數値의 2進 表現
-BINARY REPRESENTATION OF NUMERICAL VALUE-

Digital System에서 모든 Data는 High Voltage 혹은 Low Voltage의 연속적인 Combination으로 처리·표현된다. High Voltage를 "1", Low Voltage를 "0"으로 대신하여 表現하는 正論理 系統(positive Logic System)에서 數値 Data를 어떻게 表現하는지 알아 보기로 하자.

1. INTEGERS(整數)

(가) Unsigned Representation

우리는 0~9의 10가지 기호를 연속적으로 나열하여 正數를 표현한다.

예컨대, 「47063」은 3개의 $10^0(=1)$, 6개의 $10^1(=10)$, 0개의 $10^2(=100)$, 7개의 $10^3(=1{,}000)$, 4개의 $10^4(=10{,}000)$의 合을 표현한 것이다. 이런 식의 표현을 10진(Decimal)표현이라 한다.

일반적인 n 자리의 10진 표현 $Y_{n-1}\ Y_{n-2}\ \cdots\cdots\ Y_1\ Y_0$($Y_{n-1}$~$Y_0$ 는 각기 0~9)은

$Y_{n-1}\cdot 10^{n-1}+Y_{n-2}\cdot 10^{n-2}+\ \cdots\cdots\ +Y_1\cdot 10^1+Y_0\cdot 10^0$

$=\sum_{i=1}^{n-1} Y_i\cdot 10^i$을 표현하는 것이다.

Digital System에서는 각 자리가 "0" 혹은 "1"이므로 Binary(2進) Digit(1位)가 되고 이를 간단히 Bit라 부른다.

예컨대 4 Bit 표현인 1101의 각 Bit는 하위로부터 $B_0=1$, $B_1=0$, $B_2=1$, $B_3=1$이므로 이를 수치로 해석하면, 1개의 $2^0(=1)$, 0개의 $2^1(=2)$, 1개의 $2^2(=4)$, 1개의 $2^3(=8)$의 合 즉, 10진수의 13을 표현하고 있다.

일반적으로 n자리의 2進 表現인 $X_{n-1}\ \cdots\cdots\ X_0$($X_{n-1}$~$X_0$는 각기 "0" 또는 "1")은 $X_{n-1}\cdot 2^{n-1}+\cdots\cdots+X_2\cdot 2^2+X_1\cdot 2^1+X_0\cdot 2^0=\sum_{i=1}^{n-1} X_i\cdot 2^i$를 表現한다.

n자리의 2進 表現 중 가장 큰 수치의 것은 111…11($=2^{n-1}+\cdots\cdots+2^2+2^1+2^0=2^n-1$)이고, 가장적은 것은 000…00($=0$)이다.

예컨데, 4자리 2進 表現 중 가장 큰 것은 1111($=1\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1.2^0=15$)이고, 8자리의 2進 表現에서는 1111 1111($=2^7+2^6+\cdots\cdots+2^1+2^0=255$)가 가장 큰 것이다. 더 큰 수치(정수)를 표현하려면 당연히 Bit수(자리수)를 늘이면 된다.

이상의 방식으로 수치를 해석하면 0 및 양의 정수가 되며 Unsigned Representation이라 한다. Unsigned Representation에 의해 표현한 수치를 Unsigned Value라 한다.

Unsigned Representation에서 덧셈과 뺄셈은 다음과 같다.

먼저 덧셈에서

$0+0=0$, $0+1=1$은 당연하다.

$1+1=10$ 즉 당해 자리는 0이 되고 윗자리로 자리 올림이 생긴다.

4 자리에 있어서의 덧셈은 다음과 같다.

```
  0101
 +0111
------
  1100 → 1+1=10, 당해 자리는 0, 윗자리로 자리 올림발생
```

0+1=1에 아랫자리로부터 받은 자리 올림을 계산하면 1+1=10 즉 당해 자리는 0이 되고 윗자리로 자리 올림 발생.

1+1=10 윗자리로 자리 올림이 발생하고 아랫자리로 부터 받은 자리 올림을 계산하면 당해 자리는 0+1=1.

0+0=0 윗자리로부터 받은 자리 올림을 계산하면 0+1=1.

8 bit에 있어서는 다음과 같다.

```
  0110 1010(106)          0100 1101( 77)
 +0011 0011( 51)         +1001 1110(158)
---------------         ---------------
  1001 1101(157)          1110 1011(235)
```

뺄셈에서는

0−0=0

1−0=1

1−1=0은 당연하다.

0−1에서는 해당 자리에 1이 남고 그 상위에서 1을 뺀다. 즉 10−1=01, 100−1=011, 1000−1=0111 등이다.

4 자리 뺄셈의 예로

```
  1010(10)          1010(10)
 −0101( 5)         −0111( 7)
----------        ----------
  0101( 5)          0011( 3)
```

8 Bit의 예는 다음과 같다.

```
  1101 0001(209)          1000 1101(141)
 −0110 1110(110)         −0001 1011( 27)
---------------         ---------------
  0110 0011( 99)          0111 0010(114)
```

(나) 16진 표현−Hexadecimal Representation.

Digital System에서는 Binary Representation에 의해 정확히 처리, 표현되지만 우리가 이를 기술하는 데는 너무 길어서 불편하다. Digital System내에서 최근에는 8 Bit, 16 Bit 혹은 32 Bit 단위 등으로 처리하는 것이 보통이다. 그래서, 연속된 4 Bit씩 나누어 Group을 짓고 각 Group을 아래표에 따라 한 자리 기호로 나타내면 기술이 간략하게 된다.

예컨대 0101 1010은 5A

0111 1001은 79등이다. 이렇게 표현한 것을 16진 표현이라 한다. 16진 표현 79와 우리가 흔히 쓰는 10진 수 79를 구분하기 위해 16진 79는 $(79)_{16}$, 10진 79는 $(79)_{10}$, 2진 $(1101)_2$ 등으로 기술한다. $(79)_{16}$은 H79, 79H로도 기술한다.

HEXADECIMAL CONVERSION TABLE

Binary Representation	Hexadecimal Representation	Unsigned Value(Decimal)
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

예컨대 HEF은 $(1110\ 1111)_2$이고 0110 1100은 H6C가 된다.

H73AC는 4×4=16 Bit의 $(0111\ 0011\ 1010\ 1100)_2$가 된다.

16진 2位(8Bit)의 Unsigned Value는 다음과 같이 직접(Binary로 바꾸지 않고도) 쉽게 구해진다.

HE7 $(231)_{10}$

상위의 Unsigned Value 14×16=224

하위의 Unsigned Value 7× 1= 7

계 231

일반적으로 $HX_{n-1}\ X_{n-2}\cdots\cdots X_2X_1X_0$의 Unsigned Value는 $\sum_{i=0}^{n-1}$(X_i의 Unsigned Value×16^i)가 된다.

(다) Signed Representation

(가)항에서 다룬 Unsigned표현에서는 음의 정수를 나타내지 못하는 문제점이 있다.(음수를 다루지 않는 경우에는 아무런 문제점이 없다). 음의 정수를 표현하기 위하여 8 Bit의 경우 최상위 Bit는 부호를 표시하고, 하위 7 Bit는 절대치를 표시하기로 할 수 있다. 최상위 Bit가 0이면 양수, 1이면 음수로 생각하고 하위 7 Bit의 Unsigned Value가 절대값을 표시하도록 하면 된다.

〈Sign Magnitude의 예〉

Binary	Sign Magnitude Value
0000 0000	0
0000 0001	+1
1000 0001	−1
0000 0010	+2
1000 0010	−2
0111 1111	+127
1111 1111	−127

이와 같은 수치 표현을 Sign Magnitude Representation이라 한다. 이상의 Sign−Magnitude방식은 정수의 범위를 음으로 확장하기 위해 도입한 것으로 표현의 간명성에서는 성공적이다. 그러나 계산 방식에 취약성이 있다.

〈Sign Magnitude의 계산 예〉

0000 0001	Sign Magnitude Value = +1
+1000 0001	〃 = −1
0000 0000	〃 = 0

위의 예에서 보는 바와 같이 Sign Magnitude에서 (+1)과 (−1)의 덧셈 결과는 Unsigned에서의 덧셈 결과$(1000\ 0010)_2$와 판이하게 다르다. 그래서 Unsigned와 다른 방식의 덧셈 방식을 고안해야 하는 결점이 지적된다. 따라서 Sign Magnitude 방식은 Digital System에서 별로 채택되지 않는 것 같다.

정수의 범위를 음으로 확장하면서 Unsigned에서와 똑같은 방식으로 덧셈 뺄셈을 할 수 있도록 고안한 것이 「2의 補數」(Two's Complement) 방식이다.

8 Bit에서 0000 0001의 Unsigned Value는 (+1)이고 이를 1111 1111과 합하여 얻은 Binary의 하위 8 bit는 0000 0000 즉 0이 되므로 1111 1111의 값을 (−1)로 볼 수 있다.

```
  0000 0001  (+1)
+ 1111 1111  (−1)
-------------------
  0000 0000  ( 0)
```

수치를 해석함에 있어서

첫째로 최상위 Bit가 0인 경우는 양수로 보고 그 Unsigned Value를 절대값으로 해석하고,

둘째로 최상위 Bit가 1인 경우는 음수로 보되, 당해 Binary와 덧셈을 하여 0이 되는 Binary의 Unsigned Value를 절대값으로 해석하도록 한다면 Unsigned의 덧셈, 뺄셈과 같은 방식을 사용할 수 있게 된다. 이런 방식에서 8 Bit Binary의 Value는 다음과 같다.

Binary	Signed Value	Unsigned
0000 0000	0	0
0000 0001	+1	1
0000 0010	+2	2
⋮	⋮	⋮
0111 1111	+127	127
1000 0000	−128	128
1000 0001	−127	129
⋮	⋮	⋮
1111 1110	−2	254
1111 1111	−1	255

덧셈 · 뺄셈이 Unsigned에서와 같은 것을 확인해 보기 바란다.

이런 방식을 Two's Complement 방식이라 하는데, 최상위 Bit가 0이면 Unsigned에서와 마찬가지이므로 값의 계산이 용이하다.

음수인 경우는 다음 예와 같이 한다.

예를 들어 음의 Two's Complement 1101 1011의 Signed Value를 구하기 위해 각 Bit를 반전시킨(1을 0으로, 0은 1로) Binary 0010 0100을 얻어 여기에 1을 더한다.

```
  0010 0100
+ 0000 0001
-----------
  0010 0101    (Unsigned Value=37)
```

예에서 주어진 Binary의 Signed Value는 −37이 된다.

위와 같은 방식으로 해석하는 Two's Complement의 값을 Signed Value라 한다.

*Computer의 경우 거의 모든 연산은 Hardware 상에서 Two's Complement 상태로 시행하고 최종 결과를 출력하거나 할때만 Value를 계산하는 것이므로 위에서와 같이 Signed Value 혹은 Unsigned Value를 10진으로 계산해 내는 일은 빈번한 것이 아니며, 실제 출력하기 위하여 10진으로 계산해내는 일은 System Designer 등의 전문가들이 수단을 마련한다.

(라) Extending Binary Representation

8 Bit로 표현된 수치를 16 Bit 혹은 32 bit 등으로 표현하거나 16 Bit로 표현된 수치를 32 Bit 등으로 자리수를 늘려서 표현하는 것을 Bit Extension 이라 한다. Bit Extension할때는 Unsigned Representation과 Two's Complement(Signed Representation)이 서로 다르므로 주의하여야 한다. Unsigned 8 Bit를 16 Bit로 Extend할 때는 상위 8 Bit를 모두 "0"으로 채우면 된다. 8 Bit Two's Complement에 있어서는 약간 다르다. 먼저 그 Binary가 양인지 음인지 판단하여 양이면 Unsigned와

8 Bit Binary	Unsigned Value	Signed Value	Unsigned Extension	Signed Extension
0000 0001	1	+1	0000 0000 0000 0001	0000 0000 0000 0001
0111 1111	127	+127	0000 0000 0111 1111	0000 0000 0111 1111
1000 0000	128	−128	0000 0000 1000 0000	1111 1111 1000 0000
1111 1110	254	−2	0000 0000 1111 1110	1111 1111 1111 1110
1111 1111	255	−1	0000 0000 1111 1111	1111 1111 1111 1111

같이 하고 음이면 상위 8 Bit에 "1"을 채워야 한다. 다음 예를 잘 살펴 착오가 없기 바란다.

이상에서 Integer의 Binary표현 방식을 간략하게 살펴보았는데 Computer System에서 Integer처리는 불가결한 것이며 앞으로 다루게 될 실수 표현의 기초가 되는 것이므로 실제로는 매우 중요한 사항이다. 앞으로 기회가 있으면 Unsigned Integer의 Binary 표현에 좀더 숙달될 수 있는 방법을 설명하기로 한다.