BINARY REPRESENTATION

Representations of Integers

- In the modern world, we use decimal, or base 10, notation to represent integers.

- We can represent numbers using any base b, where b is a positive integer greater than 1.

Base 10

- When we write 965, this can be translated as:
  - $9 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0$

Base 10

- When we write 965, this can be translated as:
  - $9 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0$
- $n = a_k b^k + a_{k-1} b^{k-1} + \ldots. + a_1 b + a_0$

Base b

- **Theorem:** Let b be a positive integer greater than 1. Then if n is a positive integer, it can be expressed uniquely in the form:

$$n = a_k b^k + a_{k-1} b^{k-1} + \ldots. + a_1 b + a_0$$

where k is a nonnegative integer and $a_0, a_1, \ldots. a_k$ are nonnegative integers less than b.

- This representation of n is called the base b expansion of n and can be denoted by $(a_k a_{k-1} \ldots. a_1 a_0)_b$.

- We usually omit the subscript 10 for base 10 expansions.

Binary Expansions

- Computers represent integers and do arithmetic with binary (base 2) expansions of integers. In these expansions, the only digits used are $0 \text{ and } 1$.

Binary Expansions

- **Example**: What is the decimal expansion of the integer that has $(11011)_2$ as its binary expansion?
- **Solution**:

$(11011)_2$

$= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

$= 16 + 8 + 0 + 2 + 1$

$= 27.$

Binary Expansions

- **Example**: What is the decimal expansion of the integer that has $(1\ 0101\ 1111)_2$ as its binary expansion?
- **Solution**:

$$(1\ 0101\ 1111)_2$$
$$= 1{\cdot}2^8 + 0{\cdot}2^7 + 1{\cdot}2^6 + 0{\cdot}2^5 + 1{\cdot}2^4 + 1{\cdot}2^3 + 1{\cdot}2^2 + 1{\cdot}2^1 + 1{\cdot}2^0$$
$$= 256 + 0 + 64 + 0 + 16 + 8 + 4 + 2 + 1$$
$$= 351.$$

Base Conversion

To construct the base b expansion of an integer n:

- Divide n by b to obtain a quotient and remainder.

  $n = bq_0 + a_0 \quad 0 \leq a_0 < b$

Base Conversion

To construct the base b expansion of an integer n:

- Divide n by b to obtain a quotient and remainder.

  $n = bq_0 + a_0 \quad 0 \leq a_0 < b$

- The remainder, a_0, is the rightmost digit in the base b expansion of n.

Base Conversion

To construct the base b expansion of an integer n:

- Divide n by b to obtain a quotient and remainder.

  $n = bq_0 + a_0 \quad 0 \le a_0 < b$

- The remainder, a_0, is the rightmost digit in the base b expansion of n.
- Next, divide q_0 by b.

  $q_0 = bq_1 + a_1 \quad 0 \le a_1 < b$

Base Conversion

To construct the base b expansion of an integer n:

- Divide n by b to obtain a quotient and remainder.

  $n = bq_0 + a_0 \quad 0 \le a_0 < b$

- The remainder, a_0, is the rightmost digit in the base b expansion of n.
- Next, divide q_0 by b.

  $q_0 = bq_1 + a_1 \quad 0 \le a_1 < b$

- The remainder, a_1, is the second digit from the right in the base b expansion of n.

Base Conversion

To construct the base b expansion of an integer n:

- Divide n by b to obtain a quotient and remainder.

  $n = bq_0 + a_0 \quad 0 \leq a_0 < b$
- The remainder, a_0, is the rightmost digit in the base b expansion of n.
- Next, divide q_0 by b.

  $q_0 = bq_1 + a_1 \quad 0 \leq a_1 < b$
- The remainder, a_1, is the second digit from the right in the base b expansion of n.
- Continue by successively dividing the quotients by b, obtaining the additional base b digits as the remainder. The process terminates when the quotient is 0.

Base Conversion

- **Example**: Find the binary expansion of $(19)_{10}$
- **Solution**: Successively dividing by 2 gives:
  - 19 = 2 * 9 + 1
  - 9 = 2 * 4 + 1
  - 4 = 2 * 2 + 0
  - 2 = 2 * 1 + 0
  - 1 = 2 * 0 + 1

Base Conversion

- **Example**: Find the binary expansion of $(19)_{10}$
- **Solution**: Successively dividing by 2 gives:
  - 19 = 2 * 9 + 1
  - 9 = 2 * 4 + 1
  - 4 = 2 * 2 + 0
  - 2 = 2 * 1 + 0
  - 1 = 2 * 0 + 1

- The remainders are the digits: read from bottom to top to become the binary number from left to right
- $(10011)_2$.

Base Conversion

- **Example**: Find the binary expansion of $(12345)_{10}$
- **Solution**: Successively dividing by 2 gives:
  - 12345 = 2 * 6172 + 1
  - 6172 = 2 * 3086 + 0
  - 3086 = 2 * 1543 + 0
  - 1543 = 2 * 771 + 1
  - 771 = 2 * 385 + 1
  - 385 = 2 * 192 + 1
  - 192 = 2 * 96 + 0
  - 96 = 2 * 48 + 0
  - 48 = 2 * 24 + 0
  - 24 = 2 * 12 + 0
  - 12 = 2 * 6 + 0
  - 6 = 2 * 3 + 0
  - 3 = 2 * 1 + 1
  - 1 = 2 * 0 + 1
- $(12345)_{10} = (11\ 0000\ 0011\ 1001)_2$

List of Binary Numbers

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

- These are 4-bit unsigned binary numbers.

Binary Addition

- Binary number can be added the same way decimal numbers are added:

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 2 = 5

	0	0	1	1
+	0	0	1	0

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 2 = 5

	0	0	1	1
+	0	0	1	0
				1

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 2 = 5

		1		
	0	0	1	1
+	0	0	1	0
			0	1

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 2 = 5

		1		
	0	0	1	1
+	0	0	1	0
		1	0	1

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 2 = 5

	0	0	1	1
+	0	0	1	0
	0	1	0	1

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 3 = 6

	0	0	1	1
+	0	0	1	1

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 3 = 6

			1	
	0	0	1	1
+	0	0	1	1
				0

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 3 = 6

		1	1	
	0	0	1	1
+	0	0	1	1
			1	0

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 3 = 6

		1	1	
	0	0	1	1
+	0	0	1	1
		1	1	0

Binary Addition

- Binary number can be added the same way decimal numbers are added:

- 3 + 3 = 6

	0	0	1	1
+	0	0	1	1
	0	1	1	0

Binary Subtraction

- How do we subtract numbers?
- 7 – 6 = 7 + (-6) = 1
- We need a way to represent negative numbers

Signed Binary Numbers

- To represent a negative number, we use a sign bit.
- The sign bit is the most significant bit (MSB)
  - 1 represents a negative number
  - 0 represents a positive number

Signed Representations

- Sign-Magnitude: sign bit + absolute value
  - Positive 7 0111
  - Negative 7 1111
- 1's Complement: bitwise inverse
  - Positive 7 0111
  - Negative 7 1000
- Disadvantages
  - Two representations for 0: +0, -0
  - Arithmetic is more complex
- Advantages
  - The magnitude (absolute value) of the smallest number is the same as the largest number

2’s Complement Representation

- To represent a **negative** value: do a bitwise inverse, then add 1

- To determine -7:
  - 7 0111
  - Bitwise inverse 1000
    Add 1 1001

- 2’s Complement is the standard representation for negative numbers.

Binary Numbers

List of Binary Numbers in 2's Complement Representation

Decimal	Binary	Decimal	Binary
0	0000	-1	1111
1	0001	-2	1110
2	0010	-3	1101
3	0011	-4	1100
4	0100	-5	1011
5	0101	-6	1010
6	0110	-7	1001
7	0111	-8	1000

4 bits gives us the following range:
Biggest 4-bit number: 7
Smallest 4-bit number: -8

Binary Subtraction

- 2’s complement binary numbers are subtracted by adding their negative equivalent.

- 7 – 6 = 7 + (-6) = 1

	0	1	1	1
+	1	0	1	0

Binary Subtraction

- Binary numbers are subtracted by adding their 2’s complement equivalent.

- 7 – 6 = 7 + (-6) = 1

	0	1	1	1
+	1	0	1	0
				1

Binary Subtraction

- Binary numbers are subtracted by adding their 2's complement equivalent.

- 7 – 6 = 7 + (-6) = 1

		1		
	0	1	1	1
+	1	0	1	0
			0	1

Binary Subtraction

- Binary numbers are subtracted by adding their 2’s complement equivalent.

- 7 – 6 = 7 + (-6) = 1

	1	1		
	0	1	1	1
+	1	0	1	0
		0	0	1

Binary Subtraction

- Binary numbers are subtracted by adding their 2's complement equivalent.

- 7 – 6 = 7 + (-6) = 1

1	1	1		
	0	1	1	1
+	1	0	1	0
	0	0	0	1

Binary Subtraction

- Binary numbers are subtracted by adding their 2's complement equivalent.

- 7 – 6 = 7 + (-6) = 1

1				
	0	1	1	1
+	1	0	1	0
	0	0	0	1

- The extra 1, called the carry-out, is ignored

Overflow

- Overflow occurs when the result of an arithmetic operation is too large or too small to represent.
  - In our 4-bit examples, that would occur if the result is less than -8 or greater than 7.

Overflow

- Overflow occurs when adding whenever:
  - 2 positive numbers are added and the result is negative
  - 2 negative numbers are added and the result is positive

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 7 – 6 = 7 + (-6) = 1

Carry Out   Carry In

1	1	1		
	0	1	1	1
+	1	0	1	0
	0	0	0	1

Most Significant Bit

Since the Carry in of the MSB is equal to the carry out, no overflow has occurred. The carry out is discarded and “0001” is the correct result.

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 7 + 3 = 10

	0	1	1	1
+	0	0	1	1

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 7 + 3 = 10

			1	
	0	1	1	1
+	0	0	1	1
				0

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 7 + 3 = 10

		1	1	
	0	1	1	1
+	0	0	1	1
			1	0

Overflow

- An easy way to spot overflow is to compare the "Carry in" of the MSB to the "Carry out". If they are different, then overflow has occurred.

- 7 + 3 = 10

	1	1	1	
	0	1	1	1
+	0	0	1	1
		0	1	0

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 7 + 3 = 10

0	1	1	1	
	0	1	1	1
+	0	0	1	1
	1	0	1	0

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 7 + 3 = 10

Carry Out Carry In

0	1	1	1	
	0	1	1	1
+	0	0	1	1
	1	0	1	0

Most Significant Bit

Since the Carry in of the MSB is not equal to the carry out, an overflow has occurred.

The result calculated (-6) is clearly incorrect.

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- -4 - 5 = -9

	1	1	0	0
+	1	0	1	1

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- -4 - 5 = -9

	1	1	0	0
+	1	0	1	1
				1

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- -4 - 5 = -9

	1	1	0	0
+	1	0	1	1
			1	1

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- -4 - 5 = -9

	1	1	0	0
+	1	0	1	1
		1	1	1

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- -4 - 5 = -9

1				
	1	1	0	0
+	1	0	1	1
	0	1	1	1

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- -4 - 5 = -9

Carry Out → Carry In

1	0			
	1	1	0	0
+	1	0	1	1
	0	1	1	1

Most Significant Bit

Since the Carry in of the MSB is not equal to the carry out, an overflow has occurred.

The result calculated (7) is clearly incorrect.

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 3 + (-5) = -2

	0	0	1	1
+	1	0	1	1

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 3 + (-5) = -2

			1	
	0	0	1	1
+	1	0	1	1
				0

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 3 + (-5) = -2

		1	1	
	0	0	1	1
+	1	0	1	1
			1	0

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 3 + (-5) = -2

	0	1	1	
	0	0	1	1
+	1	0	1	1
		1	1	0

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 3 + (-5) = -2

0	0	1	1	
	0	0	1	1
+	1	0	1	1
	1	1	1	0

Overflow

- An easy way to spot overflow is to compare the “Carry in” of the MSB to the “Carry out”. If they are different, then overflow has occurred.

- 3 + (-5) = -2

Carry Out → Carry In

0	0	1	1	
	0	0	1	1
+	1	0	1	1
	1	1	1	0

Most Significant Bit

Since the Carry into of the MSB is equal to the carry out, no overflow has occurred. The extra zero is discarded and “1110” is the correct result.