Data Representation

컴퓨터개론

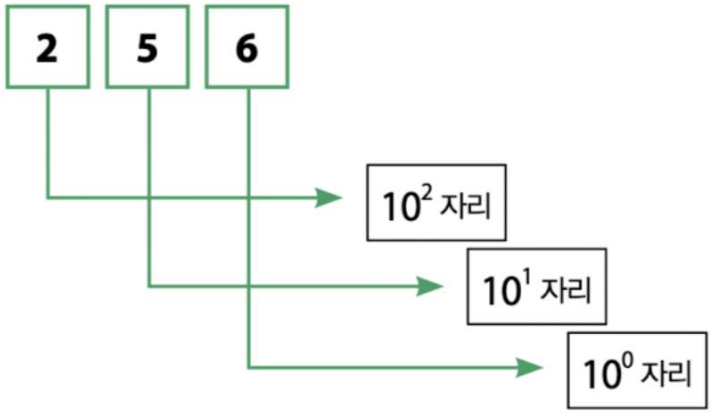
(Introduction to Computer Systems)

GEN1030

컴퓨터의 수 표현

자료 표현 원리

- **10진수(Decimal):** 사람에게 가장 익숙
  - 0부터 9까지 숫자를 사용
  - 10을 기수(base)라고 함
  - 가장 오른쪽은 10^0인 단 단위, 그 왼쪽은 10^1인 십 단위
  - 어느 자릿수는 바로 근접한 오른쪽 자릿수의 10배로 커짐



자료 표현 원리

- 컴퓨터에게 가장 적합한 수의 표현은?
- 컴퓨터 내부에는 전기가 흐르거나(**On**) 흐르지 않는(**Off**) 두 가지 전기 신호만을 표현 가능
- 트랜지스터(Transistor)
  - 전류 크기를 조절하는 반도체 소자
  - 컴퓨터의 중앙처리장치(CPU)는 수십억 개의 트랜지스터로 이루어짐
  - 전류를 켜고 끌 수 있음 (On/Off) ➔ 두 가지 상태

- 2진수(Binary)
  - 숫자 0과 1만 사용
  - 컴퓨터는 논리의 조합이 간단하고, 사용되는 소자(트랜지스터) 특성상 이진법을 사용하는 것이 효율적
  - 전기가 흐를 경우(On) '참(True)' 의미하는 '1' , 흐르지 않을 경우(Off) '거짓(False)' 의미하는 '0'

자료 표현 원리

- 2진수의 기수(base)? 2
- 가장 오른쪽은 2^0인 단, 왼쪽으로 갈수록
  - $2(2^1)$ 단위
  - $4(2^2)$ 단위
  - $8(2^3)$ 단위
  - …
- 2진수 10010
  - 10010_2

$16(2^4)$		$8(2^3)$		$4(2^2)$		$2(2^1)$		$1(2^0)$		
x		x		x		x		x		
1		0		0		1		0		
‖		‖		‖		‖		‖		
16	+	0	+	0	+	2	+	0	=	**18**

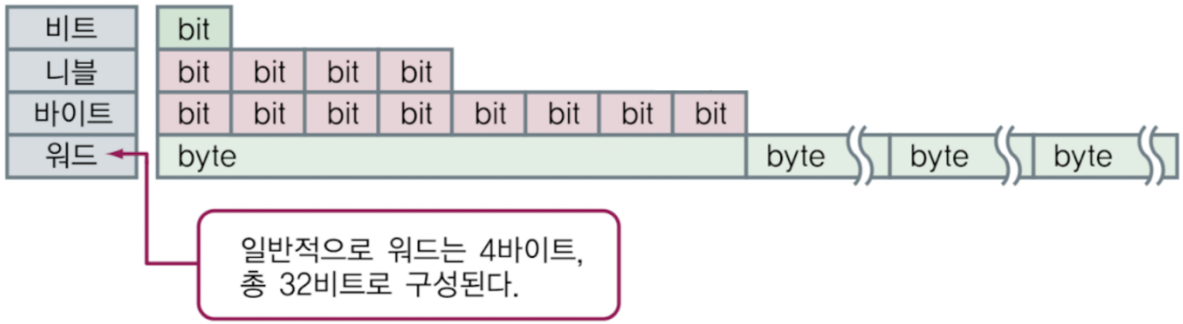
자료 표현 원리

- 2진수의 기수(base)? 2
- 가장 오른쪽은 2^0인 단, 왼쪽으로 갈수록
  - $2(2^1)$ 단위
  - $4(2^2)$ 단위
  - $8(2^3)$ 단위
  - …
- 2진수 11111
  - 11111_2

$16(2^4)$		$8(2^3)$		$4(2^2)$		$2(2^1)$		$1(2^0)$		
x		x		x		x		x		
1		1		1		1		1		
‖		‖		‖		‖		‖		
16	+	8	+	4	+	2	+	1	=	**31**

정보의 표현

- **Bit** (**Bi**nary digit): 컴퓨터 메모리의 저장 단위 또는 정보 처리 단위 중에서 가장 작은 단위
  - 두 가지 정보만 표현 - 전기의 흐름 상태인 On, Off 표현하는 단위가 Bit
- **Nibble**: 4개의 bit 묶은 단위 (4 bits)
- **Byte**: 8개의 bit 묶은 단위 (8 bits)
- **Word**: 4개 혹은 8개의 byte 묶은 단위
  - 4개는 32 bits, 8개는 64 bits
  - CPU 아키텍쳐(architecture)에 따라 달라짐
  - 32-bit CPU (32 bit를 한번에 처리할 수 있는 CPU)기준 4 byte



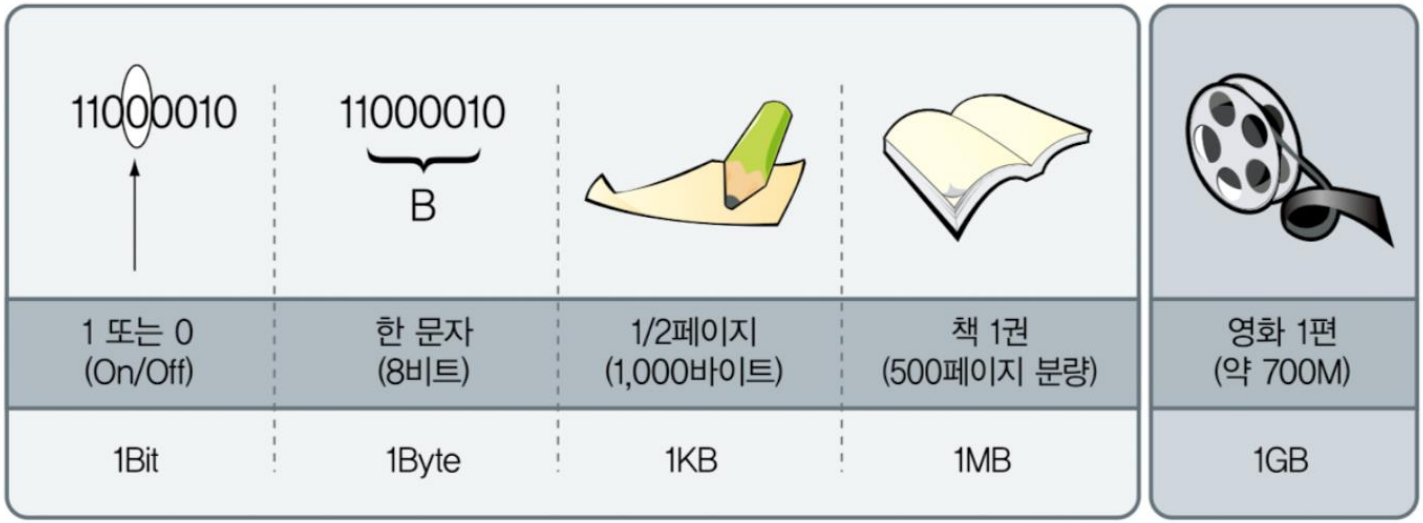
저장 용량

- 파일(File), 저장장치(Storage) 등의 크기를 나타내는 단위
  - Byte가 정보 용량의 단위
  - Kilo = 2^{10} = 1024개

표기	단위	계산	바이트 수	계량 단위
B	Byte	2^0	1	
KB	Kilo Byte	2^{10}	1,024	천
MB	Mega Byte	2^{20}	1,048,576	백만
GB	Giga Byte	2^{30}	1,073,741,824	십억
TB	Tera Byte	2^{40}	1,099,511,627,776	조
PB	Peta Byte	2^{50}	1,125,899,906,842,624	천조
EB	Exa Byte	2^{60}	1,152,921,504,606,846,976	백경
ZB	Zetta Byte	2^{70}	1,180,591,620,717,411,303,424	십해
YB	Yotta Byte	2^{80}	1,208,925,819,614,629,174,706,176	자

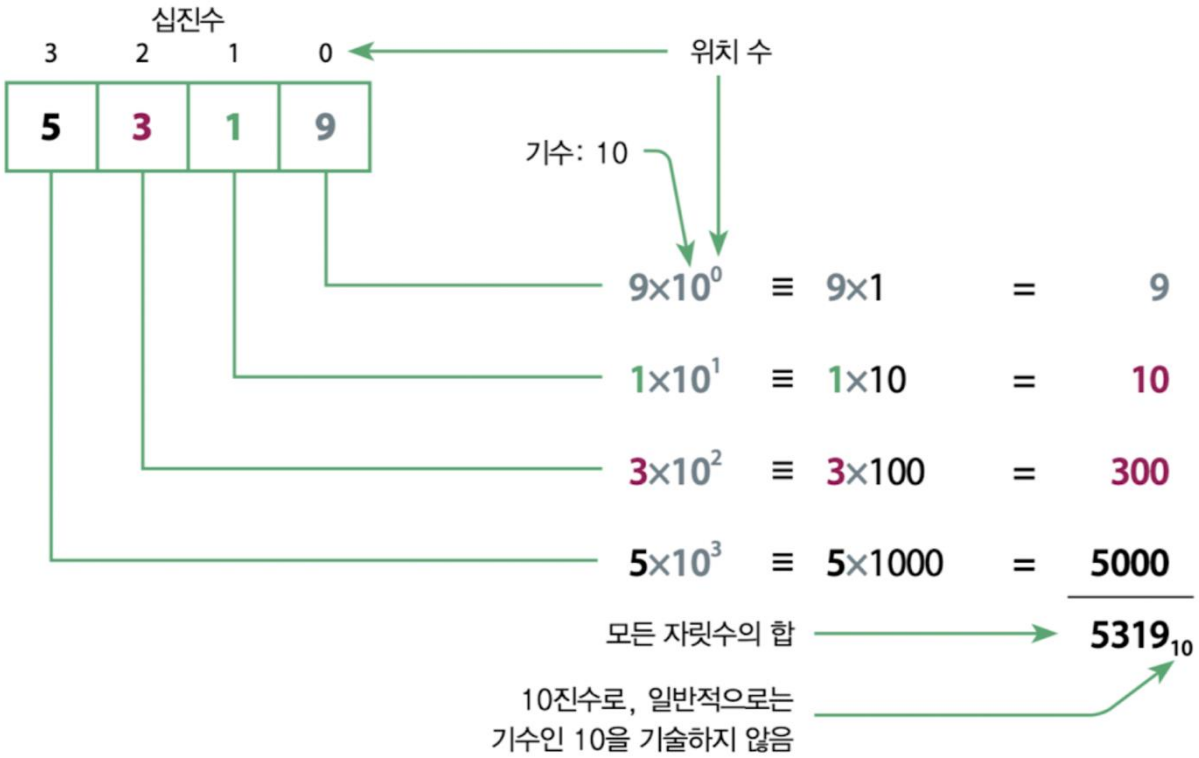
저장 용량

- 파일(File), 저장장치(Storage) 등의 크기를 나타내는 단위
- 실생활 예시



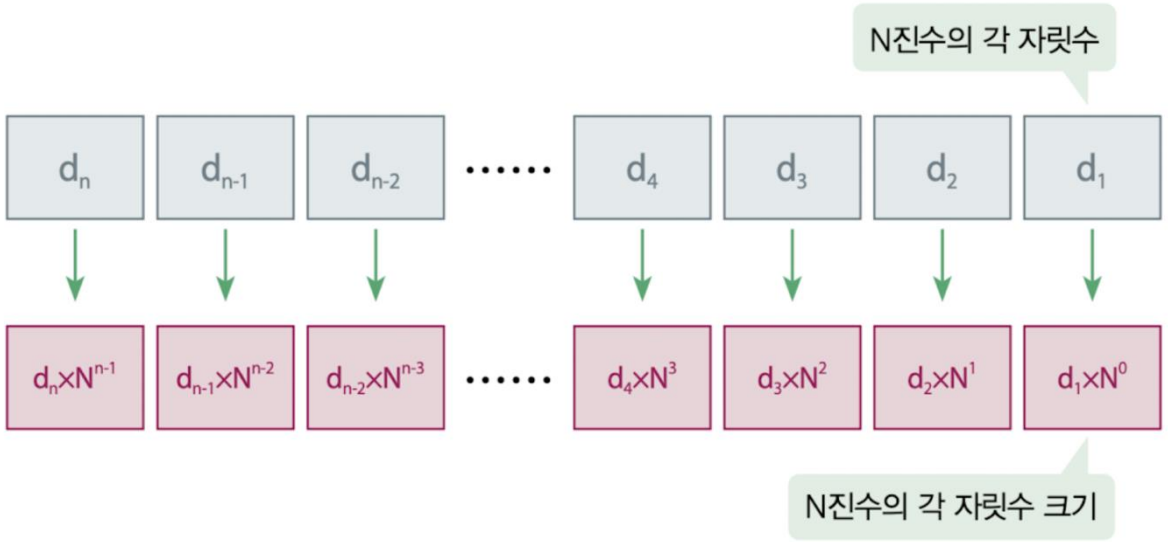
진수의 종류

- 10진수
  - 우리 인간이 일상 생활에서 이용하는 가장 친숙한 진수



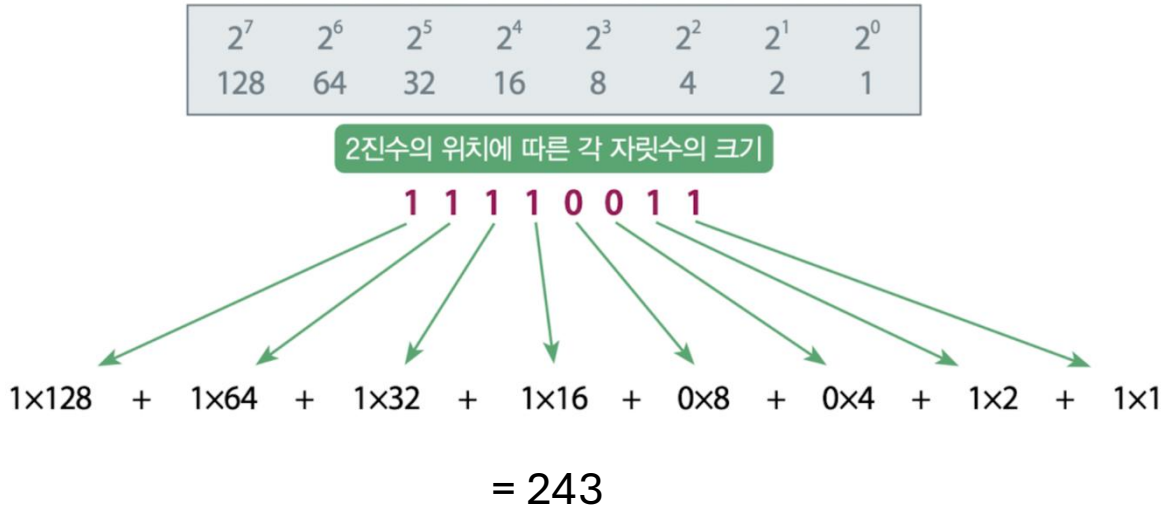
진수의 종류

- N진수
  - 0에서 N-1까지의 정수를 이용
  - 오른쪽부터 n번째 자리의 크기는 N^{n-1}



진수의 종류

- 2진수
  - 0과 1의 두 가지 표현으로 각 자릿수를 표시



= 243

진수의 종류

- 8진수
  - 0부터 7까지(8개) 이용하여 숫자 표현

$$
\begin{aligned}
&301_8 \\
&= 3 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 \\
&\quad = \quad 192 + \quad 0 + \quad 1 \\
&\quad = \quad 193
\end{aligned}
$$

- 16진수
  - 16개의 **숫자나 문자**를 이용하여 표현
  - 숫자: 0부터 9까지
  - 문자: A부터 F까지, 각 알파벳은 10~15에 대응
    - 소문자 a~f로도 표현 가능

$$
\begin{aligned}
&1\mathrm{AF}_{16} \\
&= 1 \times 16^2 + \mathrm{A} \times 16^1 + \mathrm{F} \times 16^0 \\
&\quad = \quad 256 + \quad 160 + \quad 15 \\
&\quad = \quad 431
\end{aligned}
$$

*컴퓨터 내부에서는 2진수를 사용하지만, 실제로는 16진수로 표현하는 경우 많음

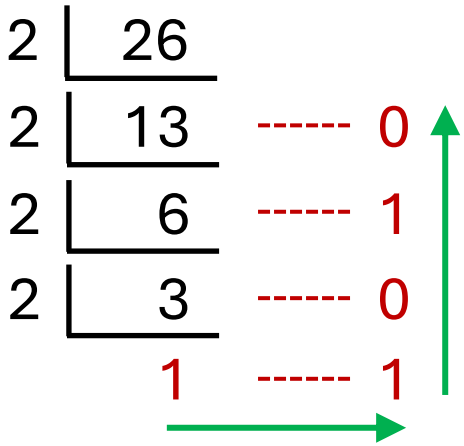
A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

진수의 변환

- 10진수를 2진수로 변환

1단계: 주어진 값을 2로 나누고 그 나머지를 기록
2단계: 몫이 0이 아니면 계속해서 새로운 몫을 2로 나누고 그 나머지는 기록
3단계: 몫이 0이면 원래 값의 2진 표현은 나머지가 기록되는 순서대로
왼쪽에서 오른쪽으로 나열

$26 = ??_2$



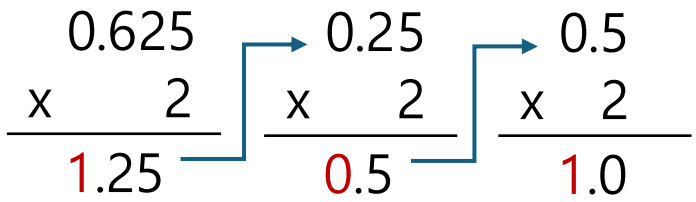
$26 = 11010_2$

진수의 변환

- 10진수의 소수를 2진수로 변환

1단계: 10진수에 2를 곱하여 나온 결과에서 정수 부분으로의 자리 올림수와 소수점 아래 부분을 따로 보관한다.
2단계: 단계 1에서 소수 부분이 0이면 단계 3으로 넘어가고, 아니면 소수점 아래 부분을 다시 새로운 10진수로 하여 1단계를 반복한다.
3단계: 구해진 정수 부분으로의 자리 올림수를 순서대로 나열한다.

$0.625 = ??_2$



$0.625 = 0.101_2$

진수의 변환

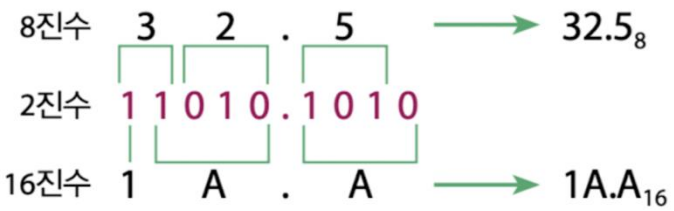
- 10진수의 소수를 2진수로 변환

1단계: 10진수에 2를 곱하여 나온 결과에서 정수 부분으로의 자리 올림수와 소수점 아래 부분을 따로 보관한다.
2단계: 단계 1에서 소수 부분이 0이면 단계 3으로 넘어가고, 아니면 소수점 아래 부분을 다시 새로운 10진수로 하여 1단계를 반복한다.
3단계: 구해진 정수 부분으로의 자리 올림수를 순서대로 나열한다.

$$0.375 = ??_2$$

진수의 변환

- 2진수로 표현된 수를 8진수, 16진수로 쉽게 변환 가능
  - $8 = 2^3$, $16 = 2^4$
- 변환 방법
  - 소수점을 기준으로 정수 부분은 왼쪽으로, 소수 부분은 오른쪽으로
  - 2진수 → 8진수
    - 2진수의 3자리 씩을 8진수로 변환
  - 2진수 → 16진수
    - 2진수의 4자리 씩을 16진수로 변환



2진수의 음수 표현

- 음수 표현을 위해서는 먼저 Bit 크기를 정하고 수를 표현

- **보수**
  - 쉽게 말하면 보충하는 수
  - 각 자릿수의 수와 보수를 더하면 진수의 자리올림이 발생하고 해당 자릿수는 0이 되도록 하는 수

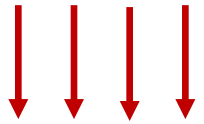
  - 10진수에서 4의 보수?
    - 6

    → 4 + 6 = 10

2진수의 음수 표현

- **1의 보수(1's complement)**
  - 주어진 이진수의 bit를 0은 1로, 1은 0으로 각각 변환하는 방법

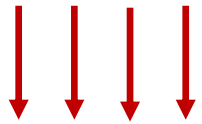
0 1 0 0



1 0 1 1

  - 단점: 0이 +0과 -0으로 각각 다르다
  - +0 = 0000 / -0 = 1111

0 0 0 0 **+0**



1 1 1 1 **-0**

2진수의 음수 표현

- **2의 보수(2's complement)**
- 변환 방법 1

N bit에서 –a의 2의 보수 계산 방법:
2^N - a

예시) 4 bit 사용
-4 = ?

$2^4 - 4$
= 12
= 1100_2

예시) 4 bit 사용
-6 = ?

2진수의 음수 표현

- **2의 보수(2's complement)**
- 변환 방법 2
  - 음수의 2의 보수는 1의 보수 값보다 1이 크다는 것을 이용

1단계: 음수의 절대값인 양의 정수의 이진수를 N bit에서 구한다
2단계: 1단계에서 얻은 이진수의 1의 보수를 N bit에서 구한다
3단계: 2단계에서 얻은 이진수에 1을 더한 N bit만을 취한다

예시) 4 bit 사용,
-4 = ?

1단계: 0100

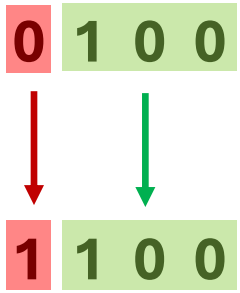
2단계: 1011

3단계: 1011 + 1 = 1100

2진수의 음수 표현

- **2의 보수(2's complement)**
- 변환 방법 3
  - 양수의 N bit 2진수에서 가장 오른쪽의 0에서 처음으로 나오는 1까지 그대로 두고 나머지 왼쪽 bit를 모두 1의 보수로

예시) 4 bit 사용,
-4 = ?



1의 보수 vs. 2의 보수

- 1의 보수
  - 단점: 1 의 보수는 0이 두 가지로 표현됨 (-0000, +0000)
  - 4 bit에서 -8을 표현 불가능
- 2의 보수
  - 1의 보수의 단점을 보완
  - 4 bit에서 -8부터 +7까지 까지 표현 가능

[방법 2] 0000 → 1111

+1

10000

0000

1의 보수 vs. 2의 보수

- 1의 보수
  - 단점: 1 의 보수는 0이 두 가지로 표현됨 (-0000, +0000)
  - 4 bit에서 -8을 표현 불가능
- 2의 보수
  - 1의 보수의 단점을 보완
  - 4 bit에서 -8부터 +7까지 까지 표현 가능

숫자	2진수(양수)	1의 보수	2의 보수
0	0000	0000 1111	0000
-1	0001	1110	1111
-2	0010	1101	1110
-3	0011	1100	1101
-4	0100	1011	1100
-5	0101	1010	1011
-6	0110	1001	1010
-7	0111	1000	1001
-8	1000	-	1000

2진수의 표현: 부호 비트

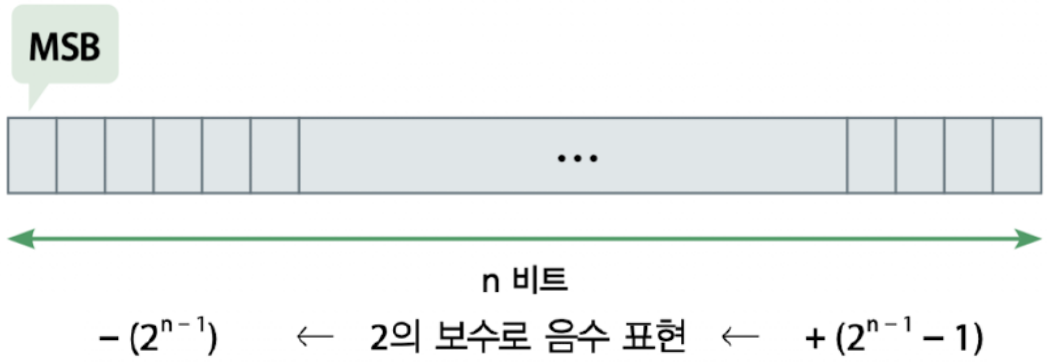
- 최상위 비트(Most Significant Bit, MSB)
  - 비트 표현에서 가장 왼쪽의 비트

- 부호 비트 (Sign bit)
  - 보수 표현에서는 MSB가 부호를 나타내므로 부호비트 라고도 부름
  - 2의 보수에서 **음수의 MSB = 1**

컴퓨터의 정보 종류

부호가 "있는" 정수 표현

- Signed 정수
  - 양수와 음수를 표현하는데 주로 2진수 및 2의 보수 방법을 이용
- n개 bit로 정수의 양수, 음수 표현 할 때
  - 수의 범위: $-(2^{n-1})$ ~ $+(2^{n-1}-1)$



부호가 "있는" 정수 표현

- Signed 정수
  - 양수와 음수를 표현하는데 주로 2진수 및 2의 보수 방법을 이용
- n개 bit로 정수의 양수, 음수 표현 할 때
  - 수의 범위: $-(2^{n-1})$ ~ $+(2^{n-1}-1)$

저장공간 크기	표현 범위
1비트	$-1(-2^0) \sim 0(2^0-1)$
2비트	$-2(-2^1) \sim 1(2^1-1)$
4비트	$-8(-2^3) \sim 7(2^3-1)$
8비트	$-128(-2^7) \sim 127(2^7-1)$
16비트	$-32,768(-2^{15}) \sim 32,767(2^{15}-1)$
32비트	$-2,147,483,648(-2^{31}) \sim 2,147,483,647(2^{31}-1)$

부호가 "있는" 정수 표현

- Signed 정수
  - 양수와 음수를 표현하는데 주로 2진수 및 2의 보수 방법을 이용

- n개 bit로 정수의 양수, 음수 표현 할 때
  - 수의 범위: $-(2^{n-1})$ ~ $+(2^{n-1}-1)$
  - ex) 8 bit

숫자	2진수	숫자	2진수
–	–	−128	10000000
+127	01111111	−127	10000001
+126	01111110	−126	10000010
…	…	…	
+5	00000101	−5	11111011
+4	00000100	−4	11111100
+3	00000011	−3	11111101
+2	00000010	−2	11111110
+1	00000001	−1	11111111
0	00000000	–	–

부호가 "없는" 정수 표현

- Unsigned 정수
  - 0과 양수만을 다루는 정수 표현
  - n개 bit 일때 수의 범위: ~ $+(2^n-1)$
  - ex) 8 bit: 2^8(=256)개 정보 표현 가능

4비트 정보	unsigned	signed 1의 보수	signed 2의 보수
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7

4비트 정보	unsigned	signed 1의 보수	signed 2의 보수
1000	8	−7	−8
1001	9	−6	−7
1010	10	−5	−6
1011	11	−4	−5
1100	12	−3	−4
1101	13	−2	−3
1110	14	−1	−2
1111	15	−0	−1

오버플로(Overflow)

- N개의 비트로는 표현 한계가 있음
  - ex) 8 bit는 256개 값만 표현 가능
- 오버플로(Overflow)
  - 컴퓨터가 표현할 수 있는 숫자 범위를 초과하는 값을 계산하거나 저장하는 경우
  - 컴퓨터는 사용하는 bit 수가 고정 범위를 넘어가는 순간 순환(wrap around)

- 예시 1 $7 + 1 = 0111_2 + 0001_2 = 1000_2$
  $= -8$
- 예시 2 $-5 + -4 = 1011_2 + 1100_2 = 10111_2$
  $\rightarrow 0111_2$
  $= 7$