어셈블리프로그램 설계및실습 보고서

6차 실습과제

학 과: 컴퓨터정보공학부

담당교수: 이형근 교수님

실습분반: 화요일 6, 7

학 번: 2019202050

성 명: 이강현

제 출 일: 2022.11.03(목)

1. Problem Statement

Floating point에 관련하여 부동소수점의 표준형식을 안다. Floating point를 연산하는 명령어가 없기에 이를 수행할 수 있는 assembly code가 필요하다. 덧셈과 뺄셈을 부호에 맞게 적절하게 수행할 수 있도록 한다.

2. Design

<problem1>

먼저 프로그램의 흐름은 다음과 같다.

메모리에 배열로서 16진수형태로 저장된 값들을 r2,r3,에 불러온다. 그 후 r4,r5에는 각각 부호비트를 담는다. R6,r7에는 exponent값을 담는다. R8,r9에는 mantissa값을 담는다. mantissa에는 1을 추가적으로 더해준다. 위와 같은 작업들은 LSR과 LSL을 이용하여 구현한다.

Exponent를 비교한 후 r10에 exponent를 큰것에서 작은 것을 빼서 저장한다.

R10은 shiftnum으로서 exponent를 동일하게 맞추어 mantissa를 계산하기 위함이다.exponent간의 차이에 따라 mantissa의 소수점 자리를 바꾼 후 저장한다.

그리고 부호비트를 비교한다.

부호비트가 같다면 바로 캐리를 계산하여 연산을 진행하고 캐리가 발생한다면 exponent를 1 더해주고 mantissa를 LSR #1 처리해주고 아니라면 그대로 endline 함수로 이동한다.

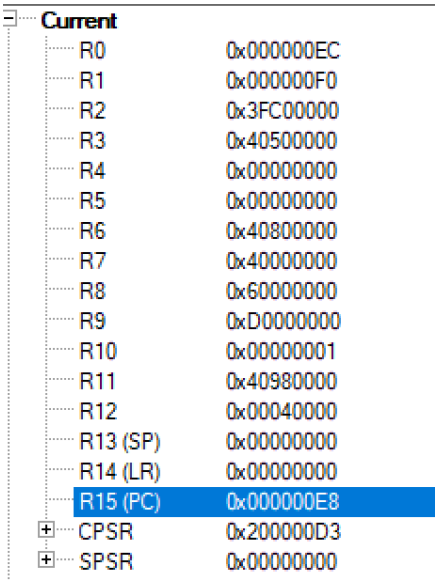
부호비트가 다르다면 먼저 둘의 mantissa값이 같은지 검사한 후 같다면 연산결과는 0으로서 프로그램이 종료될 것이고 다르다면 mantissa간의 연산 후 normalization으로 이동한다. nomalization에서는 mantissa가 하위 23비트에 제대로 정렬되어 있을 수 있게하는 연산을 담당한다 따라서 0x01000000과 비교해 캐리가 발생하면 01000000보다 크다는 것을 의미하므로 mantissa를 right로 한칸 이동해주고 0x00800000과 비교해서 더 작다면 mantissa를 left로 한칸 이동해주는것으로 연산을 진행한다. Mantissa의 정렬이 끝나면 endline으로 이동한다.

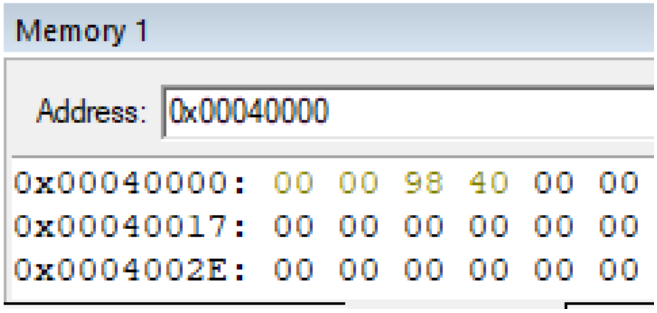
endline에서는 정렬된 모든 값들을 더해준다면 32비트의 각자리에 부호비트, exponent, mantissa가 서로 값을 침범하지 않은채 저장될 것이다.

메모리 40000번지에 최종 연산값을 저장하고 프로그램은 종료된다.

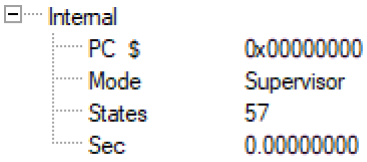
3. Conclusion

<problem1>





0x3FC00000과 0x40500000의 연산 결과인 1.5 + 3.25 = 4.75의 값이 올바르게 메모리에 저장되어 있음을 확인할 수 있고 레지스터들은 각각 필요한 값들을 저장하는 사용했다.



state수는 57이 나왔다.

4. Consideration

이번 실습에서는 floating point의 덧셈과 뺄셈을 연산하는 것을 assembly언어를 통해 구현해보았다. 이론으로 배운 것과 실제로 코드로 구현하는 것은 많이 다르고 어려웠다. 손으로 직접 써가며 값들을 구하는 것은 비교적 쉬웠지만 코드로 직접 생각하는 것을 구현하려니 쉬프트연산을 얼마나 해야하는지 어떤 조건에서는 어떤 연산을 진행해야하는지 그리고 마지막에 normalization을 하는 조건을 생각해내는 것이 문제였다. 또한 이번실습에서는 two's compliment에서 음수와 양수를 연산하는 방법은 음수를 2의 보수를 취하여 더하는 것이었는데 이번에 구현한 floating point 방법은 signbit가 따로 존재하여 오히려 간단하게 연산을 진행할 수 있었던 것 같다.

5. Reference

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