed number of bits.

- When adding two signed numbers of different signs, the sign of the sum is the correct sign.
- When adding two signed numbers of the same sign, if the sign of the sum matches the sign of the two numbers, no need for further action.
- If the two numbers are negative and the MSD of the sum is less than half of the base, the sum must be padded with a negative sign. The base minus 1.
- **If the two numbers are positive and the MSD of the sum is greater than or equal to half the base, the sum must be padded with a positive sign of 0.**

Start at LSB

Example: $41000001_{16} + 71000001_{16}$

carry:	0	0	0	0	0	0	0	0	0
		4	1	0	0	0	0	0	1
+		7	1	0	0	0	0	0	1
	0	B	2	0	0	0	0	0	2

Signed Subtraction

For signed subtraction, find the two's complement of the second number (subtrahend) and add it to the first number (minuend).

Difference = minuend - subtrahend

= minuend + (-subtrahend)

$01010_2 – 11111110_2 = 01010_2 + 00000010_2 = 01100_2$

$1111001_2 – 111110_2 = 1111001_2 + 000010_2 = 10111_2$

$10000001_2 – 10000001_2 = 10000001_2 + 01111111 = 0_2$

$00010001_2 – 00011_2 = 00010001_2 + 11101_2 = 01110_2$

$01000001_2 – 01000001_2 = 01000001_2 + 10111111 = 0_2$

$23400751_8 – 73476230_8 = 23400751_8 + 04301550_8 = 27702521_8$

$77771125_8 – 57210364_8 = 77771125_8 + 20567414_8 = 20560541_8$

$50000001_8 – 60000001_8 = 50000001_8 + 17777777_8 = 70000000_8$

$27210406_8 – 06104413_8 = 27210406_8 + 71673365_8 = 21103773_8$

$31000001_8 – 11000001_8 = 31000001_8 + 66777777_8 = 20000000_8$

$23A00F51_{16} – FA478C30_{16} = 23A00F51_{16} + 5B873D0_{16} = 29588321_{16}$

$FFFFAA1A_{16} – 989ACD0B_{16} = FFFFAA1A_{16} + 676532F5_{16} = 6764DD0F_{16}$

$90000001_{16} – A0000001_{16} = 90000001_{16} + 5FFFFFFF_{16} = F0000000_{16}$

$37A10B01_{16} – 2CE08011_{16} = 37A10B01_{16} + D31F7FEF_{16} = 0AC08AF0_{16}$

41000001 – 71000001 = 41000001 + 8FFFFFFF = D0000000