어셈블리프로그램설계및실습 보고서

과제 주차: 9주차

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습분반: 화요일 6, 7

학 번: 2021202058

성 명: 송채영

제 출 일: 2022.11.7(월)

1. Problem Statement

Arm에는 floating point number와 관련된 instruction이 존재하지 않아 floating point를 사용할 때 해당 연산을 수행할 수 있는 assembly code가 필요하다. Floating point number adder를 구현하여 floating point number의 덧셈과 뺄셈에 대해 알아본다. Mantissa normalization 과정에서 어떤 경우에 right와 left로 shift 되는지 고려해 코드를 작성한다.

2. Design

Sign bit는 양수와 음수를 판단하는 비트로, 가장 왼쪽 비트(MSB)가 0이면 양수, 1이 면 음수라 판단한다. Mantissa는 가수를 뜻하며, 실질적인 데이터를 의미한다. Exponent는 지수로, 32비트에선 8비트로 표현된다.

-problem 1

Start, equal, notequal, normalize, finish, endline으로 나누어 설명하겠다.

먼저 start에서는 r0, r1, r2, r12에 각각 메모리 주소, floating point 값 2개, mantissa를 불러온다. -> R3, r4, r5에 각각 value 1의 sign bit, exponent bit, mantissa bit를 r7, r8, r9에 value2의 sign bit, exponent bit, mantissa bit를 저장한 후 mantissa 값에 1을 더해준다. -> exponent를 비교한다. -> sign bit를 비교해 같으면 equal로, 다르면 notequal로 간다.

다음으로 equal에서는 r11에 두 mantissa값을 더해준 후 exponent에 1을 더해준 다음 mantissa의 값을 오른쪽으로 1만큼 이동 후 bit를 clear해준다. -> finish로 간다.

Notequal에서는 두 mantissa 값을 비교해 같다면 endline으로 간다. -> 다르다면 큰 값에서 작은 값을 빼고 비교해 준 후 exponent에 1을 더해준다. -> finish로 간다.

Normalize에서는 mantissa의 값을 normalize 한 후 exponent 값을 1 빼준다. -> finish로 간다.

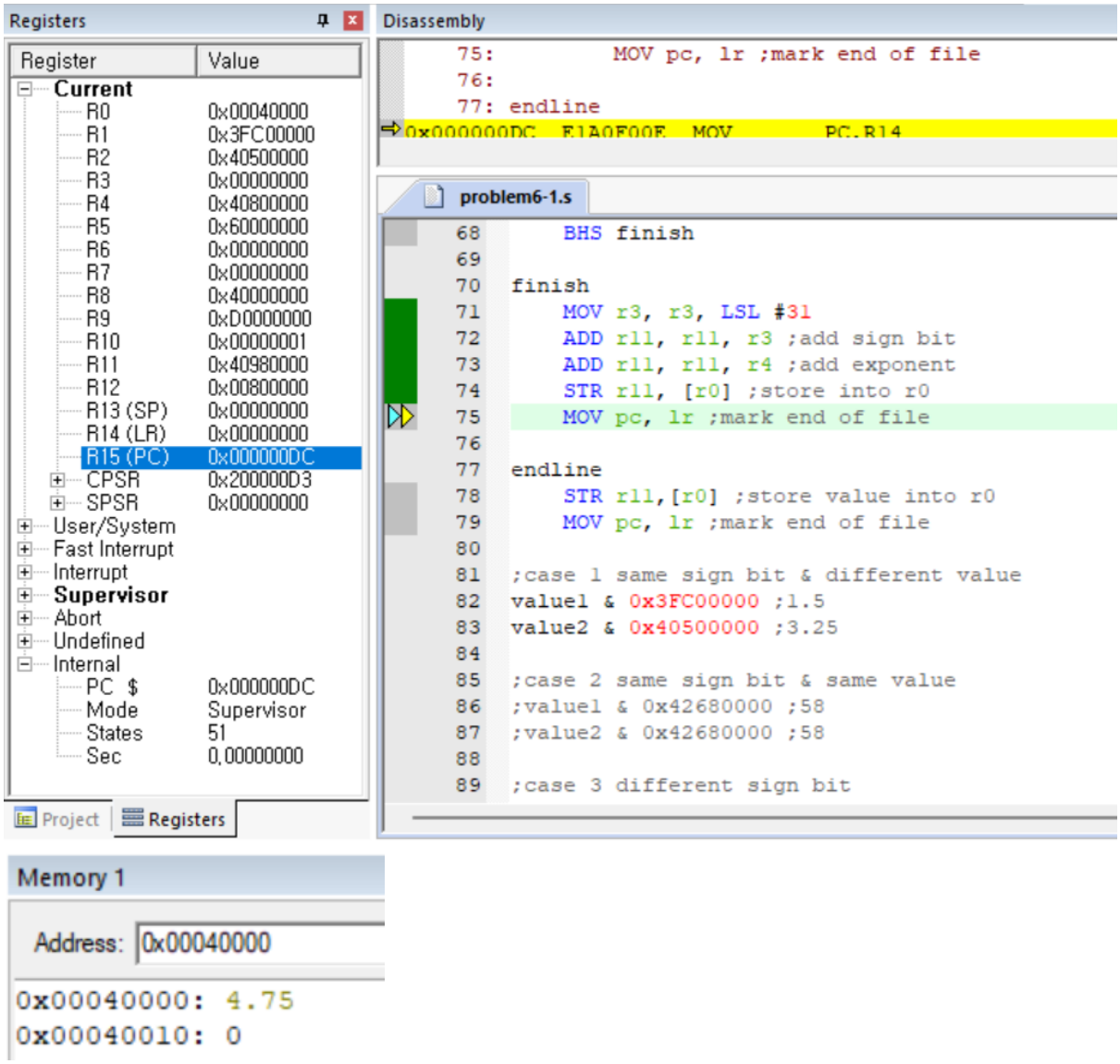
Finish에서는 sign bit와 exponent bit를 더해준 후 r0에 저장해준다. -> 프로그램을 종료한다.

Endline은 두 값이 같을 경우를 뜻하는데, 두 값이 같다면, 값을 저장하고 프로그램을 종료한다.

3. Conclusion

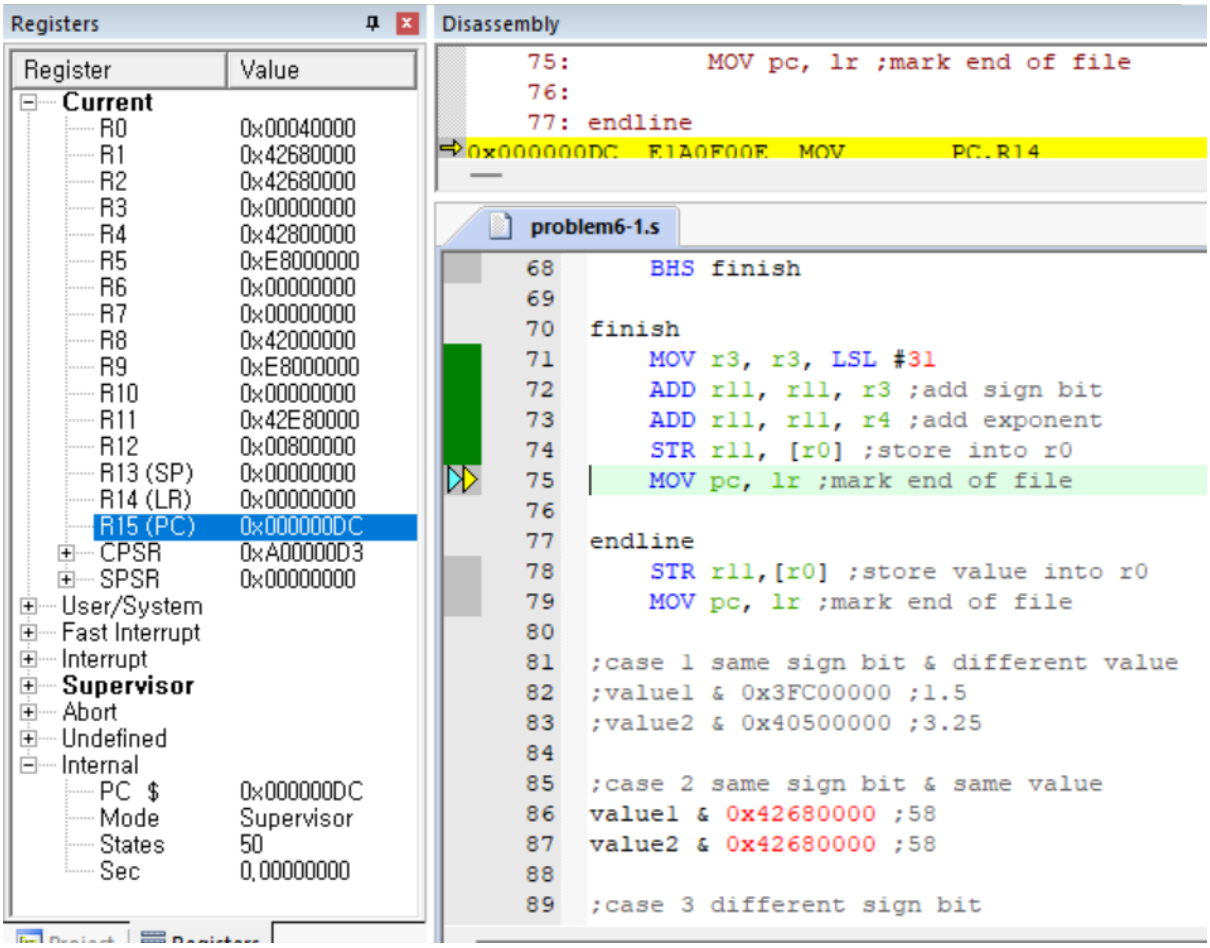
- problem 1

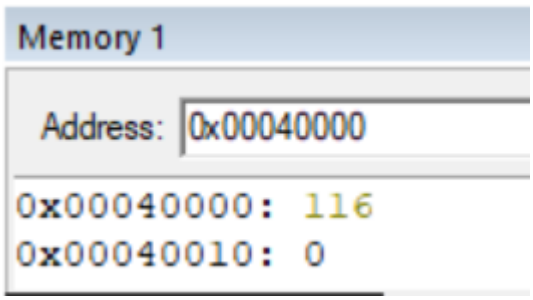
Case 1



0x3FC00000과 0x40500000를 더한 것으로, 1.5+3.25=4.75가 나온 것을 확인할 수 있다. Sign bit가 같으므로 start -> equal -> finish를 통해 결과가 잘 나왔다.

Case 2

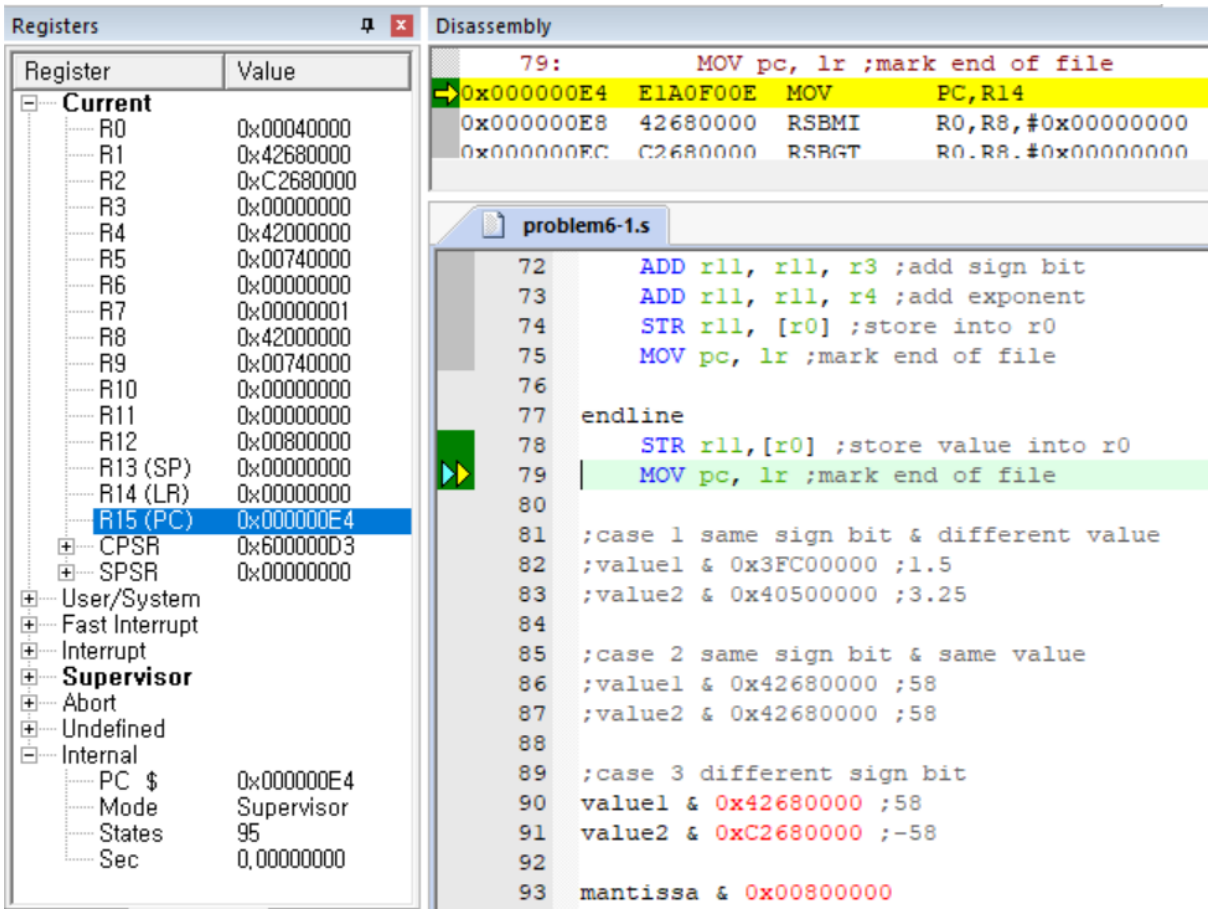


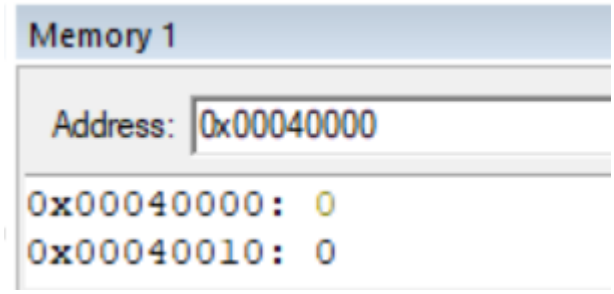


0x42680000과 0x42680000를 더한 것으로, 58+58=116이 나온 것을 확인할 수 있다.

부호가 같으므로 start -> equal -> finish를 통해 결과가 잘 나왔다.

Case 3





0x42680000과 0xC2680000를 더한 것으로, 58+(-58)=0이 나온 것을 확인할 수 있다.

부호가 다르므로, start -> notequal의 순으로 갔으며, notequal에서 숫자가 같으므로 바로 endline으로 간 것을 통해 결과가 잘 나왔음을 알 수 있다.

4. Consideration

Start 부분에서 LSL, LSR을 통해 bit shift를 진행했는데 그냥 코드를 짜는 것이 복잡해 종이에 적어가면서 했다. 실습을 진행하기 전, 58, -58을 0x42680000, 0xC2680000으로, 0x3FC00000, 0x40500000를 1.5, 3.25로 변환하는 과정을 직접 해보고 실습을 진행하니 조금 수월했던 것 같다. 전의 실습과는 달리 코드의 길이도 점점 길어지고 함수를 쓰는 부분도 많아져 복잡했고 register가 부족해 여러 번 수정을 했던 것 같다. 항상 메모리에 주소가 잘 저장되었는지 확인할 때 unsigned char형을 이용해 결과값을 확인하였는데 이번

에는 float를 통해 연산 결과를 더 쉽게 확인 할 수 있어 간편했다. Floating point의 개념은 금방 이해가 됐지만, 실습을 진행하니 다소 복잡하고 어려웠던 것 같아 프로젝트를 진행하기 위해서는 복습을 해야 할 것 같다.

5. Reference

이형근 교수님/어셈블리프로그램설계및실습/광운대학교(컴퓨터정보공학부)/2022