


$$a = 2$$

$$b = \sim a$$

의 결과가 -3인 이유

비트 연산자

정수나 글자 등을 2진수로 변환한 후 각 자리의 비트끼리 연산하는 연산자.

Python 

코드

```
a=2
```

```
b=~a
```

```
print(f'{a}의 bit not : {b}')
```

2의 bit not : -3



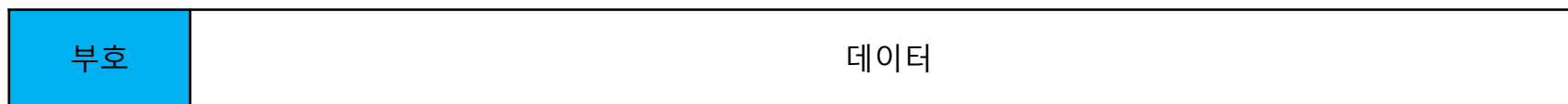
이유: 숫자에 bit not(~) 연산만 하면 순수 비트 연산을 하는게 아니라 해당 숫자의 음수값을 계산한 것을 보여준다.

컴퓨터가 음수를 표현하는 방법1

■ 부호와 절대치

- 부호 비트만 변경한다.
- 전체 비트 중 최상위 비트는 그 숫자의 부호(sign)를 나타낸다.

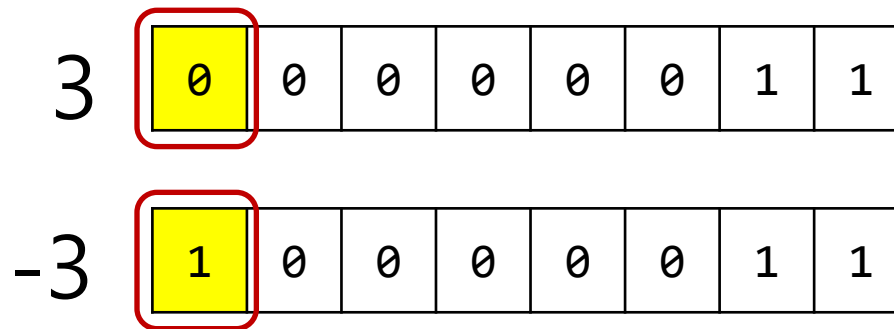
1비트



양수: 0

음수: 1

예를 들어 3과 -3을 8비트로 2진화하면 아래와 같다



■ 1의 보수

- 비트를 반전시킨다. 즉, $1 \rightarrow 0$, $0 \rightarrow 1$

예를 들어 3과 -3을 8비트로 2진화하면 아래와 같다

3	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

-3	1	1	1	1	1	0	0
----	---	---	---	---	---	---	---

■ 2의 보수

- ① 1의 보수로 만든다. 즉, 1->0, 0->1
- ② 1의 보수로 만든 값에 1을 더한다.

예를 들어 3을 8비트로 2진화하면 아래와 같다.

3	0	0	0	0	0	1	1
	↓	↓	↓	↓	↓	↓	↓
	1	1	1	1	1	0	0

① 1의 보수로 만든다.

② 1을 더한다.

							+	1
-3	1	1	1	1	1	1	0	1



컴퓨터는 음수를 나타내기 위해 2의 보수를 사용한다.

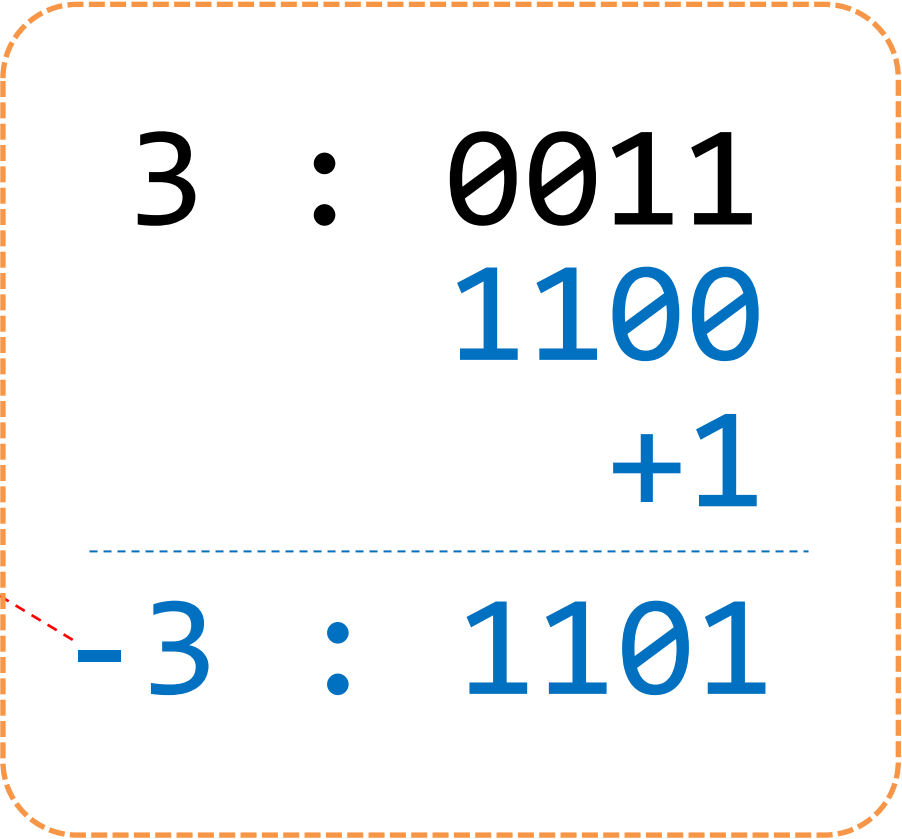
-9를 2의 보수를 사용해서 2진수로 표시하기

-9

$$\begin{array}{r} 9 : 1001 \\ 0110 \\ +1 \\ \hline -9 : 0111 \end{array}$$

5-3을 컴퓨터가 계산하는 방법 음수를 2의 보수로 만들어 더한다.

$$\begin{array}{r} 5 : 0101 \\ + \\ -3 : 1101 \\ \hline 2 : 0010 \end{array}$$


$$\begin{array}{r} 3 : 0011 \\ 1100 \\ +1 \\ \hline -3 : 1101 \end{array}$$

비트 not 연산한 값이 $1001_{(2)}$ 이라면 이 값은 10진수로 얼마? Python

not 연산을 한
값이 '1001' 이라면
10진수로
얼마인가?

역으로 연산하면 됨.
1을 뺀 후 1의 보수로 만든다.

1001
-1

1000

0111

0111은 7
따라서 2진수 '1001' 은 -7

비트 not 연산한 값이 1010 이라면 이 값은 10진수로 얼마? Python

not 연산을 한
값이 '1010' 이라면
10진수로
얼마인가?

역으로 연산하면 됨.
1을 뺀 후 1의 보수로 만든다.

1010

-1

1001

0110

0110은 6

따라서 2진수 '1010' 은 -6