3 차 과제

A. 플래시 메모리 구성

B. 섹터와 블록 매핑 알고리즘 구현

- 가정해야 할 부분이 존재한다면 추가 구현할 것 (여분 블록, 추가 DRAM 등 가정 가능)

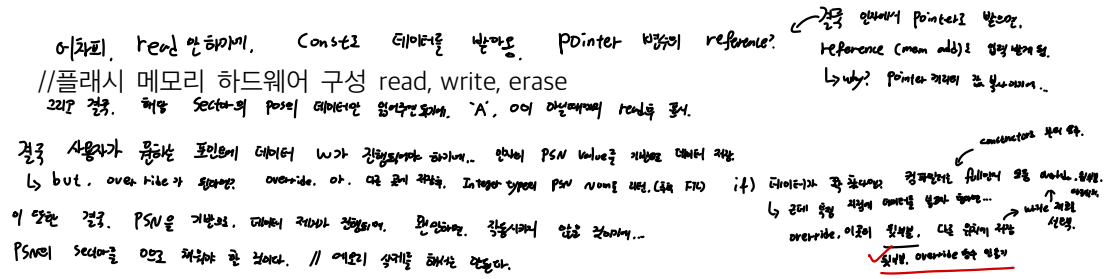
구현 함수:

init ( ); //플래시 메모리 생성

Flash_read ( ); //플래시 메모리 하드웨어 구성 read, write, erase

Flash_write ( );

Flash_erase ( );



Input 명령어:

init megabytes // x megabytes 플래시 메모리 생성

read PSN // 해당 PSN 에서 데이터 읽어오기

write PSN data // 해당 PSN 섹터에 데이터 적기

결과값:

init 의 ouput: x megabytes flash memory ;

read 의 output: PSN 의 **data return**

write 의 output: write 가 수행된 PSN, 데이터 표시

1 block = 32 Sector = 32 X 512 byte.
= 16384 byte.

1 MB = 1048576 byte. = 2048 Sector. = 64 block
= $(2^{10})^2$ = $(1024)^2$ byte. = 1024 x 2 x 512 byte.

1. 실제 스토리지 공간 처럼 RAW 메모리 할당. (Init)
Input x 1024X1024.
↳ char * data;
data = (char *) malloc (1024 X1024); ← bytes
Pointer + char (two) overhead 계산할것.
= 1 mb allocation. then... how to find per sector...

2. byte 단위의 데이터 공간을
어떻게. Sector. block 단위로
나눌 것인가?

결과는 결국 최종 사용자가 사용하는 함수에서
1 Sector 선택이 가능해야 한다...
그렇다면 512 byte 단위로 데이터 공간을 어떻게
나눌 것인가?
struct 라해도. array 단위로 잘라지기에. struct 내 char *data 512 byte설정! (← array로 할당.)
↳ 그렇다면, 추후 데이터 엑세스는 어떻게 처리 할것인가?
struct [0]. data.
read? while (const auto &i : struct [Input].data)

C. 보고서 작성

1. 요구 분석 (2장이상)

2. 설계 (3장이상)

Flowchart로 보여주고, 각 과정 설명

3. 구현 (3장이상)

각 함수 설명

input:

output:

함수 역할

중요 소스코드 라인 설명

4. 테스트 결과 (1장이상)