어셈블리프로그래밍설계및실습 보고서

실험제목: Floating point

실험일자: 2016년 10월 27일 (목)

제출일자: 2016년 11월 02일 (수)

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습분반: 화5, 목6,7

학 번: 2013722095

성 명: 최 재 은

1. 제목 및 목적
   1. 제목

Floating pointer

* 1. 목적

Floating point의 개념을 이해하고 이를 적용하여 floating point의 adder를 구현하여

Floating point의 덧셈과 뺄셈 연산에 대해 알아본다. 또한 normalize 과정에서 어떠한

경우에 right/left로 shift되는지 알아본다.

1. 설계 (Design)
   1. Pseudo code

R0, r1에 미리 선언해둔 값들을 읽어 들이고 R2에 저장할 주소 값을 저장함

R10에 mantissa에 더해줄 0x10000000 값을 저장

R11에 절대값 비교를 위한 0x80000000 값을 저장

현 위치를 lr에 저장하고 func함수 호출

최종 결과값인 r9을 r12가 가진 주소 값 위치에 저장

- func

r0, r1을 lsr #32 하여 signbit값을 추출하고 각각 r2, r3에 저장

r2, r3를 비교하여 두 입력 값의 부호가 같은지 비교

서로 다르면 r11에 1을 저장, 같으면 0을 저장

R0, r1를 lsl #1해 r4, r5에 저장하고, r4, r5를 lsr #24하여 exponent값 추출

R0, r1을 lsl #9해서 r6, r7에 저장해 mantissa를 추출하고, lsr #4해서 1을 더할 자리 마련

R6, r7에 r10을 더함(1을 더함)

R4, r5를 비교해서 두 exponent의 곱셈차를 r8에 저장

R8만큼 작은 값의 mantissa를 소수점 이동

R0, r1을 lsl #1한뒤 lsr #1하여 절대값을 추출

절대값을 비교해서 더 큰 값의 sign bit(r2 or r3)를 r9에 저장

R11을 0과 비교해서 같으면 addition 함수 호출, 아니면 그대로 subtraction 진행

- subtraction

R0, r1을 비교해서 절대값이 큰 값의 mantissa에서 작은 값의 mantissa를 뺌

뺄셈 이후 1을 더했던 자리가 0인지 확인하여 곱셈 값 변경 및 소수점 이동, Next 호출

- addition

r8에 두 mantissa를 더한 값을 저장

더한 값에서 carry 생겼는지 확인하고 생겼으면 곱셈 값을 1 올리고 r8의 소수점 이동

- next

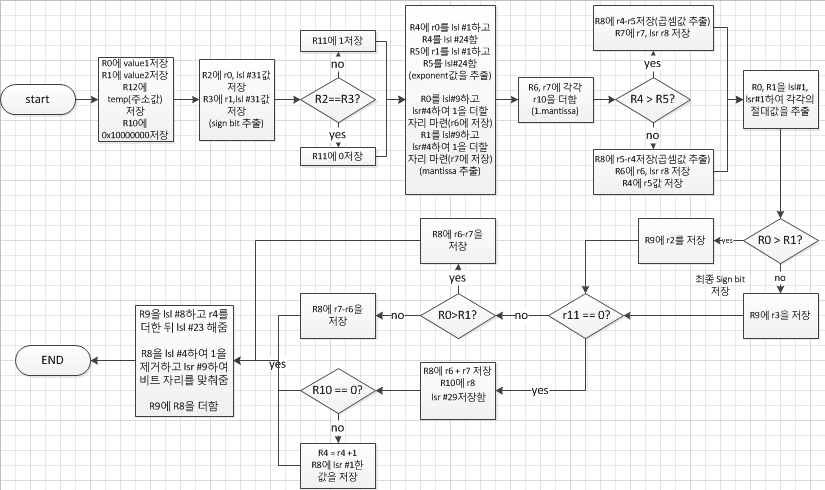
R9을 lsl #8하여 exponent를 넣을 자리를 마련

R9에 r4(exponent)를 더함

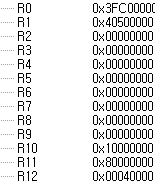
R9을 lsl #23하여 비트자리 맞춤

R8을 lsl #4하여 앞서 더했던 1을 제거하고, lsr #9하여 비트자리 맞춤

R9에 r8을 더하고 lr에 저장했던 위치로 이동

* 1. Flow chart 작성
  2. Result

- 초기값 setting



- exponent 추출



- mantissa 추출 후 1을 더해줌



- sign bit + exponent



- 최종 결과 값을 40000번지에 저장



* 1. Performance





- code size는 232이고 state는 64개입니다. Score는 code size \* state^2이므로

이 프로그램의 스코어는 950272이 나오게 됩니다.

1. 고찰 및 결론
   1. 고찰 및 결론

- 이번 과제를 진행하면서 다시 한번 high level language program이 정말로 편했구나

라는 것을 실감했습니다. 물론 그 프로그램들 또한 이러한 과정을 거쳐서 floating point를 이동시키며 계산하겠지만, 그 부분을 직접 구현하려니 고통스러웠습니다. 입력된 두 값이 서로 같은 부호를 갖는지, 같다면 양수인지 음수인지, 서로 다르다면 결과 값의 sign bit는 어떻게 주어야 하는지를 한동안 고민하다가 문득 담배를 피우며 깨달음을 얻었습니다. 바로 고은정 조교님께서 수업시간에 언급하셨던 ‘절대값’이었습니다. 절대값을 통해서 sign bit를 어떻게 줄지를 결정할 수 있었다는 것을 깨달은 뒤로는 그렇게 큰 어려움은 없었습니다. 다만 진행하는 와중에 비트를 이리저리 움직이다 보니 지금 이게 맞나 싶은 감이 있었지만 손으로 ‘직접’풀어가면서 값을 비교하여 문제를 진행해 나갔습니다.

.

1. 참고문헌