**SYSTEM PROGRAMING**

**HW1**

성균관대학교 소프트웨어학과

2019315505 이원규

**과제를 한 방식**

And(&)연산자를 이용해 필요한 비트에 마스킹을 하여 비트연산을 쉽게 하도록 하였습니다. 모든 함수의 첫 시작은 비트마스킹으로 시작합니다. 이를 통해 16bit의 sfp를 sign, exp, frac로 분해해 사용했습니다.

또한 가독성을 높이기 위해 sfp.c에 +-INT\_MAX, +-SFP\_MAX, Nan등을 정의하였습니다.

Frac를 10비트에 위치시키기 위한 방식은 0b1111111111(10개의 bit)에 마스킹을 하여 그 값을 루프문 조건으로 사용했습니다.

이 보고서의 서술 흐름은 작성한 코드 안의 함수의 순서 및 진행 방식과 동일합니다.

**1. int2sfp**

- 표현 범위를 벗어난 수부터 +-inf로 먼저 처리했습니다.

- 0 리턴과 후에 리턴 될 ret에 대해 부호를 결정하였습니다.

- input의 끝자리부터 비트로 변환하여 frac에 저장하였습니다. 나중에 리뷰를 하면서 보니 그냥 input그대로 저장해도 됐는데 제가 너무 어렵게 생각한 것 같습니다.

- input을 비트변환 한 뒤에 input 비트스트림의 MSB가 frac의 MSB바로 앞에 자리할 수있도록 남는 frac의 자리를 조절해주었습니다.

- 마지막으로 앞에서 저장한 sign비트를 추가해주었습니다.

**2. sfp2int**

- 예외 케이스부터 최대한 정리하였습니다.

- exp가 0이되는 케이스를 정리하였습니다. 0이 되지 않는 경우 frac에 1을 더해서 1.xx의 꼴로 표현하였습니다. 이렇게 표현할 경우 frac의 MSB가 0,1에 관계 없이 loop문 하나로 조절이 가능합니다.

- E값에 따라서 frac의 MSB 위치를 조정하였습니다.

- 마지막엔 부호를 설정해주었습니다.

3. float2sfp

- 예외케이스를 최대한 우선 처리하였습니다. Sign bit도 결정합니다.

- 정수부와 소수부로 나누기 위해 IntPart, DeciPart 변수를 사용합니다.

- 정수부가 존재할 경우 정수부의 MSB를 기준으로 frac를 설정할 수 있습니다. 정수부를 비트로 변환하면서 E값과 frac의 MSB를 조절합니다. 그리고 10-E의 남은 비트를 소수부를 비트변환하여 채웁니다.

- 정수부가 없을 경우 소수부만을 이용합니다. 소수부의 경우엔 매우 작은 수가 나올 수도 있어 그런 경우에 대해 0으로 예외 체크를 했습니다.

- flag를 이용해 소수부의 leading 1을 찾고 소수점에서의 위치를 E로 계산합니다.

- frac는 10 bit만 필요하기 때문에 leading1을 포함해 11개의 비트만 체크합니다.

- exp가 0인 경우와 아닌 경우로 나눠 frac조절을 해줍니다.

- 각 파트를 합친 뒤 리턴합니다.

**4. sfp2float**

- ret를 float 타입으로 설정합니다.

- 비트 연산자 대신 \*2와 /2를 사용합니다.

- frac 파트를 사용하는 대신 frac의 비트를 소수의 형대로 ret에 저장해 사용합니다.

- 노멀라이즈-> 스페셜 ->디노멀라이즈 순서대로 코드를 작성하였습니다.

- float타입을 이용한 것을 제외하고는 sfp2int 함수의 작동과 동일합니다.

**5. sfp\_add**

- 비트마스킹을 하면서 두 sfp의 sign관계를 표현할 sign0를 미리 설정합니다.

- ppt에 적혀있는 예외 조건들을 우선 배제했습니다.

- exp의 상태에 따라 1.xx꼴로 사용할지 0.xx꼴로 사용할지 결정합니다.

- 0은 항등원이기에 하나의 인자가 0이라면 나머지 파라메터를 리턴합니다.

-E의 크기에 따라서 분류한 뒤 계산합니다. E가 크다면 무조건 큰 수라 할 수 있기에 E를 기준으로 연산합니다.

-E1, E2의 차이 만큼 큰 수의 frac에 쉬프트 연산을 해주고 합연산을 해줍니다. 그런 뒤 다시 frac의 MSB위치를 10bit이내로 조절해줍니다. 동시에 E도 조절해줍니다.

- 마지막으로 E의 값에 따라서 special, Denormalized라면 그에 맞춰 sfp를 구성합니다.

- 최종 결과값을 리턴합니다.

**6.sfp\_mul**

- 예외 케이스를 먼저 배제합니다.

- a, b의 E, frac을 설정합니다.

- frac2의 비트를 기준으로 frac2의 각 비트 위치만큼 frac0에 frac1을 쉬프트 연산하여 더해줍니다.

- leading 1 bit를 찾습니다.

- i 만큼 leading 1이 움직여야하기 때문에 a, b의 E0에서 i를 뺍니다.

- 루프를 통해서 frac의 위치를 조절해줍니다.

- E0의 값을 통해 overflow를 체크합니다.

- sign0, exp0, frac0 결과값들을 통해 최종 sfp를 리턴합니다.

**7. sfp2bits**

- char [16] 배열을 선언합니다.

- 파라메터의 binary LSB부터 배열에 거꾸로 집어넣습니다(15->0)

- string.h 사용을 위해 string형태가 필요하다고 판단하여 [16]에 NULL을 저장하였습니다.