Lab Introduction & NAND Simulator

Prof. Dongkun Shin (dongkun@skku.edu)
TA – Junho Lee (crow6316@skku.edu)
TA – Somm Kim (sommkim@skku.edu)
Embedded Software Laboratory
Sungkyunkwan University
http://nyx.skku.ac.kr

About TA

- Embedded Software Laboratory
- Office : Semiconductor Bldg. #400309 (3rd floor)
- Junho Lee (이준호)
  - crow6316@skku.edu
- Somm Kim (김솜)
  - sommkim@skku.edu
- When email us, start the email subject lines with "[ICE3028]" ☺

Contents

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

Lab Overview & Notifications

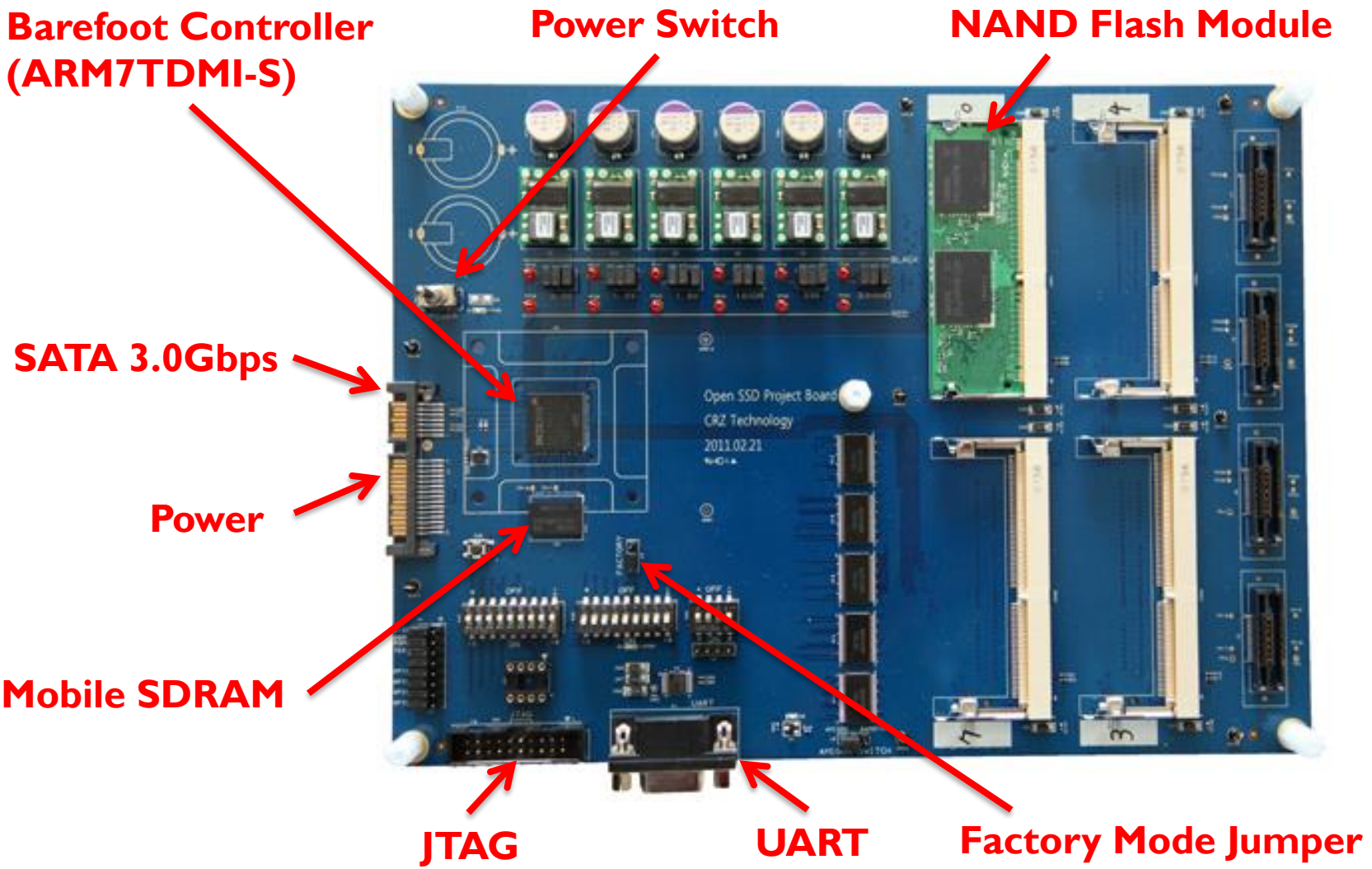
Lab Schedule

- Schedule may subject to change.

Week	Tuesday	Thursday
1	Course Overview	Introduction to Embedded Systems
2	NAND Flash Memory I	*(National Holiday)*
3	NAND Flash Memory II	Lab Introduction, NAND Simulator
4	FTL I	Jasmine Board Setting
5	Dummy FTL, Tutorial FTL	*(National Holiday)*
6	FTL II	Greedy FTL
7	*(No Lecture)*	FTL Testing (Jasmine)
8	FTL III	**(Midterm Exam)**
9	SSD Technologies I	DFTL
10	SSD Technologies II	Multi-Streamed SSD
11	New Memory Technologies	Jasmine Porting Guide
12	File Systems I	F2FS
13	File Systems II	DFTL Optimization Method
14	Reserved	Project Q&A
15	Project Presentation I	Project Presentation II
16	**(Final Exam)**	

Solid State Drive (SSD)

Jasmine OpenSSD Platform



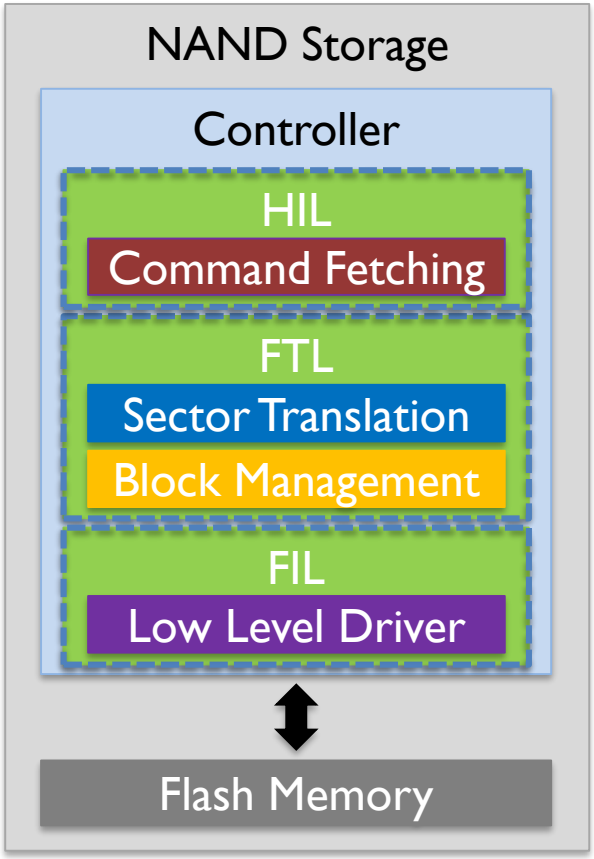
The OpenSSD Project

- It is an initiative to promote research and education on the 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)
    - DFTL and its improvements (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.

Make Groups!

- S/W FTL Simulator : Personal projects

- OpenSSD Jasmine : Team projects (2 people)
  - Choose a team member by next week.
  - If not, randomly assigns.

Notifications

- Do not 'copy'
- Please let me know if (e-mail & hangout)
  - Something goes wrong
  - Stupid questions (Welcome!)
    - We will be managing a Google Sheet for questions.
  - Suggestions
  - Someone cheating?
  - "I have better ideas"
  - The course is too easy

Reference

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

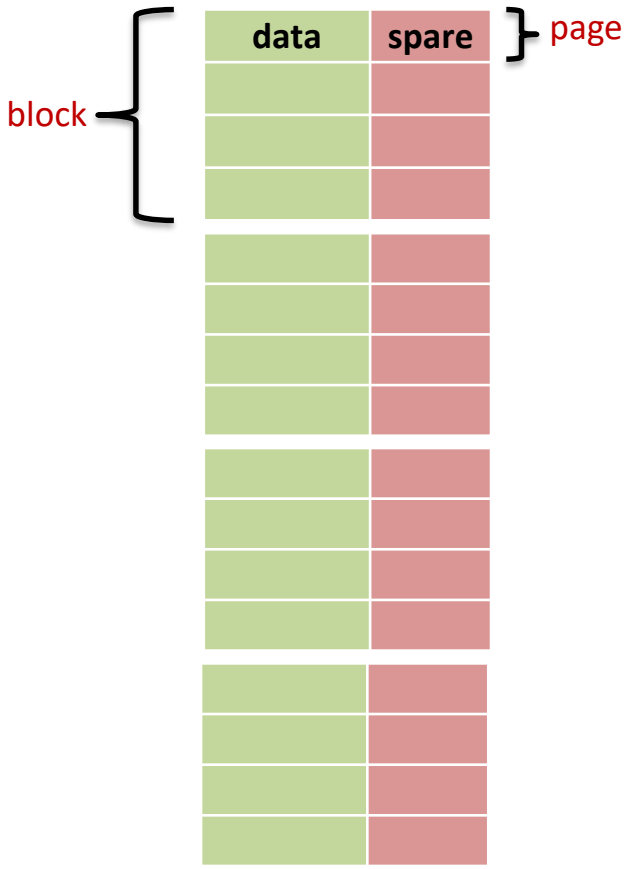
- Related Paper
  - D. Ma, J. Feng, and G. Li, "**A Survey of Address Translation Technologies for Flash Memories**" ACM Computing Surveys, Vol. 46, No. 3, January 2014.
  - A. Gupta, Y. Kim, and B. Urgaonkar, "**DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings**" Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2009.

Any Questions?

NAND Simulator

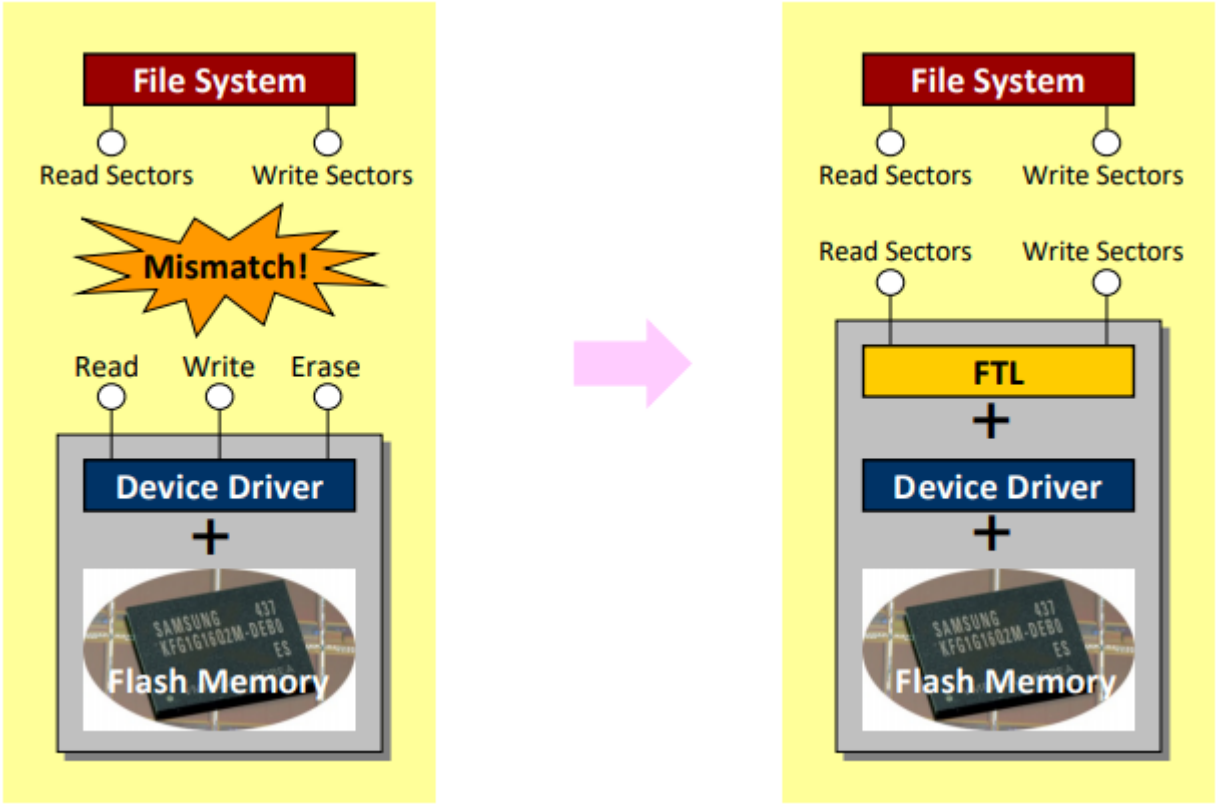
NAND Flash Memory

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



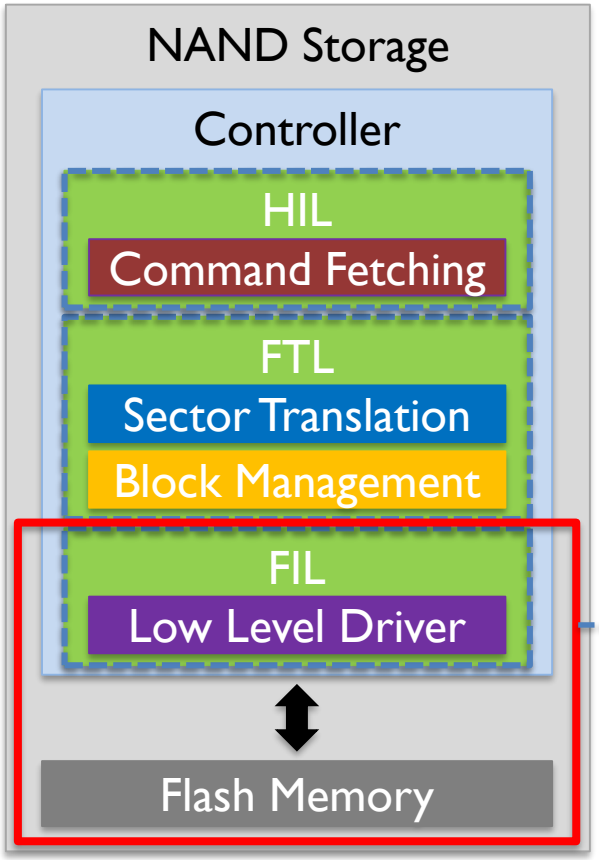
Flash Translation Layer

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



Source: Zeen Info. Tech.

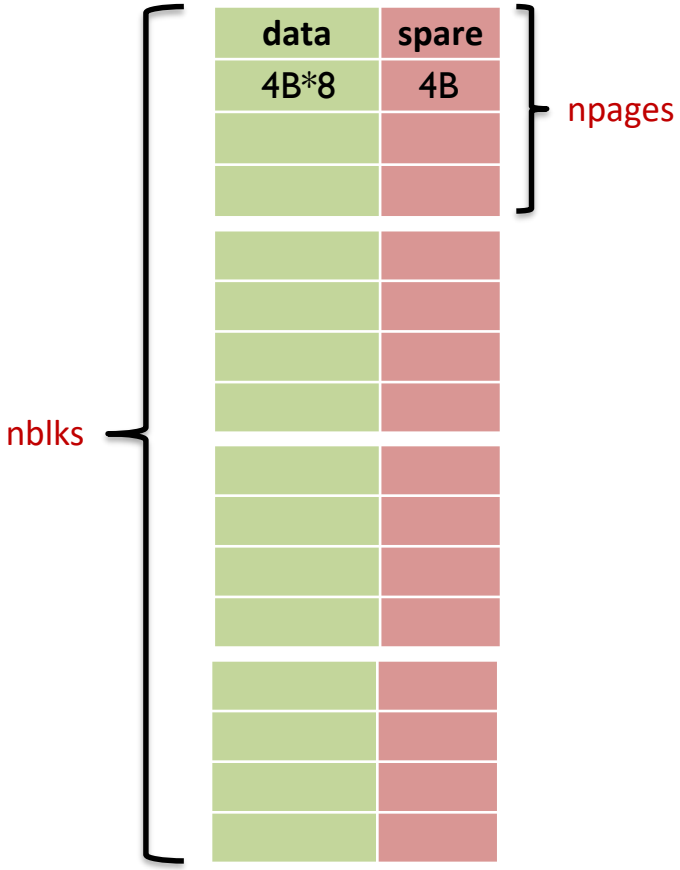
Lab Outline (without board)



Lab #1 : NAND simulator

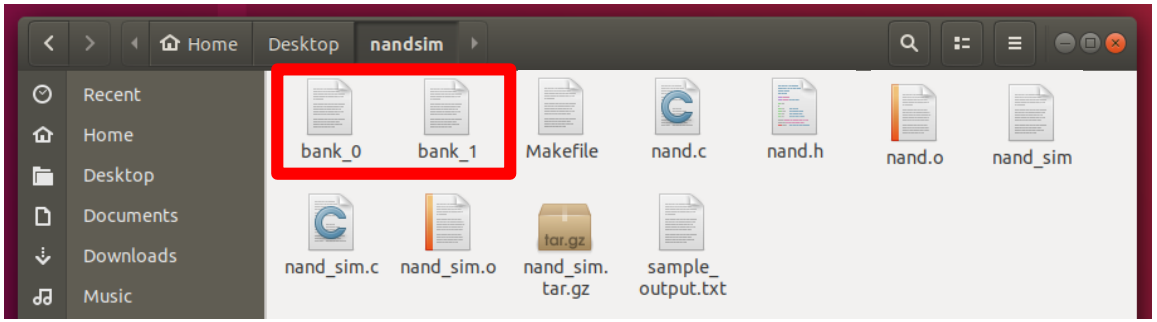
Lab 1 : NAND Simulator

- Develop a NAND simulator
  - Simulate NAND flash memory using file
    - use Linux system call (open(), read(), write(), lseek(), ...)
  - 4B(32bit) for sector size
  - Sectors per page: 8
  - 4B*8 for data / 4B for spare
  - Functions to implement:
    - nand_init()
    - nand_read()
    - nand_write()
    - nand_erase()
    - nand_blkdump()
  - The skeleton code is available at icampus.



nand_init()

- nand_init(*nbanks*, *nblks*, *npages*)
  - Description
    - Initialize your own NAND flash memory using file per 'bank'
    - Initial state of the flash memory is 'all-blks-erased'
    - If success, print initialized information of the flash memory
    - If not, print appropriate error message (reason for the error)
  - Argument
    - nbanks: the total # of banks (should be > 0)
    - nblks : # of blocks per bank (should be > 0)
    - npages : # of pages per block (should be > 0)
  - Return value
    - Return 0 on success
    - Return -1 on errors



nand_write()

- nand_write(*bank, blk, page, *data, spare*)
  - Description
    - Write 'data' and 'spare' to the file
    - If success, print written data (only first sector) and spare
    - If not, print appropriate error message (reason for the error)
  - Argument
    - bank, blk, page : address of the flash memory
    - data, spare : data to store
  - Return value
    - Return 0 on success
    - Return -1 on errors

nand_read()

- nand_read(*bank, blk, page, *data, *spare*)
  - Description
    - Read 'data' and 'spare' from the file
    - If success, print read data and spare
    - If not, print appropriate error message (reason for the error)
  - Argument
    - bank, blk, page : address of the flash memory
    - data, spare : data to load
  - Return value
    - Return 0 on success
    - Return -1 on errors

nand_erase()

- nand_erase(*bank*, *blk*)
  - Description
    - Erase the NAND flash memory block 'blk' at bank 'bank'
    - If success, print appropriate message with 'bank' and 'blk'
    - If not, print appropriate error message (reason for the error)
  - Argument
    - bank, blk : address of the NAND flash memory
  - Return value
    - Return 0 on success
    - Return -1 on errors

nand_blkdump()

- nand_blkdump(*bank*, *blk*)
  - Description
    - Dump the contents of the flash memory block 'blk' at bank 'bank' (for debug)
    - If success, print appropriate message
    - If not, print appropriate error message (reason for the error)
  - Argument
    - bank, blk : address of the NAND flash memory
  - Return value
    - Return 0 on success
    - Return -1 on errors

Sample Output

- Do not change the format of the output message.
  - See the sample_output.txt file

```
init: 2 banks, 8 blocks, 8 pages per block
write(0,3,0): data = 0x99999999, spare = 0x00000018
write(0,3,1): data = 0x22222222, spare = 0x00000020
write(0,3,2): data = 0xaaaaaaaa, spare = 0x00000028
write(0,3,3): data = 0x33333333, spare = 0x00000030
write(0,3,4): data = 0xbbbbbbbb, spare = 0x00000038
write(0,3,5): data = 0x44444444, spare = 0x00000040
write(0,3,6): data = 0xcccccccc, spare = 0x00000048
write(0,3,7): data = 0x55555555, spare = 0x00000050
read(0,3,7): data = 0x55555555, spare = 0x00000050
read(0,3,6): data = 0xcccccccc, spare = 0x00000048
read(0,3,5): data = 0x44444444, spare = 0x00000040
read(0,3,4): data = 0xbbbbbbbb, spare = 0x00000038
read(0,3,3): data = 0x33333333, spare = 0x00000030
read(0,3,2): data = 0xaaaaaaaa, spare = 0x00000028
read(0,3,1): data = 0x22222222, spare = 0x00000020
read(0,3,0): data = 0x99999999, spare = 0x00000018
write(0,4,0): data = 0x22222222, spare = 0x00000020
write(0,4,0): failed, the page was already written
write(0,4,2): failed, the page is not being sequentially written
read(0,4,3): failed, trying to read an empty page
erase(0,4): block erased
erase(1,0): failed, trying to erase a free block
```

```
write(0,4,0): data = 0x22222222, spare = 0x00000020
write(0,4,1): data = 0xaaaaaaaa, spare = 0x00000028
write(0,4,-1): failed, invalid page number
read(0,4,-1): failed, invalid page number
write(0,-1,0): failed, invalid block number
read(0,-1,0): failed, invalid block number
erase(0,-1): failed, invalid block number
erase(0,8): failed, invalid block number
write(-1,0,0): failed, invalid bank number
read(-1,0,0): failed, invalid bank number
blkdump(0,3): Total 8 page(s) written
blkdump(0,3,0): data = 0x99999999, spare = 0x00000018
blkdump(0,3,1): data = 0x22222222, spare = 0x00000020
blkdump(0,3,2): data = 0xaaaaaaaa, spare = 0x00000028
blkdump(0,3,3): data = 0x33333333, spare = 0x00000030
blkdump(0,3,4): data = 0xbbbbbbbb, spare = 0x00000038
blkdump(0,3,5): data = 0x44444444, spare = 0x00000040
blkdump(0,3,6): data = 0xcccccccc, spare = 0x00000048
blkdump(0,3,7): data = 0x55555555, spare = 0x00000050
blkdump(0,4): Total 2 page(s) written
blkdump(0,4,0): data = 0x22222222, spare = 0x00000020
blkdump(0,4,1): data = 0xaaaaaaaa, spare = 0x00000028
blkdump(1,4): FREE
```

Miscellaneous

- Recommended environment : Linux (Ubuntu is ok!)
  - You can do it in Windows, but be sure that your work also runs in Linux (I'll score all the works only in Ubuntu 16.04)
- Personal Project
- Submit to the icampus
  - Due: 9/25(Wed.) 23:59:59
  - File name: student_id.tar.gz (modify 'STUDENT_ID' at Makefile)
  - Submit as tar.gz file (use 'make tar' command)
- Late penalty : -20% per day (Up to 3 days)

Any Questions?