C언어 강의자료

문정욱

C언어 기초 다지기 2
데이터의 표현

컴퓨터의 자료 표현

- 이진수(binary)와 비트(bit)
  - 디지털(digital)
    - 데이터를 0과 1로 표현하여 전송
    - 8bits = 1byte
  - bit vs. byte
    - 비트(bit)는 데이터 전송 최소 단위
    - 바이트(byte)는 데이터 표현의 최소 단위
  - N bits의 표현 범위
    - 2^n개의 데이터 표현
  - SI prefix

SI Prefix	Value	SI Prefix	Value
K (Kilo)	$2^{10} \fallingdotseq 1{,}000^1$	P (Peta)	$2^{50} \fallingdotseq 1{,}000^5$
M (Mega)	$2^{20} \fallingdotseq 1{,}000^2$	E (Exa)	$2^{60} \fallingdotseq 1{,}000^6$
G (Giga)	$2^{30} \fallingdotseq 1{,}000^3$	Z (Zetta)	$2^{70} \fallingdotseq 1{,}000^7$
T (Tera)	$2^{40} \fallingdotseq 1{,}000^4$	Y (Yotta)	$2^{80} \fallingdotseq 1{,}000^8$

1 bit	2 bit	3 bit	4 bit
0	00	000	0000
1	01	001	0001
	10	010	0010
	11	011	0011
		100	0100
		101	0101
		110	0110
		111	0111
			1000
			1001
			1010
			1011
			1100
			1101
			1110
			1111

컴퓨터의 자료 표현

10진수(decimal number) vs. 2진수(binary number)

$$
\begin{aligned}
182_{10} &= 1\times 10^2 + 8\times 10^1 + 2\times 10^0 \\
&= 1\times 100 + 8\times 10 + 2\times 1 \\
&= 182
\end{aligned}
$$

$$
\begin{aligned}
1101_2 &= 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 \\
&= 1\times 8 + 1\times 4 + 0\times 2 + 1\times 1 \\
&= 13
\end{aligned}
$$

컴퓨터의 자료 표현

16진수(hexadecimal number)와 8진수(octal number)

$$\begin{aligned}1AF_{16} &= 1\times16^2 + A\times16^1 + F\times16^0 \\ &= 1\times256 + 10\times16 + 15\times1 \\ &= 431\end{aligned}$$

$$\begin{aligned}3171_8 &= 3\times8^3 + 1\times8^2 + 7\times8^1 + 1\times8^0 \\ &= 3\times512 + 1\times64 + 7\times8 + 1\times1 \\ &= 1657\end{aligned}$$

컴퓨터의 자료 표현

10진수(decimal number) vs. 2진수(binary number)

$$13_{10} = \boxed{1}\times 2^3 + \boxed{1}\times 2^2 + \boxed{0}\times 2^1 + \boxed{1}\times 2^0$$
$$= (1\times 2^3 + 1\times 2^2 + 0\times 2^1) + \boxed{1}\times 2^0$$
$$= (1\times 2^2 + 1\times 2^1 + 0\times 2^0)\times 2 + \boxed{1}\times 2^0$$
$$= 6\times 2 + \boxed{1}$$

$$6_{10} = \boxed{1}\times 2^2 + \boxed{1}\times 2^1 + \boxed{0}\times 2^0$$
$$= (1\times 2^2 + 1\times 2^1) + \boxed{0}\times 2^0$$
$$= (1\times 2^1 + 1\times 2^0)\times 2 + \boxed{0}\times 2^0$$
$$= 3\times 2 + \boxed{0}$$

$$3_{10} = \boxed{1}\times 2^1 + \boxed{1}\times 2^0$$
$$= (1\times 2^1) + \boxed{1}\times 2^0$$
$$= (1\times 2^0)\times 2 + \boxed{1}\times 2^0$$
$$= 1\times 2 + \boxed{1}$$

$$1_{10} = \boxed{1}\times 2^0$$
$$= (\quad 0 \quad) + \boxed{1}\times 2^0$$
$$= (\quad 0 \quad)\times 2 + \boxed{1}\times 2^0$$
$$= 0\times 2 + \boxed{1}$$

컴퓨터의 자료 표현

10진수(decimal number) vs. 2진수(binary number)

2) 13 … 1

2) 6 … 0

2) 3 … 1

2) 1 … 1

0

$$13_{10} = 1101_2$$

컴퓨터의 자료 표현

10진수(decimal number) vs. 2진수(binary number)

$$
\begin{aligned}
13_{10} &= 1\times2^3+1\times2^2+0\times2^1+1\times2^0\\
&= (1\times2^3+1\times2^2+0\times2^1)+1\times2^0\\
&= (1\times2^2+1\times2^1+0\times2^0)\times2+1\times2^0\\
&= 6\times2+1
\end{aligned}
$$

$$
\begin{aligned}
6_{10} &= 1\times2^2+1\times2^1+0\times2^0\\
&= (1\times2^2+1\times2^1)+0\times2^0\\
&= (1\times2^1+1\times2^0)\times2+0\times2^0\\
&= 3\times2+0
\end{aligned}
$$

$$
\begin{array}{r|l l l}
2 & 13 & \cdots & 1\\ \hline
2 & 6 & \cdots & 0\\ \hline
2 & 3 & \cdots & 1\\ \hline
2 & 1 & \cdots & 1\\ \hline
 & 0 & &
\end{array}
$$

$$13_{10} = 1101_2$$

$$
\begin{aligned}
3_{10} &= 1\times2^1+1\times2^0\\
&= (1\times2^1)+1\times2^0\\
&= (1\times2^0)\times2+1\times2^0\\
&= 1\times2+1
\end{aligned}
$$

$$
\begin{aligned}
1_{10} &= 1\times2^0\\
&= (\ 0\ )+1\times2^0\\
&= (\ 0\ )\times2+1\times2^0\\
&= 0\times2+1
\end{aligned}
$$

컴퓨터의 자료 표현

16진수(hexadecimal number)

001010101100

0010 1010 1100

2 A C

$2AC_{(16)}$

10진수	2진수	16진수
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

컴퓨터의 자료 표현

8진수(octal number)

001010101100

001 010 101 100

1 2 5 4

$1254_{(8)}$

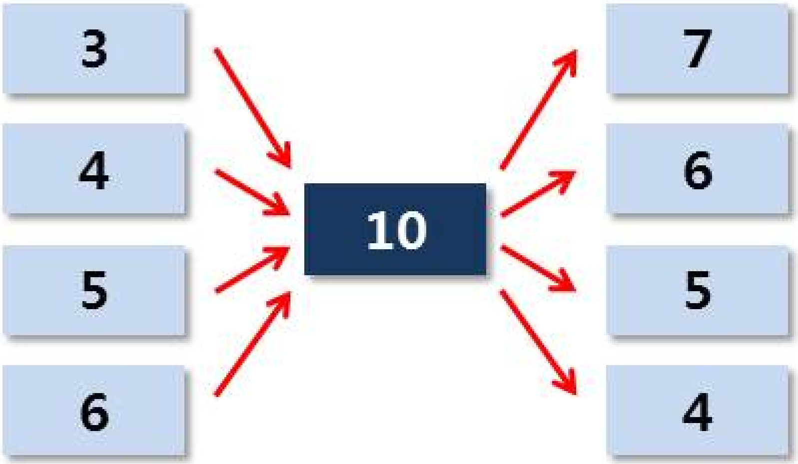
10진수	2진수	8진수
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

보수(complement)의 개념

3에 대한 10의 보수는 7이다.

3	→	10	→	7

보수(complement)의 개념



보수(complement)의 개념

13에 대한 10의 보수는 87이다.

100

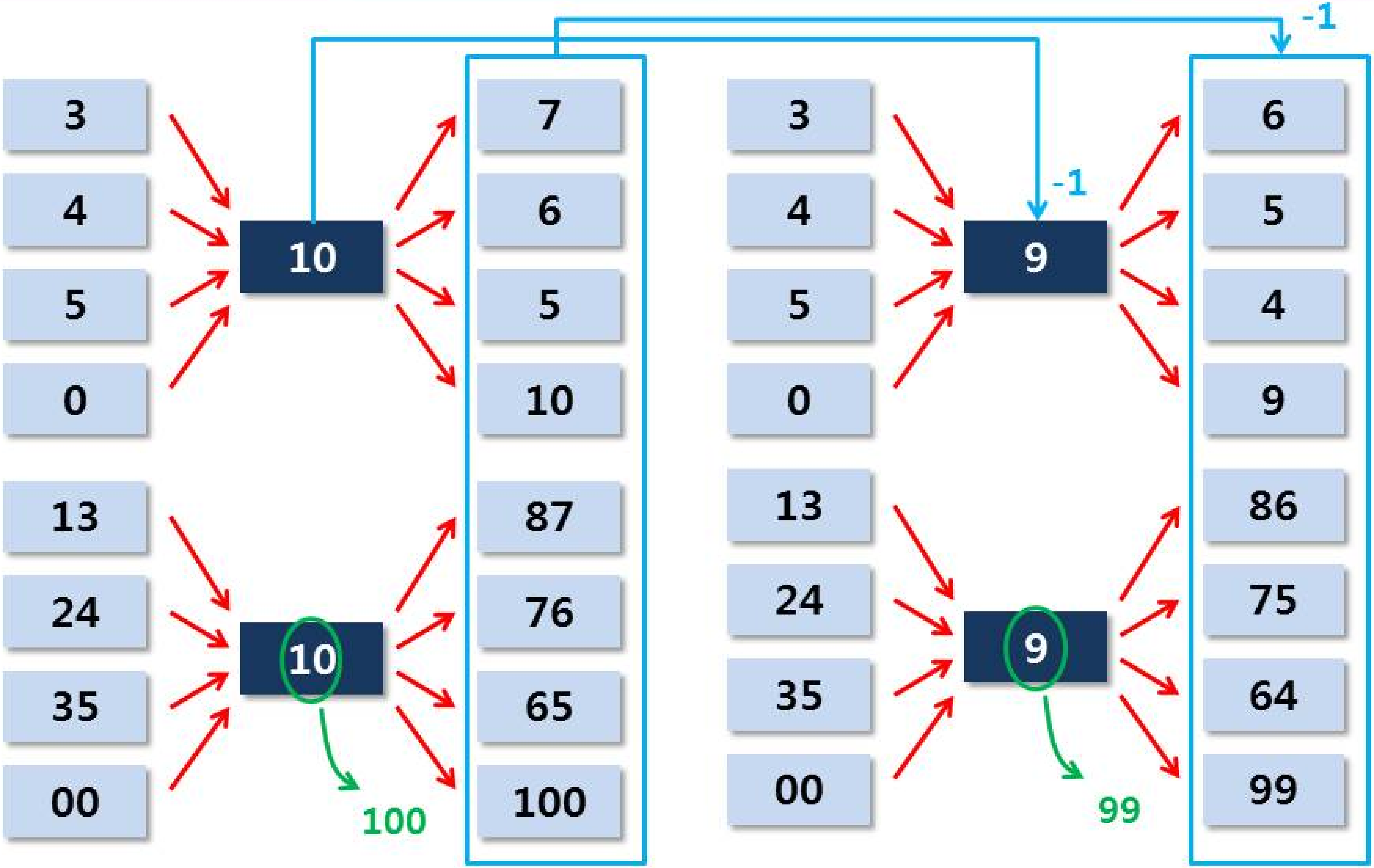
613에 대한 10의 보수는 387이다.

1000

2613에 대한 10의 보수는 7387이다.

10000

보수(complement)의 개념



보수(complement)의 개념

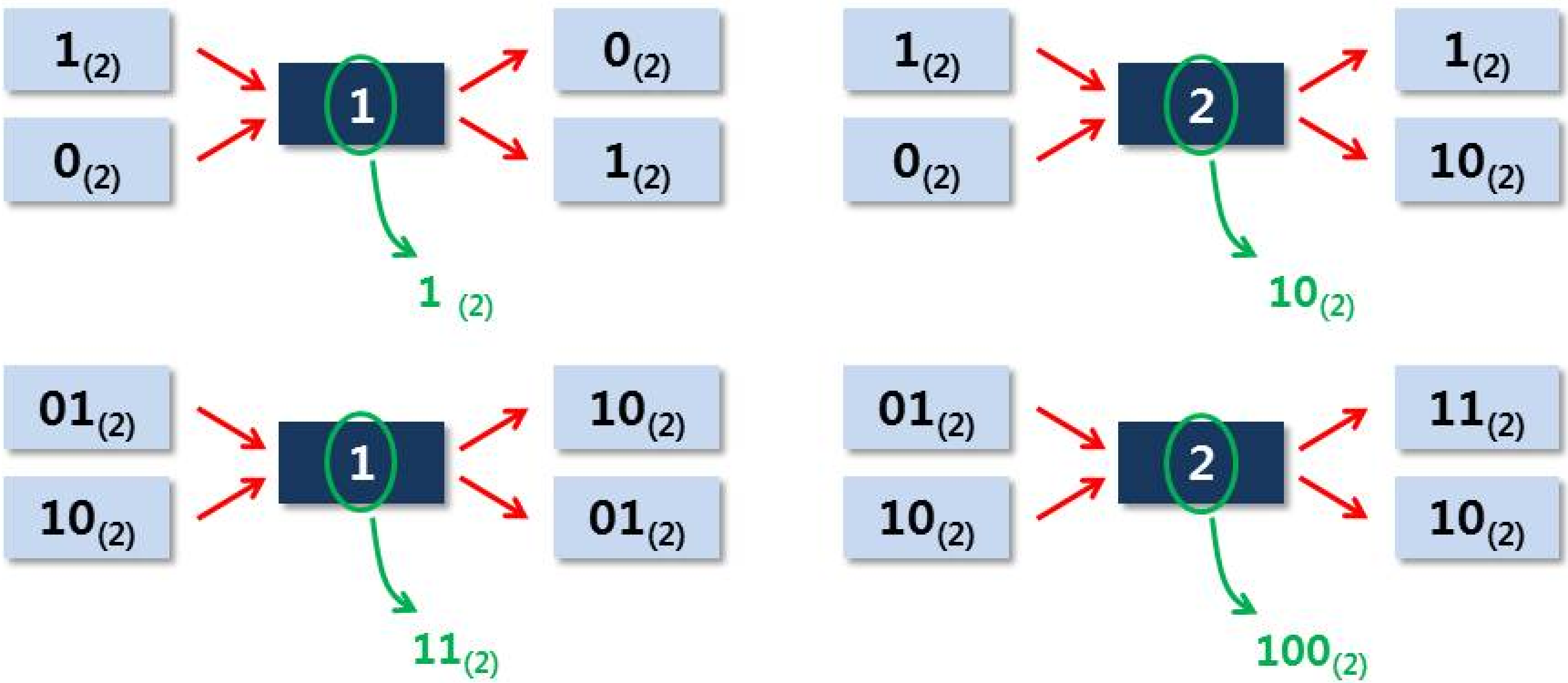
$1_{(2)}$에 대한 1의 보수는 $0_{(2)}$이다.

$1_{(2)} \rightarrow$ 1 $\rightarrow 0_{(2)}$

$1_{(2)}$에 대한 2의 보수는 $1_{(2)}$이다.

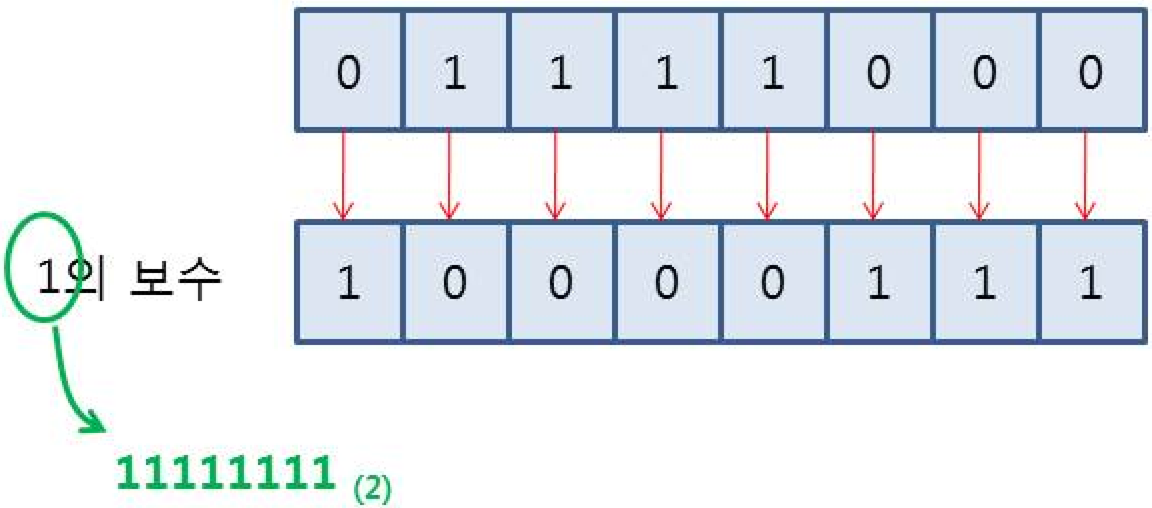
$1_{(2)} \rightarrow$ 2 $\rightarrow 1_{(2)}$

보수(complement)의 개념



이진수에서 1의 보수 계산

1의 보수 계산법



이진수에서 2의 보수 계산

2의 보수 계산법

	0	1	1	1	1	0	0	0
1의 보수	1	0	0	0	0	1	1	1
								+1
2의 보수	1	0	0	0	1	0	0	0

$100000000_{(2)}$

정수 표현

양의 정수 표현

2)	120	…	0
2)	60	…	0
2)	30	…	0
2)	15	…	1
2)	7	…	1
2)	3	…	1
2)	1	…	1
	0		

8bit Register

0	1	1	1	1	0	0	0

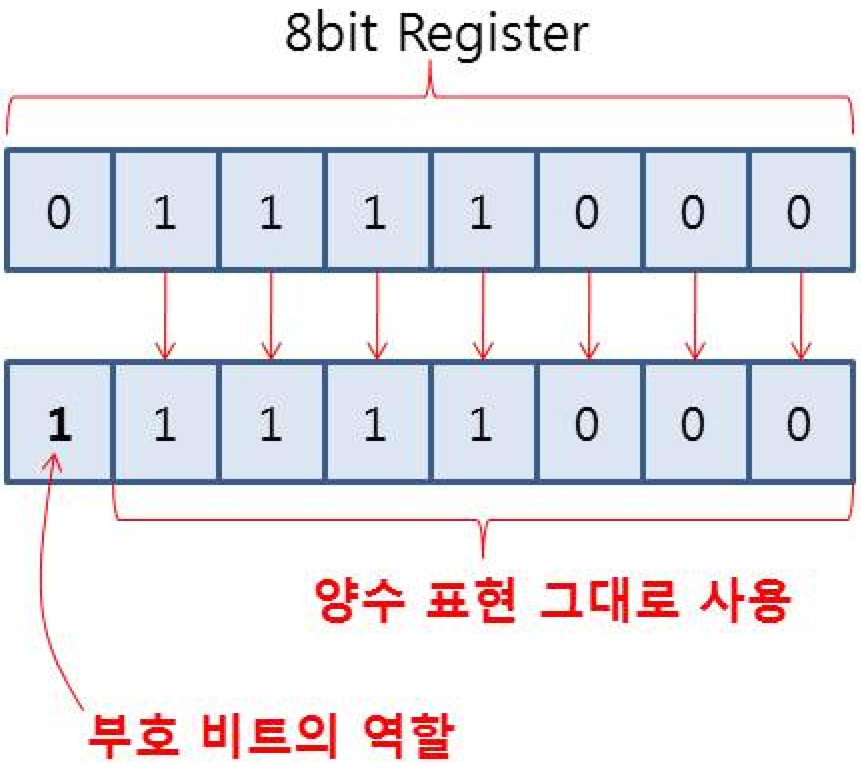
$$120_{10} = \boxed{1111000}_2$$

정수 표현

음의 정수 표현: 부호와 크기(sign and magnitude)

$$120_{10} = 1111000_2$$

부호 비트	값의 범위
0	0 ~ $(2^{n-1}-1)$
1	$-(2^{n-1}-1)$ ~ -0

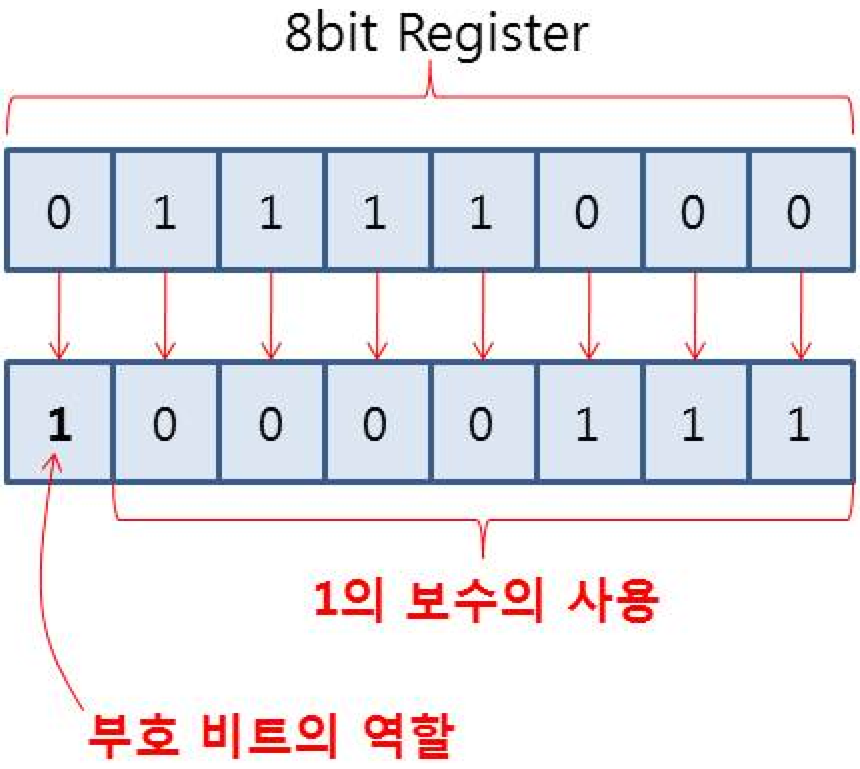


정수 표현

음의 정수 표현: 1의 보수(1's complement)

$$120_{10} = 1111000_2$$

부호 비트	값의 범위
0	0 ~ $(2^{n-1}-1)$
1	$-(2^{n-1}-1)$ ~ -0

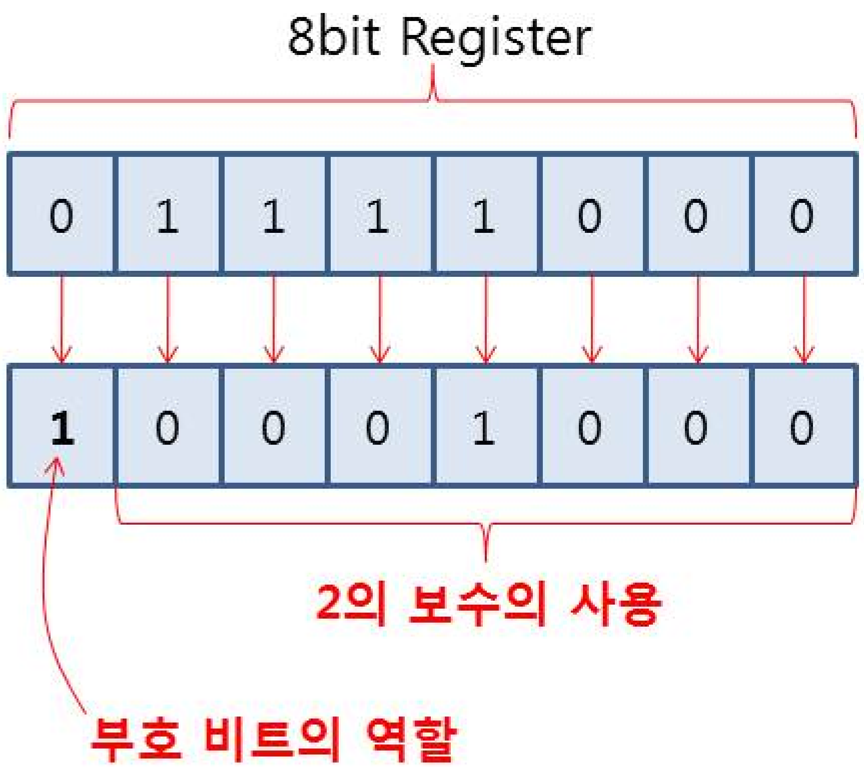


정수 표현

음의 정수 표현: 2의 보수(2's complement)

$$120_{10} = 1111000_{2}$$

부호 비트	값의 범위
0	0 ~ $(2^{n-1}-1)$
1	$-(2^{n-1})$ ~ -1



정수 표현

음의 정수 표현: 비균형(exceed n)

이진 표현	양수	Exceed 127
0000 0000	0	-127
0000 0001	1	-126
0000 0010	2	-125
...	...	...
0111 1111	**127**	**0**
1000 0000	128	1
...	...	...
1111 1111	255	128

정수 표현

음의 정수 표현: -2 기수법(base -2)

$$
\begin{aligned}
1101_2 &= 1\times(-2)^3 + 1\times(-2)^2 + 0\times(-2)^1 + 1\times(-2)^0 \\
&= -1\times 8 + 1\times 4 - 0\times 2 + 1\times 1 \\
&= -3
\end{aligned}
$$

정수 표현

정수 표현 방법 비교

Binary	Unsigned	Sign and Mag.	1's comp.	2's comp.	Exceed 127
0000 0000	0	0	0	0	-127
0000 0001	1	1	1	1	-126
0000 0010	2	2	2	2	-125
…	…	…	…	…	…
0111 1101	125	125	125	125	-2
0111 1110	126	126	126	126	-1
0111 1111	127	127	127	127	0
1000 0000	128	-0	-127	-128	1
1000 0001	129	-1	-126	-127	2
1000 0010	130	-2	-125	-126	3
…	…	…	…	…	…
1111 1101	253	-125	-2	-3	126
1111 1110	254	-126	-1	-2	127
1111 1111	255	-127	-0	-1	128

정수 표현

네 가지 정수 표현 방법 비교

표현 방법	값의 범위	0(zero) 표현	표현 가능 개수
부호 비트	$-2^{n-1}+1$ ~ $2^{n-1}-1$	+0, -0	2^n-1
1의 보수	$-2^{n-1}+1$ ~ $2^{n-1}-1$	+0, -0	2^n-1
2의 보수	-2^{n-1} ~ $2^{n-1}-1$	0	2^n
Exceed $2^{n-1}-1$	$-2^{n-1}+1$ ~ 2^{n-1}	0	2^n

정수 표현

- Exceed N를 사용한 음수 표현의 장점
  - 이진수의 값이 증가함에 따라 의미하는 값도 증가한다.
  - 그러므로 상대적인 비교가 쉽다.

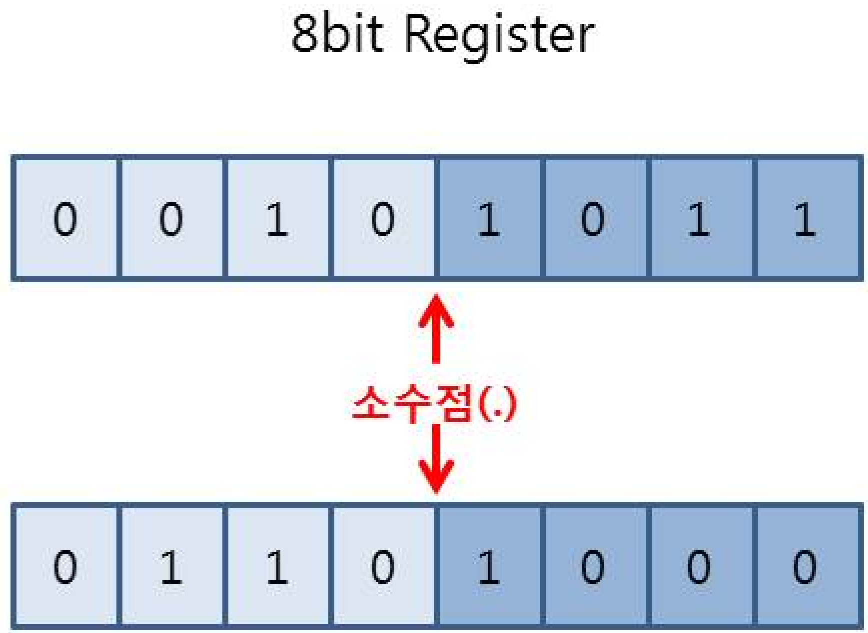
- 2의 보수를 사용한 음수 표현의 장점
  - 임의의 두수 A, B가 있고, B의 2의 보수를 B'이라고 할 때, A-B는 A+B'과 같다.
  - 그러므로 2의 보수만 사용하면 뺄셈 연산을 덧셈 연산으로 대체할 수 있다.

실수 표현

고정 소수점(fixed point)

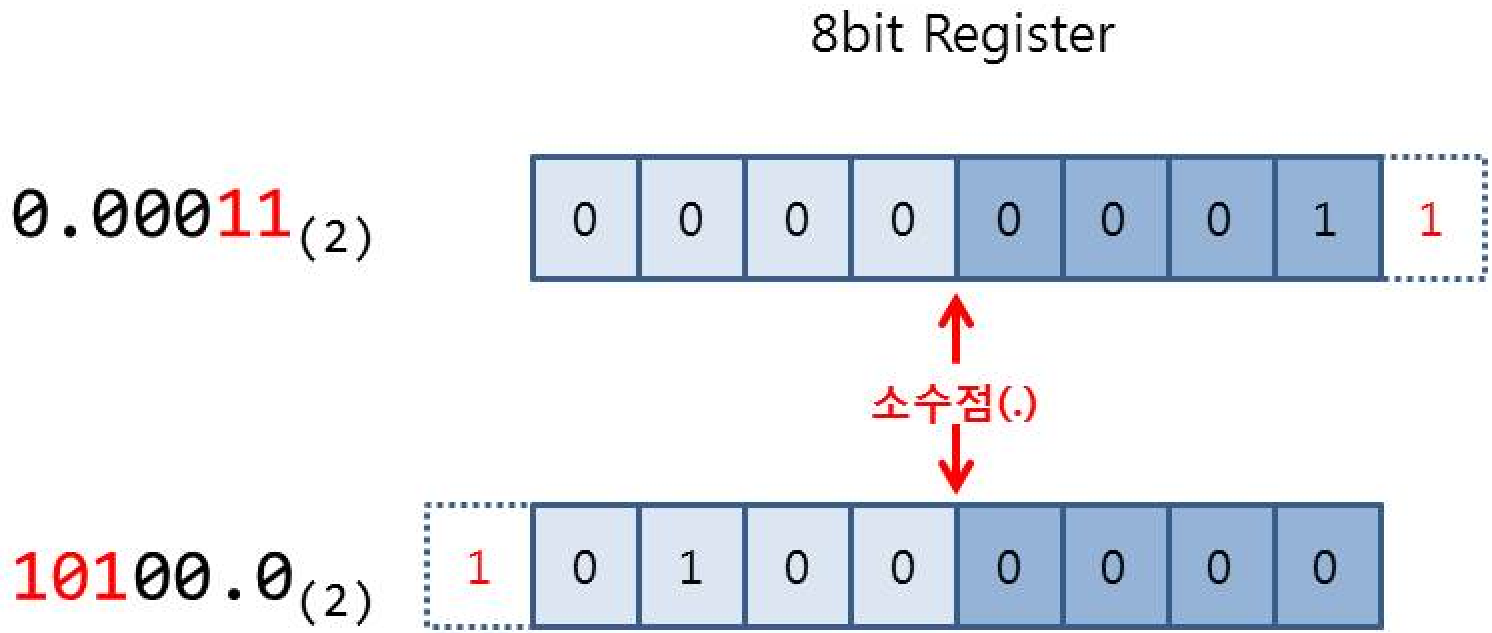
$10.1011_{(2)}$

$110.1_{(2)}$



실수 표현

고정 소수점(fixed point) – 문제점

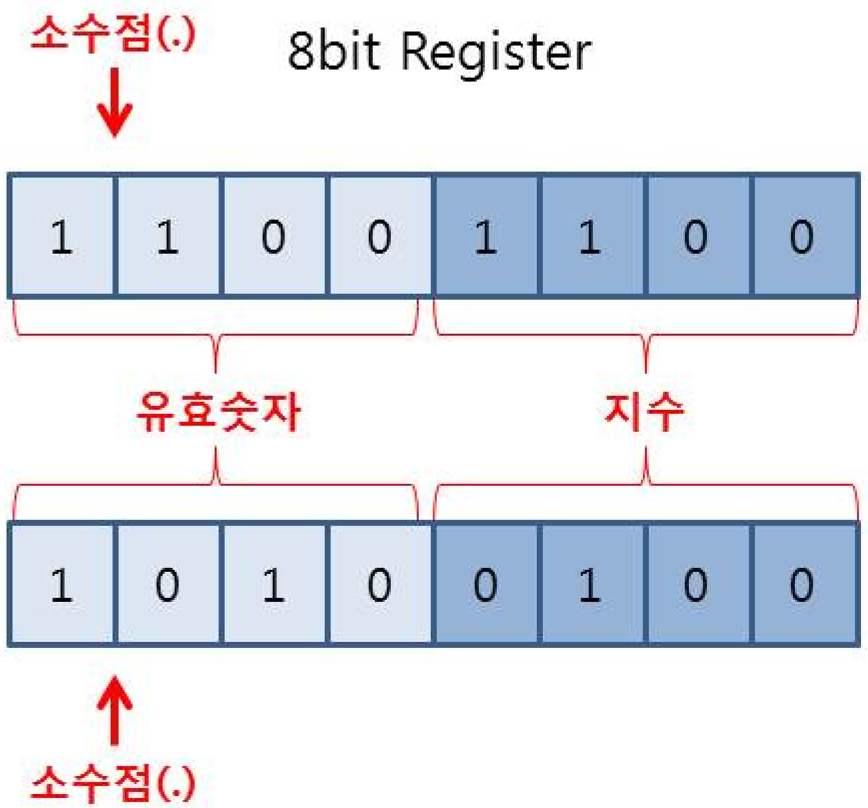


실수 표현

부동 소수점(floating point)

$0.00011_{(2)}$
$= 1.1_{(2)} \times 2^{-4}$
$= 1.1_{(2)}, -4$

$10100.0_{(2)}$
$= 1.01_{(2)} \times 2^{4}$
$= 1.01_{(2)}, +4$



실수 표현

부동 소수점(floating point) – 유효자리 확대

$0.00011_{(2)}$

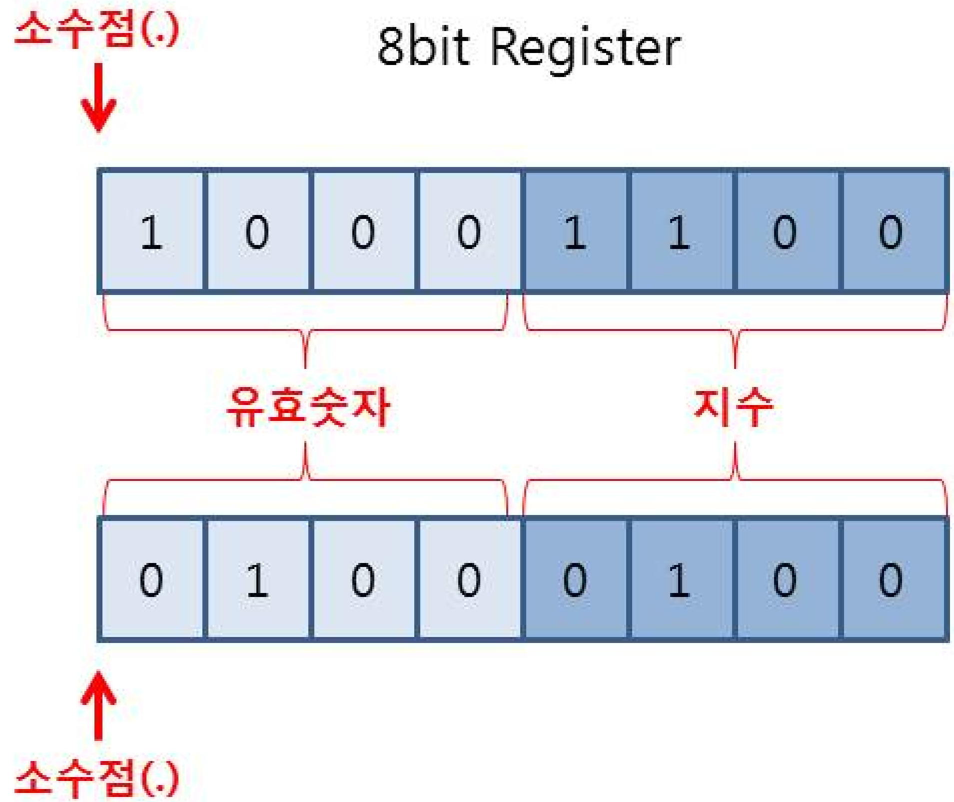
$= 1.1_{(2)} \times 2^{-4}$

$= \cancel{1}.1_{(2)}, -4$

$10100.0_{(2)}$

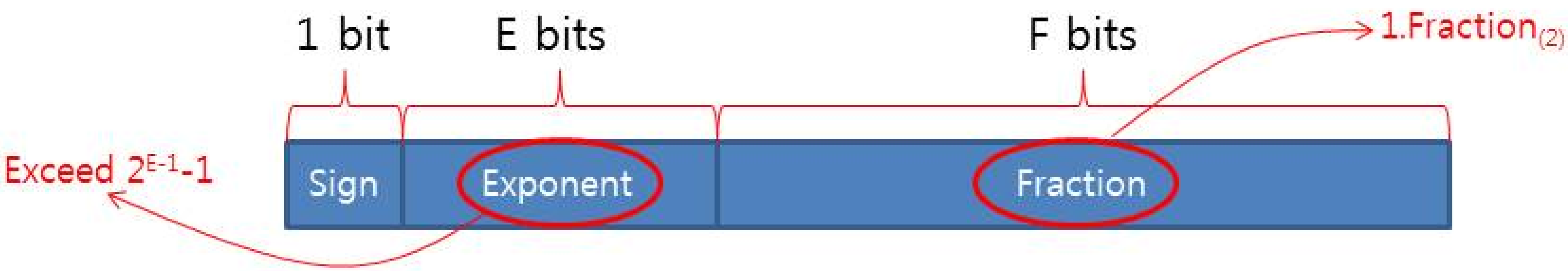
$= 1.01_{(2)} \times 2^{4}$

$= \cancel{1}.01_{(2)}, +4$



실수 표현

Floating Point (IEEE 754-1985)

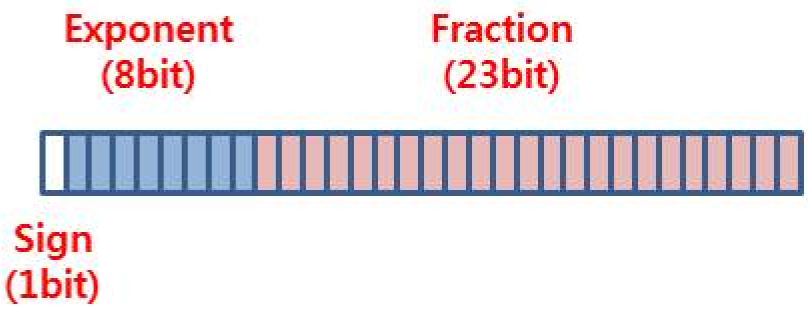


$$F = (-1)^{Sign} \times (1.Fraction_{(2)}) \times 2^{Exponent-Bias}, \quad (Bias = 2^{E-1} - 1)$$

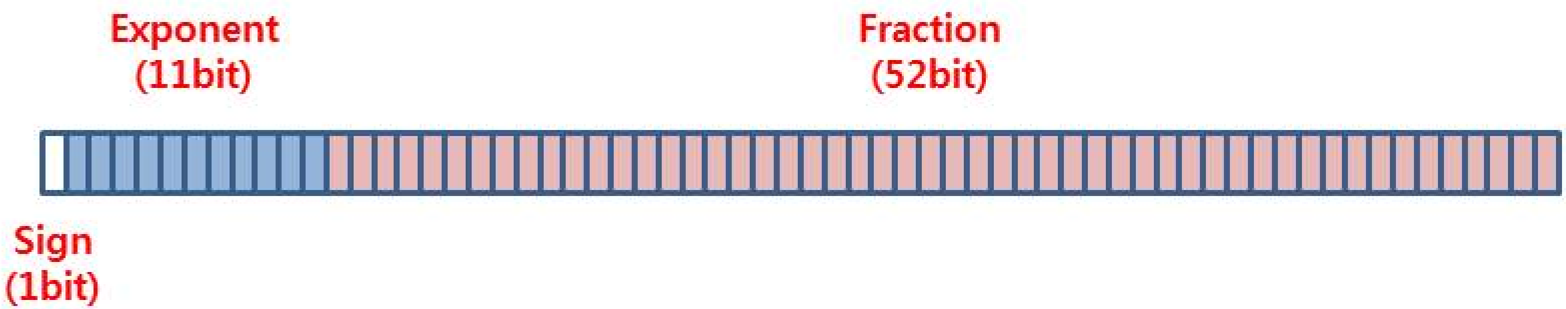
실수 표현

Floating Point (IEEE 754-1985)

- Single Precision: **float** in C language



- Double Precision: **double** in C language



실수 표현

Floating Point (IEEE 754-1985) – single precision

$$F = (-1)^{Sign} \times (1.Fraction_{(2)}) \times 2^{Exponent-127}$$