**System program - Assignment 1**

**SFP**

2020314944 PARK JINAH

1. sfp int2sfp(int input)

입력 받은 int를 sfp로 변환하는 함수이다.

가장 먼저, s, exp, frac로 구성된 16-bit를 처리하기 위해 길이가 16인 int형 배열 bitArr을 선언하였다. 그 다음으로는 sign bit 의 값을 결정하고 음수를 입력 받았을 때 부호를 변환하여 양수와 같이 처리하기 위해 입력값의 부호를 확인하였다.

이제 함수 구현에 중요한 역할을 수행하는 부분을 중심으로 설명하겠다.

우선, 입력 받은 10진수를 2진수로 변환하여야 한다. 다음 코드에서 확인할 수 있듯이 계속하여 2로 나눈 나머지를 배열에 담는 식으로 진행한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그 다음 Exp를 구하기 위해 E = exp – bias 공식을 활용하는 것이 중요하다. 이때 exp는 5 bit 길이 이므로 bias의 값은 2^(5-1) -1 로, 15임을 알 수 있다. 해당 공식을 적용하여 구한 Exp 또한 2진수로 변환하여 배열에 담는다.

텍스트, 화면이(가) 표시된 사진

자동 생성된 설명

현재 입력 받은 값을 sfp로 변환하기 위해 선언한 비트 배열에는 부호 비트 s와 exp가 담겨있고, 이제 frac 값만 담으면 된다. 앞서 2진수로 변환하여 배열을 담을 때에는 값이 거꾸로 저장되는 구조이기 때문에, 최종 배열 bitArr에 담을 때에는 역순으로 담아야 한다. 그러므로 frac 값이 저장된 Marr 배열은 마지막 -> 0번째 순서로 값을 불러올 수 있도록 반복문을 활용하여 인덱스 값을 감소시키며 bitArr 배열에 값을 저장한다. 마지막으로, s, exp, frac이 다 저장되고 남은 자리에는 0을 넣으면 된다.

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

(\* exp 가 31인 경우는, exp를 구성하는 비트가 전부 1인 상황으로 special case에 해당하여 별도의 과정을 거쳐 처리해야한다.)

1. sfp float2sfp(float input);

입력 받은 float를 sfp로 변환하는 함수로, 앞선 int2sfp 함수와 그 구조가 거의 유사하기 때문에 앞서 언급하지 않은 부분에 집중하여 설명하겠다.

이 함수에서도 가장 먼저 s,exp,frac 16-bit를 담기 위한 배열을 선언하고 입력값의 부호를 확인한다.

int2sfp 함수와 가장 큰 차이점은 소수부분이 존재하기 때문에, 정수부분과 소수부분으로 나누어 처리를 해야한다는 점이다. 입력값에서 정수부분을 빼면 소수부분을 구할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이 함수에서도 역시 10진수를 2진수로 변환하는 과정이 필수적이며, exp와 frac 계산을 위해 앞서 언급한 E = exp – 15 공식을 활용한다.

소수부분 2진수로 변환하는 방법은, 정수부분을 2진수로 변환하는 과정과 반대에 있다고 말할 수 있다. 정수는 계속하여 2를 나누며 나머지 값을 저장했다면, 소수는 소수부분이 0이 될 때까지 계속하여 2를 곱해주어야한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

또 주의해야할 점은, 정수부분이 0인 실수와 그렇지 않은 실수를 구분해야 한다는 것이다. 이는 후에 구현할 함수에도 핵심적인 역할을 하는 부분인데, V = (-1)^(s) \* M \* 2^(E) 공식을 활용하려면 해당 형태로 값을 변환해야 한다. 이를 위해 반복문을 활용하여 소수부분에 가장 먼저 등장하는 1의 위치를 알아내었다. 예를 들어 0.00101 을 소수부분을 탐색하며 가장 먼저 등장하는 1의 인덱스 값을 활용하여 1.01 x 2^(-3)으로 변환할 수 있다. 해당 인덱스 자리를 구하기 위해 pos 변수를 활용하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 사진 속 코드는 int형 배열에 담아둔 값을 sfp 타입으로 변환하는 부분이다. 반복문을 통해 shift 연산자와 임시로 선언한 i의 값을 활용하여 각 자리별로 가중치를 부여하여 변환하였다. 이 부분은 다른 함수에서도 배열을 sfp 타입으로 바꾸는데 활용된다.

1. int sfp2int(sfp input);

sfp 타입을 int 형으로 변환하는 함수이다.

비트 저장을 위해 배열을 선언하고, 부호를 확인하고, 10진수를 2진수로 변환하는 과정 모두 동일하다.

해당 함수 구현을 위해서는 E = exp – bias, V = (-1)^(s) \* M \* 2^(E) 공식을 응용하는 것이 매우 중요하다. int형으로 변환하기 위해 16자리를 s, exp, frac 세 부분으로 나누는 것이 굉장히 중요하다. (가장 처음 비트는 s, 이어지는 5개 비트가 exp, 마지막 10자리가 frac에 해당한다.)

구역을 나눈 후, 각 자리에 해당하는 가중치를 부여해주어야 한다. 이때도 역시 반복문과 shift 연산자를 활용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

보다 편리하게 처리하기 위해, 음수를 양수로 바꾸어 진행하였으니 연산을 진행한 후에, 부호 비트를 확인하여 만약 1이라면 다시 음수로 바꾸어준다.



1. float sfp2float(sfp input)

입력 받은 sfp를 float로 변환하는 함수이며, 전체적인 구조는 sfp를 int로 변환하는 함수와 유사하다. 이 함수에서도 역시 sfp 를 2진수로 이루어진 16자리 배열로 변환한 후에, s,exp,frac 세 구역으로 나누는 것이 핵심적인 부분이라 할 수 있다. 나아가 세 구역으로 나눈 뒤, shift 연산자를 적극 활용해 각 자리에 해당하는 가중치를 부여하여 계산해야 하는 것이 중요하다.

또, float을 sfp로 변환할 때와 마찬가지로, 정수부분이 0인 실수인가, 소수부분이 0인 실수인가에 따라 처리 과정이 달라지므로 이를 주의하여 구현하여야 한다. 만약 소수부분이 0인 실수라면, 앞서 구현한 sfp2int 함수와 그 과정이 완전히 동일하다고 말할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

해당 부분을 구현한 위의 코드가 이 함수에서 가장 중추적인 역할을 하고 있다고 말할 수 있는 부분이다. 위에서 언급한 공식을 활용하여 구한 E가 0보다 작다면, 정수부분이 0인 실수이다. E값에 따라 각 실수의 정수부분을 계산한 후, 가중치를 고려한 shift 연산을 통해 실수 부분 계산을 마치면 입력받은 sfp의 float 값을 구할 수 있다.

# 기타 함수

sfp spf\_add (sfp a, sfp b) , sfp sfp\_mul (sfp a, sfp b), char\* sfp\_comp(sfp a, sfp b)

덧셈 함수와 뺄셈 함수에서 int, float 없이 연산을 진행하려면 비트 이동 연산자, shift의 역할이 굉장히 중요하다. shift 연산자를 활용하여 s, exp, frac의 인덱스에 해당하는 가중치를 부여할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 코드는 비트이동을 통해 s,exp,frac 변수 값을 할당한 부분으로, 덧셈/곱셈 함수에서 동일하게 사용된다.

덧셈 함수에서는 지수 부분을 맞춰야 하기 때문에 exp 값을 동일하게 조정하여야 한다. 입력 받은 두 sfp 중에 exp가 작은 것이 큰 값으로 맞추어 연산을 진행하면 된다. 2의 보수를 활용하여 두 수의 차를 구하는 것이 핵심적이다.

이에 반해 곱셈함수에서는 지수 부분을 맞출 필요가 없기 때문에 exp 값을 동일하게 맞추지 않아도 된다는 특징이 있다.

또한 나는 두 sfp 의 값을 비교하는 char\* sfp\_comp(sfp a, sfp b) 함수를 구현할 때 덧셈 함수를 응용하였다. a에서 b를 뺐을 때 그 값이 0이면 두 값은 동일하며, 양수이면 a가 더 크고, 그렇지 않다면 b가 더 크다고 말할 수 있기 때문이다. 덧셈 함수를 활용하여 뺄셈을 구현하기 위해 b 값의 부호를 바꾸어 처리하였다.