

Binary Arithmetic

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

Announcement

- My Office Hours (Rice 310 Or Zoom: <https://virginia.zoom.us/j/3627787726>)
 - Monday: 1 PM - 2 PM
 - Wednesday: 1 PM - 3 PM
- TA Office Hours starting today
- Homework 1 available Friday, due September 15, 2025

Binary

2 digits: 0, 1

Try to turn 1100101_2 into base-10

1	1	0	0	1	0	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

$$64 + 32 + 4 + 1 = 101_{10}$$

Binary

method 2:

Any downsides to binary?

Turn 2130_{10} into base-2:

hint: find largest power of 2 and subtract

method 1:

2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	0	0	0	1	0	1	0	0	1	0

$$2130 - 2048 = 82 \quad 82 - 64 = 18 \quad 18 - 16 = 2$$

2	2130	
2	1065	0
2	532	1
2	266	0
2	133	0
2	66	1
2	33	0
2	16	1
2	8	0
2	4	0
2	2	0
2	1	0
	0	1

read from bottom

Long Numbers

How do we deal with numbers too long to read?

Long Numbers

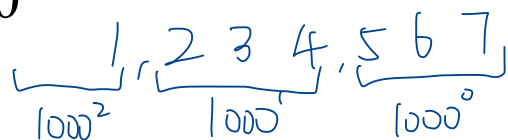
How do we deal with numbers too long to read?

- Group them by 3 (right to left)

Long Numbers

How do we deal with numbers too long to read?

- Group them by 3 (right to left)
- In decimal, use commas: ,
- Numbers between commas: 000 – 999
- Effectively base-1000


$$\underbrace{1}_{1000^2}, \underbrace{234}_{1000^1}, \underbrace{567}_{1000^0}$$

$$\Rightarrow 1 \times 1000^2 + 234 \times 1000^1 + 567 \times 1000^0$$

Long Numbers in Binary - Readability

- Typical to group by 3 or 4 bits
- No need for commas *Why?*

In binary, grouping by 3 or 4 already gives us a fix-size block that maps to another digit system.

100001010010

Long Numbers in Binary - Readability

- Typical to group by 3 or 4 bits
- No need for commas *Why?*
- We can use a separate symbol per group
- How many do we need for groups of 3? $2^3 = 8$ (000 - 111)

binary	decimal	octal
000	0	0
001	1	1
010	2	2
011	3	3
100	4	4
101	5	5
110	6	6
111	7	7

100001010010

Long Numbers in Binary - Readability

- Typical to group by 3 or 4 bits
- No need for commas *Why?*
- We can use a separate symbol per group
- How many do we need for groups of 3?
- Turn each group into decimal representation

100001010010

Long Numbers in Binary - Readability

- Typical to group by 3 or 4 bits
- No need for commas *Why?*
- We can use a separate symbol per group
- How many do we need for groups of 3?
- Turn each group into decimal representation
- Converts binary to **octal**

100001010010
4 1 2 2₈

Long Numbers in Binary - Readability

- Groups of 4 more common
- How many symbols do we need for groups of 4? $2^4 = 16$

100001010010

Long Numbers in Binary - Readability

- Groups of 4 more common
- How many symbols do we need for groups of 4?
- Converts binary to **hexadecimal**
- Base-16 is very common in computing

100001010010

Hexadecimal

Need more than 10 digits. What next?

1110

binary	decimal	hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

2 exercises:

1. hexadecimal to binary:

5b42 \Rightarrow

5	b	4	2
0101	1011	0100	0010

$$5 \times 16^3 + \cancel{11} \times 16^2 + 4 \times 16 + 2 \times 1$$

$$= 5 \times 4096 + 11 \times 256 + 4 \times 16 + 2 \times 1$$

$$= 20480 + 2816 + 64 + 2$$

$$= 23362.$$

2. binary to hexadecimal

1 010 1111 0010

<u>0001</u>	<u>1010</u>	<u>1111</u>	<u>0010</u>
1	A	F	2

Hexadecimal Exercise

Consider the following hexadecimal number:

852dab1e



check the right-most bit!

Is it even or odd?

Using Different Bases in Code

	^{like C} Old Languages	^{like Java, Python, C#} New Languages
binary	no way	0b 01010011
octal	073	0o 735 (zero and a letter 'o')
decimal	2130	2130
hexadecimal	0x 3af	0x 3af

Binary Addition

$$01101011 + 01100101$$

$$\begin{array}{r} 01101011 \\ + 01100101 \\ \hline 11010000 \end{array}$$

$$11101011 + 11100101$$

$$\begin{array}{r} 11101011 \\ + 11100101 \\ \hline 11110000 \end{array} \leftarrow \text{overflow!}$$

Range for an 8-bit
number:

0000 0000 \rightarrow 1111 1111

0 255

0 $2^8 - 1$

Binary Subtraction

$$01111011 - 01100101$$

$$\begin{array}{r} 01111011 \\ - 01100101 \\ \hline 00010110 \end{array}$$

Finally, Numbers!

Storing Integers

- Use binary representation of decimal numbers
- Usually have a limited number of bits (ex: 32, 64)
 - Depending on language
 - Depending on hardware

Finally, Numbers!

Storing Integers

- Use binary representation of decimal numbers
- Usually have a limited number of bits (ex: 32, 64)
 - Depending on language
 - Depending on hardware
- Is there something missing?

Negative Integers

Representing negative integers

- Can we use the minus sign?

Negative Integers

Representing negative integers

- Can we use the minus sign?
- In binary we only have 2 symbols, must do something else!

Two's Complement

The scheme is called Two's Complement

Why do we need Two's Complement?

- We want the computer to represent both positive and negative numbers.
- And we want addition and subtraction to use the same hardware (just one adder), instead of building a separate "subtractor."

How does it work?

- The **leftmost bit (MSB)** is treated as negative.
 - In normal binary: the leftmost bit is +128 (for 8-bit).
 - In two's complement: the leftmost bit is -128.
- That's why $10000000_2 = -128$ instead of +128.

