어셈블리프로그램설계및실습 보고서

과제 주차: 10주차

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습분반: 화요일 6, 7

학 번: 2021202058

성 명: 송채영

제 출 일: 2022.11.9(수)

1. Problem Statement

Pseudo instruction이 assembler에 의해 어떻게 실제 instruction으로 변환되는지 직접 확인하고 이해한다. Disassembly를 해석하는 방법을 이해하고 32bit 명령어들의 구조를 이해한다. Label의 이름이 실제 메모리 주소임을 이해하고 이를 이용해 어셈블리어 프로그래밍을 진행하는 능력을 습득한다.

2. Design

Pseudo instruction은 어셈블러에서 지원하는 명령을 말하며, ADR, ADRL, NOP, ALIGN, DCB, DCD, LDR 등이 있다. Disassembly는 기계어를 어셈블리어로 변환하는 컴퓨터 프로그램의 출력물을 말한다. 하지만 원본 소스코드보다 사람이 해석하기 더 어려워진다는 단점이 있다.

-problem 1

Start, loop, endline으로 나누어 설명하겠다.

먼저 start에서는 r0 레지스터에 문자열을 r1 레지스터에 주소를 각각 load한다.

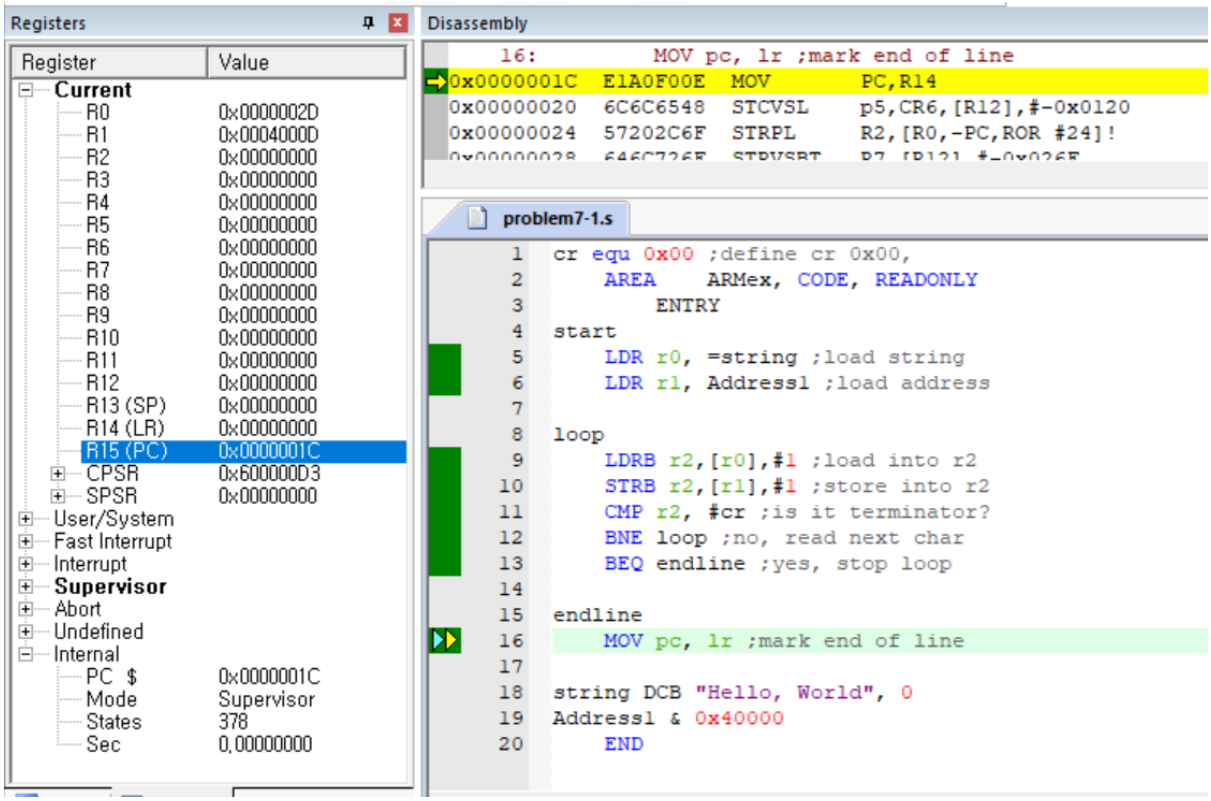
다음으로 loop에서는 r2 레지스터에 r0의 문자열을 불러와 한 칸 이동시킨다. -> r2 레지스터를 r1 레지스터에 저장 후 한 칸 이동시킨다. -> r2와 cr을 비교한다. -> 문자열의 끝이 아니라면 loop를 반복하며, 끝이라면 endline으로 간다.

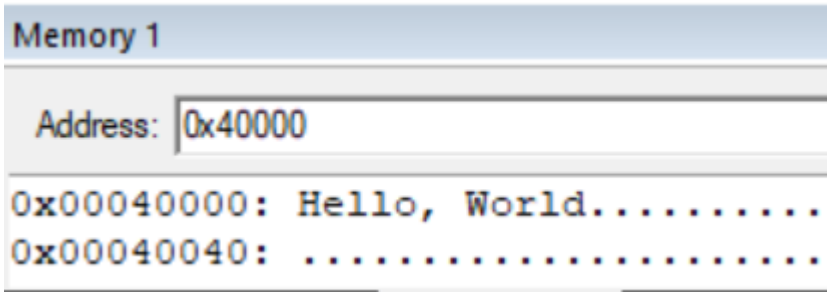
마지막으로 endline에서는 프로그램을 종료한다.

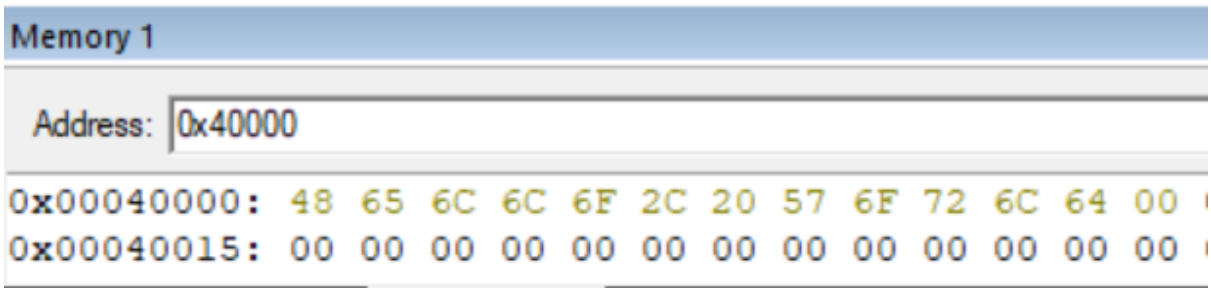
3. Conclusion

- problem 1

<Assembly>

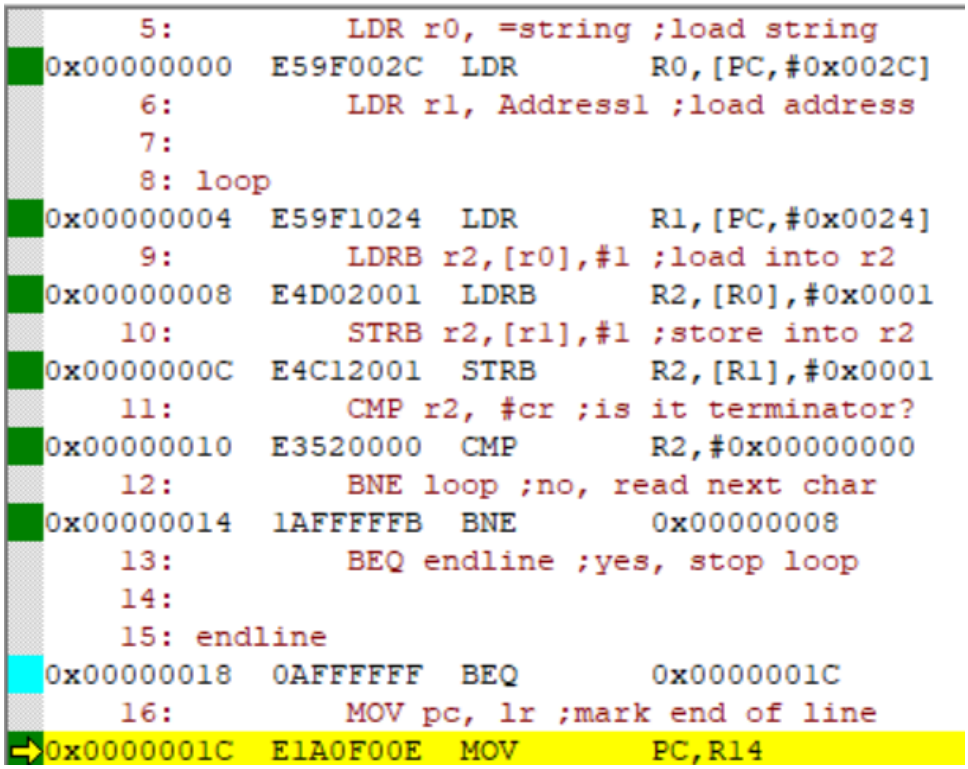






두 번째 사진에서 볼 수 있듯이, 아스키코드 값으로 봤을 때 Hello, World가 잘 저장된 것을 알 수 있다. 세 번째 사진은 unsigned char형으로 아스키 코드로 변환하면 "Hello, World",0이므로 잘 저장된 것을 알 수 있다.

<Disassembly>



Pseudo instruction은 32bit로 다음과 같으며, 이진수로 나타냈다.

LDR E59F002C : 1110 0101 1001 1111 0000 0000 0010 1100

LDR E59F1024 : 1110 0101 1001 1111 0001 0000 0010 0100

LDRB E4D02001 : 1110 0100 1101 0000 0010 0000 0000 0001

STRB E4C12001 : 1110 0100 1100 0001 0010 0000 0000 0001

CMP E3520000 : 1110 0011 0101 0010 0000 0000 0000 0000

BNE 1AFFFFFB : 0001 1010 1111 1111 1111 1111 1111 1011

BEQ 0AFFFFFF : 0000 1010 1111 1111 1111 1111 1111 1111

MOV E1A0F00E : 1110 0001 1010 0000 1111 0000 0000 1110

예를 들어 LDR은 E59F002C로 나타낼 수 있으며 instruction을 해석해 아래와 같이 동작하도록 한다.

Pseudo instruction은 작성한 코드를 다음과 같이 프로그램이 인식할 수 있도록 변경된다.

LDR r0, =string -> LDR R0, [PC, #0x002C]

LDR r1, =Address1 -> LDR R1, [PC, #0x0024]

LDRB r2, [r0], #1 -> LDRB R2, [R0], #0x0001

STRB r2, [r1], #1 -> STRB R2, [R1], #0x0001

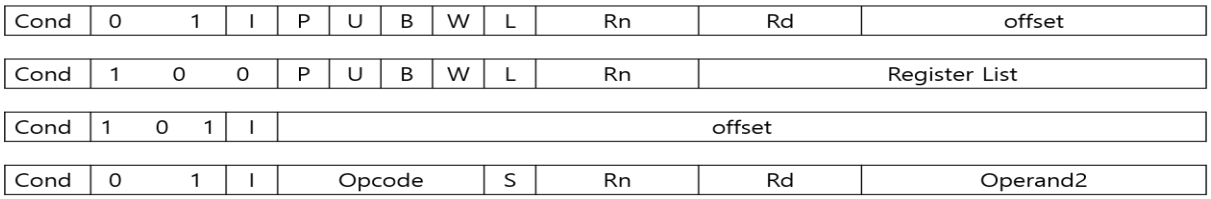
CMP r2, #cr -> CMP R2, #0x00000000

BNE loop -> BNE 0x00000008

BEQ endline -> BEQ 0x0000001C

MOV pc, lr -> MOV PC, R14

이를 통해 어셈블러가 주소를 직접 가져와 동작하는 것을 알 수 있다.



Cond	0	1	I	P	U	B	W	L	Rn	Rd	offset
Cond	1	0	0	P	U	B	W	L	Rn	Register List	
Cond	1	0	1	I	offset						
Cond	0	1	I	Opcode				S	Rn	Rd	Operand2

위의 표는 강의자료에 있는 ARM Instruction set format으로 차례대로 Load/Store Byte/word, Load/Store Multiple, Branch, Data processing/PSR transfer로, 32bit의 코드가 위의 표처럼 이루어져 있는 것을 알 수 있다.

4. Consideration

실습 ppt의 예시를 참고해 코드는 금방 짤 수 있었다. 처음으로 equal을 뜻하는 equ를 써서 cr equ 0x00으로 cr을 0x00으로 정의해주고 코드를 짰다. 문자열의 끝에 0인 cr을 만날 때 까지 비교하는 것을 반복해주어 해결하였다. 반면 Disassembly 과제를 수행하는 데 많은 시간이 걸렸던 것 같다. Instruction마다 다른 format을 갖고 있으며, 각 Instruction의 format을 Disassembly에서 확인할 수 있었고, 어떠한 방식으로 변경되어 프로그램이 명령을 수행하는 지 알 수 있었다.,

5. Reference

이형근 교수님/어셈블리프로그램설계및실습/광운대학교(컴퓨터정보공학부)/2022