Why are computers inaccurate?

Fixed-point Representation in Computer

소프트웨어 꼰대 강의

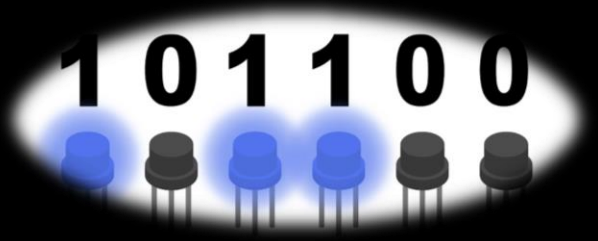
노기섭 교수

(kafa46@gmail.com)

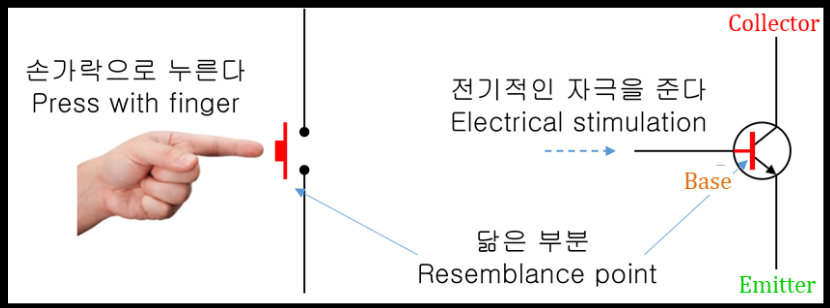
Course Overview

Topic	Contents
01. Orientation 오리엔테이션	Course introduction, motivations, final objectives 과정 소개, 동기부여, 최종 목표
02. Converting floating point 실수 변환	How to convert float from decimal to binary 어떻게 십진수를 이진수로 변환하는가?
03. Fixed-point Representation 고정 소수점 방식	How to represent float in fixed representation 어떻게 고정 소수점 방식으로 실수를 표현하는가?
04. Floating-point Representation 부동 소수점 방식	How to represent float in floating representation 어떻게 부동 소수점 방식으로 실수를 표현하는가?
05. Handling Negative Numbers 음수 처리	Complement, Radix, n-ary System, etc. 보수, 기수, 진법 등

Bits in Computer



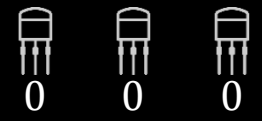
Why are only Zeros & Ones used in computers?



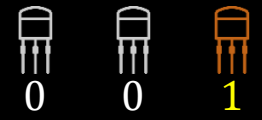
이미지 출처: https://javalab.org/ko/transistor/

n개의 트랜지스터를 사용하면?

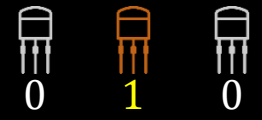
2^n개 정보를 표현할 수 있다.



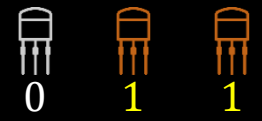
$0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 0$



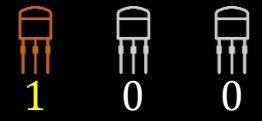
$0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1$



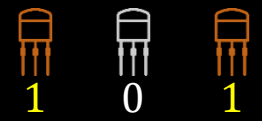
$0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 2$



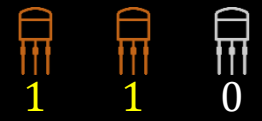
$0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 3$



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



$1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5$



$1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6$

$1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 7$

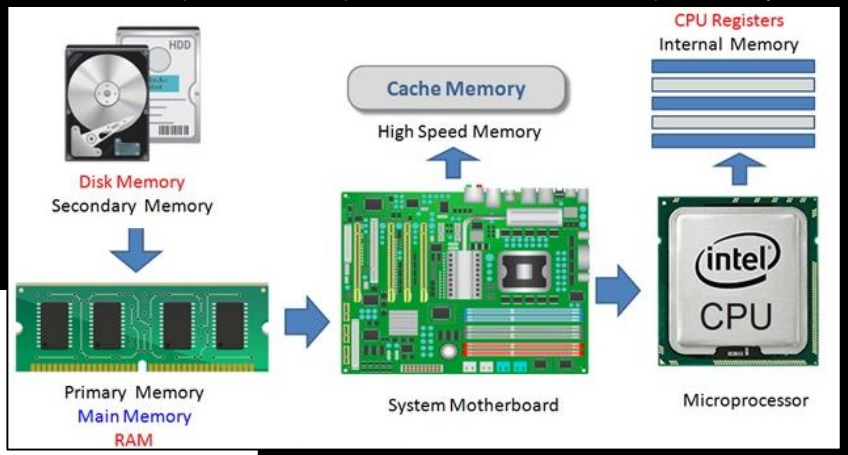
Memory System

Computer Memory?

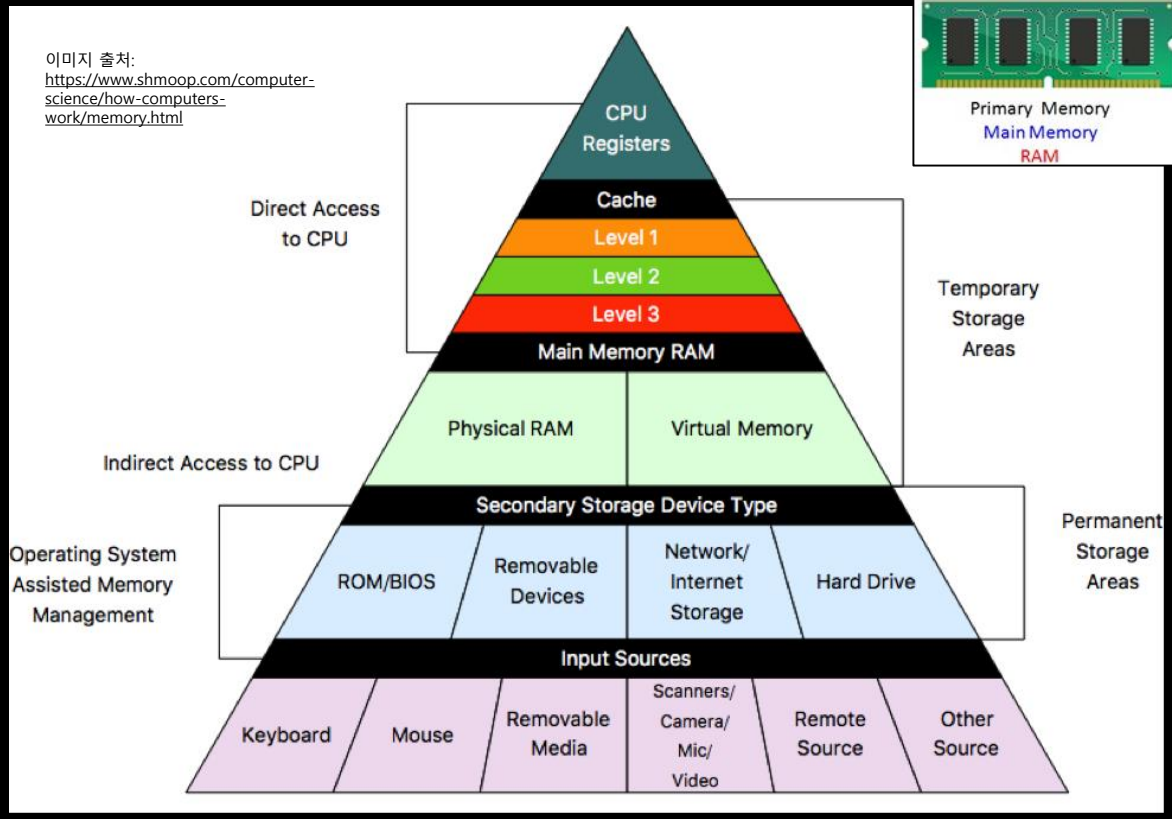
Memory Hierarchy

Goal:
CPU can as quickly accesses data & program instructions as possible!

이미지 출처: https://www.learncomputerscienceonline.com/what-is-computer-memory/



이미지 출처:
https://www.shmoop.com/computer-science/how-computers-work/memory.html



CPU fetches the data & program instructions into the CPU's internal memory.

CPU operates on the data as per program instructions.

Processed data either sent to output devices or permanent memory (HDD, USB, etc.)

Anyway, we need to store data into memory device!

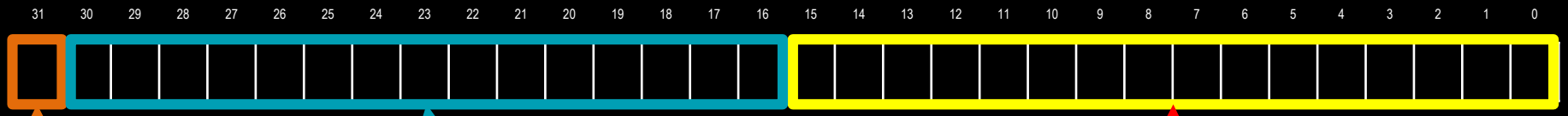
How to allocate Float data into Memory?

메모리 유치원 ^^

고정 소수점 방식
(Fixed-Point Number Representation)

Fixed-point Number Representation

실수 저장할 때 32비트 사용한다고 가정~



실수는 양수, 음수가 있잖아?

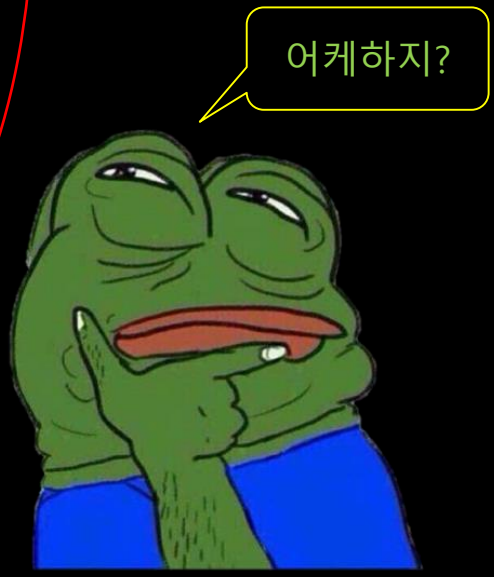
1개 비트를 사용하면
양수(0), 음수(1) 표현이 가능하겠군!
우리가 하던 대로 맨 앞 비트를 적용하자!

일단 정수부, 실수부는 어떻게 채우지?

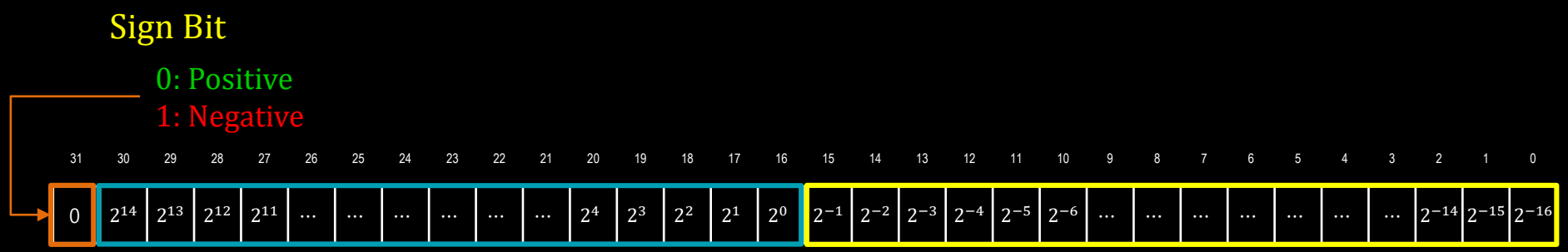
반띵 하자!

15 bits ➔ 정수(integer part)를 표현하자!

16 bits ➔ 소수(fractal part)를 표현하자!



Converting to Binary System



[Example]

5 . 6 2 6

2 | 5

2 | 2 ---------- 1

1 ---------- 0

$1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

0 . 6 2 5

× 2

1 . 2 5 0

0 . 2 5 0

× 2

0 . 5 0 0

× 2

1 . 0 0 0

0 . 0 0 0

0 . 1 0 1

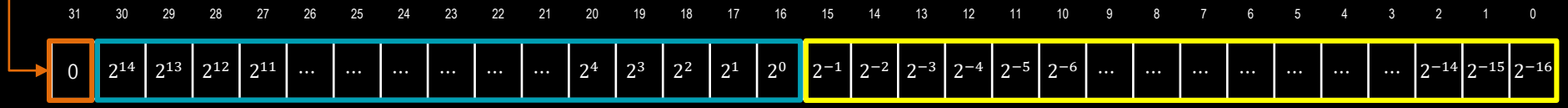
$1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-2}$

Assign bits in Fixed-point Memory System

Sign Bit

0: Positive

1: Negative

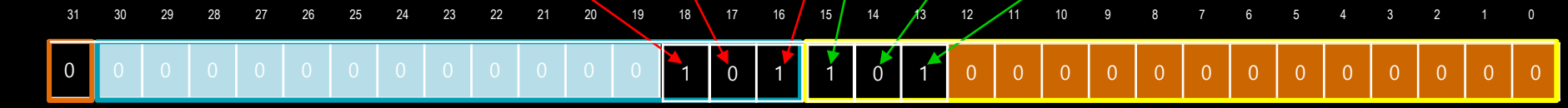


[Example]

5.626

$1 \cdot 2^{2} + 0 \cdot 2^{1} + 1 \cdot 2^{0}$

$1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-2}$



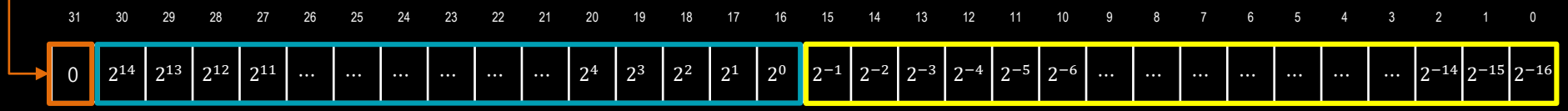
Approximation in Fixed-point Memory System

Sign Bit

0: Positive

1: Negative

Representing **0.1** in a Fixed-point Memory System



[Example]

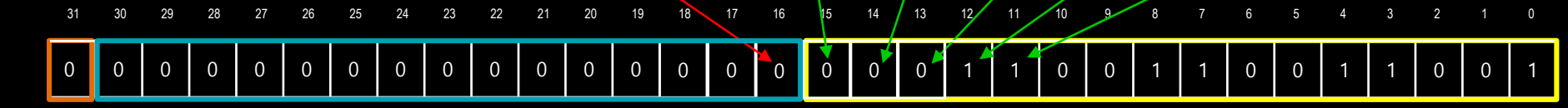
0 . 1

$0 \cdot 2^0$

0.0001100110011001100110011 ... (2)

$0 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} \ldots$

고정된 24비트 소수부까지만 잘라서 근사 ……



0 0000000 00011001100110011001101

$= 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + \cdots \approx 0.1000000238$

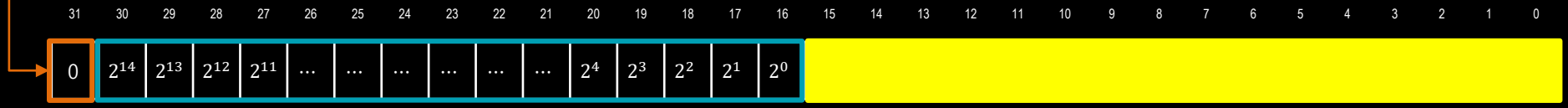
실제값 0.1과의 오차 $= 0.1000000238 - 0.1 =$ 0.0000000238

Min/Max Value in Integer Part

Sign Bit

0: Positive

1: Negative



How many positive (or negative) integers?

$2^{15} = 32{,}768$ *including zero* (0)

$$
\begin{array}{rl}
 & 2x = \quad 2^1 + 2^2 + \cdots + 2^{14} + 2^{15} \\
- & x = 2^0 + 2^1 + 2^2 + \cdots + 2^{14} \\
\hline
 & x = -2^0 + 2^{15} = 2^{15} - 1
\end{array}
$$

Which value is the minimum?

15 digits

000000000000001

$= 0 \cdot 2^{14} + 0 \cdot 2^{13} + \cdots + 1 \cdot 2^0$

$= 1 \cdot 1 = 1$

Which value is the maximum?

15 digits

111111111111111

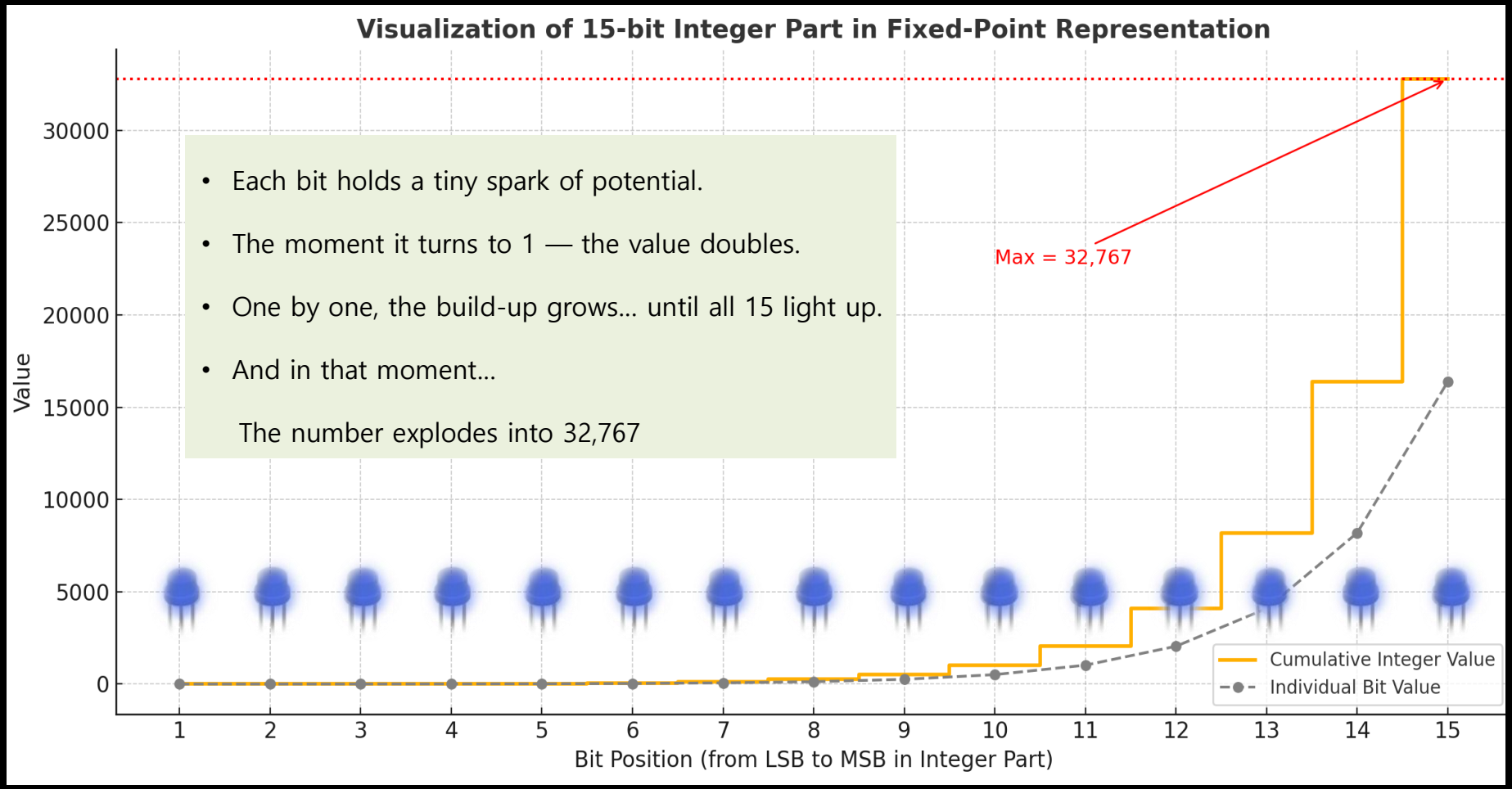
$= 1 \cdot 2^{14} + 1 \cdot 2^{13} + \cdots + 1 \cdot 2^0$

$= 2^0 + 2^1 + \cdots + 2^{14}$

$= 2^{15} - 1$

$= 32{,}767$

Visualization of Positive Integers with 15 Bits

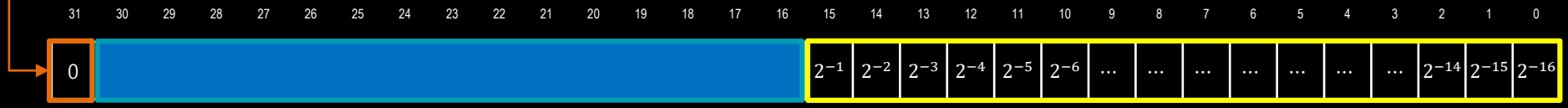


Min/Max Value in Fractional Part

Sign Bit

0: Positive

1: Negative



$$
\begin{aligned}
2x &= 2^0 + 2^{-1} + 2^{-2} + \cdots + 2^{-15} \\
x &= 2^{-1} + 2^{-2} + 2^{-3} + \cdots + 2^{-15} + 2^{-16} \\
\hline
x &= 2^0 - 2^{-16} = 1 - 2^{-16}
\end{aligned}
$$

How many positive (or negative) Fractional Part?

$2^{16} = 65{,}536$ *including zero* (0.0)

Which value is the minimum?

16 digits

0000000000000001

$= 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + \cdots + 1 \cdot 2^{-16}$

$= 2^{-16} = \frac{1}{2^{16}} = \frac{1}{65{,}536}$

$\approx 0.0000152587890625$

Which value is the maximum?

16 digits

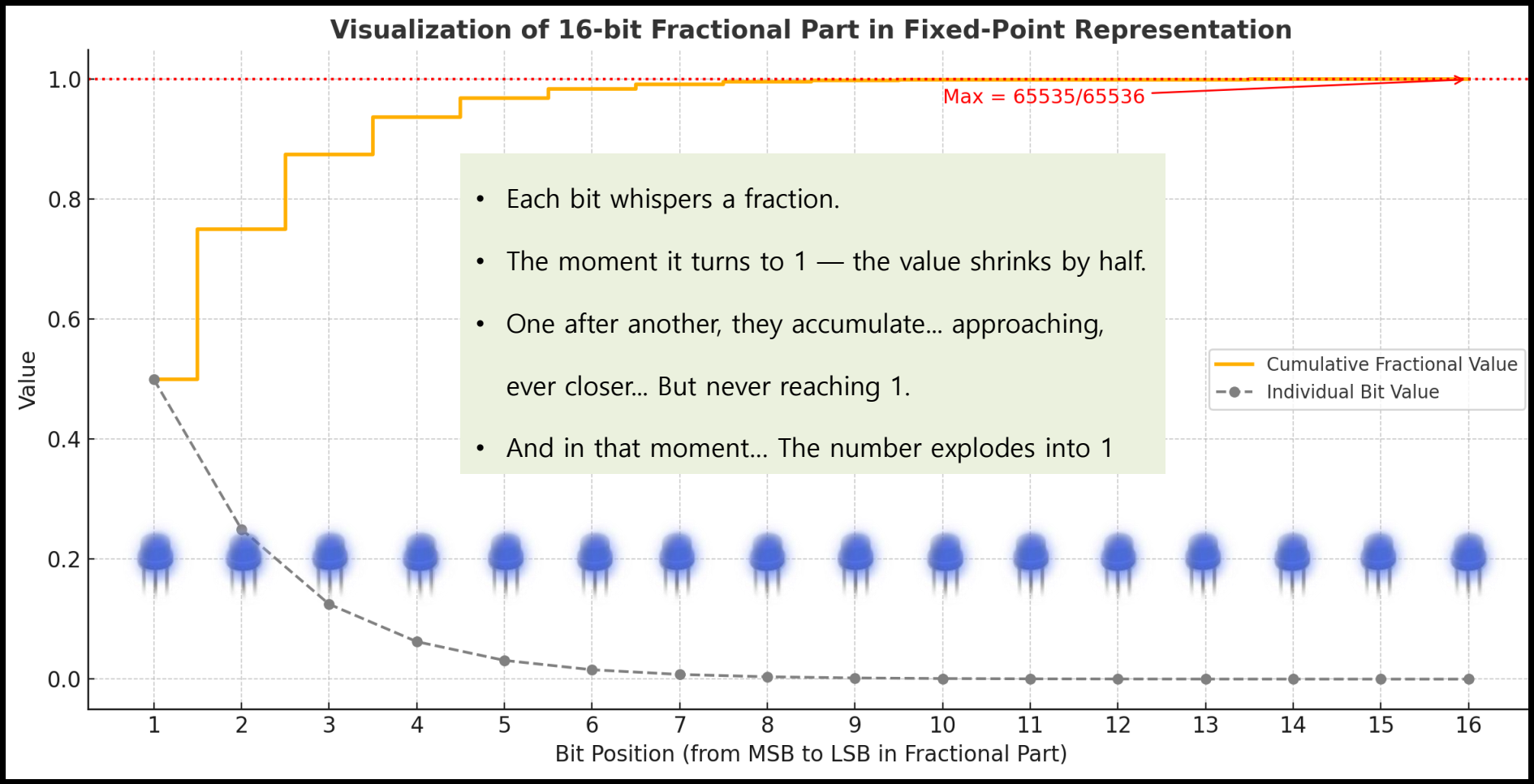
1111111111111111

$= 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + \cdots + 1 \cdot 2^{-16}$

$= 1 - 2^{-16} = 1 - \frac{1}{65{,}536}$

≈ 0.999985

16-bit Fractional Part in Fixed-Point Representation

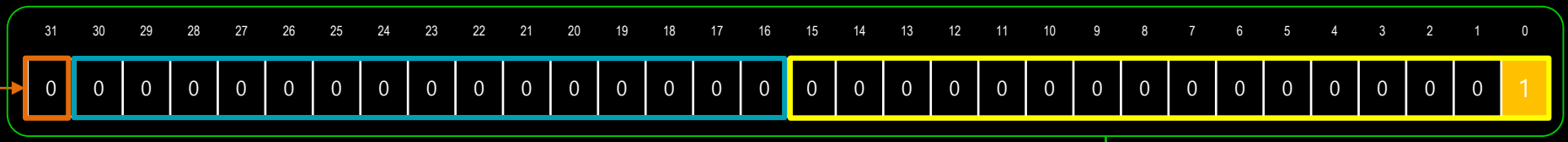


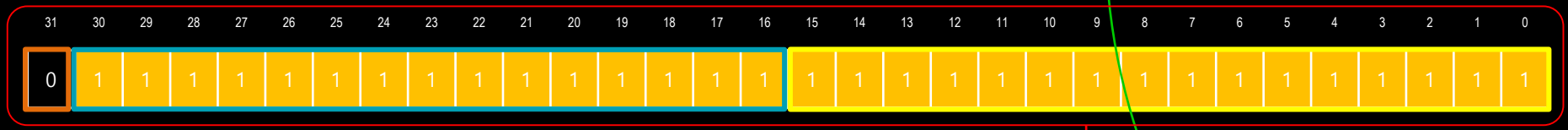
Disadvantage of Fixed-point Representation

Disadvantage 1. Limited Range

32,767보다 큰 수는 표현 못함

Sign Bit

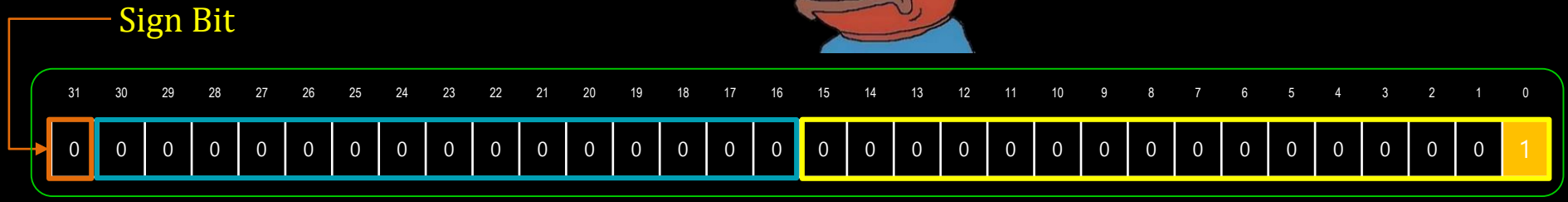




	Integer	Fraction Part	Float
Minimum (> 0)	0	$2^{-16} = \frac{1}{65536} \approx 0.000015$	0.000015
Maximum	$2^{15} - 1 = 32{,}767$	$1 - 2^{-16} = \frac{65535}{65536} \approx 0.999985$	32767.999985

Disadvantage of Fixed-point Representation

Disadvantage 2~4



Disadvantage 2. Rigid Bit Allocation

Wasted bits when representing numbers that don't require high precision or a large range.

Disadvantage 3. Poor Flexibility in Precision vs. Range

Wasted bits when representing numbers that don't require high precision or a large range.

Disadvantage 4. Unused Capacity (Memory)

In many cases, the allocated bits for may remain mostly unused.
→ leading to inefficient memory usage.

What is a possible solution?

교수님 ~~
고정 소수점 방식에
단점이 많네요 ㅠㅠ
대안은 없나요?

대안이 있어요 ^^
부동 소수점 방식을
적용하면 쉽게
해결할 수 있어요 ^^

수고하셨습니다 ..^^..