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:
  - 965
  - =900+60+5
  - =9.100+6.10+5.1

Digit

  - $=9*\mathbf{10^2} + 6*\mathbf{10^1} + \mathbf{5}*\mathbf{10^0}$ where $(10^n)=(\underbrace{10.10.10 \dots .10.10.10}_{n})$

Weight

  - {0,1,…,9} is called **digit set**.
  - 10 is base.
  - The numbers 9,6 and 5 in 965 are called **digits**.

Base 10

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

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

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.

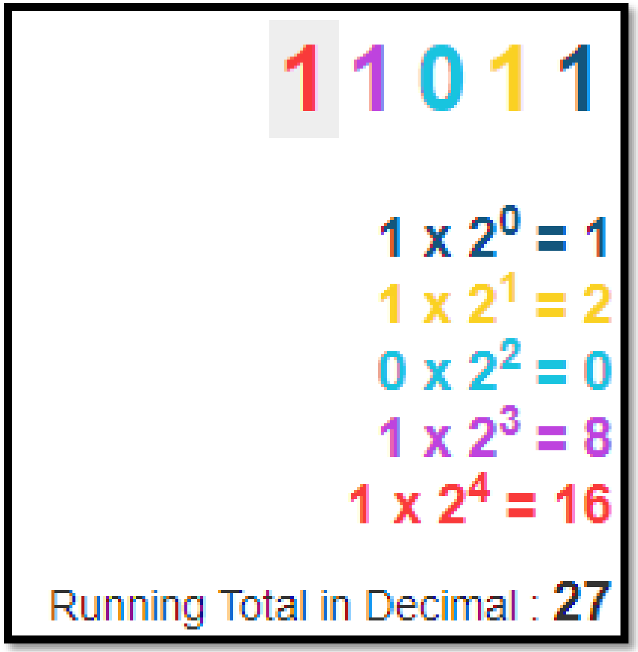
Binary Expansions

- Computers represent integers and do arithmetic with binary **(base 2)** expansions of integers. In these expansions, the only digits used are **0 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**:

$(1\ 0101\ 1111)_2$

$= 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

$= 256 + 0 + 64 + 0 + 16 + 8 + 4 + 2 + 1$

$=351.$

[Online Practice: Binary to Decimal]

Base Conversion

To construct the base *b* expansion of an integer *n*:

- Divide *n* by *b* to obtain a **quotient (q_0)** and **remainder (a_0).**

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

Base Conversion

To construct the base *b* expansion of an integer *n*:

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

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

- The remainder, $\mathbf{a_0}$, 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.

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

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

  $q_0 = bq_1 + \mathbf{a_1} \quad 0 \leq a_1 < b$

Base Conversion

To construct the base *b* expansion of an integer *n*:

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

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

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

  $q_0 = bq_1 + \boldsymbol{a_1} \quad 0 \leq a_1 < b$

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

Base Conversion

To construct the base *b* expansion of an integer *n*:

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

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

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

  $q_0 = bq_1 + \mathbf{a_1} \quad 0 \leq a_1 < b$

- 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
- $(12345)_{10} = (11\ 0000\ 0011\ 1001)_2$

[Online Practice: Decimal to Binary]

List of Binary Numbers

Decimal	Binary
0	0000
1	000**1**
2	0010
3	0011
4	0100
5	0101
6	0110
7	0**111**

- These are 4-bit unsigned binary numbers.

Octal (base 8)

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

- **Digit Set in Octal:** {0,1,2,3,4,5,6,7}

- **Example**: Find the octal expansion of $(12345)_{10}$
- Solution:
- Convert decimal to binary
- $(12345)_{10}$ = $(11\ 0000\ 0011\ 1001)_2$
- Separate groups of 3 bits from right
- = **011000000111001**
- Find the equivalence of each digit from the table
- = $(30071)_8$

Hexadecimal (base 16)

Decimal	Binary	Decimal	Binary
0	000	8	1000
1	001	9	1001
2	010	10	1010
3	011	11	1011
4	100	12	1100
5	101	13	1101
6	110	14	1110
7	111	15	1111

- **Example**: Find the octal expansion of $(12345)_{10}$
- Solution:
- Convert decimal to binary
- $(12345)_{10} = (11000000111001)_2$
- Separate groups of 4 bits from right
- = **0011000000111001**
- Find the equivalence of each digit from the table
- = $(3039)_{16}$

- **Digit Set in Hexadecimal:** {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

Binary Addition

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

Binary Addition

- The process is that we line the two numbers up (one under the other),
- then, starting at the far right, add each column,
- recording the result and possible carry as we go.

- Here are the possibilities:

- 0 + 0 = 0
- 1 + 0 = 1
- 1 + 1 = 2 which is 10 in binary which is 0 with a carry of 1
- 1 + 1 + 1 (carry) = 3 which is 11 in binary which is 1 with a carry of 1

- The carry is involved whenever we have a result larger than 1 (which is the largest amount we may represent with a single binary digit).

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 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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

0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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

0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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

0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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

0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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

0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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

0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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

0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

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

0 + 0 = 0
1 + 0 = 1
1 + 1 = 10
1 + 1 + 1 =11

See this link for more practice on binary-arithmetic (Online)

Binary Subtraction

- How do we subtract numbers?
- 7 – 6 = 7 + (-6) = 1
- There are two approaches.

Binary Subtraction

Approach 1:

- Similar to binary addition, we will work through the numbers, column by column, starting on the far right.
- Instead of **carrying forward** however, we will **borrow backwards** (when necessary).
- Here are the possibilities:

- 0 - 0 = 0
- 1 - 0 = 1
- 1 - 1 = 0
- 0 - 1 (we can't do so we borrow 1 from the next column. This makes it 10 - 1 which is 1.)

See this link for more practice on binary-arithmetic (Online)

Binary Subtraction

- How do we subtract numbers?

- 31 – 13 =

$$\begin{array}{rr} 31 & 11111 \\ 13 & 1101\ - \\ \hline \end{array}$$

Binary Subtraction

- How do we subtract numbers?

- 31 – 13 =

$$
\begin{array}{lr}
31 & 11111 \\
13 & 1101\ - \\
\hline
 & 0
\end{array}
$$

Binary Subtraction

- How do we subtract numbers?

- 31 – 13 =

31	11111	
13	1101	-
	10	

Binary Subtraction

- How do we subtract numbers?

- 31 – 13 =

```
31      11111
13       1101 -
        ------
          010
```

Binary Subtraction

- How do we subtract numbers?

- 31 – 13 =

31	11111	
13	1101	-
	0010	

Binary Subtraction

- How do we subtract numbers?

- 31 – 13 = 18

$$
\begin{array}{rr}
31 & 11111 \\
13 & 1101\ - \\
\hline
18 & 10010
\end{array}
$$

See this link for more practice on binary-arithmetic (Online)

Binary Subtraction

- How do we subtract numbers?

- 7 – 6 = 7 + (-6) = 1

- **Approach 2:**
- We need a way to represent negative numbers
  - Sign Magnitude
  - 1's Complement
  - 2's Complement

Signed Binary Numbers

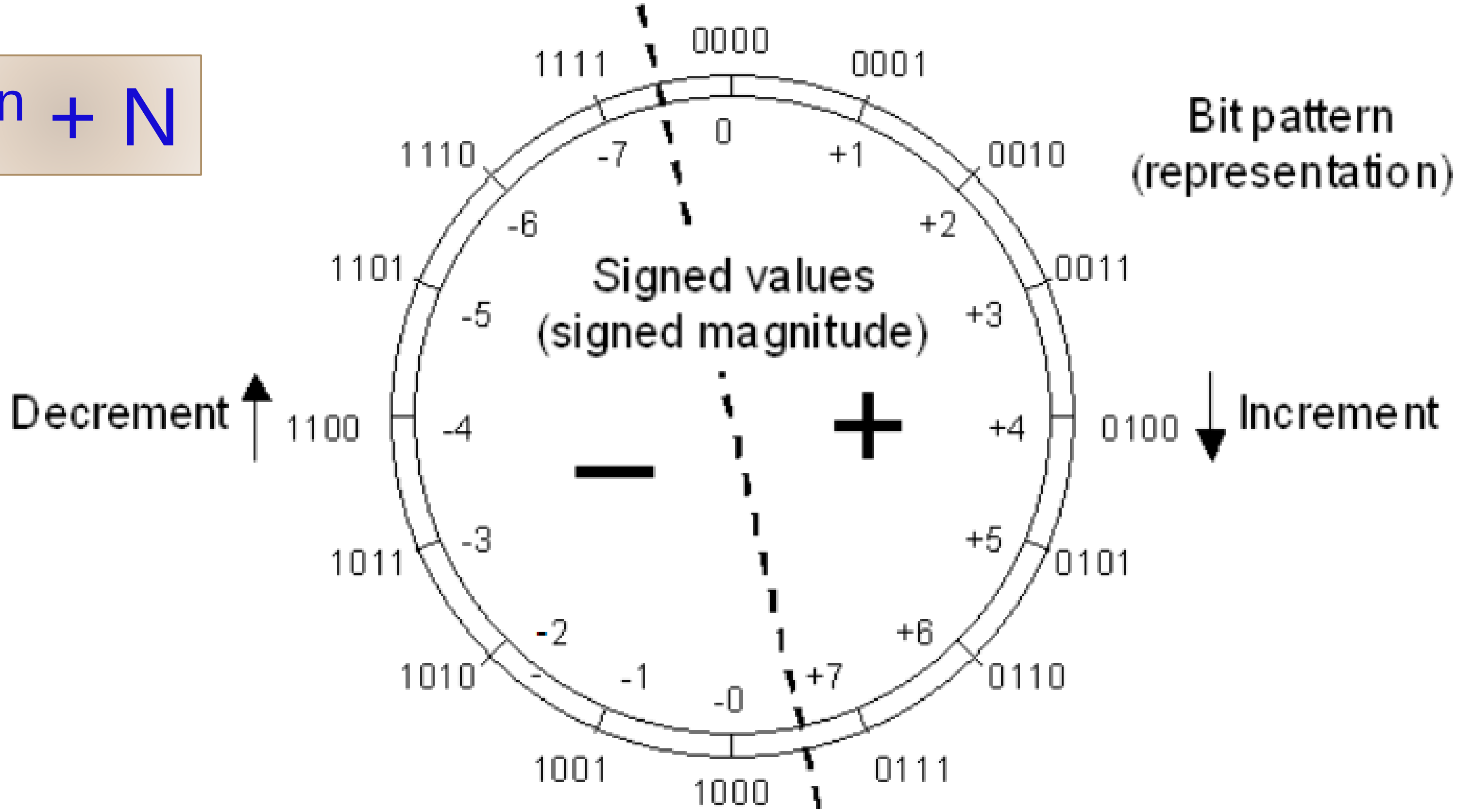
- To represent a negative number, we use a **sign bit**.
- The sign bit is the **most significant bit** (MSB)
  - **1** represents a **negative number**
  - **0** represents a **positive number**

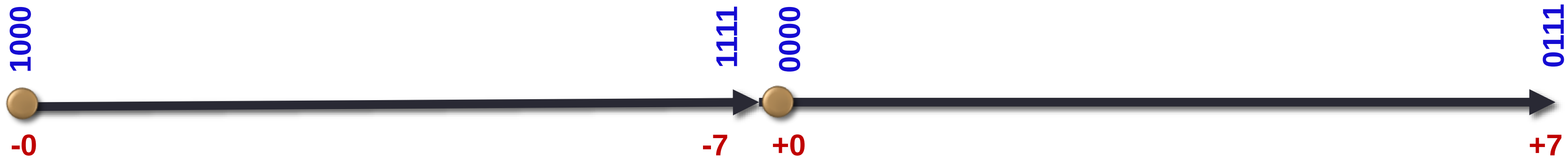
Signed Binary Numbers (Sign Magnitude)

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

Signed Magnitude Representation

$2^n + N$





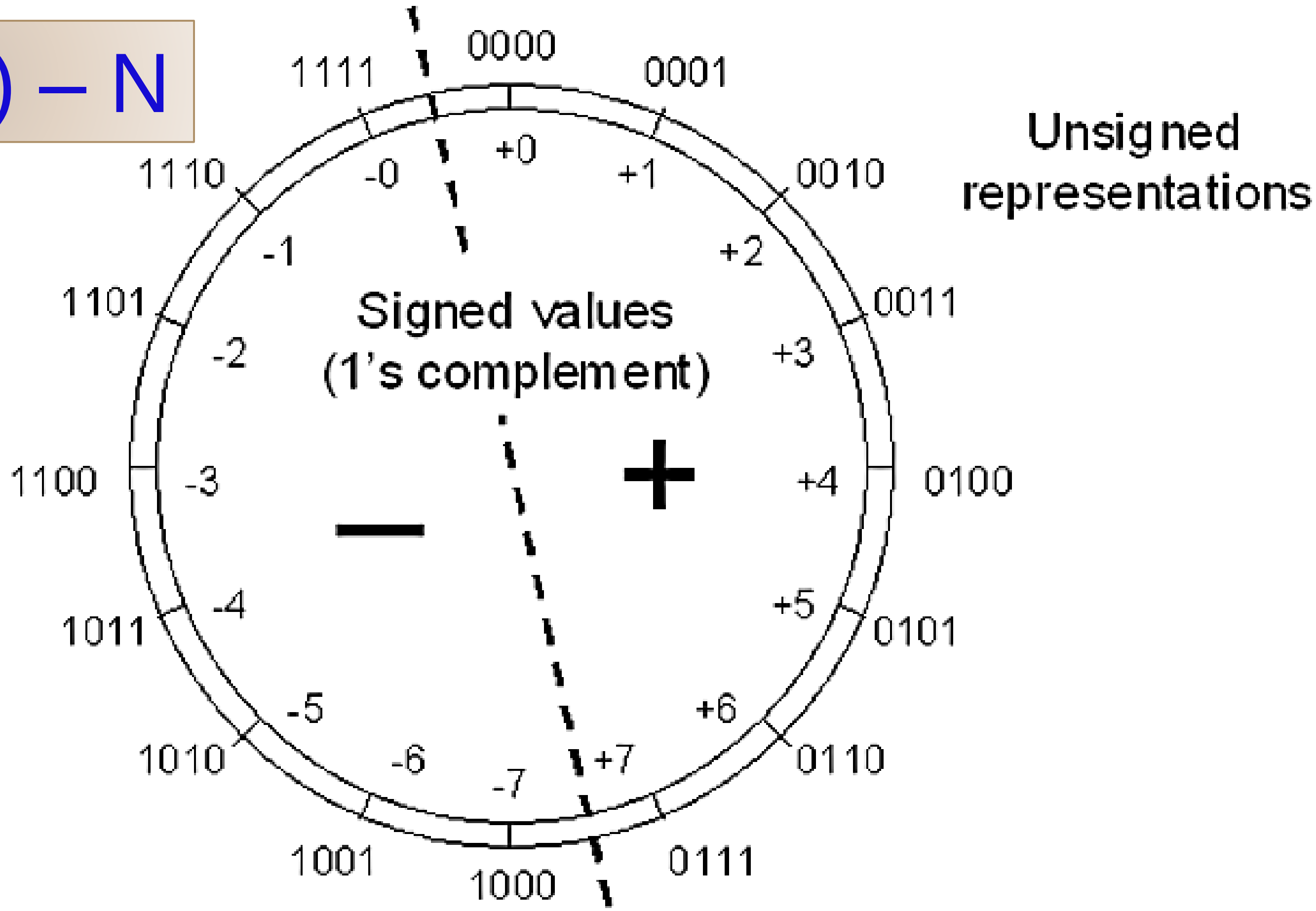
List of Binary Numbers (1's complement)

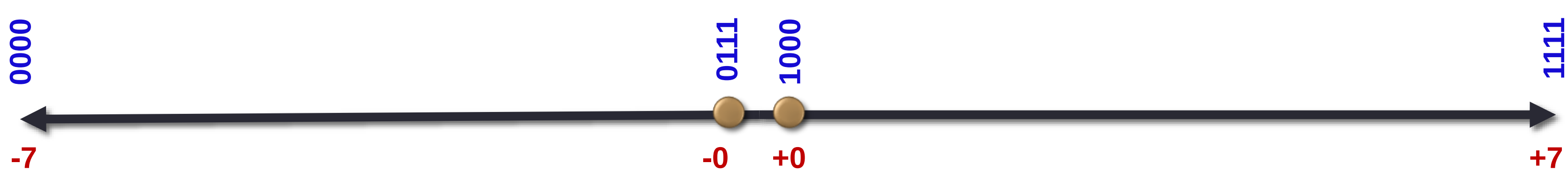
Decimal	Binary	Decimal	Binary
0	0000	-0	**1**111
1	0001	-1	**1**110
2	0010	-2	**1**101
3	0011	-3	**1**100
4	0100	-4	**1**011
5	0101	-5	**1**010
6	0110	-6	**1**001
7	0111	-7	**1**000

- Similar to sign magnitude the **most significant bit indicates the sign** of the number.
- For negative numbers, however, we **invert the bits** from what they would normally be.

1’s Complement Representation

$(2^{n+1} - 1) - N$





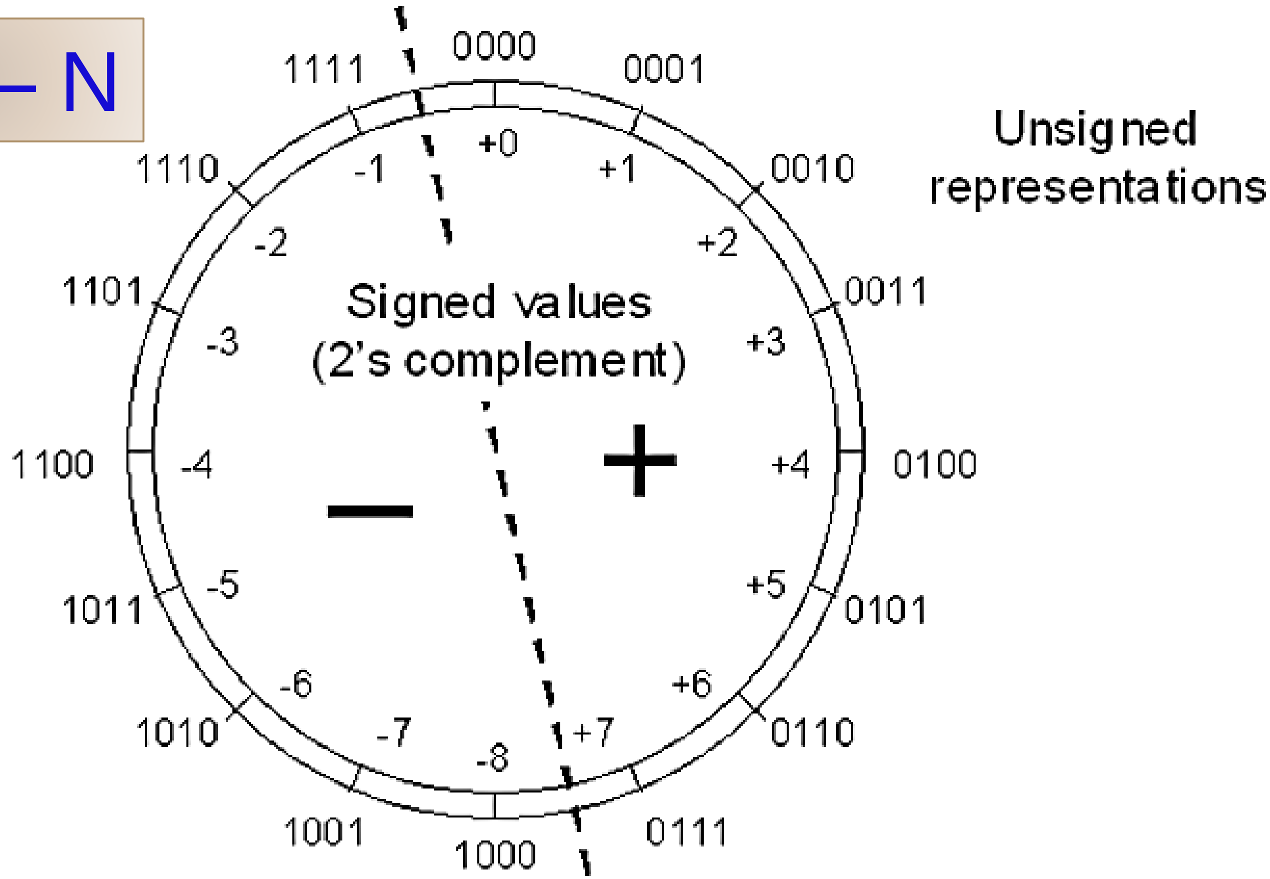
List of Binary Numbers (2's complement)

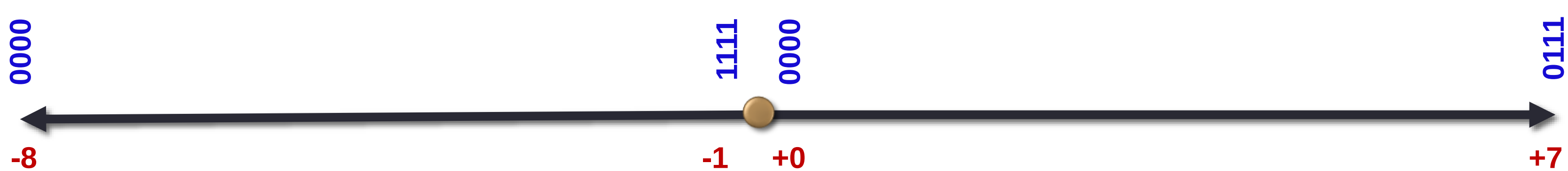
Decimal	Binary	Decimal	Binary
0	0000	-1	1111
1	0001	-2	1110
2	0010	-3	1101
3	0011	-4	1100
4	0100	-5	1011
5	0101	-6	1010
6	0110	-7	1001
7	0111	-8	1000

These are 4-bit signed numbers using the 2's complement representation of negative numbers
Biggest 4-bit number: 7
Smallest 4-bit number: -8

2’s Complement Representation

$2^{n+1} – N$





2’s Complement Representation

- To calculate: do a bitwise inverse, then add 1

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

2's Complement Representation

-48

48	Take positive of number
110000	Convert to Binary (unsigned)
00110000	Pad out to required number of bits
11001111	Invert the digits
11010000	Add 1, and you are done :)

2’s Complement Representation

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

- Alternative 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

Online Practice: negative numbers

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

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

- 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

- 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**

35	00100011
22	00010110 +
	1 1
57	00111001

Finished

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**

125	0 1 1 1 1 1 0 1
68	0 1 0 0 0 1 0 0 +
	1 1 1 1 1
193	1 1 0 0 0 0 0 1

Overflow

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**

-99 10011101

-115 10001101 +

1 1 1 1 1

-214 100101010

Overflow

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**

-49		1	1	0	0	1	1	1	1	
-31		1	1	1	0	0	0	0	1	+
	1	1			1	1	1	1		
-80	1	1	0	1	1	0	0	0	0	

Discard final bit

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.

Online practice: Negative numbers and overflow