[CSED211] Introduction to Computer Software Systems

Lab 2: Data Lab2

Yonggon Park

POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY 1986 포항공과대학교

CAOS
COMPUTER ARCHITECTURE & OPERATING SYSTEMS LABORATORY

2024.09.12

Before We Begin: Next Week Schedule

- We have a make-up class in next week lab session (9/19)

Week	Lab Date	Lab	Deadline	Weight
1	09.05	Data Lab1	09.11	**20**
2	09.12	Data Lab2	09.18	**40**
3	09.19	Make Up Class		
4	09.26	Bomb Lab	10.09	**100**
5	10.03			
6	10.10	Attack Lab	***10.27***	**100**
7	10.17			
8		Midterm week		
9	10.31	Optimization Lab	-	-
10	11.07	Cache Lab	11.20	**100**
11	11.14			
12	11.21	Shell Lab	12.03	**100**
13	11.28			
14	12.04	Malloc Lab	***12.18***	**100**
15	12.11			
16	-	Final week	-	

Before We Begin: Change Your Password!

- Most of you did not change the password of the programming server
  - Everyone can check your povis id, and the default password is [povis_id]!@#

- Please change your password!
  - `$ passwd`

Today's Agenda

- Two's Complement
- Floating Point
  - Fractional Binary Numbers
  - IEEE Floating Point Standard
- Data Lab2

Two's Complement

Two's Complement

$$B2T(X) = -x_{w-1} \cdot 2^{w-1} + \sum_{i=0}^{w-2} x_i \cdot 2^i$$

Sign Bit

- Range of two's complement values
  - $TMin = -2^{w-1}$

    10…00
  - $TMax = 2^{w-1} - 1$

    01…11

- Inversion: $-x == \sim x + 1$
  - $-TMin = TMin$

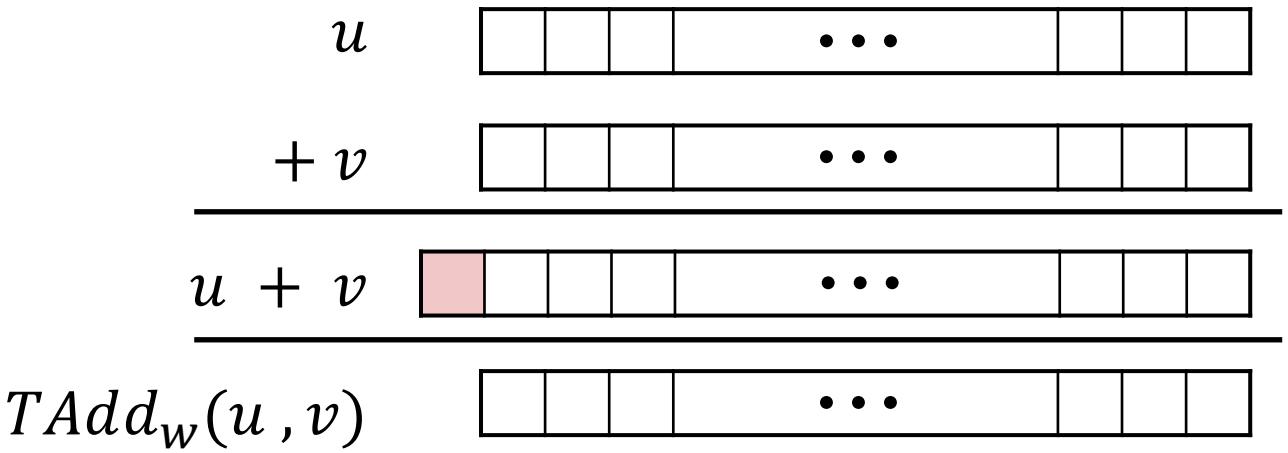
$bits$	$B2U(x)$	$B2T(x)$
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

Two's Complement Addition

Operands: w bits

True Sum: $(w + 1)$ bits

Discard Carry: w bits



- $TAdd$ and $UAdd$ have identical bit-level behavior
  - Signed vs. unsigned addition in C

```
int s, t, u, v;
s = (int) ((unsigned) u + (unsigned) v);
t = u + v;
```

  - Will give `s == t`

Today's Agenda

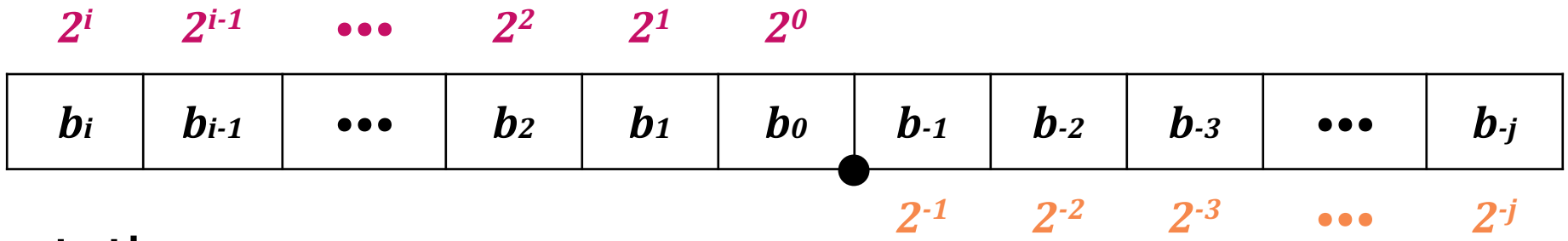
- Two's Complement

- Floating Point
  - Fractional Binary Numbers
  - IEEE Floating Point Standard

- Data Lab2

Fractional Binary Numbers

- What is $1011.101_{(2)}$?
  - The same principle as base 10 numbers



- Representation
  - Bits to right of binary point represent fractional powers of 2
  - Represents rational number: $\sum_{k=-j}^{i} b_k \times 2^k$
- Limitations
  - Limitation #1. Can only exactly represent numbers of the form $x/2^k$
  - Limitation #2. Limited range of numbers

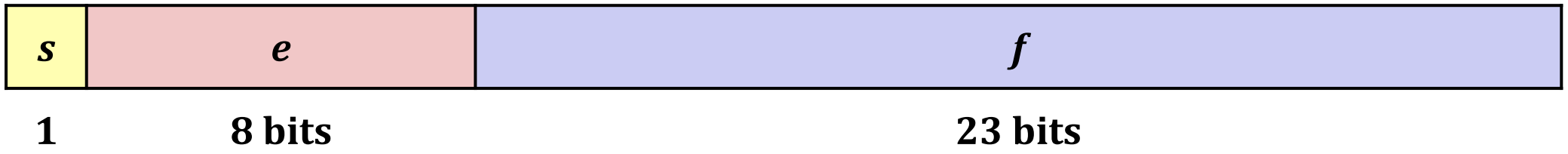
Floating Point: Representation

- Numerical form: $(-1)^s \times M \times 2^E$
  - Sign bit (s) determines whether number is negative or positive
  - Significand (M) normally a fractional value in range $[1.0, 2.0)$
  - Exponent (E) weights value by power of two

- Encoding
  - MSB s is sign bit s
  - Exp field e encodes E (not exactly the same as E)
  - Frac field f encodes M (not exactly the same as M)

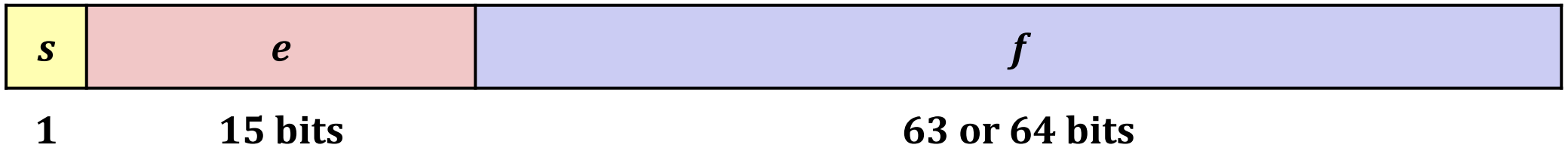
Floating Point: Precisions

- Single precision: 32 bits



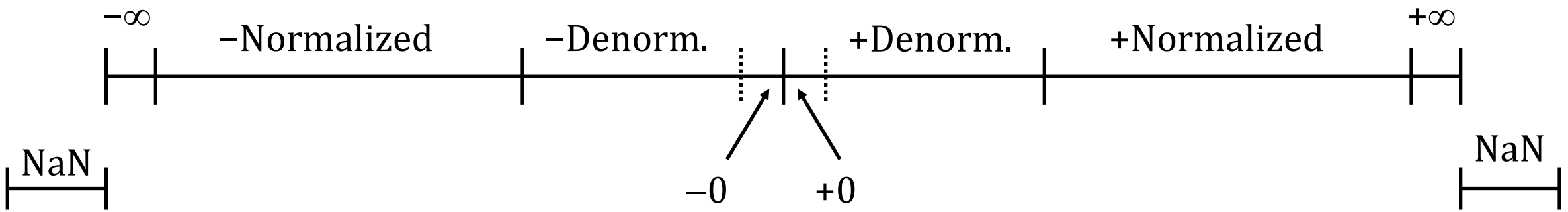
- Double precision: 64 bits

- Extended precision (Intel only): 80 bits



Floating Point: Encoding

- Normalized value
- Denormalized value
- Special value



Normalized Values

- When: $e \neq 000\ldots0$ and $e \neq 111\ldots1$

$$v = (-1)^s \times M \times 2^E$$

- Exponent coded as biased value: $E = Exp - Bias$
  - EXP: Unsigned value e
  - $Bias$: $2^{k-1} - 1$, where k is the number of exponent bits
    - Single precision (8-bit exp): 127 (EXP: 1, ···, 254 map to E: -126, ···, 127)
    - Double precision (11-bit exp): 1023 (EXP: 1, ···, 2046 map to E: -1022, ···, 1023)

- Significand coded with implied leading 1: $M = \mathbf{1.xxx\ldots x}_{(2)}$
  - $f_{i-1}, f_{i-2}, \cdots, f_0$ represent the mantissa part **xxx…x**
  - Minimum when 000…0 ($M = 1.0$)
  - Maximum when 111…1 ($M = 2.0 - \varepsilon$)
  - Get extra leading bit for free

Normalized Encoding: Example

$$v = (-1)^s \times M \times 2^E$$
$$E = Exp - Bias$$

- Value: `float f = 15213.0;`
  - $15213_{(10)} = 11101101101101_{(2)}$
    $= 1.1101101101101_{(2)} \times 2^{13}$
- Significand
  - $M = 1.\underline{1101101101101}_{(2)}$
  - $f = \underline{1101101101101}0000000000_{(2)}$
- Exponent
  - $E = 13$
  - $Bias = 127$
  - $Exp\ (e) = 140 = 10001100_{(2)}$
- Result

0	1000 1100	110 1101 1011 0100 0000 0000
s	e (exp)	f (frac)

Denormalized Values

$$v = (-1)^s \times M \times 2^E$$
$$E = 1 - Bias$$

- When $e = 000\ldots0$
- Exponent $E = 1 - Bias$ (instead of $E = -Bias$)
- Significand coded with implied leading 0: $M = 0.xxx\ldots x_{(2)}$
  - $f_{i-1}, f_{i-2}, \cdots, f_0$ represent the mantissa part **xxx…x**
- Two purposes
  - To represent zero value: $f = 000\ldots0$
    - Note distinct values: +0 and –0 (why?)
  - To represent numbers very close to 0: $f \neq 000\ldots0$
    - Lose precision as get smaller
    - Gradual underflow: For numbers smaller than the minimum normalized value

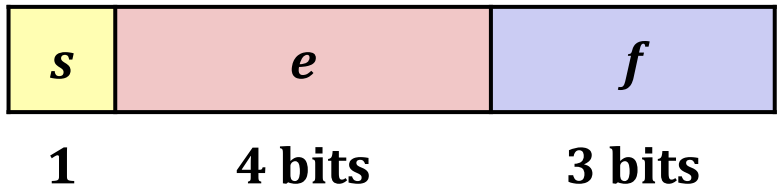
Special Values

- When $e = \mathbf{111 \ldots 1}$

- If $f = \mathbf{000 \ldots 0}$
  - Represents value ∞ (infinity)
  - Operation that overflows
  - Both positive and negative
  - E.g., $1.0/0.0 = -1.0/-0.0 = +\infty$, $1.0/-0.0 = -\infty$

- If $f \neq \mathbf{000 \ldots 0}$
  - Not-a-Number (NaN)
  - Represents case when no numeric value can be determined
  - E.g., `sqrt(-1)`, $\infty - \infty$, $\infty \times 0$

Example: Tiny Floating Point



- 8-bit floating point representation
  - The sign bit is in the most significant bit
  - The next four bits are the exponent, with a bias of 7
  - The last three bits are the fraction part

- The same general format as IEEE format
  - For normalized and denormalized numbers
  - For special values to represent 0, NaN, and infinity

Dynamic Range (Positive Only)

	s	exp	frac	E	Value	
Denormalized numbers	0	0000	000	-6	0	
	0	0000	001	-6	1/8*1/64 = 1/512	Closest to zero
	0	0000	010	-6	2/8*1/64 = 2/512	
	…					
	0	0000	110	-6	6/8*1/64 = 6/512	
	0	0000	111	-6	7/8*1/64 = 7/512	Largest denorm.
	0	0001	000	-6	8/8*1/64 = 8/512	Smallest norm.
Normalized numbers	0	0001	001	-6	9/8*1/64 = 9/512	
	…					
	0	0110	110	-1	14/8*1/2 = 14/16	
	0	0110	111	-1	15/8*1/2 = 15/16	Closest to 1 below
	0	0111	000	0	8/8*1 = 1	
	0	0111	001	0	9/8*1 = 9/8	Closest to 1 above
	0	0111	010	0	10/8*1 = 10/8	
	…					
	0	0111	110	7	14/8*128 = 224	
	0	0111	111	7	15/8*128 = 240	Largest norm
Special Value	0	1111	000	n/a	inf	+∞

Today's Agenda

- Two's Complement
- Floating Point
  - Fractional Binary Numbers
  - IEEE Floating Point Standard
- Data Lab2

Data Lab2: Overview

- You should implement some functions that uses integer/floating-point while:
  - 1. Following other programming rules (will be discussed later)
    - Programming rules for integer questions and floating point questions are different
  - 2. Using only allowed operations
  - 3. Using less than a limited number of operations
  - Any violation will be regarded as 0 point

- Due: 09/18 23:59 (Late submission will not be accepted)

- Submit a code file and your lab report (in pdf)
  - Source code name: [student id].c (e.g., 20231234.c)
  - Report name: [student id].pdf (e.g., 20231234.pdf)

Data Lab2: Problems (Integer)

- Q1. `int negate(int x)`
  - Return `-x`
  - Legal operations: `!, ~, &, ^, |, +, <<, >>`
  - Max ops: 5

- Q2. `int isLess(int x, int y)`
  - Return `1` if `x < y` else return `0`
  - Legal operations: `!, ~, &, ^, |, +, <<, >>`
  - Max ops: 24

Data Lab2: Programming Rules (Integer)

- You are allowed to use only the following:
  - 1. Integer constants `0` through `255 (0xFF)`
  - 2. Function arguments and local variables
  - 3. Unary integer operations: `!, ~`
  - 4. Binary integer operations: `&, ^, |, +, <<, >>`

- You are not allowed to:
  - 1. Use any control constructs: `if, do, while, for, switch`
  - 2. Define or use any macros
  - 3. Call any other functions
  - 4. Use any other operations: `&&, ||, -,` …
  - 5. Use any data type other than int (cannot use arrays, structs, or unions)

Data Lab2: Problems (Floating Point)

- Q3. `unsigned float_abs(unsigned uf)`
  - Return absolute value of `uf`
  - Legal operations:Any integer/unsigned operations including `||, &&, if, while`
  - Max ops: 10

- Q4. `unsignedint float_twice(unsigned uf)`
  - Return `2×uf`
  - Legal operations:Any integer/unsigned operations including `||, &&, if, while`
  - Max ops: 30

Data Lab2: Problems (Floating Point, Cont.)

- Q5. `unsigned float_i2f(int uf)`
  - Return `float(uf)`
  - Legal operations:Any integer/unsigned operations including `||, &&, if, while`
  - Max ops: 30

- Q6. `int float_f2i(unsigned uf)`
  - Return `int(uf)`
  - Legal operations:Any integer/unsigned operations including `||, &&, if, while`
  - Max ops: 30

Data Lab2: Programming Rules (Floating Point)

- You are allowed to:
  - Use standard control structures: Conditional, loops
  - Use both int and unsigned data types

- You are not allowed to:
  - Use unions, structs, or arrays
  - Use any floating point data types, operation, or constants

- Any floating-point operand will be passed to the function as having type unsigned, and any returned floating-point value will be of type unsigned

Data Lab2: How to Do

- You may assume that your machine:
  - 1. Uses 2's complement, 32-bit representations of integers
  - 2. Performs right shifts arithmetically
  - 3. Has unpredictable behavior when shifting an integer by more than the word size

- Open the `bits.c` file in `datalab2.tar` file and implement 6 functions
  - Use minimum number of operators as you can
  - Follow the programming rules (0 point if violated)

- We provided grading program (`btest`) and programming rule checker (`dlc`)
  - TA will use this two programs to grade your works

Questions?

[CSED211] Introduction to Computer Software Systems

Lab 3: Bomb Lab (Test)

Yonggon Park

POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY 1986 포항공과대학교

CAOS
COMPUTER ARCHITECTURE & OPERATING SYSTEMS LABORATORY

2024.09.12

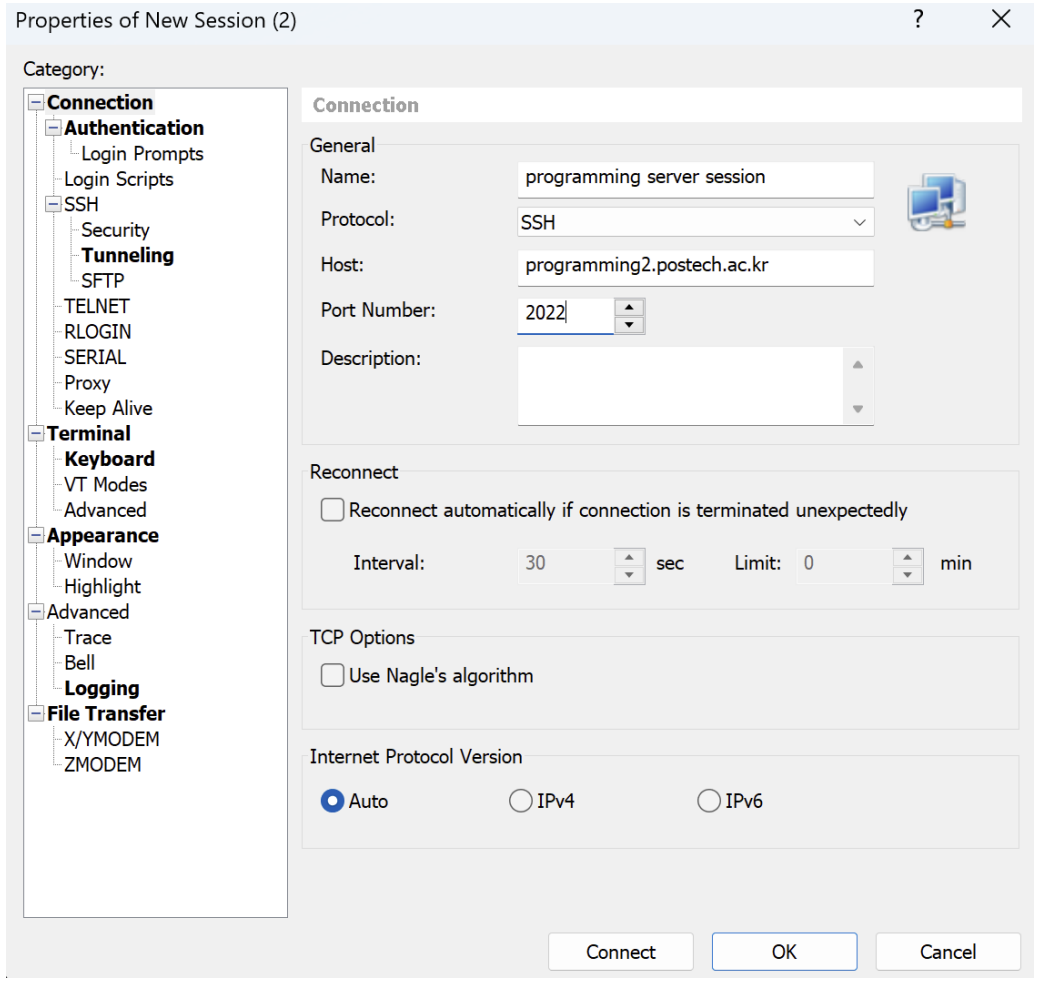
How to Download Bomb? (Terminal Version)

- Connect to the programming server with SSH command
  - `$ ssh -p 2022 -L 15213:127.0.0.1:15213 [YourServerID]@programming2.postech.ac.kr`
  - `-p: SSH port number to connect server`
  - `-L: Port tunneling`

- You can access the web server only when the SSH session is alive

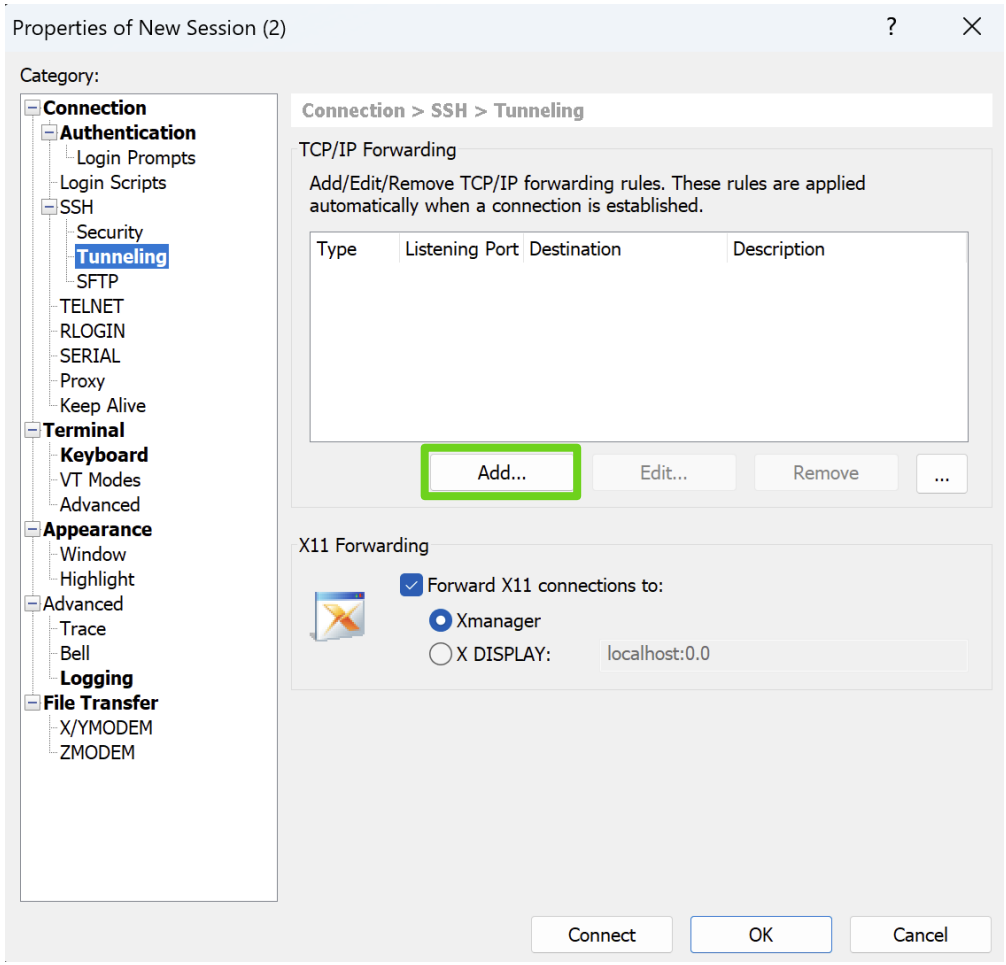
How to Download Bomb? (Xshell Version)

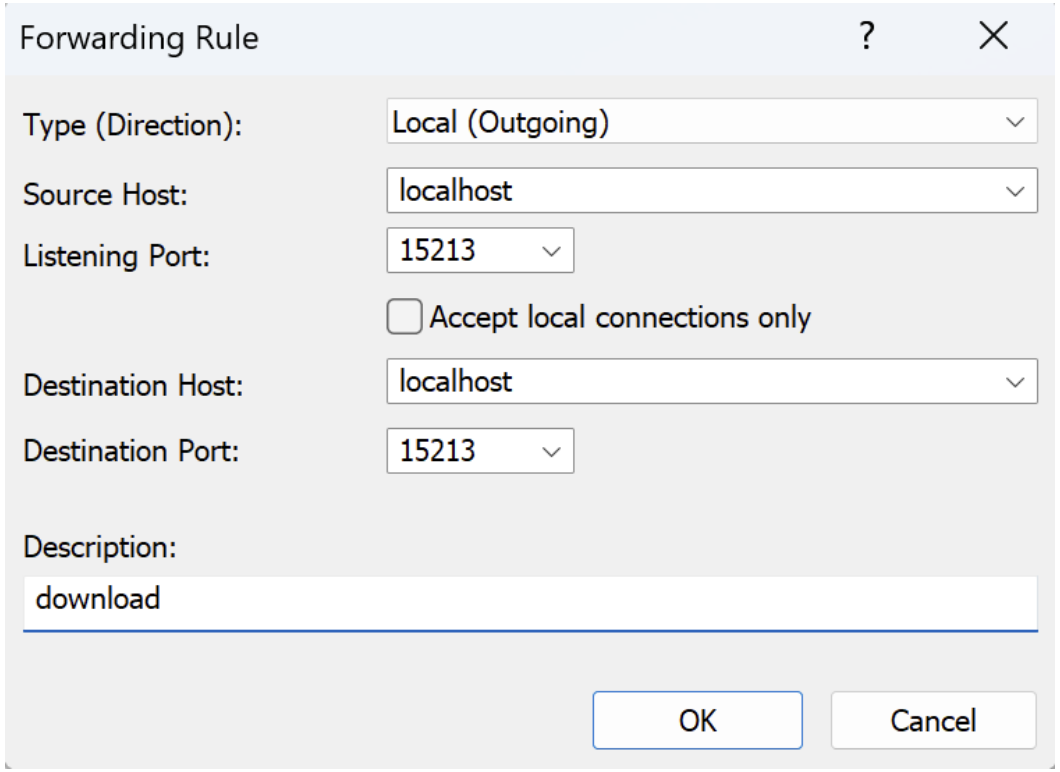
- Create new session



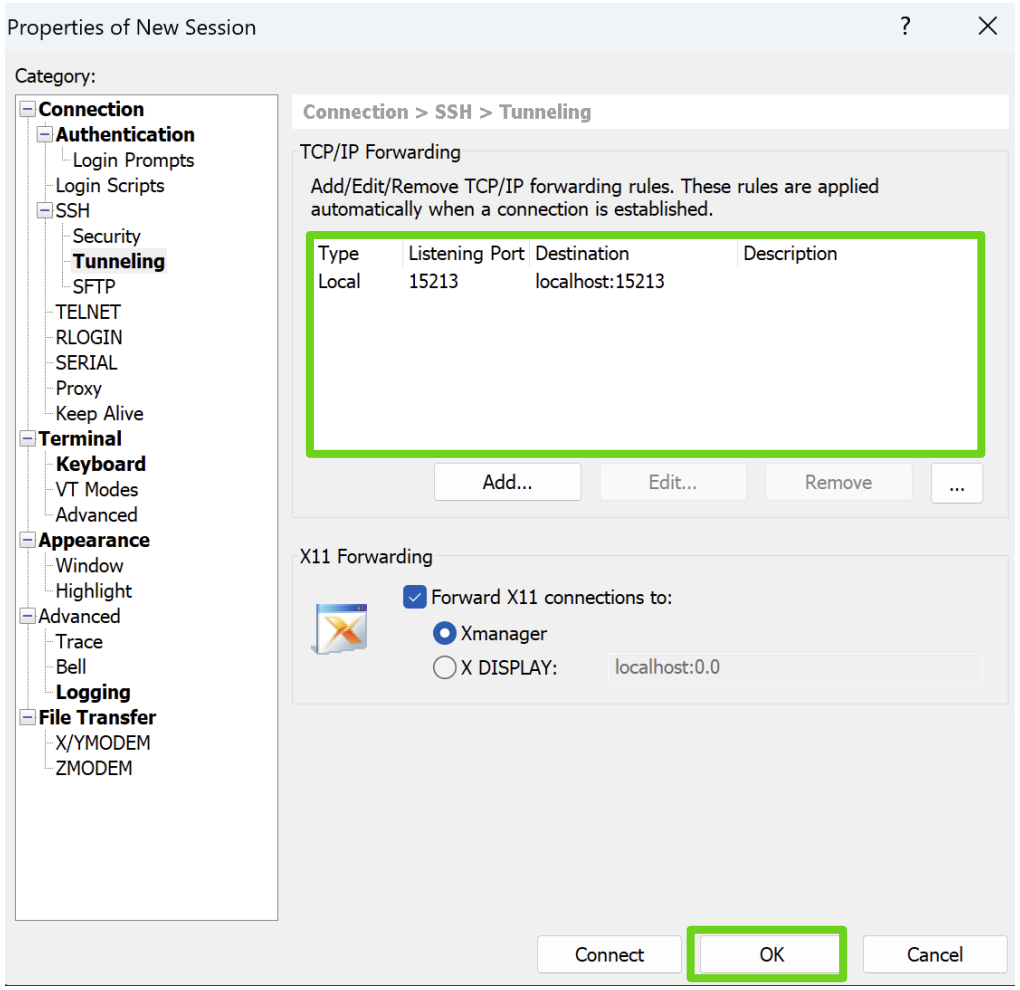
How to Download Bomb? (Xshell Version)

- Go to SSH → Tunneling to setup port forwarding
  - Add the forwarding rule for 15213





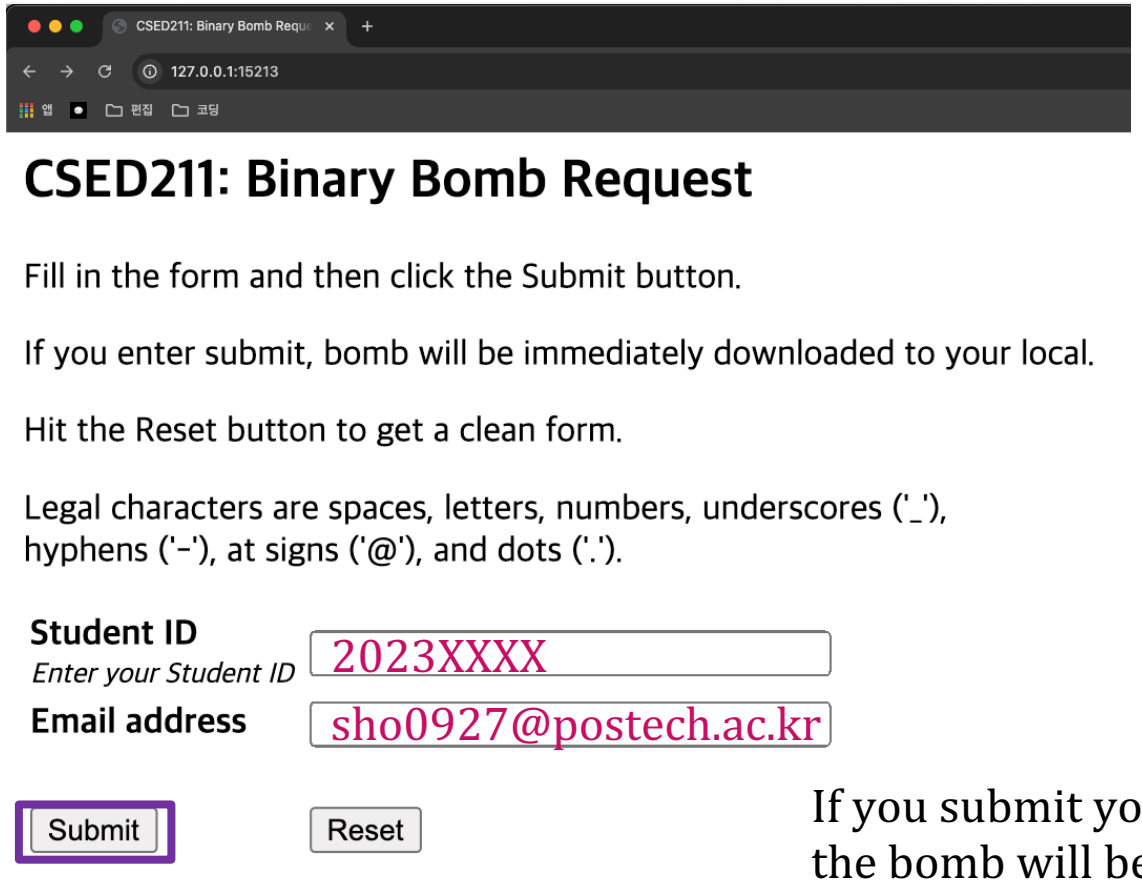
How to Download Bomb? (Xshell Version)



Accessing the Bomb Server

- You can access the web server only when the SSH session is alive

- In your web browser (e.g., Chrome), search 127.0.0.1:15213



If you submit your information,
the bomb will be downloaded immediately

Quiz

- Go to PLMS, start the quiz
  - For fairness, quiz will be shut down after everyone leaves the classroom

- Selects every things that are correct from 6 statements
  - There is a penlaty for selecting wrong statement,
    but your minimal score would be 0 (will not affect code & report score)

[CSED211] Introduction to Computer Software Systems

Lab 2: Data Lab2

Yonggon Park

POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY 1986 포항공과대학교

CAOS
COMPUTER ARCHITECTURE & OPERATING SYSTEMS LABORATORY

2024.09.12