컴퓨터 구조

PA2 Write-up

2013-11826 임주경

1. Introduction

제한된 자료 타입과 연산자의 수를 이용해, Bit 함수를 구현하며, Bit representation을 이해한다.

2. Implementation

- bitOr

x OR y 는 ~(~x AND ~y)와 같으므로, 이를 바로 return 하였다.

- isAsciiDigit

ASCII Digit에 해당하는 '0' ~ '9'는 0x30 ~ 0x39와 같다. 따라서, 16진수의 LSB로부터 두번째에 있는 4개의 bit는 0x3과 같음을 확인하기 위한 변수와 LSB로부터 첫 4개의 bit에 6이 더해졌을 때, 16진법 표현에 의해 0~9라면 5번째 bit에 자릿수 올림이 되지 않음을 확인하기 위한 변수를 지정하였다. 이 두가지 조건을 만족하면 ASCII Digit이 되므로, 두 변수의 &를 return 한다.

- getByte

하나의 변수에는 3 – n을 저장하고, 나머지 변수에는 첫 변수를 3만큼 Shift Left, 즉 8을 곱한다. Argument x를 두번째 변수만큼 Shift Left해서 MSB로부터 8개의 bit에 원하는 결과를 모은 뒤, 다시 24만큼 Shift Right해서 LSB로부터 8개의 bit에 저장한다. 이와 0xFF의 &연산을 통해 원하는 결과를 return 한다.

- byteSwap

n번째 byte와 m번째 byte를 변수로 초기화한다. 다음, x에서 이 두 byte와 ^연산을 이용해 해당 위치의 bit들을 모두 0으로 저장한다. 이렇게 저장한 bit들과 n번째 byte를 m번째 byte에 위치하게 하고, m번째 byte를 n번째 byte에 위치하게 만든 총 3개의 32bit 변수들을 OR 연산을 통해 return 한다.

- bang

하나의 변수에 -x = ~x + 1의 성질을 이용해 저장한다. 또 하나의 return에 사용하기 위한 변수는 x와 -x의 OR연산한 값을 31만큼 Arithmetic Shift Right한 결과를 저장한다. 이때, x가 0이라면 0을 0이 아니라면, -1을 저장하게 된다. 이 변수에 +1 한 값을 return 하면, 원하는 결과를 얻게 된다.

- fitsShort

변수 a는 x의 16번째 bit 이후로 MSB까지 모두 0을 가질 경우, 1을 저장하며, 그렇지 않으면 0을 저장한다 (!연산 사용). 변수 b도 마찬가지로 !연산을 사용해 x의 16번째 bit 이후로 MSB까지 모두 1을 가질 경우, 1을 저장하며, 그렇지 않으면 0을 저장한다. 이 두가지 조건을 다 만족하지 못할 경우 16bit 표현이 불가능하므로, a | b를 return 하도록 한다.

- isTmax

변수 a는 -x를 저장한다. 변수 b는 !(x ^ a)를 저장한다. 만약 x가 Tmax거나 -1이어야만 1을 갖는다. 변수 c는 !(x + 1)을 저장한다. 이는, x가 -1일 경우 1을 저장하도록 한다. Return은 b & !c를 이용해, x가 -1이 아니고 Tmax일때만, 1을 돌려받도록 한다.

- negate

단순히 2의 보수법에 의해 -x = ~x +1임을 이용해 이를 return 한다.

- sign

변수 a는 (x >> 31) &0x1로 x가 0이상이면 0을 그렇지 않으면 1을 저장한다. 변수 b는 ~a + 1로 x가 0이상이면 0을 그렇지 않으면 -1을 저장한다. 변수 c는 !!x이며, 이는 x가 0이면 0을 그렇지 않으면 1을 저장한다. Return은 b | c를 통해, 원하는 결과를 얻어낸다.

- addOK

변수 a는 x + y의 MSB가 0이면 1, 아니면 0을 저장한다. 변수 b, c는 각각 x와 y의 MSB가 0이면 1을 아니면 0을 저장하도록 한다. 변수 d는 x와 y의 MSB가 같으면 0을 아니면 1을 저장한다. d | !(a ^ b)를 return하며, x와 y각각의 MSB가 같을 경우, overflow가 일어났는지를 판단해서 원하는

결과를 얻어낸다.

- isPositive

변수 a는 x가 0이상일 때, 1을 아니면, 0을 저장한다. 변수 b는 x가 0이면 0을 아니면 1을 저장한다. a & b를 return하여, x가 양수일 때만, 1을 return하도록 한다.

- satMul2

변수 a는 x가 0이상이면 0을 아니면 -1을 저장한다. 변수 b는 x를 1만큼 Shift Left한 값을 저장한다. 변수 c는 변수 b가 0이상이면 0을 아니면 -1을 저장한다. 변수 d에는 변수 a와 c가 같은 값이라면, 즉 x와 x의 1만큼 Shift Left한 값의 부호가 같다면 0을 저장하고, 그렇지 않다면 -1을 저장한다. Return은 ~d & b : 2x가 overflow되지 않은 경우 b의 값, d & ((~a & e) | (a & ~e))) : 2x가 overflow되었을 때, x가 음수라면, 0x7FFFFFFF을 아니라면, 0x80000000의 값, 이 두개를 OR 연산하여 return하면, 원하는 결과를 얻을 수 있다.

- absVal

변수 a를 통해 x가 0이상이라면 0을, 아니라면 -1을 저장한다. Return은 (~a & x) | (a & (~x + 1)) 을 통해, x가 0이상이면 그대로 return하고, 음수라면 negate한 값을 return하도록 구현한다.

- float_neg

변수 a에는 0xFF 8bit를 저장하며, 변수 b에는 0x7FFFFF를 저장한다. if문을 이용해, 만약 uf가 NaN이라면, 자기 자신을 return하도록 하고, 그렇지 않으면 MSB의 값만 변경해서 return하도록 한다.

- float_half

변수 a에는 uf의 sign과 exponent의 9bit를 저장하며, b에는 MSB를 c에는 exponent bit를 d에는 fraction bit를 저장한다. e에는 Round-to-nearest가 필요한지를 판별해서, 필요하다면 1을 저장하도록 한다. if문을 사용해 변수 c를 비교하여, uf가 NaN이거나, Infinity인 경우 자기 자신을 그대로 return하며, exponent가 0또는 1인경우, 변수 e를 사용해서 Round한 값을 1만큼 Shift Right하여 return한다. 나머지의 경우, 지수에서 1을 뺀 값을 return 한다.

- float_f2i

변수 a에는 exponent bit를 b에는 MSB를 저장한다. if문으로 변수 a의 값을 비교해서 만약 32bit 표현의 범위가 넘어간다면, 0x80000000u값을 return하며, exponent가 127보다 작으면 0을 return 하고, 127이라면, MSB에 따라서 1이나 -1을 return 하도록 한다. 나머지의 경우에는 fraction을 exponent – 127 즉, 실제 지수만큼 Shift Left하여 값을 반환한다.

3. Result

Dlc 명령어를 통해 규칙에 적합한 구현임을 확인하였으며, Btest 결과, 모든 함수가 Error없이 통과하였다.