4190.308: Computer Architecture Lab. 2

Jaehoon Shim
Seongyeop Jeong
Ilkueon Kang
Wookje Han
Jinsol Park
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

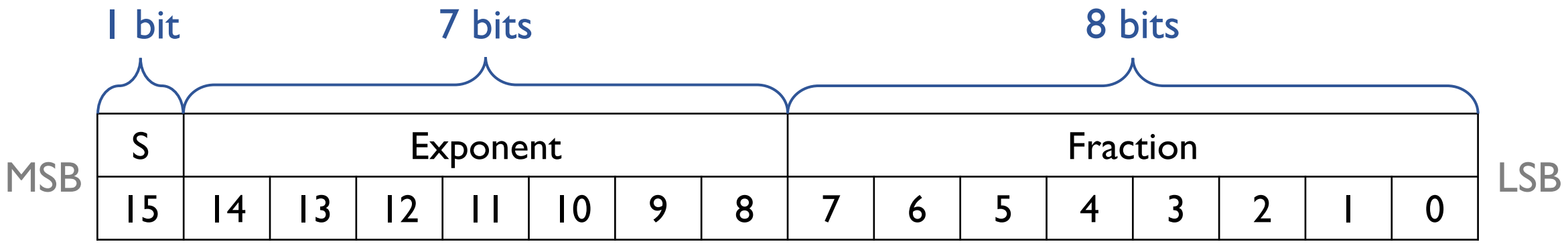
Seoul National University

Fall 2022

FP Addition

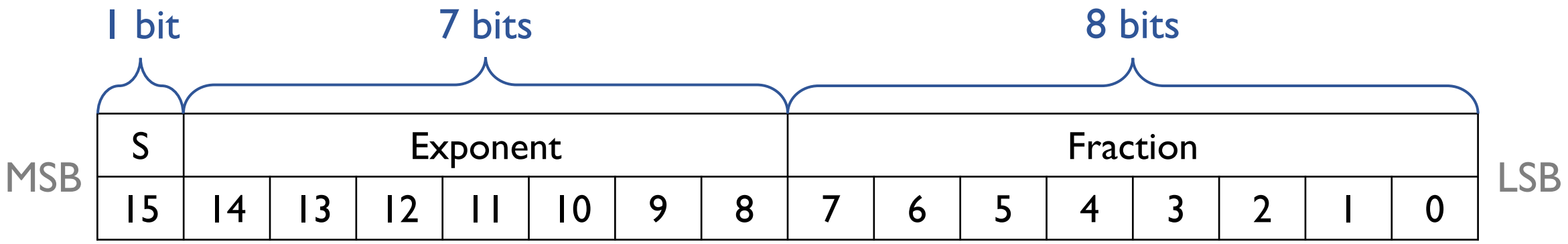
SnuFloat16 (SFP16)

- 16-bit floating point representation that follows the IEEE 754 standard for floating point arithmetic
- It consists of 1-bit sign bit, 7-bit exponent, and 8-bit fraction



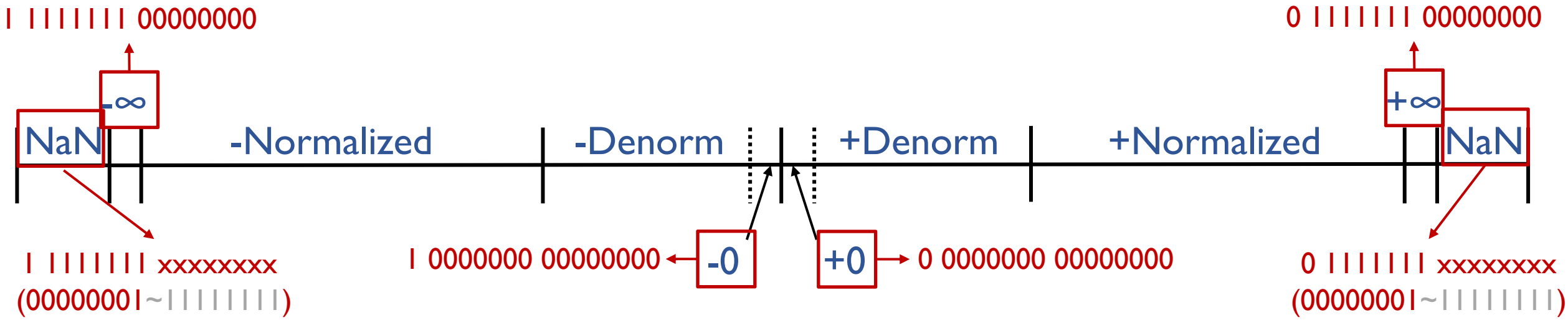
SFP16 Bias

- Bias for 7-bit exponent: $2^{7-1} - 1 = 63$
  - Smallest positive number: 0 0000000 00000001 → $0.00000001 \times 2^{-62}$
  - Largest positive number: 0 1111110 11111111 → 1.11111111×2^{63}



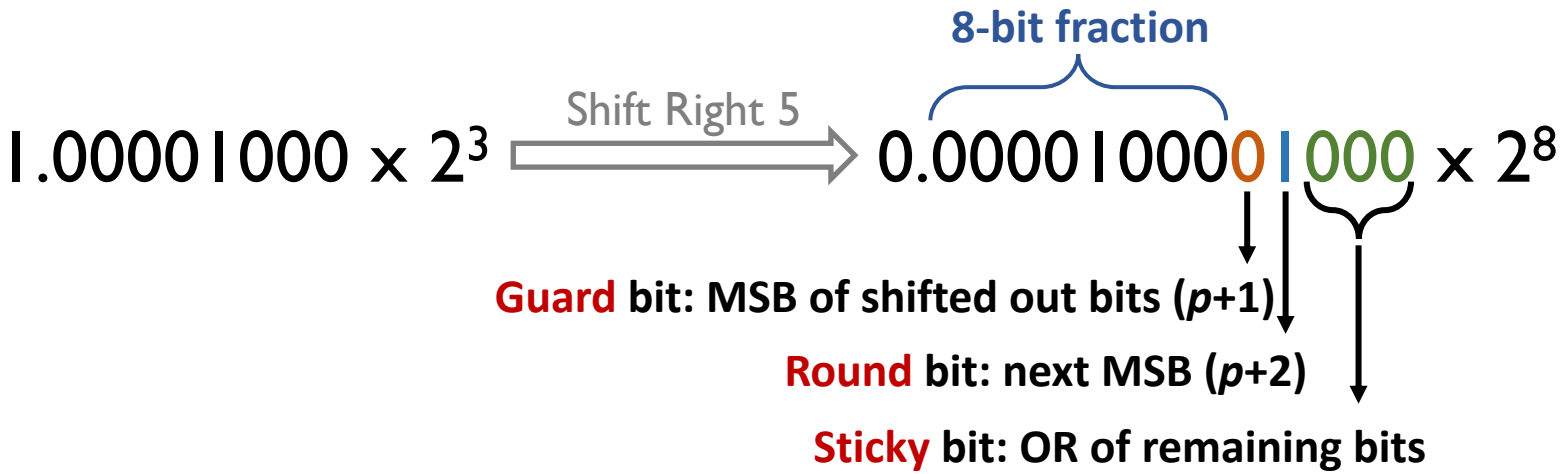
SFP16 Values

- You should follow the IEEE standard rules and representations
  - Normalized values: exp ≠ 0000000 and exp ≠ 1111111
  - Denormalized values: exp = 0000000
- For rounding, you should use the **round-to-even** scheme



FP Addition using Guard, Round and Sticky

- Ideal algorithm would perform operation first, and then round the result to p bits
  - This requires a very wide adder and very long registers to compute and keep the intermediate result
- We can produce the same result by maintaining only three extra bits
  - Guard (G), Round (R), and Sticky (S)



Steps for FP Addition using GRS

- E_x: exponent of x, M_x: significand of x (likewise for y)

1. If |x| < |y|, swap the operands
2. Extend M_x and M_y to include extra three bits (GRS) after 8-bit fraction and shift right M_y by *d* bits so that E_x == E_y
   - Initialize GRS bits to zero
   - $d = E_x - E_y$

```
                S EEE EEEE FFFF FFFF GRS
x: 0x17f2 = 0 001 0111 1111 0010 000   Sx=0, Mx=1.11110010, Ex=23-bias=-40
y: 0x154f = 0 001 0101 0100 1111 000   Sy=0, My=1.01001111, Ey=21-bias=-42
```

E_x != E_y

```
My: 1.01001111 000      Ey=-42
```

↓ Shift Right 2 (*d* = 23-21)

```
My: 0.01010011 110      Ey=-40
```

Steps for FP Addition using GRS

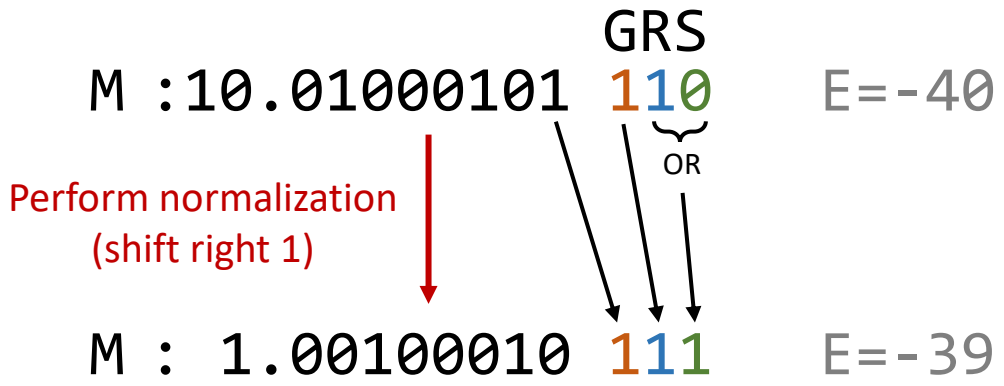
3. If S_x and S_y are same, compute preliminary significand M by adding M_y to M_x using integer arithmetic
   - If S_x and S_y are different, subtract M_y from M_x

```
  Mx: 1.11110010 000      Ex=-40, Sx=0
+ My: 0.01010011 110      Ey=-40, Sy=0
-------------------------------------
  M :10.01000101 110      E=-40, S=0
```

Steps for FP Addition using GRS

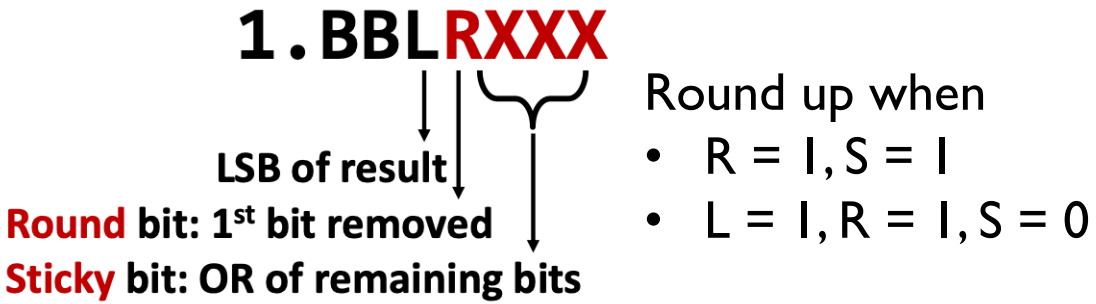
4. Normalize the preliminary significand M

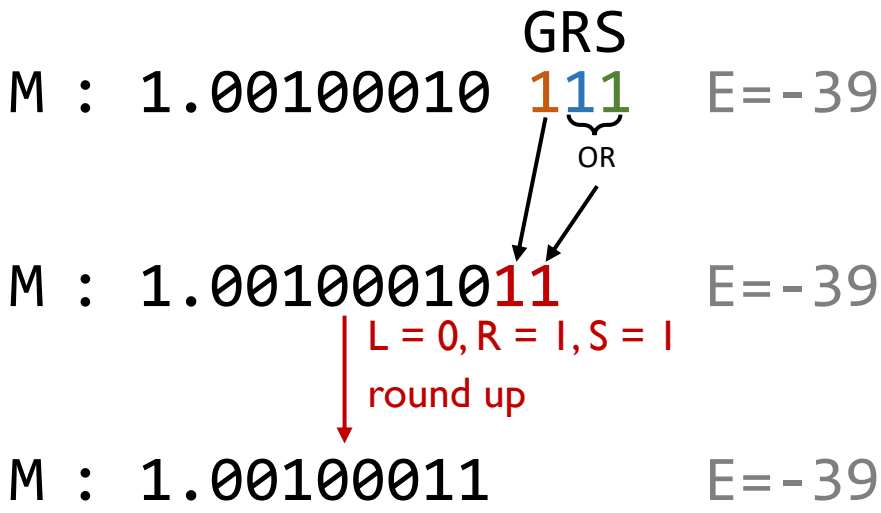
- Whenever M is shifted left, GRS bits are shifted together like part of fraction bits
- Whenever M is shifted right, GRS bits are updated as follows
  - Guard bit: previous LSB of 8-bit fraction
  - Round bit: previous guard bit
  - Sticky bit: previous round bit | previous sticky bit



Steps for FP Addition using GRS

5. Perform rounding according to round-to-even scheme
   - Role of guard bit is over after the normalization step
   - Adjust the round bit and sticky bit
     - Round bit: previous guard bit
     - Sticky bit: previous round bit | previous sticky bit





Steps for FP Addition using GRS

6. Perform normalization/rounding again if necessary
7. Encode the result into binary numbers according to SFP16 format

```
  M : 1.00100011     E=-39, S=0
      S = 0
      f = 00100011
      e = -39 + bias = 24

      S EEE EEEE FFFF FFFF
x+y : 0 001 1000 0010 0011  = 0x1823
```

Example (2)

- Normal + Normal values

```
              S EEE EEEE FFFF FFFF GRS
x: 0x0afe = 0 000 1010 1111 1110 000   Sx=0, Mx=1.11111110, Ex=10-bias=-53
y: 0x0288 = 0 000 0010 1000 1000 000   Sy=0, My=1.10001000, Ey=2-bias=-61

Mx: 1.11111110 000    Ex=-53, Sx=0
My: 1.10001000 000    Ey=-61, Sy=0
```

Shift Right 8 ⇒

```
My: 0.00000001 10001000 → When not using GRS
My: 0.00000001 101      Ey=-53, Sy=0
```

```
   Mx: 1.11111110 000    Ex=-53, Sx=0
+  My: 0.00000001 101    Ey=-53, Sy=0
-------------------------------------
   M : 1.11111111 101    E=-53, S=0
```

rounding ↓

```
   M : 1.11111111 11     E=-53, S=0
```

round up? O ↓ L=1, R = 1, S = 1

```
   M :10.00000000        E=-53, S=0
```

normalize ↗

```
M : 1.000000000   E=-52, S=0
```

round up? x ↓

```
M : 1.00000000    E=-52, S=0
```

```
S = 0
f = 00000000
e = -52 + bias = 11
```

```
          S EEE EEEE FFFF FFFF
x+y: 0 000 1011 0000 0000  = 0x0b00
```

Example (3)

- Normal + Denormal values

```
                S EEE EEEE FFFF FFFF GRS
x: 0x05E1 = 0 000 0101 1110 0001 000   Sx=0, Mx=1.11100001, Ex=5-bias=-58
y: 0x80F3 = 1 000 0000 1111 0011 000   Sy=1, My=0.11110011, Ey=1-bias=-62

Mx: 1.11100001 000     Ex=-58, Sx=0
My: 0.11110011 000     Ey=-58, Sy=1

Mx: 1.11100001 000     Ex=-58, Sx=0
-My: 0.00001111 001    Ey=-58, Sy=1
-------------------------------------
M : 1.11010001 111     E=-58, S=0
rounding
M : 1.11010001 11      E=-58, S=0
round up? O   L=1, R = 1, S = 1
M : 1.11010010         E=-58, S=0
```

Shift Right 4

```
My: 0.00001111 001     Ey=-58, Sy=1
```

```
                                  S = 0
                                  f = 11010010
M : 1.11010010    E=-58, S=0      e = -58 + bias = 5

                S EEE EEEE FFFF FFFF
x+y: 0 000 0101 1101 0010  = 0x05d2
```

Example (4)

- Denormal + Denormal values

|x|<|y|

```
            S EEE EEEE FFFF FFFF GRS
x: 0x0024 = 0 000 0000 0010 0100 000   Sx=0, Mx=0.00100100, Ex=1-bias=-62
y: 0x0037 = 0 000 0000 0011 0111 000   Sy=0, My=0.00110111, Ey=1-bias=-62

Mx: 0.00110111 000    Ex=-62, Sx=0
My: 0.00100100 000    Ey=-62, Sy=0

Mx: 0.00110111 000    Ex=-62, Sx=0
+ My: 0.00100100 000  Ey=-62, Sy=0
-------------------------------------
M : 0.01011011 000    E=-62, S=0
```

Ex- Ey == 0

rounding

```
M : 0.01011011 00     E=-62, S=0
```

round up? X L=1, R = 0, S = 0

```
M : 0.01011011        E=-62, S=0
```

```
S = 0
f = 01011011
e = 0
```

```
M : 0.01011011    E=-62, S=0
```

```
     S EEE EEEE FFFF FFFF
x+y: 0 000 0000 0101 1011  = 0x005b
```

Specification

- *SFP16 fpadd(SFP16 x, SFP16 y);*
  - Return value should be also represented in the SFP16 format
  - Should make use of the three extra bits (GRS) without retaining the unnecessary fraction bits
  - We do not distinguish between +0 and -0
    - If the result is 0, you can return any bit pattern corresponding to +0(0x0000) or -0(0x8000)
  - For NaN, we only allow bit pattern where fractional part is 0b00000001
    - You can assume that it has the bit pattern of 0x7f01 or 0xff01, and nothing else
  - We do not distinguish between +NaN and -NaN
    - If the result is NaN, you can return any of them
  - We DO distinguish between +inf and –inf
    - The result should be converted to +inf or –inf depending on its sign

Specification – Special Cases

- For special cases where +int/-inf, +0/-0, and +NaN/-NaN are involved, the return value should be the same as follows

	+inf	-inf	NaN	zero	other
+inf	+inf	NaN	NaN	+inf	+inf
-inf	NaN	-inf	NaN	-inf	-inf
NaN	NaN	NaN	NaN	NaN	NaN
zero	+inf	-inf	NaN	zero	
other	+inf	-inf	NaN		

Grading Guideline

- Types of test cases and their relative points during grading

Test Cases		Points
Normal + Normal Values	Addition	15
	Subtraction	15
Denormal + Normal Values	Addition	15
	Subtraction	15
Normal + Denormal Values	Addition	15
	Subtraction	15
Handling of Special Values		10

Restrictions

- You should not use any array even in the comment lines

- You are not allowed to use following data types
  - Floating-point data types
  - Any integer data type whose bit width is greater than 16 bits

- Following is the list of symbols and keywords that are not allowed
  - [,], int, long, float, double, struct, union, static

Restrictions

- Do not include any header file in the pa2.c file
- You are not allowed to use any external library functions
  - Please make sure to remove it before submission
- Your code should finish within a reasonable time
- If your implementation violates the intention of this project assignment, you will get penalty

Submission

- Due: 11:59PM, October 16 (Sunday)
  - 25% of the credit will be deducted for every single day delay
- Only submit the pa2.c file to the submission server
  - You don't have to write a report in this assignment
- Submitted code will NOT be graded instantly
  - It will be graded twice a day at noon and midnight
  - Only the last version submitted before 12:00pm or 12:00am will be graded

Thank You!

- Don't forget to read the detailed description before you start your assignment

- If you have any questions about the assignment, feel free to ask via KakaoTalk

- This file will be uploaded after the lab session☺