Lab Intro & NAND Simulator

Prof. Dongkun Shin (dongkun@skku.edu)
TA – Youngjae Lee(yjlee4154@gmail.com)
TA – Jeeyoon Jung(wjdwldbs1@skku.edu)
Embedded Software Laboratory
Sungkyunkwan University
http://nyx.skku.ac.kr

About TA

- Embedded Software Laboratory
- Office : Semiconductor Bldg. #400309 (3rd floor)
- Youngjae Lee(이영재)
  - yjlee4154@gmail.com
- Jeeyoon Jung(정지윤)
  - wjdwldbs1@skku.edu
- When email us, start the email subject lines with "[ICE3028]" 😀

Contents

- Q&A google docs
  - https://docs.google.com/spreadsheets/d/1N4h9DyIDWYdLa6oIPyEnr1By11Zxsnk-L517lQzh6G0

- Lab Overview & Notifications
  - Jasmine OpenSSD Platform
  - Schedule may subject to change

- NAND simulator

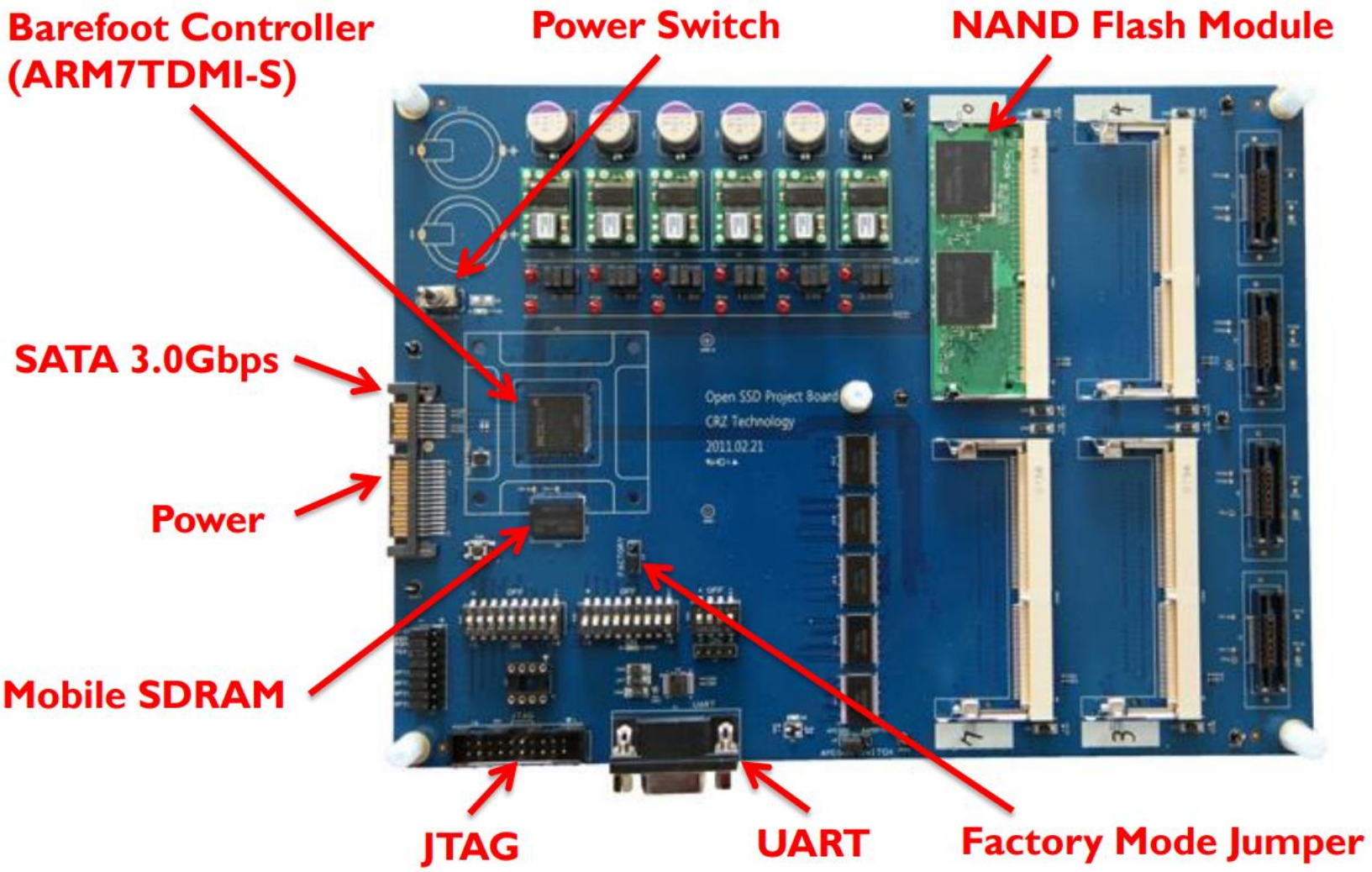
Lab Schedule

- Schedule may subject to change

week	Tuesday	Thursday
1	Course Overview	Intro to Embedded Systems
2	NAND Flash Memory I	NAND Flash Memory II
3	FTL I	Lab Intro. & NAND Simulator
4	FTL II	Page Mapping FTL 1
5	FTL III	Page Mapping FTL 2
6	SSD Technologies I	DFTL
7	SSD Technologies II – Advanced Topics	ZNS
8	SSD Technologies III – NVMe/eMMC/UFS	Project 1: ZNS+ Simulator
9	New Memory Technologies	File Systems I
10	File Systems II	Jasmine Intro (11/4)
11	Dummy FTL	Page Mapping FTL 3
12	DFTL at Jasmine	Project 2: ZNS at Jasmine
13	Project 3: ZNS+ at Jasmine	Project Q&A
14	Project Presentation I	Project Presentation II
15	Final Exam	

이론수업 실습수업 (보드 미사용) 실습수업 (보드 사용)

Jasmine OpenSSD Platform



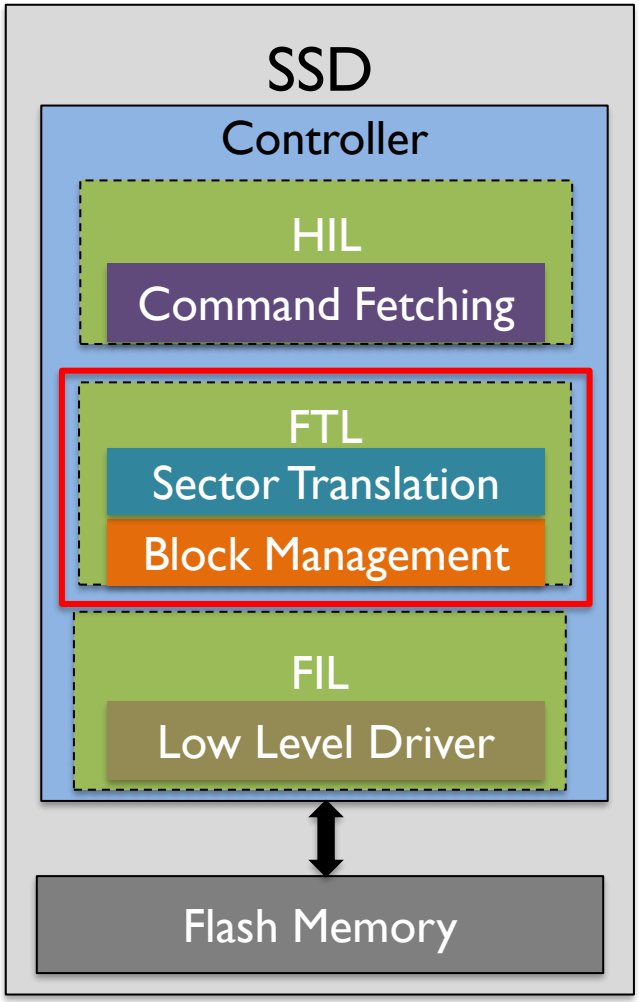
The OpenSSD Project

- It is an initiative to promote research and education on recent SSD technology

- Providing OpenSSD platforms on which open source SSD firmware can be developed

Jasmine OpenSSD Platform

- We will cover the firmware embedded in SSDs
  - Flash Translation Layer(FTL)
    - ZNS and its improvement (Project)
  - One of the most important layers in SSD firmware
- A large part of DS Division's software engineers.
  - Memory Business



Prerequisites

- You should be fluent in C Programming!!
  - Including basic system Programming
  - Ability to read and understand prewritten source code
- All labs are linked until the end of the semester.
- It's not easy and you have to spend a lot of time

Team project

- Lab & Project using SW Simulator
  - Personal project

- Lab & Project using Jasmine OpenSSD
  - 3 people in one group
  - Before start OpenSSD lab(week 10, offline class) , make team and write your team name & member in Google sheet!
  - If not, randomly assigns.

Notifications

- Do not COPY
  - One copy mean 'F' score
- Please let me know if (e-mail, hangout)
  - Something goes wrong
  - Stupid question (Good!!)
  - Suggestion
  - Someone cheating?
  - Better Idea
  - About Project or Lab
  - Use google sheet to share your good question!! 

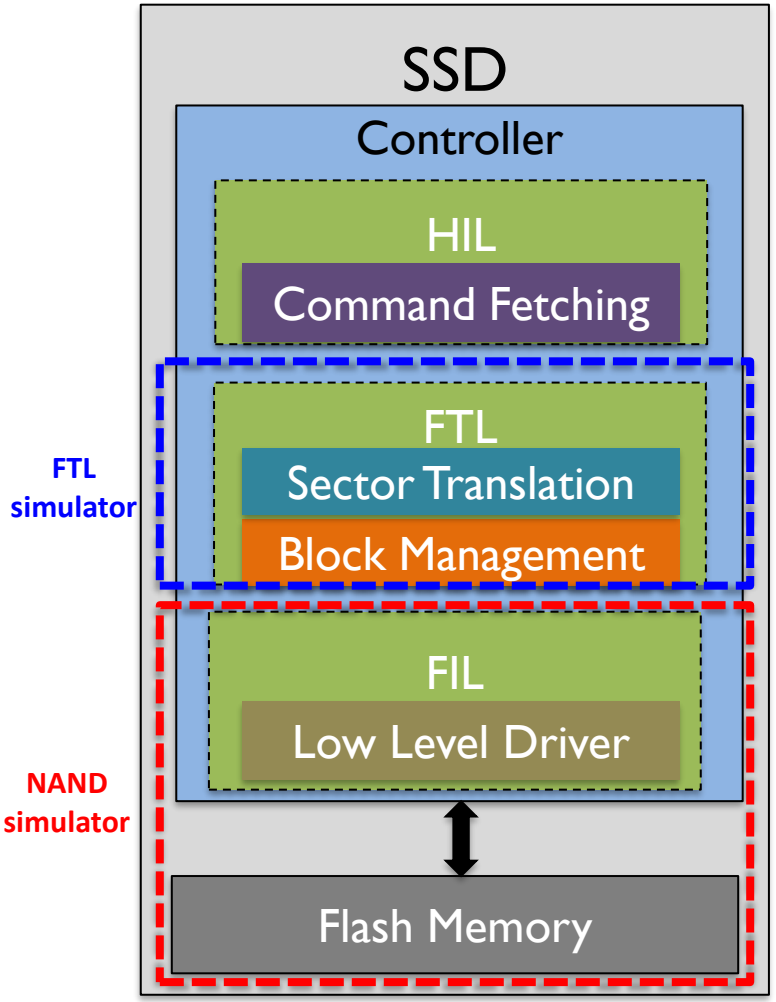
Reference

- Interesting Posts
  - https://tech.kakao.com/2016/07/13/coding-for-ssd-part-1/

- Related Paper
  - **A Survey of Address Translation Technologies for Flash Memories**
  - **DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mapping**
  - **ZNS+: Advanced Zoned Namespace Interface for Supporting In-Storage Zone Compaction**

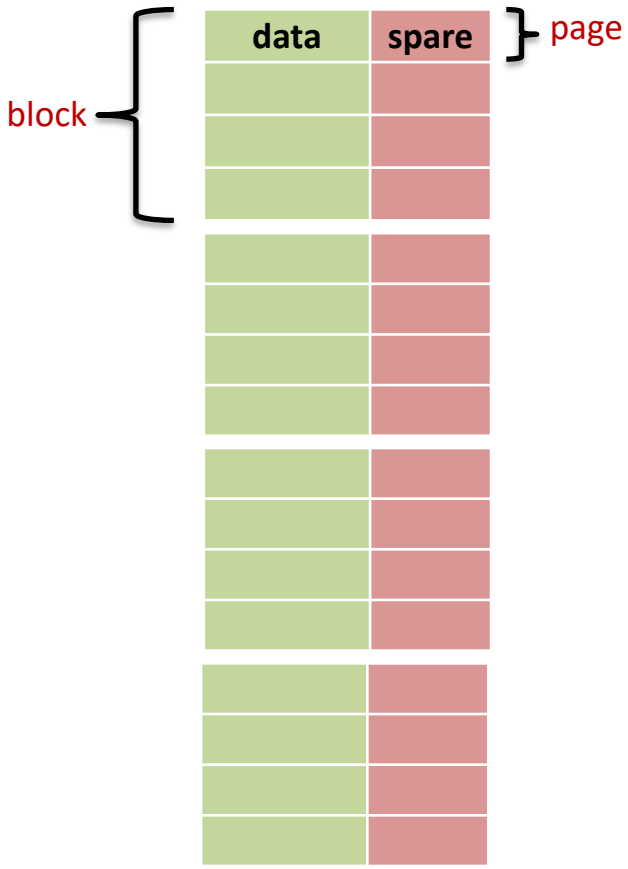
SSD Simulator

- NAND simulator
  - Basic NAND operations
    - Program
    - Erase
    - Read
- FTL simulator
  - Provide block device interface
    - Logical to physical mapping
    - Garbage collection



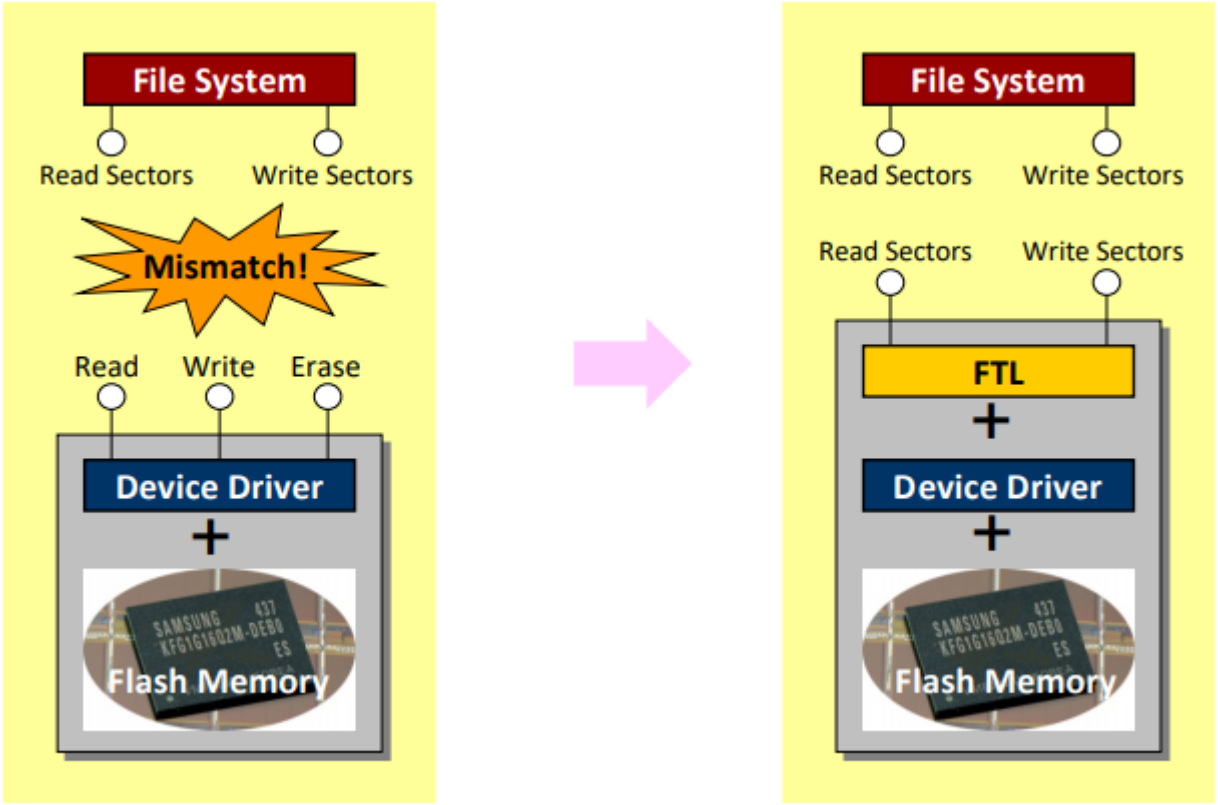
NAND Flash Memory

- Erase-before-write
- Bulk erase
  - Program unit: page
  - Erase unit: block
- Sequential write in a block



Flash Translation Layer

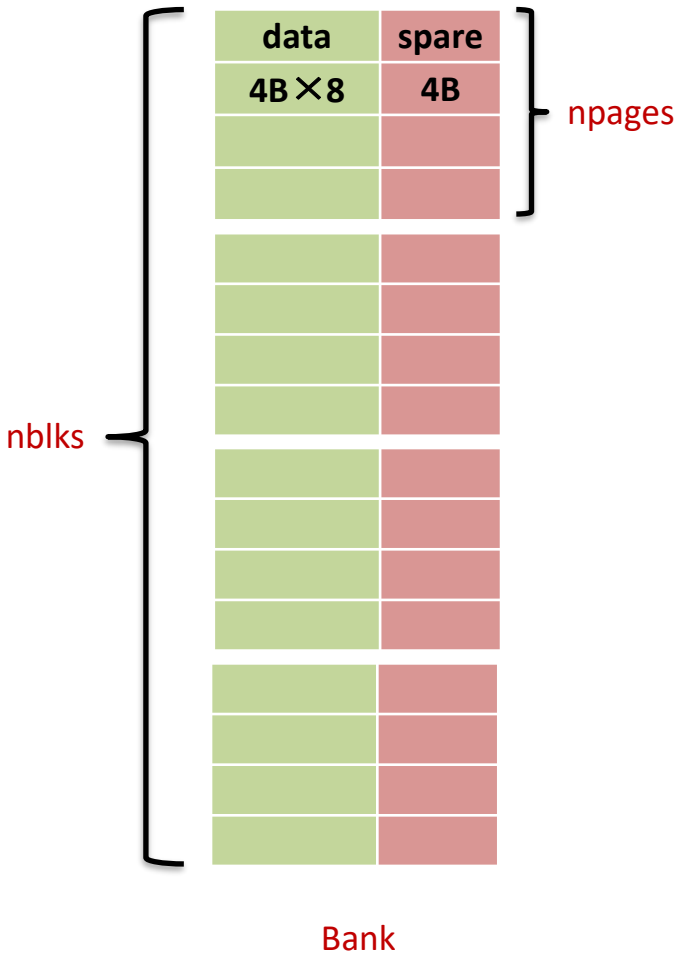
- A software layer to make NAND flash fully emulate traditional block devices



Source: Zeen Info. Tech.

Lab 1: NAND Simulator

- Develop a NAND simulator
  - Simulate NAND flash memory
    - Page size: 32B for data / 4B for spare
  - Functions to implement:
    - nand_init()
    - nand_read()
    - nand_write()
    - nand_erase()
  - The skeleton code is available at icampus



nand_init()

- nand_init(*nbanks*, *nblks*, *npages*)
  - Description
    - Initialize your own NAND flash memory using DRAM
    - Initially, all flash memory blocks are in erased state.
    - Return appropriate error code
  - Argument
    - nbanks: the total # of banks (should be > 0)
    - nblks : the total # of blocks (should be > 0)
    - npages : # of pages per block (should be > 0)
  - Return value (defined in `nand.h`)
    - `NAND_SUCCESS`: success
    - `NAND_ERR_INVALID`: invalid dimension (negative value)

nand_write()

- nand_write(*bank, blk, page, data, spare*)
  - Description
    - Write 'data' and 'spare' to the NAND page pointed by 'blk' and 'page'
    - Return appropriate error code
  - Argument
    - bank, blk, page : address of the flash memory
    - data, spare : data to store
  - Return value
    - `NAND_SUCCESS`: success
    - `NAND_ERR_INVALID`: invalid address
    - `NAND_ERR_OVERWRITE`: page is already written
    - `NAND_ERR_POSITION`: write position is wrong (not sequential)

nand_read()

- nand_read(*bank, blk, page, data, spare*)
  - Description
    - Read 'data' and 'spare' from NAND page pointed by 'blk' and 'page'
    - Return appropriate error code
  - Argument
    - bank, blk, page : address of the flash memory
    - data, spare : data to load
  - Return value
    - `NAND_SUCCESS`: success
    - `NAND_ERR_INVALID`: invalid address
    - `NAND_ERR_EMPTY`: empty page (not written yet)

nand_erase()

- nand_erase(*bank*, *blk*)
  - Description
    - Erase the NAND memory block pointe by `bank` and `blk`
    - Return appropriate error code
  - Argument
    - bank, blk : Address of the NAND flash memory
  - Return value
    - `NAND_SUCCESS`: success
    - `NAND_ERR_INVALID`: invalid address
    - `NAND_ERR_EMPTY`: already erased block

Test

- Use `nand_test` tool to verify your code
  - Use `make` to build `nand_test`
  - Usage: nand_test [input_file] [output_file]

```
yj:lab1_sol ᚠ master ⅄ ./nand_test input.txt
Init(2,8,8): success
Write(0,3,0): success
Write(0,3,1): success
Write(0,3,2): success
Write(0,3,3): success
Write(0,3,4): success
Write(0,3,5): success
Write(0,3,6): success
Write(0,3,7): success
Write(-1,0,0): failed, invalid address
Read(0,3,0): success, data = [       11       22       33       44       55       66       77       88 ] spare = [       ff ]
Read(0,3,1): success, data = [       11       22       33       44       55       66       77       88 ] spare = [       ff ]
Read(0,3,2): success, data = [       11       22       33       44       55       66       77       88 ] spare = [       ff ]
Read(0,3,3): success, data = [       11       22       33       44       55       66       77       88 ] spare = [       ff ]
Read(0,3,4): success, data = [       11       22       33       44       55       66       77       88 ] spare = [       ff ]
Read(0,3,5): success, data = [       11       22       33       44       55       66       77       88 ] spare = [       ff ]
Read(0,3,6): success, data = [       11       22       33       44       55       66       77       88 ] spare = [       ff ]
Read(0,3,7): success, data = [       11       22       33       44       55       66       77       88 ] spare = [       ff ]
Erase(0,3): success
Read(0,3,0): failed, trying to read empty page
Read(0,3,1): failed, trying to read empty page
Read(0,3,2): failed, trying to read empty page
Read(0,3,3): failed, trying to read empty page
Read(0,3,4): failed, trying to read empty page
Read(0,3,5): failed, trying to read empty page
Read(0,3,6): failed, trying to read empty page
Read(0,3,7): failed, trying to read empty page
Write(0,2,0): success
Write(0,2,1): success
Write(0,2,3): failed, the page is not being sequentially written
Write(0,2,2): success
```

Grading Policy

- Recommended environment : GCC on Linux
  - You can do it in Windows, but be sure that your work also runs in Linux
  - Use only standard C library or POSIX C library functions
- Personal Project
- Submissions will be graded based on the number of test cases passed
  - We will use larger test cases for scoring
- Submit to the icampus
  - Due: 9/22(Wed.) 23:59:59
  - Submission file name: `<student_id>.tar.gz` (includes `nand.c` only)
  - Modify student id in `Makefile` and use `make submit` command
- Late penalty : -20 % / day (Up to 3 days)

Any Questions?