어셈블리프로그램 설계및실습 보고서

7차 실습과제

학 과: 컴퓨터정보공학부

담당교수: 이형근 교수님

실습분반: 화요일 6, 7

학 번: 2019202050

성 명: 이강현

제 출 일: 2022.11.09(수)

1. Problem Statement

Pseudo instruction이 assembler에 의해 어떻게 변환되어 저장되는지를 확인한다. 32bit 명령어들이 어떻게 구성되어 있는지 알고 disassembly를 해석하는 법을 배운다. Lable의 이름이 실제 메모리 주소임을 이해하고 이를 이용해서 어셈블리어 프로그래밍을 진행하는 능력을 습득한다.

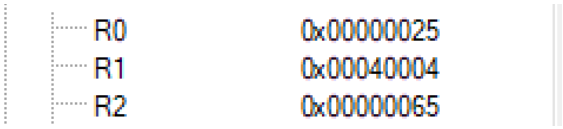
2. Design

<problem1>

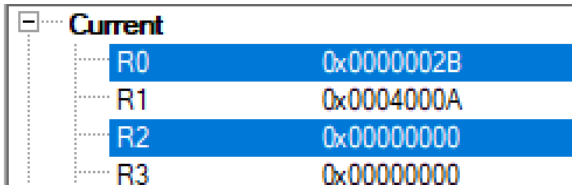
1. ADR명령어를 통하여 r0에 string을 label로 하는 배열의 주소를 저장한다.
2. LDR명령어를 통하여 r1에 tempaddr에 저장된 주소값을 저장한다.
3. LDRB명령어를 통해 r2에 r0에 저장된 주소가 가리키는 값을 1바이트 단위로 가져온다.
4. CMP명령어를 통해 r2의 값과 0을 비교한다.(문자열의 끝을 알기 위해)
5. BEQ명령어를 통해 문자열의 끝에 도달했다면 종료한다.
6. 5번에서 끝에 도달하지 못했다면 STRB명령어를 통해 0이 아니라면 r2값을 r1에 저장된 주소에 바이트단위로 저장한다.
7. BNE명령어를 통해 다시 3번을 반복한다.

3. Conclusion

<problem1>



r0에서 가져온 값들을 r2에 저장하고 r1에 저장된 주소(40000)부터 1씩증가하면서 메모리에 저장하는 모습



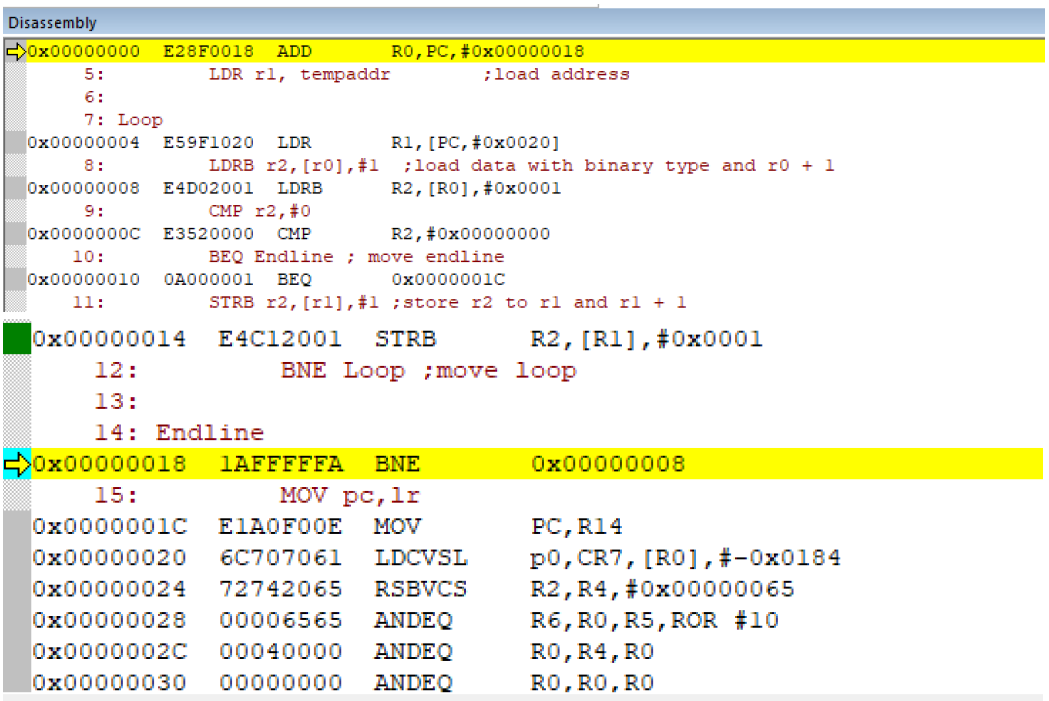
R2가 0이 되어 문자열의 끝에 도달했음을 확인

```
0x00040000: 61 70 70 6C 65 20 74 72 65 65 00
```

주소 40000에 저장된 문자열

```
0x00040000: apple tree....
```

Ascii형식으로 확인하면 위와 같다.



위에서부터 해석하면

1. ADR명령어가 ADD로 바뀌어 r0에 pc값과 0x00000018을 더하여 저장하는 것으로 바뀌었다.

   32bit환산시-E28F0018-1110_0010_1000_1111_0000_0000_0001_1000

2. LDR명령어는 그대로 LDR명령어로 유지되어 r1에 pc에 0x0020을 더한 주소를 저장한다.

   32bit환산시-E59F1020-1110_0101_1001_1111_0001_0000_0010_0000

3. LDRB명령어는 그대로 LDRB명령어로 유지되고 r0에 저장된 값이 가리키는 값을 r2에 저장하고 r0를 1증가한다.

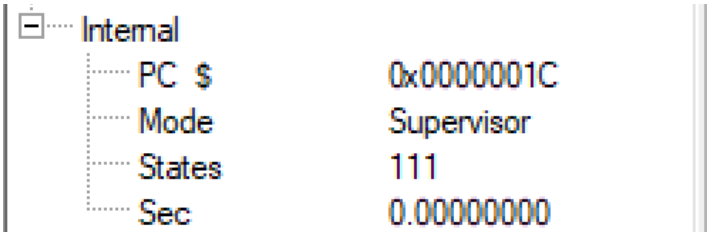
   32bit환산시-E4D02001-1110_0100_1101_0000_0010_0000_0000_0001

4. CMP명령어는 그대로 CMP명령어로 유지되고 r2와 0을 비교한다.

32bit환산시-E3520000-1110_0011_0101_0010_0000_0000_0000_0000

5. BEQ명령어는 BEQ명령어로 유지되고 Endline이라는 label은 주소값을 바뀌어 0x0000001C가 된다.
   32bit환산시-0A000001-0000_1010_0000_0000_0000_0000_0000_0001
6. STRB명령어는 STRB명령어로 유지되고 r2값을 r1에 저장된 값이 가리키는 메모리에 저장하고 r1을 1증가한다.
   32bit환산시-E4C12001-1110_0100_1100_0001_0010_0000_0000_0001
7. BNE명령어는 BNE명령어로 유지되고 Loop라는 label은 주소값을 바뀌어 0x00000008이 된다.
   32bit환산시-1AFFFFFA-0001_1010_1111_1111_1111_1111_1111_1010
8. MOV명령어는 MOV명령어로 유지되고 lr이 r14로 바뀌어 lr값을 pc에 저장한다.
   32bit환산시-E1A0F00E-1110_0001_1010_0000_1111_0000_0000_1110

Disassembly를 분석한 결과 각 단계마다 label은 주소값으로 바뀌었고 명령어는 ADR명령어가 ADD명령어로 바뀌는 모습을 확인할 수 있었다.
각각의 명령어마다 가지는 ARM에서의 format에 맞추어 32bit로 바뀌어 저장되고 이를 실행하는 것을 알 수 있다.



Strcpy를 완료하고 나온 state는 111이다.

4. Consideration

이번 실습시간에는 사용자가 구현한 명령어가 어떻게 바뀌어 저장되는지를 알 수 있는 실습이었다. Label은 실제로는 주소로서 인식되어 자동으로 assembler에 의해 주소값으로 변경되어 프로그램이 실행되는 것을 알 수 있었고 pseudo instruction은 원하는 처리에 맞게 assembler가 자동으로 변경해준 후 프로그램이

진행되는 것을 알 수 있었다. 또한 명령어 셋들이 32비트의 용량을 가지고 작동한다는 것을 이론상으로 배웠지만 실습에서 처음으로 확인하게 되었다. 이번 문제에서는 명령어가 바뀌어 저장되는 것의 형태를 ADR명령어에서만 확인했지만 더 많은 pseudo instruction set들이 존재할 것이고 이를 좀 더 알아보고 싶게 했다. ARM명령어의 구조를 시각적으로 더 잘 이해하게 된 실습이었다.

5. Reference

이형근/어셈블리프로그램 설계 및 실습/광운대학교(컴퓨터정보공학부)/2022