數値의 2進 表現(2)

—Binary Representation of Numerical Value—

2. REAL NUMBER(整數)

整數를 BINARY로 표현하기 위하여 UNSIGNED 및 SIGNED REPRESENTATION을 생각하였는데 이를 기초로 實數(간단하게 整數 및 小數의 집단으로 생각하면 된다.)의 표현 방식을 검토해 보자.

(가) FIXED POINT (固定小數)

8 BIT BINARY 0111 0001의 SIGNED VALUE는 +113이었는데 이는 정수의 표현 방식이었다. 8 BIT로 실수를 표현하기 위해서 그 상위 몇 BIT는 정수로 해석하고 나머지 하위의 BIT를 소수로 해석할 수 있다. 예컨데 $(0111\ 0001)_2$의 상위 5 BIT를 정수, 하위 3 BIT를 소수 즉 01110. 001이라 생각하자.

정수 부분$(01110)_2$의 UNSIGNED VALUE=14

소수 부분$(001)_2$의 UNSIGNED VALUE=1

그래서 실수$(01110.\ 001)_2$는 $14+1/2^3$으로 해석한다.(결과적으로 $113/2^3$이 된다)

일반적인 경우를 보면

$X_{n-1}\cdots X_0.Y_1Y_2\cdots Y_m$의 UNSIGNED VALUE는 $(X_{n-1}\cdot 2^{n-1}+\cdots+X_1\cdot 2^1+X_0\cdot 2^0)+(Y_1\cdot 2^{-1}+Y_2\cdot 2^{-2}+\cdots+Y_m\cdot 2^{-m})$이 된다.

즉, $2^{-m}(X_{n-1}\cdot 2^{n+m-1}+\cdots+X_1\cdot 2^{m+1}+X_0\cdot 2^m+Y_1\cdot 2^{m-1}+Y_2\cdot 2^{m-2}+\cdots+Y_m\cdot 2^0)$가 된다.

괄호 안의 VALUE는 주어진 BINARY의 소수점을 없애버린 것의 UNSIGNED VALUE가 됨을 알 수 있다. 따라서 소수점이 있는 BINARY는 그 BINARY 전체를 정수로 생각하여 얻는 UNSIGNED VALUE를 2^m으로 나눈 값으로 해석할 수 있다. (단, m은 소수자리숫자) 음의 수까지 확장하려면 전체 BINARY를 2'S Complement로 해석하면 된다. 예로서 정수 부분과 소수 부분이 각기 8 BIT인 다음의 16 BIT의 SIGNED VALUE를 계산해보자.

0010 1010. 1101 0011의 SIGNED VALUE는 $(2^{13}+2^{11}+2^9+2^7+2^6+2^4+2^1+2^0)\times 2^{-8}$

1100 1111. 0110 1010의 SIGNED VALUE는 음수이므로 먼저 절대값을 구하기 위하여

1100 1111 0110 1010을 반전시켜

0011 0000 1001 0101

+1

0011 0000 1001 0110 이 절대값이 된다.

그러므로 SIGNED VALUE는

$$-(2^{13}+2^{12}+2^7+2^4+2^2+2^1)\times 2^{-8}$$

이상과 같이 어떤 BINARY의 정해진 일부분을 소수로서 해석하는 방법을 Fixed Point Representation이라 한다. 물론 표시할 수 있는 수의 범위를 넓게 하려면 BIT 수를 크게하면 된다.

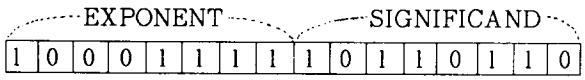
(나) FLOATING POINT(浮動小數)

전항에서 언급한 고정 소수 표현 방식에서 표시할 수 있는 가장 큰 수와 가장 적은 수의 범위를 크게 하려면 BIT 수를 한정없이 늘여야 한다. 이 문제를 해결하는데 앞서서 10진(일반적인 수치 표현)의 예를 들어 보자.

과학 기술상의 수치 표현에서

47300000 대신에 4.73×10^{7}

0.000525 대신에 5.25×10^{-4}등의 표현을 쓰기도 하며 더 큰 수 혹은 더 작은 수의 예로 $+1.53\times10^{40}$ 혹은 -4.25×10^{-29}등도 있다. 이 때 1.53, 4.27을 유효자리 숫자(Significand), 40, −29를 지수 부분(Exponent)라 부르고 물론 +, −는 부호가 된다. 이렇게 함으로써 상당히 큰 수 혹은 작은 수를 간략하게 표현한다. 이런 방식을 BINARY REPRESENTATION에서도 적용할 수 있도록 서로 약속을 할 수 있다. 예컨데 BINARY의 최상위 BIT를 부호, 다음 7개 BIT를 EXPONENT,나머지 8 BIT를 SIGNIFICAND로 하자. EXPONENT는 SIGNED INTEGER로 해석하고 SIGNIFICAND 8 BIT는 최상위 BIT가 정수, 그 아래 7 BIT를 소수로 하는 UNSIGNED FIXED POINT REAL로 보기로 하자.



먼저 SIGNIFICAND VALUE는

$(1011\ 0110)_2$ UNSIGNED VALUE=182

SIGNED VALUE=$182\times1/2^{7}$

EXPONENT VALUE는

$(000\ 1111)_2=+15$

S=−1

따라서 REPRESENTED VALUE는

$-1\times(182/2^{7})\times2^{15}=-46{,}592$가 된다.

위의 수치 표현 방식에서 가장 큰 양수는

0011 1111 1111 1111

$(255\times2^{55}\fallingdotseq9.2\times10^{18})$

가장 작은 양수는

0100 0000 0000 0001

$(1\times2^{-71}\fallingdotseq4.3\times10^{-22})$

위의 표현에서 SIGNIFICAND FIELD를 다시 살펴보면 8 BIT 중 최상위 BIT를 정수로 하였는데(이런 방식의 SIGNIFICAND FORMAT 방식을 DENORMAL MODE라 한다.) PRECISION을 살리기 위하여 SIGNIFICAND 부분을 모두 소수로 보고 정수 부분을 "1"로 생각할 수 있다. 이런 FORMAT 방식을 NORMAL MODE라 한다. 이후로부터는 설명의 편의를 위해 INTEL社의 연산용 MICRO-PROCESSOR에서 채택하고 있는 실수 FORMAT방식을 예로 들어 본다.

INTEL社의 8086 Family MICROPROSSOR가 채택하고 있는 Single Precision Floating Point의 FORMAT는 다음과 같다.

1 BIT	8 BIT	23 BIT
S	EXPONENT	SIGNIFICAND

TOTAL 32 BIT

① SIGN FIELD – 1 BIT(최상위)

S FIELD가 0이면 전체가 양수, 1이면 전체가 음수를 표시한다. ZERO의 표현에 대해서는 별도 설명하기로 한다.

② SIGNIFICAND FIELD – 23 BIT

NORMAL FORMAT에서 23 BIT는 모두 소수 부분인 것으로 해석하고 SIGNIFICAND VALUE는 그 소수의 UNSIGNED VALUE에 1을 더하는 것으로 본다. 일반적으로 SIGNIFICAND FIELD가 X_1 X_2 …… X_{23} 이면 SIGNIFICAND VALUE는 1+(X_1 X_2……X_{23}의 UNSIGNED VALUE)×2^{-23}로 본다. Denormal FORMAT와 ZERO의 Representation에 대해서는 별도로 설명하고자 한다.

* NORMALIZED FORMAT에서는 SIGNIFICAND VALUE가 Zero가 될 수 없음에 유의하기 바란다.

③ EXPONENT FIELD – 8 BIT

EXPONENT라 함은 SIGNIFICAND에 2를 몇번 곱하여 Real Number의 절대값을 구하느냐 하는 것이다. 어떤 컴퓨터, 예컨데 IBM범용 등에서는 2의 몇 제곱 대신에 16의 몇 제곱인가하는 것을 EXPONENT로 삼는다. MICROPROCESSOR에 있어서는 소규모의 Hardware로 연산을 행하기 때문에 2를 Power Factor로 잡는다.

EXPONENT가 음의 값을 가지게 하기 위하여는 EXPONENT FIELD에 실제의 EXPONENT와 어떤 상수의 합을 나타내도록 하는데, 더해 주는 상수를 BIAS라 한다. INTEL의 Single Precision FORMAT의 BIAS는 127로 한다. 그러므로 EXPONENT 값은 EXPONENT FIELD의 UNSIGNED VALLE에서 127을 빼면 된다. 즉 EXPONENT=E−127. 여기에서 E는 8 BIT의 UNSIGNED VALUE이므로 0~255이다. 따라서 EXPONENT=−127~128이 되나 E=0 및 E=255는 별도의 EXCEPTION 처리를 위하여 할당되었기 때문에 실제 EXPONENT는 −126~127까지 가능하다.

④ Real Value of Single Precision Format

이상의 3개 Field를 종합하여 어떤 32 BIT의 Real Value는 다음과 같이 계산된다.

$(-1)^S \times (\text{SIGNIFICAND}) \times 2^{E-127}$

단, S : S FIELD의 UNSIGNED VALUE(O or 1)

SIGNIFICAND=1+(SIGNIFICAND FIELD의 UNSIGNED VALUE)×2^{-23}

E=EXPONENT FIELD의 UNSIGNED VALUE

⑤ Zero 및 Infinity Representation

②항에서 설명한 SIGNIFICAND는 VALUE가 1보다 크므로 NORMALIZED FORMAT에서 Zero를 표시할 수 없게 된다. 또한 ± ∞도 표시할 수 없다. 기타 여러가지 이유로 EXPONENT FIELD의 8 BIT가 모두 "0" 혹은 모두 "1"인 경우는 (E=0 or 255) NORMALIZED FORMAT의 해석 방법(②항 및 ③항)을 벗어나서 별도의 의미를 부여키로 한다.

EXPONENT FIELD와 SIGNIFICAND FIELD가 모두 "0"인 경우는 Zero로 해석하고 EXPONENT FIELD가 모두 "1"이고 SIGNIFICAND FIELD가 모두 "0"이면 ∞(Infinity : 무한대)로 해석한다. 그렇게 하면 Zero에는 +Zero 및 −Zero를 생각할 수 있고 ∞도 +∞와 −∞가 존재할 수 있다. +Zero와 −Zero를 비교하는 경우에는 물론 Equal로 처리될 것이다.

⑥ 표현 가능한 실수 범위

Normalized Format에서 표현 가능한 가장 작은 양수와 가장 큰 양수는 다음과 같다.

○가장 작은 양수

0 000 …1 (8 BIT) 000…00 (23 BIT)

$E=1-127=-126$

SIGNIFICNAD VALUE=1

REPRESENTED

REAL VALUE=$+1\times2^{-126}\fallingdotseq1.17\times10^{-38}$

○가장 큰 양수

0 111……10 (8 BIT) 1111……11 (23 BIT)

$E=254-127=127$

SIGNIFICAND VALUE=$1+(0.\underbrace{1111\cdots11}_{23\ BIT})$

$=1+(1-2^{-23})=2-2^{-23}$

REPRESENTED

REAL VALUE = $+2^{+127}\times(2-2^{-23})$

$=+(2^{128}-2^{104})\fallingdotseq3.4\times10^{38}$

⑦ Denomal

⑥항에서 본 바와 같이 Normalized Format으로 표시할 수 있는 가장 작은 양수는 $+2^{-126}$인 것을 알 수 있다. 문제가 되는 것은 대단히 작은 양수 2개를 서로 곱하면 더 작 은 양수가 되어서 2^{-126}보다 작은 경우가 생기는데 간략하게 나마(Precision의 상실에도 불구하고) 이를 표현하는 방법으로, EXPONENT FIELD가 모두 "0"인 경우의 SIGNIFICAND VALUE는 SIGNIFICAND FIELD의 UNSIGNED VALUE(1을 더하지 않은 것)로 해석한다.

이렇게 하면 가장 작은 양의 실수는 0 0000 0000 0 …… 1이 된다.

$E=0-127=-127$

SIGNIFICAND VALUE=2^{-23}

그러므로 REAL VALUE=$+2^{-127}\times2^{-23}$

$=2^{-150}\fallingdotseq7.0\times10^{-46}$

이와 같이 EXPONENT FIELD가 모두 0인 경우에는 SIGNIFICAND FIELD가 Normalized 되지 않은 것으로 해석하며 이런 것을 Denormals라 한다.

⑧ EXCEPTION의 발생

BINARY에 의하여 모든 연속된 실수를 표시할 수는 없다. 그 BINARY에 의하여 표시된 수치로 연산을 행하여 얻은 결과를 BINARY로 정확하게 표시할 수는 없다. Hardware가 행한 연산 결과가 주어진 FORMAT으로 표시될 수 없는 것을 Exception이 발생하였다고 하며 여기에 대한 각종 대책을 세워야 하는 것이 Computer에서는 매우 어려운 일중의 하나이다. 예를 들어 (−1)의 제곱근은 존재하지 않으므로 이런 경우의 계산명령을 받았을 때에 대비한 것이 Exception 처리인 것이다.

Floating point Real에 대해 좀더 자세한 내용은 the Proposed IEEE Standard Floating Point Arithmetic for Microprocessors를 참고하기 바란다.

註 : 본기사를 이해하기 위해서는 방화정보 45호의 "수치의 2진 표현"을 먼저 읽어보아야 한다.