



새내기를 위한 최신 컴퓨터 개론

# 컴퓨터 사이언스



개정판

## CHAPTER 02. 정보의 표현

정보 체계\_컴퓨터 내부의 정보 표현과 정보  
처리

# 목차

01 수의 체계

02 진법 변환

03 정보의 표현

04 문자 표현

05 정수 표현

06 실수 표현

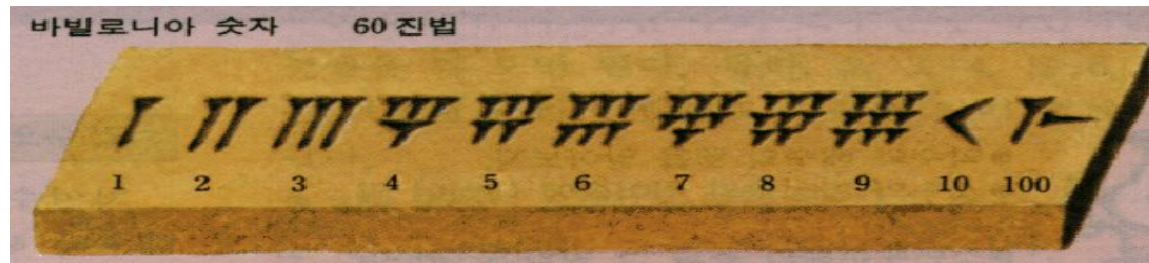
# 학습목표

- 컴퓨터에서 사용하는 수 체계와 종류를 알아본다.
- 진수 변환 방법을 알아본다.
- 컴퓨터의 정보 표현 방법을 알아본다.
- 컴퓨터에서 문자, 정수, 실수의 표현 방법을 알아본다.

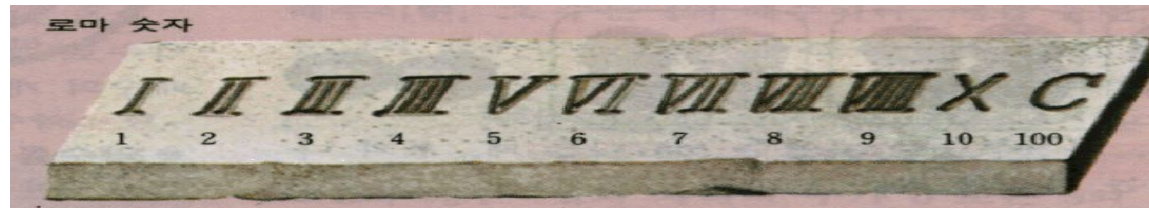
# 1.1 진수의 종류

## ◆ 고대 수 사용

바빌로니아 숫자



로마 숫자



마야족 숫자



## 1.1 진수의 종류

- ◆ 진법 : 임의의 수를 숫자로 표현하는 방법
- ◆ 디지털 컴퓨터는 두 개의 전기 신호(0 또는 1)를 이용해 정보를 표현



2진수 표현

# 1.1 진수의 종류

표 2-1 각 진수의 수 표현

진수	10진수	2진수	8진수	16진수
사용 숫자	0	0	0	0
	1	1	1	1
	2	10	2	2
	3	11	3	3
	4	100	4	4
	5	101	5	5
	6	110	6	6
	7	111	7	7
	8	1000	10	8
	9	1001	11	9
	10	1010	12	A
	11	1011	13	B
	12	1100	14	C
	13	1101	15	D
	14	1110	16	E
	15	1111	17	F
표현 예	5234 <sub>(10)</sub>	1011 <sub>(2)</sub>	146 <sub>(8)</sub>	5C31 <sub>(16)</sub>

## 1.2 자릿값

- ◆ 자릿값 : 진법에 따라 각 숫자는 별도의 자릿값을 가지며,  
해당 진수에 제곱수를 적용하여 자릿값을 계산

$10^2$	$10^1$	$10^0$		$10^{-1}$	$10^{-2}$	$10^{-3}$
1	2	3	.	4	5	6

(a) 10진수의 자릿값

$2^2$	$2^1$	$2^0$		$2^{-1}$	$2^{-2}$	$2^{-3}$
1	0	1	.	1	0	1

(b) 2진수의 자릿값

그림 2-1 진수별 자릿값

### ❖ 10진수 5234의 자릿값

- $5234_{(10)} = 5 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$

### ❖ 2진수 101.1의 자릿값

- 1의 자릿값 =  $2^2$ , 0의 자릿값 =  $2^1$ , 1의 자릿값 =  $2^0$ , 1의 자릿값 =  $2^{-1}$

### ❖ 8진수 146의 자릿값

- 1의 자릿값 =  $8^2$ , 4의 자릿값 =  $8^1$ , 6의 자릿값 =  $8^0$

### ❖ 16진수 5C3의 자릿값

- 5의 자릿값 =  $16^2$ , C의 자릿값 =  $16^1$ , 3의 자릿값 =  $16^0$



## 2. 진법 변환

◆ 진법 변환 : 주어진 수를 다른 진법으로 변환하는 것



2진수	1	1	0	1	0	1	.	1	0	0	1	1
8진수	6			5			.	4			6	
16진수	3		5				.	9				8

$$(110101.10011)_2 = (65.46)_8 = (35.98)_{16}$$

## 2.1 2진수, 8진수, 16진수 → 10진수

◆ 각 자리의 숫자에 자릿값을 곱한 후 모두 더한다.

- $1011_{(2)} = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11_{(10)}$
- $0.1_{(2)} = 1 \times 2^{-1} = 1 \times 1/2 = 0.5_{(10)}$
- $135_{(8)} = 1 \times 8^2 + 3 \times 8^1 + 5 \times 8^0 = 64 + 24 + 5 = 93_{(10)}$
- $20C_{(16)} = 2 \times 16^2 + 0 \times 16^1 + 12(C) \times 16^0 = 512 + 0 + 12 = 524_{(10)}$

## 2.2 10진수 → 2진수, 8진수, 16진수

### ❖ 정수 부분의 변환

- ① 10진수의 정수 부분을 2진수의 밑수 2로 나누어 몫과 나머지를 구한다.
- ② 몫이 더 이상 나누어지지 않을 때까지 밑수 2로 계속해서 나눈다.
- ③ 각 단계의 나머지를 역순으로 나열한다.

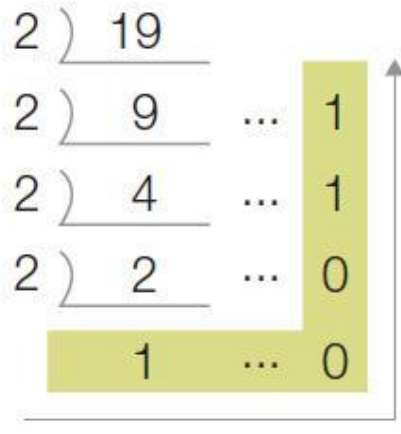

$$\begin{array}{r} 2 \overline{) 19} \\ 2 \overline{) 9} \quad \dots 1 \\ 2 \overline{) 4} \quad \dots 1 \\ 2 \overline{) 2} \quad \dots 0 \\ \hline 1 \quad \dots 0 \end{array}$$
$$19_{(10)} = 10011_{(2)}$$

그림 2-2 10진수 19를 2진수로 변환

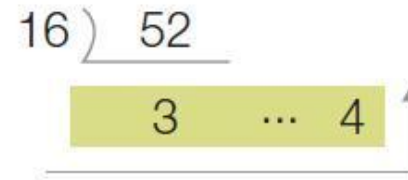
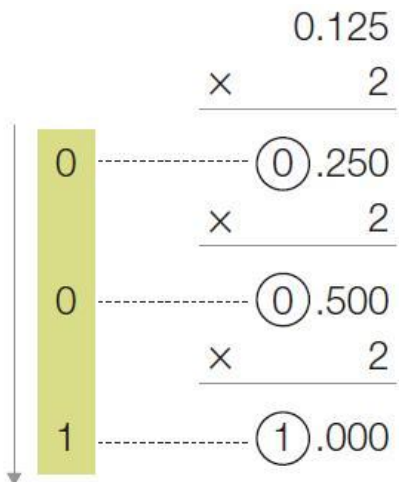

$$\begin{array}{r} 16 \overline{) 52} \\ \hline 3 \quad \dots 4 \end{array}$$
$$52_{(10)} = 34_{(16)}$$

그림 2-3 10진수 52를 16진수로 변환

## 2.2 10진수 → 2진수, 8진수, 16진수

### ❖ 소수 부분의 변환

- ① 10진수의 소수 부분에 2진수의 밑수 2를 곱한다.
- ② 곱셈 결과로 소수 부분이 0이 될 때까지 밑수 2를 계속 곱한다.
- ③ 각 단계에서 발생하는 정수 부분(자리올림)을 순서대로 나열한다.



$$0.125_{(10)} = 0.001_{(2)}$$

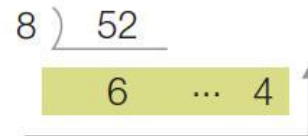
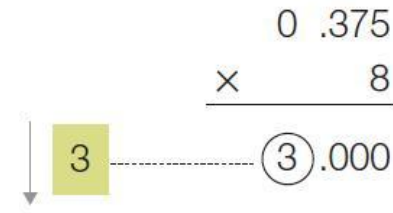


그림 2-5 10진수 52,375를 8진수로 변환



$$52.375_{(10)} = 64.3_{(8)}$$

### 3. 정보의 표현

- ◆ 디지털 컴퓨터는 문자나 숫자 등의 정보를 0과 1의 2진 체계로 부호화한 디지털 데이터로 처리

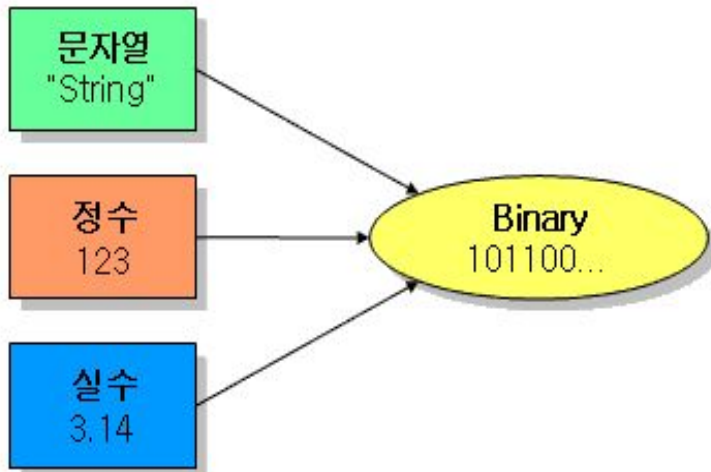


표 2-2 문자 A와 숫자 10의 부호화

정보	2진 체계 부호화
A	01000001
10	00001010

### 3. 정보의 표현

#### ◆ 비트

- Binary digit는 컴퓨터에서 정보를 나타내는 최소 단위
- 2진수 0 또는 1을 의미
- N비트로 표현할 수 있는 정보는  $2^N$ 개

#### ◆ 바이트

- 문자를 나타내는 최소 단위로 영문자나 숫자
- 특수문자는 1바이트로 표현
- 한글이나 한자는 2바이트로 표현

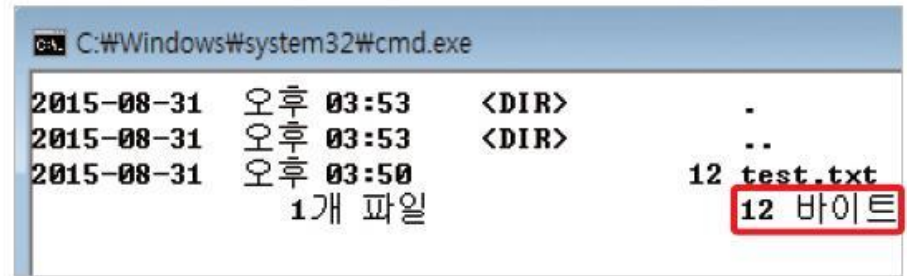
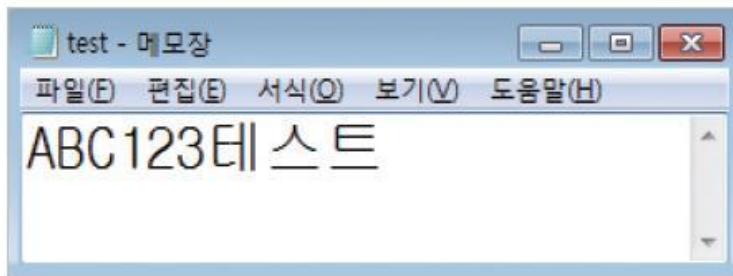


그림 2-6 test.txt 파일의 내용(왼쪽)과 크기(오른쪽)

#### ❖ 워드

- 명령어나 연산을 처리하는 기본 단위
- 기억장치에 한 번 접근하여 얻을 수 있는 데이터의 양

#### ❖ 기억 용량의 단위

기억 용량 단위	활용 예
KB(Kilo Byte)	20KB의 엑셀 파일
MB(Mega Byte)	4MB의 MP3 파일
GB(Giga Byte)	32GB의 USB 메모리
TB(Tera Byte)	2TB의 외장 하드디스크

- ◆ 미국표준협회(ANSI)가 데이터를 처리하거나 통신 시스템 간에 정보를 교환할 때 쓸 표준 코드로 제안한 것
- ◆ 표현할 수 있는 문자는  $128(2^7)$ 개



# 4.1 아스키 코드

## ◆ 아스키 코드표

- 0~31번과 127번 : 제어 문자
- 32~64번 : 특수문자와 숫자
- 65~96번 : 알파벳 대문자와 특수문자
- 97~126번 : 알파벳 소문자와 특수문자

표 2-4 아스키 코드표

10진수	2진수	ASCII	10진수	2진수	ASCII	10진수	2진수	ASCII	10진수	2진수	ASCII
0	0000000	NULL	32	0100000	SP	64	1000000	@	96	1100000	~
1	0000001	SOH	33	0100001	!	65	1000001	A	97	1100001	a
2	0000010	STX	34	0100010	"	66	1000010	B	98	1100010	b
3	0000011	ETX	35	0100011	#	67	1000011	C	99	1100011	c
4	0000100	EOT	36	0100100	\$	68	1000100	D	100	1100100	d
5	0000101	ENQ	37	0100101	%	69	1000101	E	101	1100101	e
6	0000110	ACK	38	0100110	&	70	1000110	F	102	1100110	f
7	0000111	BEL	39	0100111	'	71	1000111	G	103	1100111	g
8	0001000	BS	40	0101000	(	72	1001000	H	104	1101000	h
9	0001001	HT	41	0101001	)	73	1001001	I	105	1101001	i
10	0001010	LF	42	0101010	*	74	1001010	J	106	1101010	j
11	0001011	VT	43	0101011	+	75	1001011	K	107	1101011	k
12	0001100	FF	44	0101100	,	76	1001100	L	108	1101100	l
13	0001101	CR	45	0101101	-	77	1001101	M	109	1101101	m
14	0001110	SO	46	0101110	.	78	1001110	N	110	1101110	n
15	0001111	SI	47	0101111	/	79	1001111	O	111	1101111	o
16	0010000	DLE	48	0110000	0	80	1010000	P	112	1110000	p
17	0010001	DC1	49	0110001	1	81	1010001	Q	113	1110001	q
18	0010010	SC2	50	0110010	2	82	1010010	R	114	1110010	r
19	0010011	SC3	51	0110011	3	83	1010011	S	115	1110011	s
20	0010100	SC4	52	0110100	4	84	1010100	T	116	1110100	t
21	0010101	NAK	53	0110101	5	85	1010101	U	117	1110101	u
22	0010110	SYN	54	0110110	6	86	1010110	V	118	1110110	v
23	0010111	ETB	55	0110111	7	87	1010111	W	119	1110111	w
24	0011000	CAN	56	0111000	8	88	1011000	X	120	1111000	x
25	0011001	EM	57	0111001	9	89	1011001	Y	121	1111001	y
26	0011010	SUB	58	0111010	:	90	1011010	Z	122	1111010	z
27	0011011	ESC	59	0111011	;	91	1011011	[	123	1111011	{
28	0011100	FS	60	0111100	<	92	1011100	\	124	1111100	
29	0011101	GS	61	0111101	=	93	1011101	]	125	1111101	}
30	0011110	RS	62	0111110	>	94	1011110	^	126	1111110	~
31	0011111	US	63	0111111	?	95	1011111	_	127	1111111	DEL

- ◆ 전 세계의 언어를 일관된 방법으로 표현하고 다룰 수 있는 국제적인 문자 코드 규약
- ◆ 문자 하나를 16비트로 표현, 65,536( $2^{16}$ )개의 문자와 기호를 나타냄
- ◆ 인코딩 방식은 UTF-8, UTF-16, UTF-32(UTF 뒤의 숫자는 문자 인코딩에 사용되는 비트 수)
- ◆ 언어별 유니코드 차트 자료 : <http://www.unicode.org/charts>

## 4.4 유니코드

	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	ACA	ACB	ACC	ACD	ACE	ACF
0	가 AC00	감 AC10	감 AC20	갓 AC30	갈 AC40	각 AC50	갸 AC60	거 AC70	검 AC80	겐 AC90	갹 ACA0	결 ACB0	격 ACC0	겻 ACD0	고 ACE0	곰 ACF0
1	각 AC01	갑 AC11	갸 AC21	갹 AC31	갈 AC41	갸 AC51	갹 AC61	걱 AC71	겁 AC81	갸 AC91	겻 ACA1	결 ACB1	겻 ACC1	겻 ACD1	곡 ACE1	곱 ACF1
2	갸 AC02	갸 AC12	갸 AC22	갸 AC32	갸 AC42	갸 AC52	갸 AC62	겻 AC72	겻 AC82	갸 AC92	갸 ACA2	겻 ACB2	겻 ACC2	겻 ACD2	곡 ACE2	곶 ACF2

그림 2-11 한글 유니코드

표 2-6 훈민정음의 유니코드 표현

1101 0110 1100 1000	1011 1011 1111 1100	1100 1000 0001 0101	1100 0111 0100 1100
훈	민	정	음

- ◆ 두 수의 합이 진법의 밑수(N)가 되게 하는 수
- ◆ 음의 정수를 표현하기 위해 고안한 개념
- ◆ 컴퓨터 내부에서는 사칙연산을 할 때 덧셈을 담당하는 가산기를 이용하기 때문에 뺄셈은 덧셈 형식으로 변환하여 계산해야 함
  - $A-B$ 는  $B$ 의 보수( $-B$ )를 구한 후  $A+(-B)$ 로 계산

## 5.1 보수

### ◆ 1의 보수

2진수 1010의 1의 보수는  $0101_{(2)}$  →

$$\begin{array}{r} 1111 \\ - 1010 \\ \hline 0101 \end{array}$$

그림 2-12 1의 보수

### ◆ 2의 보수

2진수 1010의 2의 보수는  $0110_{(2)}$  →

$$\begin{array}{r} 1111 \\ - 1010 \\ \hline 0101 \\ + \quad 1 \\ \hline 0110 \end{array}$$

그림 2-13 2의 보수

◆ 두 수의 합이 2가 되면 자리올림이 발생

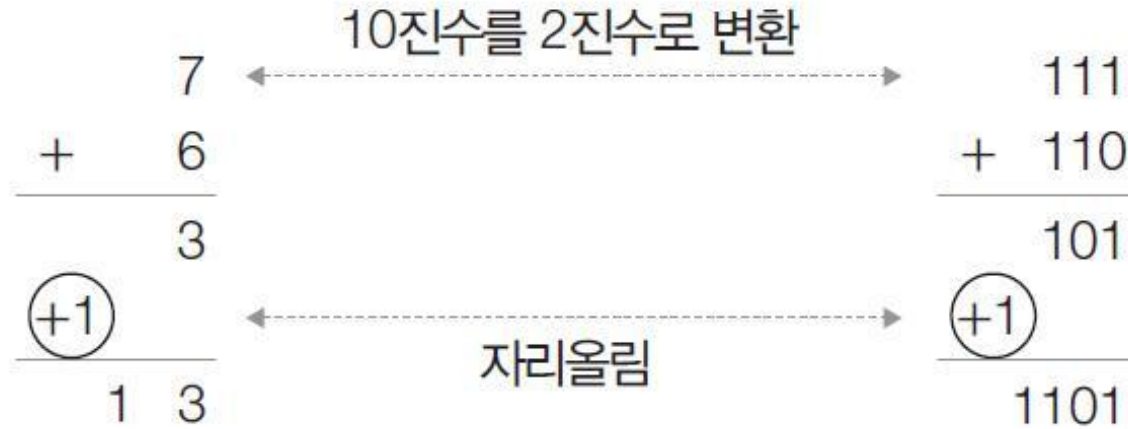


그림 2-14 자리올림이 발생한 덧셈

### ◆ 1의 보수 뺄셈

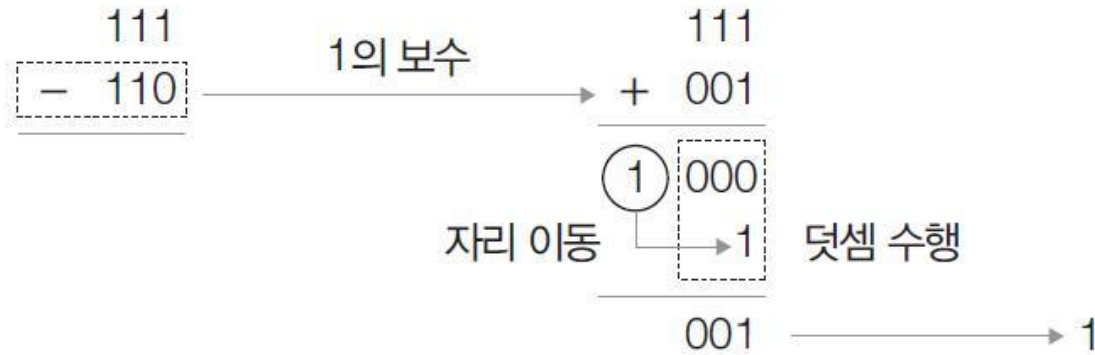


그림 2-15 자리올림이 생긴 1의 보수 뺄셈 : 7 - 6



그림 2-16 자리올림이 생기지 않는 1의 보수 뺄셈 : 4 - 6

### ◆ 2의 보수 뺄셈

$$\begin{array}{r}
 111 \\
 - 110 \\
 \hline
 \end{array}
 \xrightarrow{\text{2의 보수}}
 \begin{array}{r}
 111 \\
 + 010 \\
 \hline
 \end{array}$$

자리올림 제외 ① 001  $\longrightarrow$  1

그림 2-17 자리올림이 생긴 2의 보수 뺄셈 : 7 - 6

$$\begin{array}{r}
 100 \\
 - 110 \\
 \hline
 \end{array}
 \xrightarrow{\text{2의 보수}}
 \begin{array}{r}
 100 \\
 + 010 \\
 \hline
 \end{array}$$

$\begin{array}{r} 110 \end{array} \xrightarrow{\text{2의 보수}} 010 \xrightarrow{-\text{부호}} -2$

그림 2-18 자리올림이 생기지 않은 2의 보수 뺄셈 : 4 - 6



- ◆ 피승수에 승수의 각 수를 곱하여 부분 곱을 구함
- ◆ 각 부분 곱은 직전 단계의 부분 곱보다 왼쪽으로 한 비트만큼 시프트한 후 더함

	110011	(피승수)
×	110	(승수)
<hr/>		
	000000	(부분 곱)
	110011	(부분 곱)
+	110011	(부분 곱)
<hr/>		
	100110010	(결과)

그림 2-19 2진수 110011과 110의 곱셈

- ◆ 피제수에서 제수를 뺄 수 없을 때까지 뺄셈을 계속해서 횡수는 뭉이 되고 남은 것은 나머지가 됨

$$\begin{array}{r}
 \text{(제수)} \quad 110 \overline{) 100110} \quad \begin{array}{l} 110 \text{ (몫)} \\ \text{(피제수)} \end{array} \\
 \underline{-110} \\
 111 \quad \text{(부분 나머지)} \\
 \underline{-110} \\
 10 \quad \text{(나머지)}
 \end{array}$$

그림 2-20 2진수 100110과 110의 나눗셈

## 5.6 고정 소수점 표현

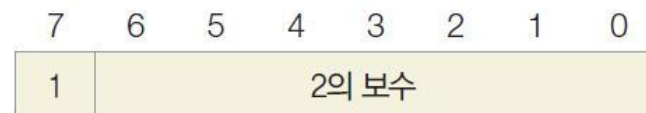
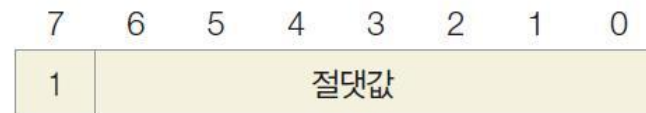
- ❖ 소수점이 고정된 위치에 있다는 뜻
- ❖ 정수 표현에 사용



그림 2-21 고정 소수점 표현



(a) 양의 정수 표현



(b) 음의 정수 표현

그림 2-22 양수와 음수의 고정 소수점 표현

## 5.6 고정 소수점 표현

표 2-7 정수 +13과 -13에 대한 8비트 표현

표현 방식	+13	-13
부호화 절댓값	0 0001101	1 0001101
1의 보수	0 0001101	1 1110010
2의 보수	0 0001101	1 1110011

표 2-8 3비트로 표현할 수 있는 수의 범위

10진수	부호화 절댓값	1의 보수	2의 보수
-4	-	-	100
-3	111	100	101
-2	110	101	110
-1	101	110	111
-0	100	111	-
+0	000	000	000
+1	001	001	001
+2	010	010	010
+3	011	011	011

## 6. 실수 표현

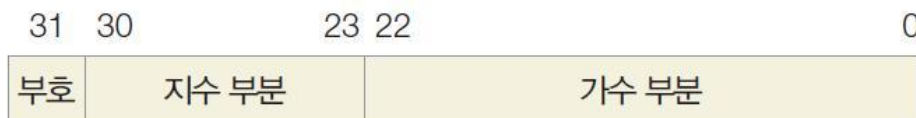
### ❖ 고정 소수점 방식

- 소수점이 항상 고정된 위치에 있다는 의미로, 정수 표현에 주로 사용

### ❖ 부동 소수점 방식

- 소수점의 위치가 변하기 때문에 실수 표현에 주로 사용
- 고정 소수점 방식보다 넓은 범위의 수를 표현

$m \times r^e$  (  $m$  : 가수  $r$  : 밑수,  $e$  : 지수 )



(a) 단일 정밀도 형식



(b) 이중 정밀도 형식

그림 2-23 부동 소수점 표현 형식