프로그래밍 역량 강화 전문기관, 민코딩

SSD 프로젝트 소개

목차

SSD Test Shell 개요

SSD Test Shell Project 시작

Test Shell

SSD 제품을 테스트 할 수 있는 Test Shell 을 제작

- SSD 를 가상으로 프로그래밍으로 구현한다.
- Test Shell 프로그램을 제작하여 SSD 동작을 테스트 할 수 있다.
- 다양한 Test Script를 제작한다.

Test 수행

프로젝트 구성

1. 가상 SSD
   - HW 대신, Software로 구현한다.

1. Test Shell Application
   - 테스트 프로그램

2. Test Script
   - 테스트 프로그램 內 Test Code

SSD 사전 지식

저장할 수 있는 공간

- 저장할 수 있는 최소 공간의 사이즈는 4KB
  (한 글자 = 약 1 Byte 으로 간주했을 때, 4,096 글자 저장 가능 공간)

- 각 공간마다 LBA (Logical Block Address) 라는 주소를 가짐

- SSD 는 OS에서 File System 을 거쳐
  Read / Write / Sync / Unmap 등 다양한 명령어를
  수행한다.

가상 SSD 제작

HW가 해야할 역할을 대신하여, 가상의 SSD를 SW로 구현한다.

구현해야 할 가상 SSD

최소화된 기능 수행

- Read 명령어와 Write 명령어만 존재
- LBA 단위는 4 Byte
- LBA 0 ~ 99 까지 100 칸을 저장할 수 있다.

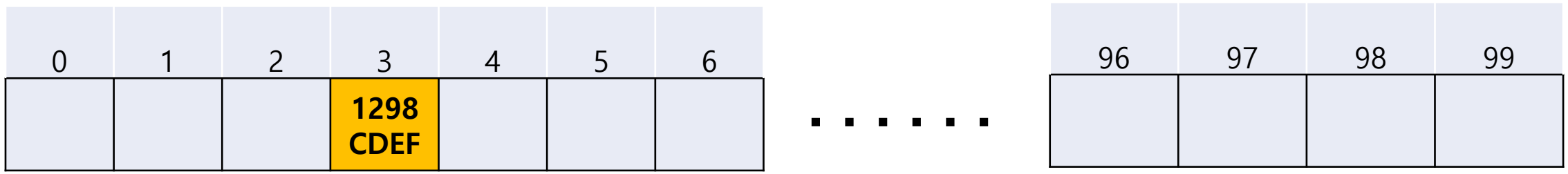
총 400 Byte를 저장 할 수 있는 가상 SSD 를 구현

Write 명령어 동작 예시 1

APP 이름 : ssd

Write 명령어 사용 예시 1

- ssd W 3 0x1298CDEF : 3 번 LBA 영역에 값 0x1298CDEF 를 저장한다.

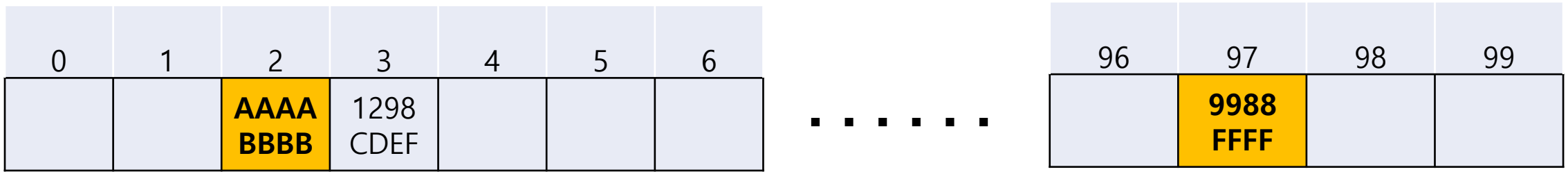


Write 명령어 동작 예시 2

Write 명령어 사용 예시 2

- ssd W 2 0xAAAABBBB
- ssd W 97 0x9988FFFF

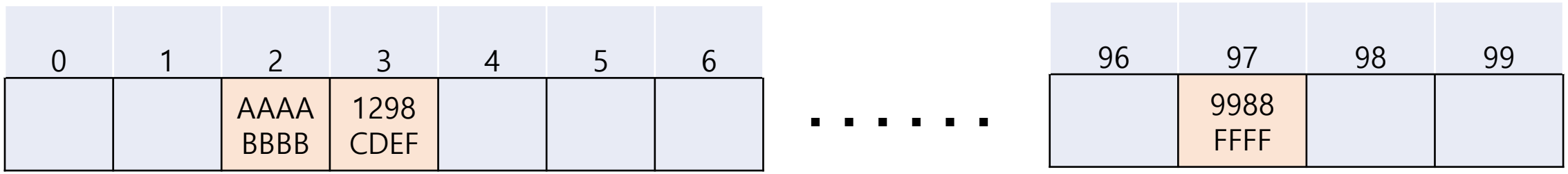
출력결과 : 없음 (저장만 수행)



Read 명령어 동작 예시

Read 명령어 사용 예시

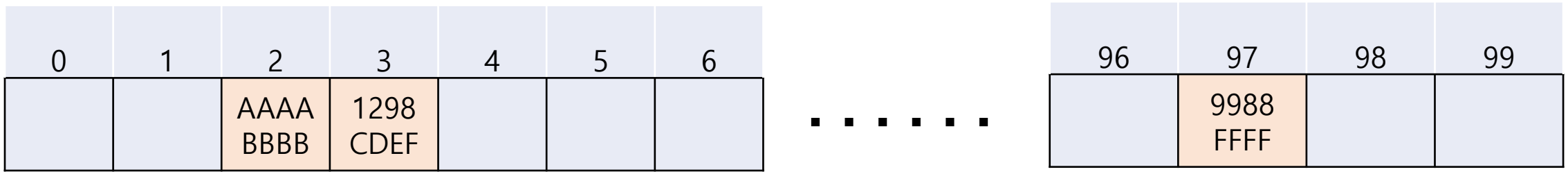
- ssd R 2
  - 출력결과 : 0xAAAABBBB
- ssd R 97
  - 출력결과 : 0x9988FFFF



실제로 저장하는 위치

nand.txt 파일을 생성

- 사용자가 Write 할 때 마다, SSD 내부 (Nand) 에 기록이 된다.
- 이를 모사하여, nand.txt 파일에 값을 저장 해 둔다.



출력결과

화면 출력하지 않고, result.txt 파일에 결과를 저장

- Write 명령어 수행시
  - result.txt 파일 건드리지 않음
  - Write는 내부적으로 기록만 수행한다.
- Read 명령어 수행시
  - result.txt 파일 내용이 교체 된다.

ssd 동작 예시

Read 명령 시 결괏값이 result.txt 파일에 저장
Write 명령 시 nand.txt 파일에 값을 저장

입력	출력
W 20 0x1289CDEF	-
R 20	0x1289CDEF
R 19	0x00000000
W 10 0xFF1100AA	
R 10	0xFF1100AA

세부 규정

데이터 범위

- LBA : 0 ~ 99, 10진수
- 값 : 항상 0x가 붙으며 10 글자로 표기한다. (0x00000000 ~ 0xFFFFFFFF)

Read 명령어

- ssd R [LBA]
- result.txt 에 읽은 값이 적힌다. (기존 데이터는 사라진다.)
- 한번도 안적은 곳은 0x00000000 으로 읽힌다.

Write 명령어

- ssd W [LBA] [값]
- nand.txt 에 저장한 값이 기록된다.

nand.txt 구현 방법

명령어를 사용할 때 마다,
nand.txt 내용 전체를 읽어온다.

W 명령어 사용시

- nand.txt 파일 내용 전체 읽어온 후,
  특정 부분을 변경하고 새로 Write를 수행한다.

R 명령어 사용시

- nand.txt 파일 내용 전체 읽어온 후,
  필요한 부분을 찾아내어 Return 한다.

Test Shell – App 개발

가상 SSD에게 명령을 내릴 수 있는 Shell 제작

Test Shell Application

SSD를 테스트 할 수 있는 프로그램

- Shell 이 동작하여 사용자 입력을 받는다.
- 사용 가능 명령어
  - write
  - read
  - exit
  - help
  - fullwrite
  - fullread

read / write 명령어 수행시,
제작한 "ssd" app을 실행시켜 값 읽기 / 저장 명령을 수행한다.

Test Shell 동작 예시

사용자 입력 예시

- write 3 0xAAAABBBB
  - 3번 LBA 에 0xAAAABBBB 를 기록한다.
  - ssd 에 명령어를 전달한다.

- read 3
  - 3번 LBA 를 읽는다.
  - ssd 에 명령어를 전달한다.
  - result.txt 에 적힌 결과를 화면에 출력한다.

exit / help

exit 명령어

- Shell 이 종료된다.

help 명령어

- 각 명령어당 사용 방법을 출력한다.

fullwrite / fullread

fullwrite 명령어

- LBA 0 번부터 99 번 까지 Write를 수행한다.
- ssd 전체에 값이 써진다.
- ex) fullwrite 0xABCDFFFF
  - 모든 LBA에 값 0xABCDFFF 가 적힌다.

fullread 명령어

- LBA 0 번부터 99 번 까지 Read를 수행한다.
- ssd 전체 값을 모두 화면에 출력한다.
- ex) fullread
  - 모든 LBA의 값들이 화면에 출력된다.

유의사항

기능 구현시 유의사항

- 입력받은 매개변수가 유효성 검사 수행
  - 파라미터의 Format이 정확해야 함
  - LBA 범위는 0 ~ 99
  - A ~ F, 0 ~ 9 까지 숫자 범위만 허용

- 없는 명령어를 수행하는 경우 "INVALID COMMAND" 을 출력
  - 어떠한 명령어를 입력하더라도 Runtime Error가 나오면 안된다.

Test Shell – Test Script 제작

제작한 Test Shell 안에서 동작되는 Test Script 제작하기

Full Write 후 ReadCompare 수행

TestApp1 제작하기

- Test Shell 에서 “testapp1” 명령어를 입력하면 Script가 수행된다.
- 먼저 fullwrite를 수행한다.
- fullread를 하면서, write 한 값대로 read가 되는지 확인한다.
  - SSD가 정상 동작하는지 확인하는 테스트 스크립트

Write Aging 후 Read Compare

TestApp2 제작하기

- 0 ~ 5 번 LBA 에 0xAAAABBBB 값으로 총 30번 Write를 수행한다.
- 0 ~ 5 번 LBA 에 0x12345678 값으로 1 회 Over Write를 수행한다.
- 0 ~ 5 번 LBA Read 했을 때 정상적으로 값이 읽히는지 확인한다.

프로젝트 전체 일정 정리

1 ~ 2일차 / 3 ~ 5 일차 프로젝트 진행 안내

CRA과정 프로젝트 진행 스케쥴

<table>
<tr><th colspan="2">시간</th><th>1일차</th><th>2일차</th><th>3일차</th><th>4일차</th><th>5일차</th></tr>
<tr><td>8:30</td><td>9:30</td><td rowspan="2">Team Building &
Communication</td><td rowspan="4">Team Project</td><td rowspan="4">프로젝트 안내
Team Project</td><td rowspan="4">Team Project</td><td rowspan="2">Team Project
*발표준비</td></tr>
<tr><td>9:30</td><td>10:30</td></tr>
<tr><td>10:30</td><td>11:30</td><td rowspan="2">Team Project
개발환경 셋업 및 안내</td><td rowspan="2">Team Project
발표, 평가 준비</td></tr>
<tr><td>11:30</td><td>12:30</td></tr>
<tr><td>12:30</td><td>13:30</td><td colspan="5">점심 시간</td></tr>
<tr><td>13:30</td><td>14:30</td><td rowspan="2">개발환경 안내
Team Project</td><td rowspan="4">Team Project</td><td rowspan="4">Team Project</td><td rowspan="4">Team Project</td><td rowspan="2">발표</td></tr>
<tr><td>14:30</td><td>15:30</td></tr>
<tr><td>15:30</td><td>16:30</td><td rowspan="2">Team Project</td><td rowspan="2">현업활동 Guide</td></tr>
<tr><td>16:30</td><td>17:30</td></tr>
</table>

1 ~ 2일차 – SSD 프로젝트 기반 개발

1. 프로젝트 소개
   (난이도 : 하 ~ 중)

1. 요구사항
   - TDD 개발 필수
   - Mocking 1개 이상 필수
   - 클린코드로 리팩토링

2. 3일차에 큰 기능 추가를 요청을 할 예정.
   깔끔한 코드로 리팩토링하여 클린코드로 만들 것.
   (기능추가 대비를 해두어야함.)

3 ~ 4일차 - 기능 추가요청

미션 (난이도 : 중 ~ 상)

1. SSD 명령어 추가
2. Log 관련 기능 추가
3. Runner 기능 추가
4. SSD 구조 개선
5. 재빌드 이슈 해결

5일차 : 발표준비 및 발표

오전

- ppt 준비 및 발표 준비

오후

- ppt 발표 및 마무리

팀 프로젝트 진행

1 ~ 2일차 / 3 ~ 5 일차 프로젝트 진행 안내

팀 프로젝트 이름 선정

✔각 팀들의 이름을 정해주세요.
✔5일차 발표 PPT 제목으로 활용예정입니다.

코딩마스터즈 팀의 PPT 표지 예시

코딩컨벤션 선정

- ✔ 팀에서 사용할 코딩컨벤션 (코딩가이드룰)을 설정합니다.
- ✔ 팀장님과 팀원 분들의 역할을 분배합니다.

TDD 개발

✔**1 ~ 2일차에는 TDD 개발 필수**

- 1일차 ~ 2일차 : TDD 개발
- 3일차 ~ 4일차 : TDD 없이 개발, 기능 추가 개발시에는 TDD를 하지 않습니다.

Test Double 사용하기

✔ **Unit Test 작성시, Test Double을 한번 이상 꼭 사용해주세요.**

- 점수에 반영됩니다.

SSD 프로젝트 시작

✔SSD 프로젝트를 시작해주세요

- 프로젝트 진행 / 기능 구현 / 디버깅 질문

  프로젝트 코치에게 언제든 질문주세요.

- 1주차 ~ 2주차 수업 내용 관련 질문

  담당 강사에게 email로 질문주세요.

  - 최인호 강사 : inho.choi@mincoding.co.kr
  - 서정환 강사 : jeonghwan.seo@mincoding.co.kr