시스템 프로그램 Assignment 1 Report

2017313213 박경태

1. Define부

TMin과 TMax값을 정의하였으며, typedef union 을 통해 비트필드 공용체를 선언했다.

2. 부속함수부

Int2sfp에서 쓰이는 부속함수들이다.

2-1. fraclength

X의 frac의 길이가 n이 맞다면 1, 틀리면 0을 반환한다.

2-2. getfraclength

Fraclength 함수를 이용해 frac길이를 찾아내서 반환한다. 만일 frac길이를 찾지 못할 경우 -1을 반환한다.

2-3. getfracbit

Getfraclength로 얻어낸 frac의 길이를 인자로 넣어서, frac 비트를 추출하여 반환한다.

3. int2sfp

Input이 0이라면 양의 0을 반환하도록 별도처리를 한다.

Round to zero이므로, sign을 따로 저장한 뒤 나머지 비트로써 양수로 취급하게 한 뒤 단지 버림 연산만 수행하면 된다.

음수면 input에 -1을 곱해주며,

Getfraclength의 에러를 방지하기 위해 input이 1인 경우 shiftnum을 0으로 따로 처리를 한다. 그 외의 경우에는 shiftnum을 getfraclength 의 반환값을 사용한다.

당연히 int는 0이나 양의 정수일텐데, 0일 경우는 예외처리를 해 뒀으므로 exp는 shiftnum에 63을 더하면 된다.

리턴될 sfp의 frac 부는 shiftnum만큼 input에서 getfracbit으로 frac비트를 추출한 것으로,

최종적으로 얻은 sign, exp, frac 비트들을 전부 조립한 뒤 sfp로 리턴하면 된다.

4. sfp2int

예외처리를 해야하는데, 양의 무한, 음의 무한, 최대값 초과치, 음의 최대값 초과치, NaN 의 경우를 각각 다뤄야 한다.

각각 TMAX, TMIN, TMAX, TMIN, TMIN 으로 리턴하도록 예외처리했다.

EXP가 63 미만이라면 반드시 1보다 작으므로 0을 리턴하도록 했고,

63 이상이라면 exp-bias(==63) 값을 한 값이 지수부가 되며 가수부는 frac에 주어져있으므로 연산하되, exp-bias값이 31 이상이라면, signed int의 표현범위를 명백히 초과하므로 오버플로우로 인지하여 리턴하도록 했다.

마지막으로, 항상 leading 1을 붙여주고(denormal 값이었다면 진작에 예외처리되었을 것이다), 부호를 반영한 뒤 리턴한다. 처음부터 양수로 취급하고 했으므로, 2의 보수형태로 음의 비트로 바꿔주면 된다.

5. float2sfp

입력된 Float 값이 sfp의 범위 안에 들어가는지, 안들어간다면 그것이 overflow인지 underflow인지를 판별하여 예외처리를 해주어야 한다.

NaN이라면 생각할 것도 없이 NaN을 반환토록 하면 되며,

무한이라면 sfp 또한 해당 부호의 무한대로,

만일 0이라면 sfp 또한 해당 부호의 0으로,

만약 exp가 190을 넘거나 같으면 exp-bias=190-127=63이 되어 sfp의 최대지수값을 초과하므로 overflow 처리를 해야하고

Exp가 64라면 -63, 즉 float에서는 normal이었던 값이 sfp에서는 denormal로밖에 표현되지 않는 예외적 상황이므로 따로 처리를 해주어야한다. 이 경우, leading1을 생략하지 않고 그대로 가수부에 반영해야한다.

그 외의 경우는 round to zero를 적용하여 frac을 버림해주고, exp에 64를 빼준다.

6. sfp2float

sfp에서 float로 변환할 때는 별다른 제한사항이 없다. 단지 0일 때, 무한일 때, NaN일 때를 고려해주기만 하면 된다.

Exp가 0이면, 0 또는 0에 근접한 작은 수인데 모두 float에 속한 범위이므로

만일 0이면 해당 부호의 0을 그대로 리턴

만일 0에 근접한 작은 수이면 float에게 있어서 해당 수는 denormal이 아니라 normal일 것이므로 이를 처리하고

그 외의 경우에는 exp를 64 더해주고 frac을 7 좌시프트 해줘서 float의 size에 맞춰주면 된다.

7. sfp_add

두 인자 중 하나라도 NaN이면 무조건 NaN을 리턴한다.

두 인자 중 하나라도 무한이라면, 다른 인자가 같은 부호의 무한이거나 일반값이라면 해당 부호의 무한을, 다른 인자가 다른 부호의 무한이라면 NaN을 리턴한다.

마찬가지로 두 번째의 인자의 경우에도, 두 번째 인자가 무한이라면 해당 두 번째 인자의 값을 무조건 따른다. 왜냐하면 첫 번째 인자가 특수값이었다면 if문에서 걸러졌을 것이기 때문이다. 다만, 두 번째의 인자가 무한이 아니라 NaN이라면 NaN을 리턴한다.

Sign 비트를 제외한 대소를 비교한 뒤, 절댓값이 큰 sfp를 ms1, 작은 sfp를 ms2로 지정한다.

우시프트하는 정도를 표시하는 shiftnum은 exp가 0인 경우와 1인 경우 e값이 동일함을 유의하여, shiftnum을 구하고

Shiftnum에 따른 우시프트연산을 Round to Even 에 맞춰서 행한다.

그 결과를 다시 Renormalize 한 뒤 리턴한다.

8. sfp_mul

두 인자 중 하나라도 NaN이면 무조건 NaN을 리턴하고,

무한과 0을 곱해도 NaN을 리턴한다.

무한과 일반값, 무한과 무한 등의 나머지 곱셈의 경우, 수학에서의 것과 직관적으로 동일하게 부호를 처리하여 계산하면 된다.

Sign의 경우, in1과 in2의 부호비트를 XOR 해주면 된다.

Exp를 계산할 때는, unsigned이므로 빼고 0과 비교를 하는게 아니라 빼기 전에 대조를 수행해야 한다. 만일 ms1의 exp값과 ms2의 exp값의 합이 63 미만일 경우, underflow이므로 무한이 아니라 0을 리턴한다.

만일 exp값의 합이 63을 넘을 경우, 그 합에 63을 빼서 exp값을 새로 만든다.

Ms1이 denormalized일 경우, frac에 leading 1을 붙이지 않은 채로 significand를 만들며, normalized라면 leading1을 붙인다. Ms2도 똑같이 한다.

리턴할 값의 significand의 값은 significand 1에 significand 2를 곱한 값이다.

그리고, round를 수행해야하는데 이 때 Round to Even이 적용된다.

마지막으로, Exponent가 7비트 한도치를 넘지 않았는지 체크를 해야한다. 만일 7비트 전부가 1이거나 그 이상의 값이라면, exponent overflow 이므로 해당 부호의 0을 리턴한다.

최종적으로 Round to Even을 따라 Round된 significand에 leading 1이 혹시 남아있다면 생략해주고 이렇게 하여, significand에서 frac을 구할 수 있다.

Exp와 frac과 sign 비트를 조합하여 return 값을 조립하고, 반환하면 된다.