Binary Arithmetic

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

UNIVERSITY of VIRGINIA | ENGINEERING

Announcement

- My Office Hours (Rice 310)
  - Tuesdays, 11-12 pm
  - Thursdays, 11-12 pm
- TA Office Hours starting today, on website (location coming soon)
- Homework 1 available later this week
- Updated Final Exam: April 30, 2026 at 7-10pm

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 2 $_8$

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

$2^4 = 16$

100001010010

8 5 7 $_{16}$

Hexadecimal

Need more than 10 digits. What next?

1110

$2^3 2^2 2^1 2^0$

8 4 2 1

8+4+2 = 14

↓

E

1
2
3
4
5
6
7
8
9
a → 10
b → 11
c → 12
d → 13
e → 14
f → 15

Hexadecimal Exercise

Consider the following hexadecimal number:

852dab1e

Is it even or odd?

See the last digit.

Exercise

Turn 1101011110010_2 into base-10:

1 1 0 1 0 1 1 1 1 0 0 1 0

$2^{12}+2^{11}+2^9+2^7+2^6+2^5+2^4+2^1$

$=4096+2048+512+128+64+32+16+2$

$=6898$

Exercise

Turn 342_{10} into binary:

	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
	1	0	1	0	1	0	1	1	0

342 - 256 = 86

86 - 64 = 22

22 - 16 = 6

6 - 4 = 2

2 - 2 = 0

Exercise

Turn 1101011110010_2 into **hexadecimal**:

1 1010 1111 0010

0001 1010 1111 0010

1 A F 2

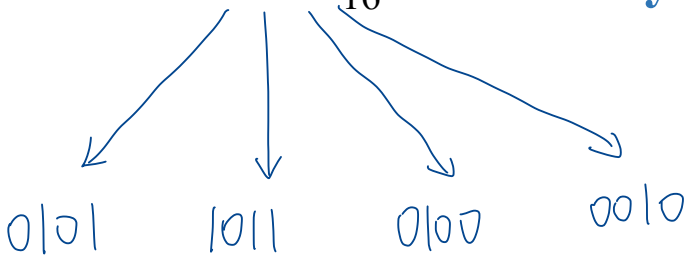
$1AF2_{16}$

//Turn 1101011110010_2 into **8-bit hexadecimal**:

How to extend it ?

Exercise

Turn $5b42_{16}$ into **binary** :



$5b42_{16} = 0101\ 1011\ 0100\ 0010_2$

Exercise

Turn $5b42_{16}$ into **decimal** :

5 b 4 2

16^3 16^2 16^1 16^0

$5\times16^3 + 11\times16^2 + 4\times16^1 + 2\times16^0$

$= 20480 + 2816 + 64 + 2$

$= 23362$

Exercise

Turn 2130_{10} into **hexadecimal** :

Directly convert it to a hexadecimal maybe too difficult:

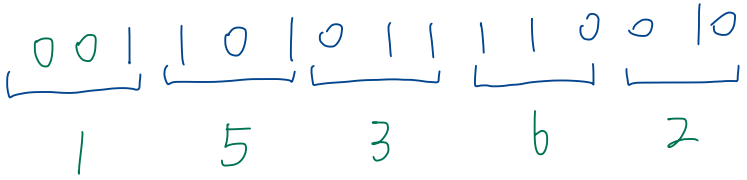
Another method :

① To binary

② Binary to hexadecimal

Exercise

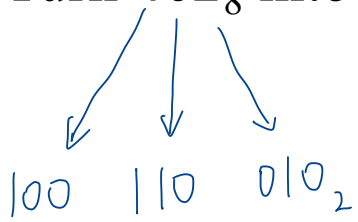
Turn 1101011110010_2 into **octal** :



So, $1101011110010_2 = 15362$

Exercise

Turn 462_8 into **binary** :



Using Different Bases in Code

	Old Languages (like C, 1972)	New Languages (like Java, Python, C#, 1995, 1991, 2000)
binary	no way	0b0101000 11 (letter b)
octal	073	0o 735 (letter o)
decimal	2130	4382
hexadecimal	0x 3af	0x 3af

Binary Addition

01101011 + 01100101

```
  0 1 1 0 1 0 1 1   (107)
+ 0 1 1 0 0 1 0 1   (101)
-----------------
  1 1 0 1 0 0 0 0   (208)
```

11101011 + 11100101

```
  1 1 1 0 1 0 1 1   (235)
+ 1 1 1 0 0 1 0 1   (229)
-----------------
1 1 1 0 1 0 0 0 0   (208)
```

The correct answer should be 464.
But I only have 8 bits for my result (read as 208)

range for 8 bits: (0 ~ 255)

$2^8 - 1 = 256 - 1 = 255$

Binary Subtraction

01111011 - 01100101

```
  0 1 1 1 1 0 1 1   (123)
- 0 1 1 0 0 1 0 1   (101)
-----------------
  0 0 0 1 0 1 1 0   (22)
```

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

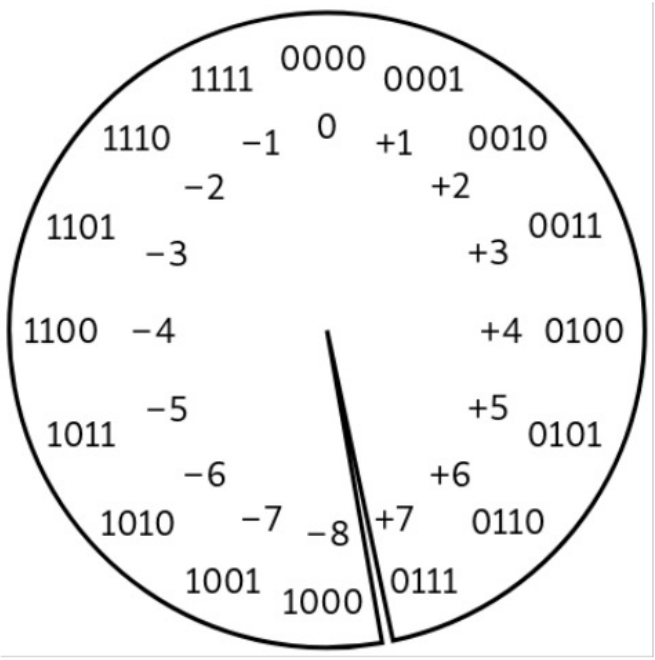
The scheme is called Two's Complement

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.



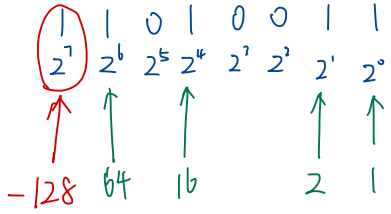
Values of Two's Complement Numbers

Consider the following 8-bit binary number in Two's Complement:

11010011

What is its value in decimal?

Method 1:



$-128 + 64 + 16 + 2 + 1 = -45$