

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
2212 결국, 해당 Sector의 PSN의 데이터만 읽어오면 되는데, 'A', 0이 아닌 값이면 read를 못해.

`Flash_write();` 결국 사용자와 관련된 데이터 w가 진행되어야 하는데.. 한이 PSN 번호를 기반으로 데이터 저장
↳ but, override가 필요함. override. 아. 다들 굳이 왜냐. Intra-type PSN 번호로 하면 (즉 SW) if) 데이터가 꼭 존재했

`Flash_erase();` 이 같은 경우, PSN을 기반으로, 데이터 제바라 진행되어야. 관용적인. 검증시켜야 함을 뜻하네...
PSN의 Sector를 0으로 재작성 가능. // 메모리 상계를 해산 가능함.

← 결국 인덱스에서 pointer로 참조함, reference (mem addr)로 접근 가능함.
↳ why? pointer 처리가 좀 복잡하네...
← constant가 문제야.
← 데이터가 꼭 존재했. 컴파일러가 시계열 흐름 시키는 것임. 최신형, 아날로그는 항상 역순
↳ 근데 특정 작업에 memory를 할당 할때... 나가는 처리 신경
override, 이론이 뒷받침, 다른 영역에 저장

Input 명령어:

`init megabytes` // x megabytes 플래시 메모리 생성

`read PSN` // 해당 PSN에서 데이터 읽어오기 'A', 0이 아닌 값이면 처리.

`write PSN data` // 해당 PSN 섹터에 데이터 적기

결과값:

init 의 output: x megabytes flash memory ;

read 의 output: PSN 의 **data return**

write 의 output: write 가 수행된 PSN, 데이터 표시

| block = 32 sector = 32 x 512 byte.
= 16384 byte.

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

1. 실제 스토리지 공간 사용 RAM 메모리 중. (Init)
↳ char * data; Input x 1024 x 1024.
data = (char *) malloc (1024 x 1024); bytes
= 1mb allocation. then... how to find per sector...
pointer + char (too) overhead 계산함.

2. byte 단위의 데이터 공간을

어떻게. sector. block 단위로

나눌 것인가?

결과를 결국 활용 사용자를 사용하는 환경에서

| sector 선택이 가능해야 함...

그렇다면 512 byte 단위로 데이터 공간을 어떻게

나눌 것인가?

struct 자료는 array 형태로 존재하며, struct 이 char * data. 1024 bytes임! ← 배열로 잘못

↳ 그렇다면, 특히 데이터 역산은 어떻게 해야 할까요?

struct [0]. data.

head? 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)