

# Lab 01

## Bit Lab

Euhyun Moon, Ph.D.  
Machine Learning Systems (MLSys) Lab  
Computer Science and Engineering  
Sogang University



# About the Labs

- **Labs account for 30% of the total score for the semester**
- **Three lab assignments**
  - **Lab 1:** Bit Lab (6%)
  - **Lab 2:** Reversing Lab (12%)
  - **Lab 3:** Cache Lab (12%)
- **Today: Lab 1 (Bit Lab)**
  - Three small C programming exercises
  - Puzzles using *bit-level* operations (a.k.a. *Data Lab* in *CSAPP*)
- **Another goal of lab 1 is to become familiar with Linux and learn how to work with the skeleton code**

# General Information

- **Check the *Assignment* tab on *Cyber Campus***
  - Please find the attached skeleton code (`Lab1.tgz`)
  - Submit your work within the same post
- **Deadline: 4/3 Wednesday 23:59**
  - Late submission deadline: 4/4 Thursday 23:59 **(-20% penalty)**
  - A delay penalty is uniformly applied **(not problem by problem)**
- **Please carefully read the instructions on this slide deck**
  - This slide deck provides a step-by-step tutorial for the lab
  - Also includes important submission guidelines
    - **Failure of follow the submission guidelines will result in a penalty**

# Skeleton Code Structure

- **Copy Lab1.tgz to the CSPRO server and then decompress it**
  - Recommend to use [csp2.sogang.ac.kr](https://csp2.sogang.ac.kr) (Ubuntu 20.04)
  - **Do not decompress-and-copy**; Do copy-and-decompress
- **1-1~1-3: Each directory contains a problem**
- **validate**: Verifies if your code satisfies the constraints
- **check.py**: Script for self-grading (explained later)
- **config**: Used by grading script (you may ignore)

```
jason@ubuntu:~$ tar -xzf Lab1.tgz
jason@ubuntu:~$ ls Lab1/
1-1  1-2  1-3  check.py  config  validate
```

# Problem Directory (Example: 1-1)

- **bitMask.c**: This is the only file that you have to fill in
  - **Do NOT** make any modification to other files
- **main.c**: This program will test your code in **bitMask.c**
- **Makefile**: You can build the program by typing **make**
  - If you are unfamiliar with **make** or **Makefile**, please take a brief look at [makefiletutorial.com/](http://makefiletutorial.com/)
- **testcase**: Contains test cases and expected outputs

```
jason@ubuntu:~/Lab1/1-1$ ls
bitMask.c  main.c  Makefile  testcase
jason@ubuntu:~/Lab1/1-1$ ls testcase/
ans-1  ans-2  tc-1  tc-2
```

# Tasks

- For each problem, you have to implement a function
  - **Read the comments in each file carefully**: it provides examples and gives the assumptions regarding the input
- **Problem 1-1 (bitMask.c)**
  - `bitMask(x)`: return a mask that has 32-x number of 0's followed by x number of 1's
- **Problem 1-2 (absVal.c)**
  - `absVal(x)`: return the absolute value of x
- **Problem 1-3 (conditional.c)**
  - `conditional(x, y, z)`: return z if x is 0, return y otherwise

# Constraints

- There are some **constraints** that your code must satisfy
  - If your code does not satisfy them, you will get **0 point**
  - Allowed operators: **! ~ & ^ | + << >>**
    - Do **NOT** use other operators such as **&& || - == < > ?**
  - Use **int** type only
    - Do **NOT** use other primitive types, structure, array, etc.
  - Write **straight-line code**
    - Do **NOT** use any control constructs such as **if, do, while, for, switch**, etc.
  - Do **NOT** define or call any additional functions
  - Do **NOT** include any headers such as **#include <stdio.h>**

# Using the Validator

- You can use `validate` to confirm whether your code satisfies the previous constraints
  - It will print any illegal points found within the code you've written
  - If it does not print anything, your code has passed the validation

```
jason@ubuntu:~/Lab1$ cat 1-2/absVal.c
int absVal(int x) {
    if (x > 0)
        return x;
    else
        return -x;
}
jason@ubuntu:~/Lab1$ ./validate 1-2/absVal.c
dlc:1-2/absVal.c:2:absVal: Illegal operator (>)
dlc:1-2/absVal.c:5:absVal: Illegal operator (-)
dlc:1-2/absVal.c:5:absVal: Illegal if
```



# Running Test Cases

- After compiling the program with the `make` command, you can run it by providing the path of the test case file
- Some test cases and their expected outputs are already provided in the `testcase/` directory
  - Output of running `tc-N` must match with `ans-N`

```
jason@ubuntu:~/Lab1/1-1$ make
gcc bitMask.c main.c -o main.bin
jason@ubuntu:~/Lab1/1-1$ cat testcase/tc-2
31
jason@ubuntu:~/Lab1/1-1$ cat testcase/ans-2
0x7fffffff
jason@ubuntu:~/Lab1/1-1$ ./main.bin testcase/tc-2
0x7fffffff
```

*Must  
match*



# Self-Grading

- **After you believe everything is complete, run `check.py` to verify that you pass all the provided test cases**
  - Each character in the result has following meaning:
    - 'O': correct, 'X': wrong,
    - 'C': compile error, 'T': timeout
    - 'I': failed to pass the validator, 'E': runtime error
  - So it is important to ensure that `./check.py` prints 'O' for all cases

```
jason@ubuntu:~/Lab1$ ./check.py
[*] 1-1: OO
[*] 1-2: II
[*] 1-3: XX
```

# Test Cases for Grading

- **We will use various test case sets to evaluate your code**
  - This means that even if you successfully pass all the provided test cases, it does not guarantee that you will receive full points.
- **Thus, you are encouraged to test your own code with various inputs**
- **Some students request additional test cases from us, but it is important to practice this independently**

# ChatGPT

- Remember that using ChatGPT or searching for solutions on the web is **not permitted**
- Nevertheless, you may still feel tempted to do so
  - Indeed, the problems in Lab 1 are not difficult, and you can easily get the solution through ChatGPT (or *Googling*)
- However, remember that if you start relying on ChatGPT from now on, it will eventually limit your capabilities
  - And you may end up believing that ChatGPT is smarter than you
- In contrast, if you continue to address challenging problems independently, you will soon surpass ChatGPT

# Problem Information

- **Three problems in total**
  - Problem 1-1: **30 points**
  - Problem 1-2: **35 points**
  - Problem 1-3: **35 points**
- **You will receive points for each problem based on the number of test cases that your code passes**

# Submission Guideline

- You should submit three C source files
  - Problem 1-1: `bitMask.c`
  - Problem 1-2: `absVal.c`
  - Problem 1-3: `conditional.c`
- If the submitted file does not compile using the "make" command, **no points are given** for that problem
- **Submission format**
  - Upload the three files listed above directly to *Cyber Campus* (**do not compress them into a zip file**)
  - **Do not change the file names** (e.g., by adding any prefixes or suffixes)
  - If your submission format is incorrect, you will get a **-20% penalty**