# 어셈블리 프로그래밍 설계 및 실습
# 보고서

실험제목: Report Sample

실험일자: 2018년 09월 11일 (화)

제출일자: 2018년 09월 18일 (화)

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습 분반: 화 6,7 , 수 5

학 번: 2017202010

성 명: 박유림

# 1. 제목 및 목적

## A. 제목

Data transfer from/to memories

## B. 목적

예제를 통해 이해한 기본 명령어 사용법, ARM 조건부 실행 코드 보는 방법, 어셈블리어 프로그래밍 능력, 원하는 데이터를 메모리로 저장 및 가져올 수 있도록 어셈블리어 프로그래밍 능력을 사용하여 아래 두 과제를 구현한다.

첫번째는, 메모리에 저장된 숫자를 데이터와 비교하는 것이다.

두번째는, 메모리를 이용하여 원하는 값을 레지스터에 저장하는 것이다.

# 2. 설계 (Design)

## A. Pseudo code

1. Problem 1

```
If r1>0x0A
   r5=1
else if r1<0x0A
   r5=2
else
   r5=3

If r2>0x0A
   r5=1
else if r2<0x0A
   r5=2
else
   r5=3

If r3>0x0A
   r5=1
else if r3<0x0A
   r5=2
```

else

r5=3

2. Problem 2

r0=0x01

r1=0x02

r2=0x03

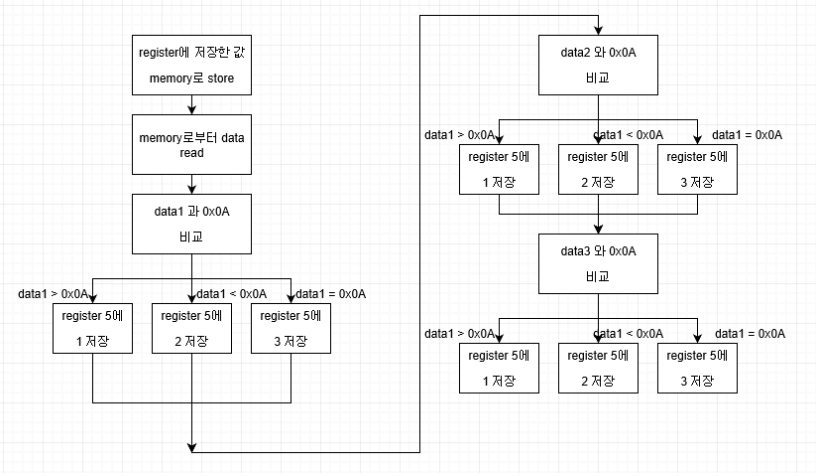
r3=0x04

r5={r3, r2, r1, r0}=0x04030201

r6={r0, r1, r2, r3}=ox01020304

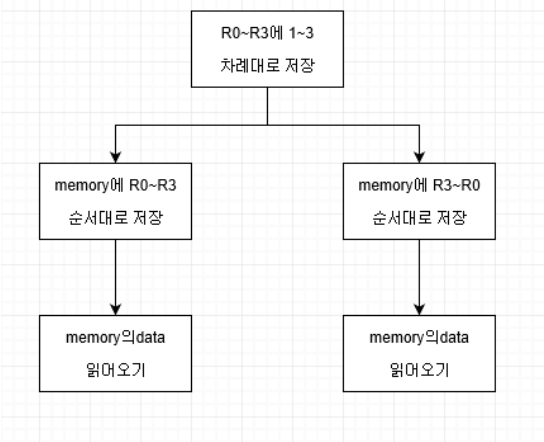
## B. Flow chart 작성

1. Problem1




Register-> memory-> register로 데이터를 옮기는 과정을 거친 후 각각 0x0A와 비교하여 알맞게 R5의 값을 바꾼다.

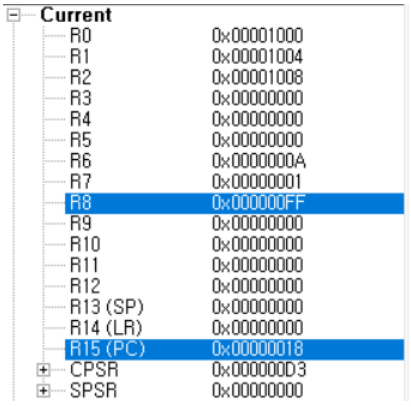
2. Problem2



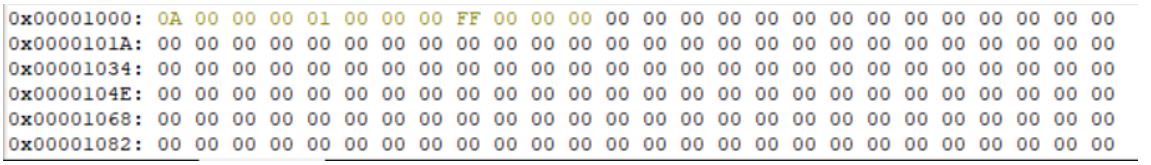

Little Endian 이라는 점을 이용해서 순서대로 혹은 역순대로 STRB로 1바이트씩 memory에 register에 들어 있는 data를 옮긴 후 LDR을 통해 한 word단위로 data를 빼 오면 문제가 해결된다.

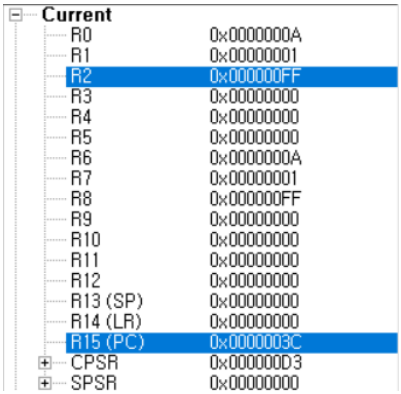
## C. Result

1. Problem1

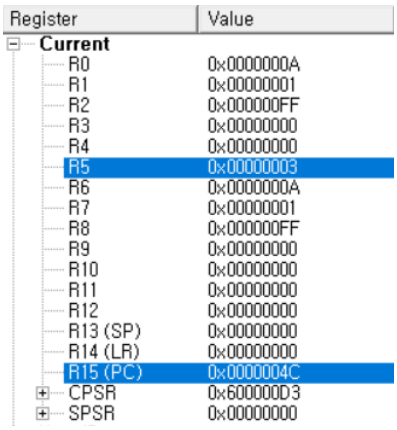



R0~R2까지 memory의 address값, R6~R8까지 memory에 저장할 값을 저장시켰다. 그리고 R0~R2에 저장된 address에 R6~R8값을 차례대로 넣어주었고 아래와 같이 memory에 데이터가 넣어졌다.

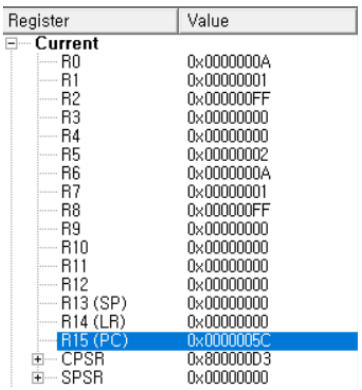




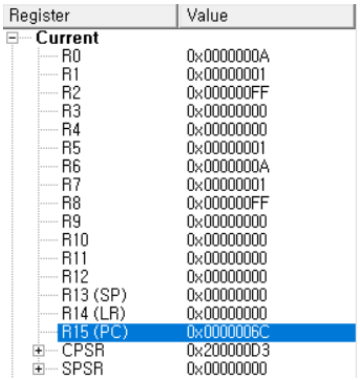

Memory에 저장한 데이터를 R0~R2 레지스터에 받아왔다.




0x0A는 0x0A와 비교했을 때, 같기 때문에 R5에 3이 들어갔다.

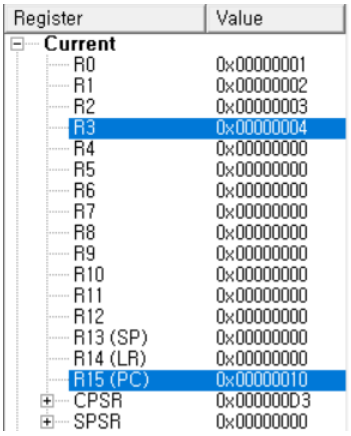



0x01은 0x0A와 비교했을 때, 작기 때문에 R5에 2가 들어갔다.

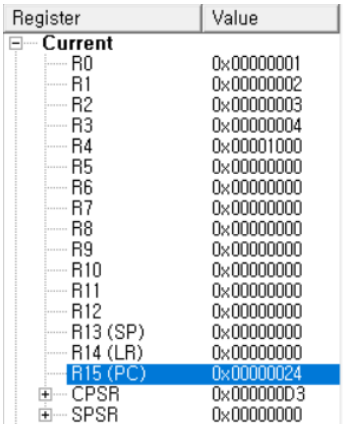



0xFF는 0x0A와 비교했을 때, 크기 때문에 R5에 1이 들어갔다.

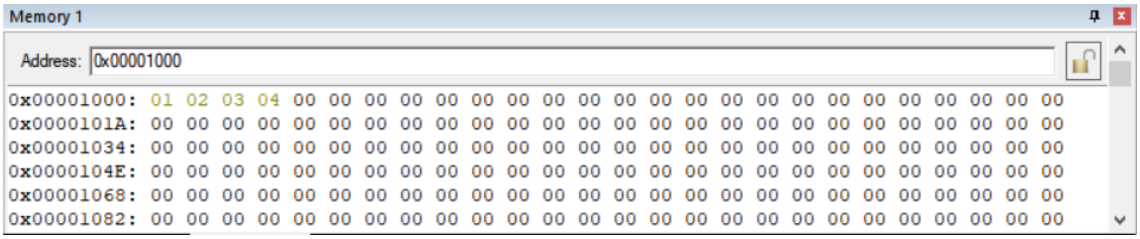
2. Problem2

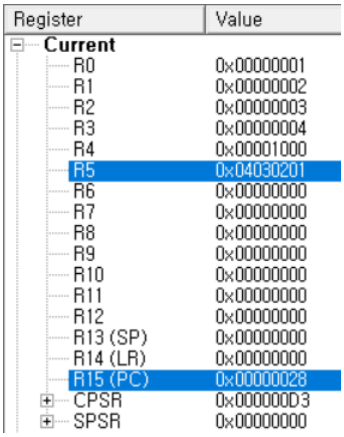



R0~R3에 1~4를 차례대로 넣어준다.



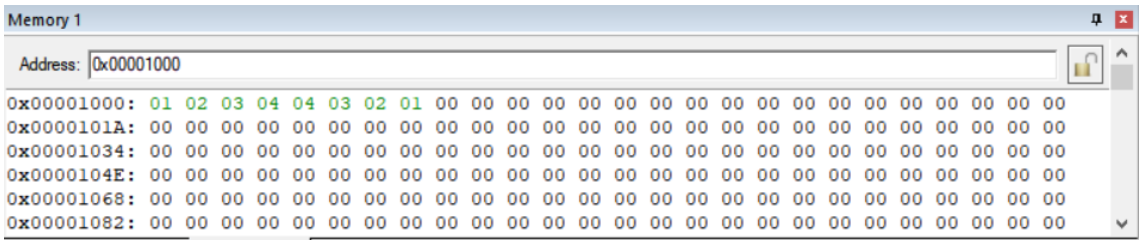

R4에 0x00001000을 넣어주고 이를 토대로 memory의 0x00001000~0x00001003에 차례대로 R0~R3에 저장된 data를 넣어준다. 결과적으로 memory에는 아래와 같이 저장된다.

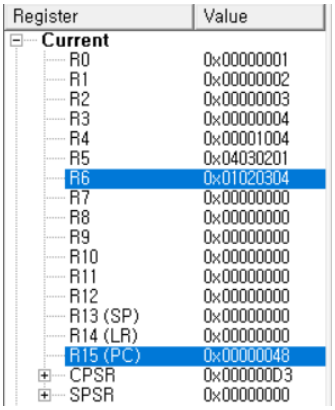



Little Endian에 따라서 0x00001000을 넣으면 32bits를 끝부터 읽으므로 위와 같이 R5에는 0x04030201이 들어간다.

R4에 0x00001004를 넣어주고 이를 토대로 memory의 0x00001004~0x00001007에 차례대로 R3~R0에 저장된 data를 넣어준다. 결과적으로 memory에는 아래와 같이 저장된다.



Little Endian방식에 따라 R6에는 01020304가 들어간다.

## 3. 고찰 및 결론

### A. 고찰

첫번째 과제에서 비교하는 과정에서 겪은 오류는 아래와 같다. 어떤 수를 memory에 저장할 때에, 숫자를 레지스터에 저장하지 않고 바로 memory에 넣으려는 시도를 했다. 하지만 이 부분에서 Error가 발생했다. 빌드 실패 화면을 통해 바로 넣는 것은 불가하다는 것을 알았다. 따라서 숫자를 레지스터에 저장을 한 후, 이 레지스터를 사용하여 memory에 저장을 했다.

어셈블리어 구현은 처음이었기에 서툴렀기 때문에 숫자를 레지스터에 16진수로 저장할 때, #을 사용하지 않고 바로 0x를 썼다. 이부분에서 오류가 발생하지는 않았지만 warning이 생기는 것을 확인하였고, 설계 강의자료를 살펴보았다. 숫자 혹은 문자 하나를 레지스터에 넣을 때, 앞에 #이 필요하다는 것을 잊고 사용했다는 것을 알게 되었다. 이에 숫자 앞에 #을 넣어 warning을 없앴다.

두번째 과제에서 저장한 data를 memory로부터 불러오는 과정에서 겪은 오류는 아래와 같다. Memory로부터 data를 읽어오는 load중, Addressing mode에 대한 부분에서 Post-index addressing을 처음 사용했다. 하지만, 이 방법을 사용하면 레지스터에 저장했던 address의 값이 바뀐다는 것을 미처 알지 못해서 결과값인 r5의 값이 생각대로 나오지 않았다. 이에 debug를 통해 원치 않는 부분의 address가 사용되는 것을 확인하였고, 실습 강의자료를 다시 보았다. 본인이 구현한 코드에서 address에 접근할 때는 post-index addressing 방법이 아닌, pre-index addressing 방법을 사용해야 한다는 것을 깨닫고 이 점을 수정하였다. 그 결과, 원하는 r5의 값이 나온 것을 확인하였다.

아직, 어셈블리어에 대한 이해가 부족하다는 것을 실감했다. 과제를 하기 전에 설계, 실습 강의자료를 참고해서 충분한 공부 뒤에 구현을 해야 예상치 못한 오류가 발생하지 않는 것 같다. 다음 과제는 구현보다 공부를 우선시 할 것이다.

### B. 결론

메모리에 함부로 접근하게 되면 위험한 상황이 발생할 수 있다. 이 때문에 메모리에 접근을 할 때는, 접근할 범위를 지정해서 정해진 범위를 벗어난 주소에 다가가지 못하게 해야 한다. 이 범위를 설정하는 파일이 ini파일이다. 메모장에 MAP 접근할 주소의 시작점, 끝점 줄 권한(READ, WRITE, EXEC)을 적고 저장할 때, .ini로 저

장하면 주소에 넣은 권한으로 접근할 수 있게 된다.

Byte Ordering은 데이터를 저장하는 방식을 뜻한다. 그 중 하나가 리틀 엔디언(Little Endian) 이다. 이 방식은 데이터를 저장할 때, 메모리의 끝에서부터 0까지 즉, 역순으로 저장한다. str문자열은 endian의 형식에 상관없이 0부터 메모리의 끝까지 저장하게 되어 예외이다. Little Endian 방식을 사용하면 산술연산과 데이터의 타입이 확장 혹은 축소될 때 Big Endian 방식보다 효율적이다.

2번째 문제에서 Big Endian 방식을 사용하여 구현, 실행해보니 데이터 저장 방식에 대해서 더 많은 이해를 할 수 있었다.

## 4. 참고문헌

Little Endian/ https://blog.naver.com/aaasssddd25/220904471846

이형근/ 어셈블리프로그래밍/ 광운대학교/ 2018