2.1 2진법 (Binary Notation)

1. **컴퓨터는 0과 1의 수를 사용하여 수를 표현함**
2. **10진법: 숫자 369 - 3자리 수**
   A. 10의 0승, 10의 1승, 10의 2승에 해당하는 값을 각 자리의 수에 곱하여 합산한 값을 의미
3. **2진법: 숫자 1001 – 4자리 수**
   A. 2의 0승, 2의 1승, 2의 2승, 2의 3승에 해당하는 값을 각 자리의 수에 곱하여 합산한 값을 의미

10진법	2진법
369 $=3\times100+6\times10+9\times1$ $=3\times10^2+6\times10^1+9\times10^0$	1001 $=1\times8+1\times4+1\times2+1\times1$ $=1\times2^3+0\times2^2+0\times2^1+1\times2^0$ $=9$ $=0\times9$ (16진수) $=011$ (8진수)

2.2 10진수와 2진수의 변환

1. 10진법 수(10진수)를 2진법 수(2진수)로 변환하는 방법
2. 10진수 13을 2진수로 변환

- 단계1 : 주어진 값을 2로 나누고 나머지를 기록
- 단계2 : 단계1의 몫이 0이 아니면 단계1로 이동
- 단계3 : 몫이 0이므로, 나머지에 해당하는 값들을 오른쪽에서 왼쪽으로 나열

13	13/2		6/2		3/2		1/2
몫	6	→	3	→	1	→	0
나머지	1		0		1		1
	←						

2.2 10진수와 2진수의 변환

1. 10진수 27을 2진수로 변환 11011

			나머지
1회	27 / 2	→	1
2회	13 / 2	→	1
3회	6 / 2	→	0
4회	3 / 2	→	1
5회	1 / 2	→	1
	0		

나머지 : 1 1 0 1

간단하게 변환의 결과값이 맞는지 테스트하기 위해서는 윈도우의 계산기를 활용하면 된다. 10진수Decimal digit를 입력한 후에 2진수Binary digit를 선택하면 해당 10진수에 대한 2진수 표현을 나타내준다.

2.3 2진수의 덧셈

1. 10진수 덧셈과 동일하게 가장 오른쪽 자리 숫자를 더하고
   그 합에서 작은 자리 숫자를 가장 오른 자리에 적고
   합의 큰 자리에 올림수가 있으면,
   왼쪽으로 넘긴 다음 덧셈을 계속

0 + 0 0	1 + 0 1	0 + 1 1	1 + 1 10

2.4 16진법(Hexadecimal Notation)

1. **16진법: 긴 비트 패턴을 위한 간이 표기법**
   A. 비트 패턴을 4 비트 그룹들로 분할
   B. 각 그룹을 한 개의 기호로 표현

예) 8비트: 10100011 → 1010 0011 → A3

- 16진법은 4비트 패턴에 대하여 1개의 기호를 사용
  예) 1000101011010001 : 8AD1
- 16진법을 10진법과 구분하기 위하여 앞에 0x를 표시함
  예) 1001 (10진법), 0x1001 (16진법)

참고) 10진법, 16진법, 8진법, 2진법

2.5 16진 인코딩 체계

비트패턴	16진법 표현	비트패턴	16진법 표현
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

2.6 16진법

1. 16진수 연산

- $0x09 + 0x01 = 0x0A$
- $0x0F + 0x01 = 0x10$

2. 16진수의 10진수 변환

- $0xFF = F(15) \times 16 + 15 = 255$
- $0x11 = 1(1) \times 16 + 1 = 17$

2.7 8진법

1. 8개의 문자를 이용하여 수를 표현
2. 잘 사용되지는 않음

- 8진법은 3비트 패턴에 대하여 1개의 기호를 사용
  예) 100010101101000 : 42550
- 8진법을 10진법과 구분하기 위하여 앞에 0을 표시함
  예) 1001 (10진법), 01001 (8진법)

비트패턴	8진법 표현
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

2.8 진법 사이의 관계

2진법 표현	8진법 표현	10진법 표현	16진법 표현
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
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

3.1 2진 체계

1. 전통적인 10진 체계는 10의 멱승에 기초하고 있다.
2. 2진 체계는 2의 멱승에 기초하고 있다.

예) 10진수 체계와 2진수 체계

10진수 375 – 3x100 + 7x10 + 5x1

2진수 1011 – 1x8 + 0x4 + 1x2 + 1x1

거듭제곱(Exponentiation) -
밑(Base, 기저)이 **a** , 지수(Exponent)가 **n** 인 거듭제곱을 **a**의 **n**제곱이라고 하고, 그 기호는 a^n 이다.

3.2 정수의 저장 (양수 및 음수)

1. 인간이 0을 포함한 양의 정수만 사용하면,
   컴퓨터도 2진법을 사용하여 표현하기 매우 편리함

2. 인간은 필연적으로 음수를 활용하고 있고,
   컴퓨터에서도 음수를 포함한 정수를 표현

3. **컴퓨터에서 양수, 음수 등 표현은 어떻게?**

3.3 정수의 저장 (양수 및 음수)

1. 아이디어? 제일 앞의 한 비트를 부호 비트로 활용

0011 -> 3

1011 -> -3

문제점은? 연산의 어려움

\+ = 1110 -> -6 ?

비트패턴	표현값	비트패턴	표현값
0111	7	1111	−7
0110	6	1110	−6
0101	5	1101	−5
0100	4	1100	−4
0011	3	1011	−3
0010	2	1010	−2
0001	1	1001	−1
0000	0	1000	−0

실질적인 활용 불가

3.3 정수의 저장 (양수 및 음수)

2. 2의 보수(Two's complement) 표기법:
가장 널리 사용되는 정수 표현 체계

3. 초과(Excess) 표기법:
또 다른 정수 표현 체계로 실수 표현에 사용

→ **두 표기법**(2, 3의 표기) **모두에서**
오버플로우(Overflow) 문제가 발생할 수 있다.

3.4 2의 보수 표기 체계

1. 4비트로 정수를 표현하는 경우 2의 보수 표기 예

비트패턴	표현값
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

2. 3 + -3 = 0

   0011 + 1101 = (1)0000

   = 0000

   (1은 자리초과 / 버림)

3.5 2의 보수 덧셈

1. 2의 보수를 사용할 경우 뺄셈 연산을 덧셈으로 표현 가능
   전자 회로의 단순화가 가능함

10진법 연산	3 + 2 5	−3 + −2 −5	7 + −5 2
2의 보수 연산	0011 + 0010 0101	1101 + 1110 1011	0111 + 1011 0010

3.6 2의 보수 표기법을 통한 정수의 표현 범위

1. 2진수의 표현 범위

A. 2의 보수를 사용한 3비트 이진수 표현의 예

$+3 = (011)_2$

$+2 = (010)_2$

$+1 = (001)_2$

$+0 = (000)_2$

$-1 = (111)_2$

$-2 = (110)_2$

$-3 = (101)_2$

$-4 = (100)_2$

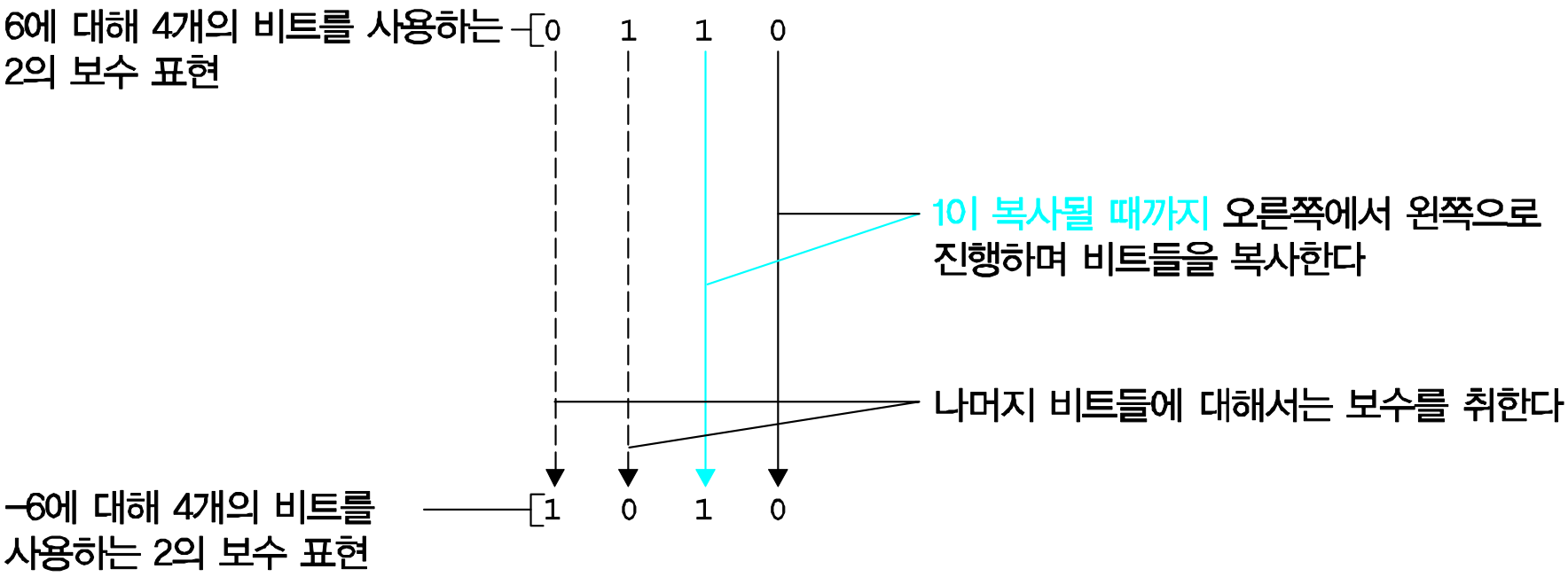
- 표현할 수 있는 수의 **범위는 -4 ~ 3**이 된다.
  이것은 -2^3-1 ~ 2^3-1-1로 표현된다.

B. **n비트** 데이터 경우로 일반화할 경우 수의 범위

❖ $-2^{n-1} \leq N \leq 2^{n-1}-1$

3.7 2의 보수 계산 방법

1. 4개 비트를 사용해 -6을 2의 보수 표기법으로 인코딩



2. 모든 비트를 1 ↔ 0 으로 변환 후에 +1 을 수행

0110 1001

+0001

1010

3.8 오버플로우 및 언더플로우

1. 컴퓨터는 인간이 사용하는 모든 숫자 표현할 수 있을까?
2. 4비트를 사용하는 경우를 가정
   양의 정수로만 사용하면 0에서 15까지 …
   2의 보수로 사용하면 -8에 서 7까지 표현이 가능 …

3. **오버플로우(Overflow)**
   숫자값이 표현할 수 있는 범위의 최대값을 넘어가는 경우
4. **언더플로우(Underflow)**
   숫자값이 표현할 수 있는 범위의 최소값을 넘어가는 경우