

## **Chapter 03. 디지털 정보의 표현**

# 목차

1. 컴퓨터의 단위
2. 진법 변환
3. 컴퓨터의 데이터 표현
4. 연산과 논리 게이트

# 학습목표

- 컴퓨터의 용량 단위와 속도 단위를 알아본다.
- 2진법, 16진법의 특징과 진법 변환 방법을 이해한다.
- 정수, 실수, 문자 등 데이터의 종류와 처리 방식을 이해한다.
- 연산자를 이해하고,  
논리 연산자를 사용하는 간단한 논리 게이트를 살펴본다.

**01**

컴퓨터의 단위

# 01. 컴퓨터의 단위

## 1. 용량 단위

- 2진법: 모든 수를 0과 1(숫자 2개)로 표현
- 10진법: 모든 수를 0부터 9까지(숫자 10개)로 표현
- 10진법을 사용하는 컴퓨터를 만들지 않는 이유

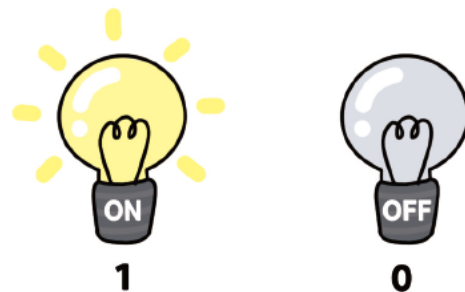


그림 3-1 진공관과 2진법



그림 3-2 2진 식당과 10진 식당

# 01. 컴퓨터의 단위

## 1. 용량 단위

### ■ 비트와 바이트

- **비트** bit : 컴퓨터에서 사용하는 데이터의 최소 단위
- **바이트** byte : 비트 8개를 묶은 단위



그림 3-3 비트와 바이트

# 01. 컴퓨터의 단위

## 1. 용량 단위

### ■ 워드

- 64비트 CPU에는 64비트 운영체제가 설치됨
- 워드 word : 컴퓨터가 한 번에 처리할 수 있는 데이터 크기의 단위



그림 3-4 32비트 CPU와 64비트 CPU의 워드

# 01. 컴퓨터의 단위

## 1. 용량 단위

표 3-1 컴퓨터의 용량 단위

용량 단위	설명
비트	데이터를 표현하는 최소 단위
바이트	8비트를 묶은 단위
워드	컴퓨터가 한 번에 처리할 수 있는 데이터 단위



# 01. 컴퓨터의 단위

## 1. 용량 단위

### ■ 큰 용량을 나타내는 단위

표 3-2 큰 용량을 나타내는 단위

용량 단위	표기	2진 크기	10진 크기	바이트 단위로 나타낸 크기	10진 단위
바이트	B	1	1	1B	일
킬로바이트	KB	$2^{10}$	$10^3$	1,000B	천
메가바이트	MB	$2^{20}$	$10^6$	1,000,000B	백만
기가바이트	GB	$2^{30}$	$10^9$	1,000,000,000B	십억
테라바이트	TB	$2^{40}$	$10^{12}$	1,000,000,000,000B	일조
페타바이트	PB	$2^{50}$	$10^{15}$	1,000,000,000,000,000B	천조

- 큰 용량 단위는 바로 한 단계 낮은 단위보다  $1,024(2^{10})$ 배 큼
- 1킬로바이트(1KB)는 정확히  $1,024$ 바이트( $2^{10}B$ )
- 그러나 사람은 2진수보다 10진수에 더 익숙하므로 보통 1KB를 약  $1,000B(10^3B)$ 로 씀

# 01. 컴퓨터의 단위

---

## 1. 용량 단위

### ■ 파일과 패킷

- 파일 file : 데이터가 모인 단위
- 패킷 packet : 네트워크가 다루는 일정 크기의 데이터

# 01. 컴퓨터의 단위

## 2. 속도 단위

### ■ 헤르츠

- 클록 clock : 컴퓨터에서 일정한 박자를 만들어내는 장치
- 클록 틱 clock tic
- 컴퓨터 내 모든 부품은 클록이 일정한 간격으로 틱을 만들면 거기에 맞추어 작업



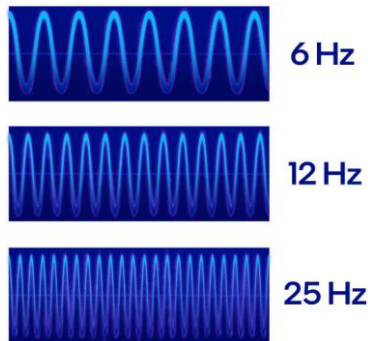
그림 3-5 틱

# 01. 컴퓨터의 단위

## 2. 속도 단위

### ■ 헤르츠

- 헤르츠 Hertz : CPU 성능의 단위, 1초 동안 클록 틱 발생 횟수
  - HZ로 표기
    - 1초 동안 클록 틱이 몇 번 발생했는지 나타냄(1번 : 1Hz, 1,000번 : 1KHz)
    - CPU 속도 3GHz → 1초 동안 작업이 약  $3 \times 10^9$ (30억)번
    - 메인메모리 속도 1.6GHz → 1초 동안  $1.6 \times 10^9$ 번 데이터를 저장
    - 주어진 시간 내에 이루어지는 더 많은 작업을 의미



\* 출처 : intel.co.kr

- 클럭속도가 빠르면 좋은 CPU?
  - 싱글코어가 주류를 이루던 시절에는 클럭속도가 CPU의 성능을 나타내는 척도
  - 단, 같은 제품군의 CPU일 때..
  - i7 900제품군에서는 클럭속도가 높은 CPU가 성능이 높음
    - i7 920 - 2.66 GHz, i7 930 - 2.8 GHz, i7 950 - 3.06 GHz ...
- 클럭 속도를 올릴수록 발열과 전력소비가 커짐
  - 제조사가 생각한 것이 바로 코어의 개수를 늘리는 것
  - CPU 내부에 코어 개수에 따라, 듀얼코어, 쿼드코어, 헥사코어 출시
- 코어가 많고 클럭이 높은 CPU를 구입?
  - 코어가 많으면서도 클럭이 높다면 고성능 CPU
  - 코어의 개수는 적어도 클럭이 높은 CPU
    - 인터넷이나 영화감상, 문서 작성 등 단순한 작업
  - 클럭속도가 좀 낮더라도 코어의 개수가 많은 CPU
    - 고사양의 게임이나 인코딩 작업 등
- 프로그래머는 랩탑, 모니터.. 램.. SSD... 정도? 머신러닝은 AWS..

# 01. 컴퓨터의 단위

## 2. 속도 단위

- 헤르츠의 활용
  - 모니터의 주사율
  - 가정용 전기 규격



그림 3-7 모니터의 성능과 전기 규격을 나타내는 헤르츠

# 01. 컴퓨터의 단위

## 2. 속도 단위

- **bps** bit per second

- 1초 동안 보내는 데이터의 양
- 10MB인 파일을 10Mbps 네트워크에서 전송하면 몇 초?

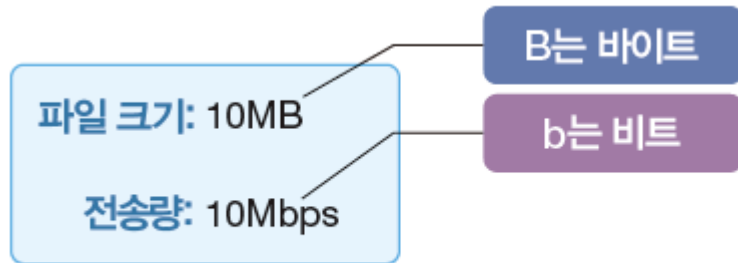


그림 3-8 파일 크기와 네트워크 전송량의 표기 차이

- **rpm** rotate per minute

- 하드디스크가 데이터를 저장하거나 읽는 속도의 단위
- 디스크 원반이 1분 동안 회전하는 수

**02**

**진법 변환**



## 02. 진법 변환

### ■ 컴퓨터가 2진법을 사용하는 이유

- 컴퓨터는 0과 1로 표현하는 2진법 사용
- 인간은 0부터 9까지 숫자 10개로 표현하는 10진법 사용
- 컴퓨터가 2진법을 사용하게 된 것은 최초의 컴퓨터가 켜기와 끄기만 할 수 있는 진공관을 사용했기 때문(꺼지면 0, 켜지면 1로 인식)
- 10진법을 사용하는 컴퓨터를 만들 수도 있으나 빠르게 계산하려면 2진법을 사용하는 것이 유리

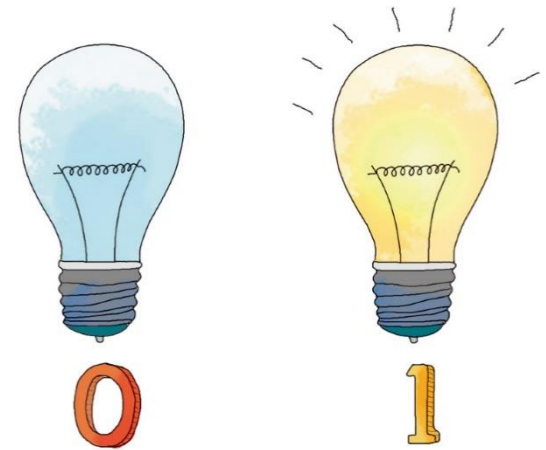


그림 3-6 2진법 개념

### ■ 양자 컴퓨터는?

- 비트가 아닌 큐비트 사용
- 큐비트는 중첩현상 활용
  - 중첩현상? 여러 가능성을 동시에 갖는 성질
  - 0과 1의 값을 동시에 가지며, 1큐비트는 2개를 동시에 계산
- 일반컴퓨터 8비트 : 256개의 상태 중 1개 선택하여 표현
- 양자컴퓨터 8큐비트 : 한 번에 256개 상태 표현

## 02. 진법 변환



그림 3-9 10진수, 2진수, 8진수, 16진수

### 1. 2진법

#### ■ 2진수 → 10진수 변환

- 각 자릿수를 곱한 후 모두 더하기

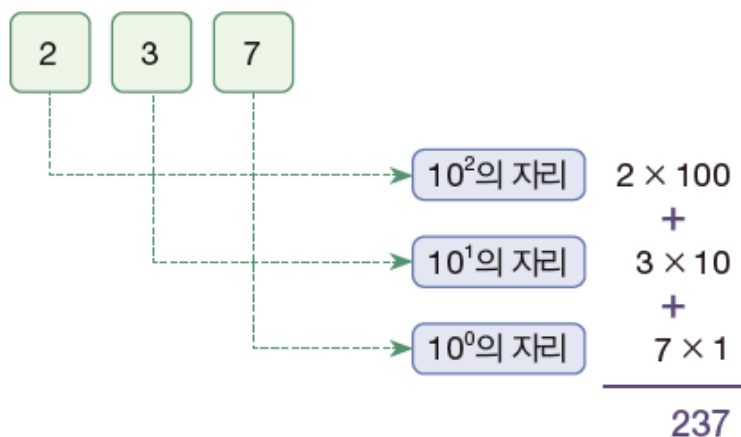


그림 3-10 10진수 표현

## 02. 진법 변환

### 1. 2진법

#### ■ 2진수 → 10진수 변환

- 10진수 표현

1	1	1	0	1	1	0	1
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
128	64	32	16	8	4	2	1
128 + 64 + 32 +            8 + 4 +            1 = 237							

그림 3-11 2진수 → 10진수 변환

$$\begin{aligned} 11101101_2 &= 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 \\ &= 128 + 64 + 32 + 8 + 4 + 1 \\ &= 237 \end{aligned}$$

## 02. 진법 변환

### 1. 2진법

#### ■ 10진수 → 2진수 변환

- 10진수를 계속 2로 나누면서 몫은 아래에, 나머지는 오른쪽에 기록
- 더 이상 나누어지지 않을 때 나머지를 거꾸로 읽기

$$2 \overline{) 237}$$

$$2 \overline{) 118} \text{ ---- } 1$$

$$2 \overline{) 59} \text{ ---- } 0$$

$$2 \overline{) 29} \text{ ---- } 1$$

$$2 \overline{) 14} \text{ ---- } 1$$

$$2 \overline{) 7} \text{ ---- } 0$$

$$2 \overline{) 3} \text{ ---- } 1$$

$$1 \text{ ---- } 1$$

$$237 = 11101101_2$$

10진수를 2로 계속 나누고  
나머지를 밑에서부터  
거꾸로 읽으면 돼.



그림 3-12 10진수 → 2진수 변환

## 02. 진법 변환

### 2. 16진법

#### ■ 16진수를 사용하는 이유

- 더 적은 비트로 숫자를 표현
- 1~9는 10진수와 같고 이후 숫자 6개는 알파벳 사용  
(10은 A, 11는 B, 12는 C, 13은 D, 14는 E, 15는 F로 표기)

2진수 4비트가  
16진수 1비트에  
해당하지.



0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

16진수 2자리로  
1바이트를 나타낼  
수 있어.



그림 3-13 8자리 2진수에 대응하는 2자리 16진수

## 02. 진법 변환

### ■ 문제

- 2진수 11101101을 16진수로 변환하면?

0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1
0	0	1	0	1	0	0	1	0
0	0	1	1	1	0	0	1	0
0	1	0	0	1	0	0	1	0
0	1	0	1	1	0	0	1	0
0	1	1	0	1	0	0	1	0
0	1	1	1	1	0	0	1	0
1	0	0	0	1	0	0	1	0
1	0	0	1	1	0	0	1	0
1	0	1	0	1	0	0	1	0
1	0	1	1	1	0	0	1	0
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	0
1	1	1	0	1	0	0	1	0
1	1	1	1	1	0	0	1	0

## 02. 진법 변환

### 2. 16진법

#### ■ 16진수 → 10진수 변환

$$\begin{aligned} ED_{16} &= E \times 16^1 + D \times 16^0 \\ &= 14 \times 16 + 13 \times 1 \\ &= 224 + 13 \\ &= 237 \end{aligned}$$

그림 3-14 16진수 → 10진수 변환

#### ■ 16진수 → 10진수 변환

$$16 \overline{) 237}$$

$$14 \text{ ..... } 13$$

$$237 = ED_{16}$$



거꾸로!

그림 3-15 10진수 → 16진수 변환

## 02. 진법 변환

### 2. 16진법

#### ■ 16진수의 활용

- 컴퓨터의 색상 표현: 빛의 삼원색인 **RGB**를 조합
  - R, G, B 각 자리가 1바이트이며 값은 0~255
  - RGB 값 (255, 0, 255)는 어떤 색상?
  - #(ff00ff)

RGB		10진수 RGB	16진수 RGB
	White	255 255 255	ffffff
	Red	255 0 0	ff0000
	Blue	0 0 255	0000ff
	Black	0 0 0	000000

그림 3-16 RGB



## 02. 진법 변환

### 2. 16진법

#### ■ 16진수의 활용

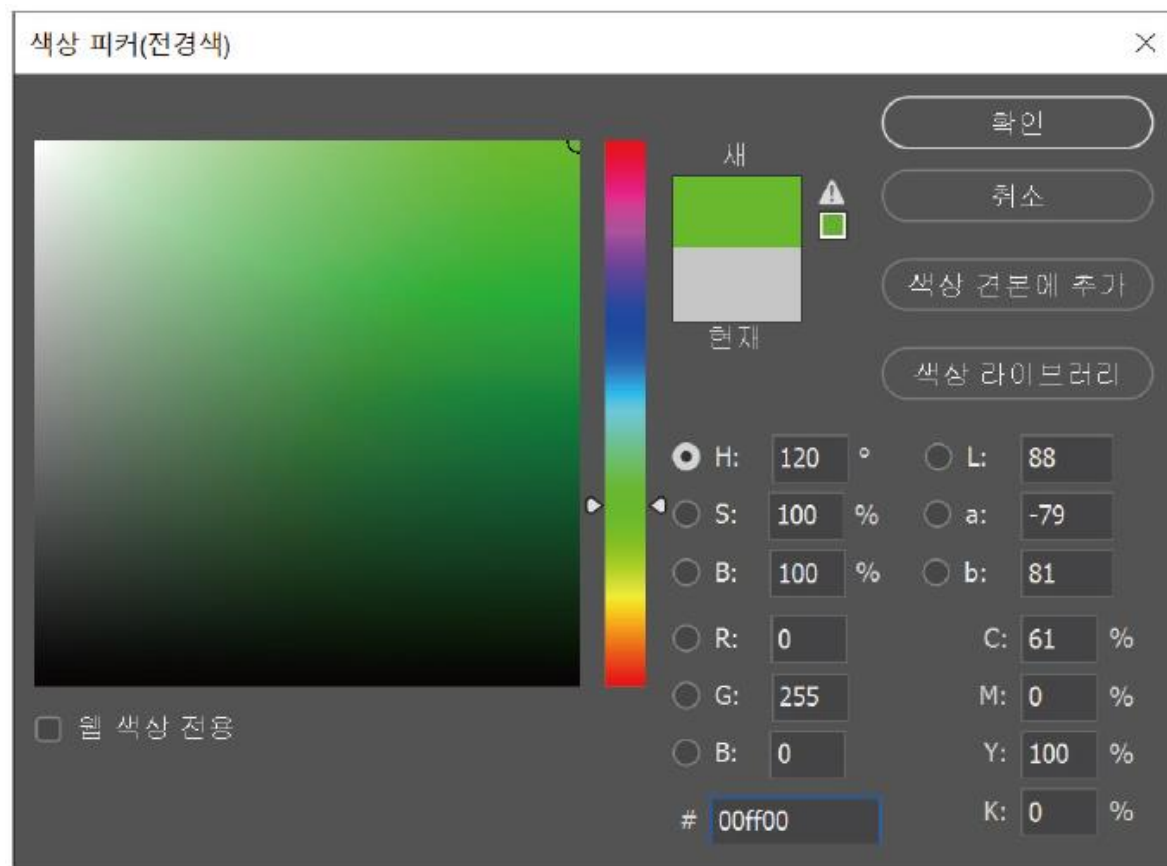


그림 3-17 포토샵에서 녹색을 RGB 값으로 선택

# 03

## 컴퓨터의 데이터 표현

### 03. 컴퓨터의 데이터 표현

- 뛰어서 100미터 11.5초에 가능?
- 문자, 정수, 실수, 불린

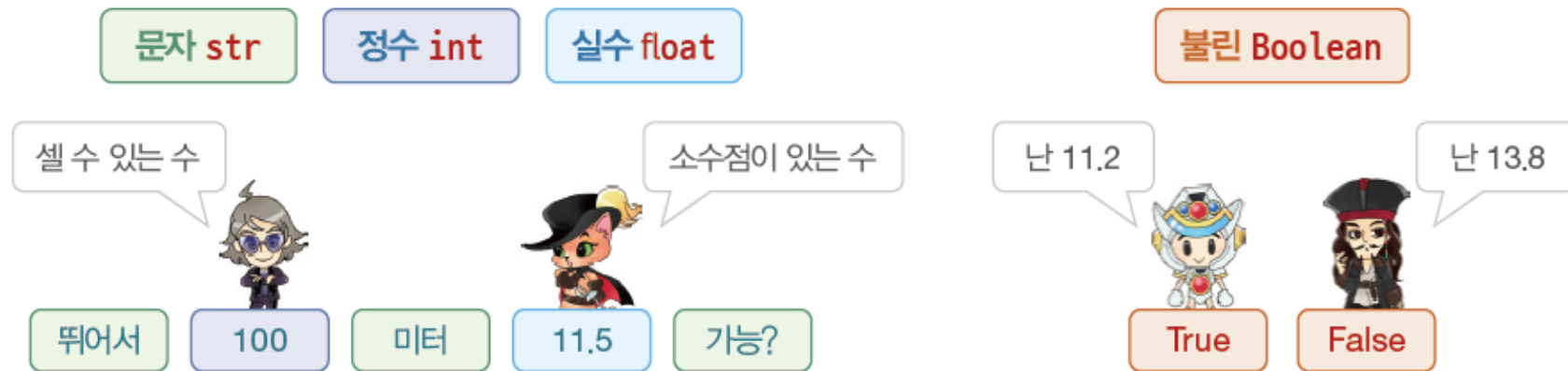


그림 3-18 문자, 정수, 실수, 불린

# 03. 컴퓨터의 데이터 표현

## 1. 숫자 표현

### ■ 정수

- 1의 보수 one's complement : 양수 2진수에서 0과 1을 바꾸어 음수 표현
  - 맨 앞의 비트가 0이면 양수, 1이면 음수

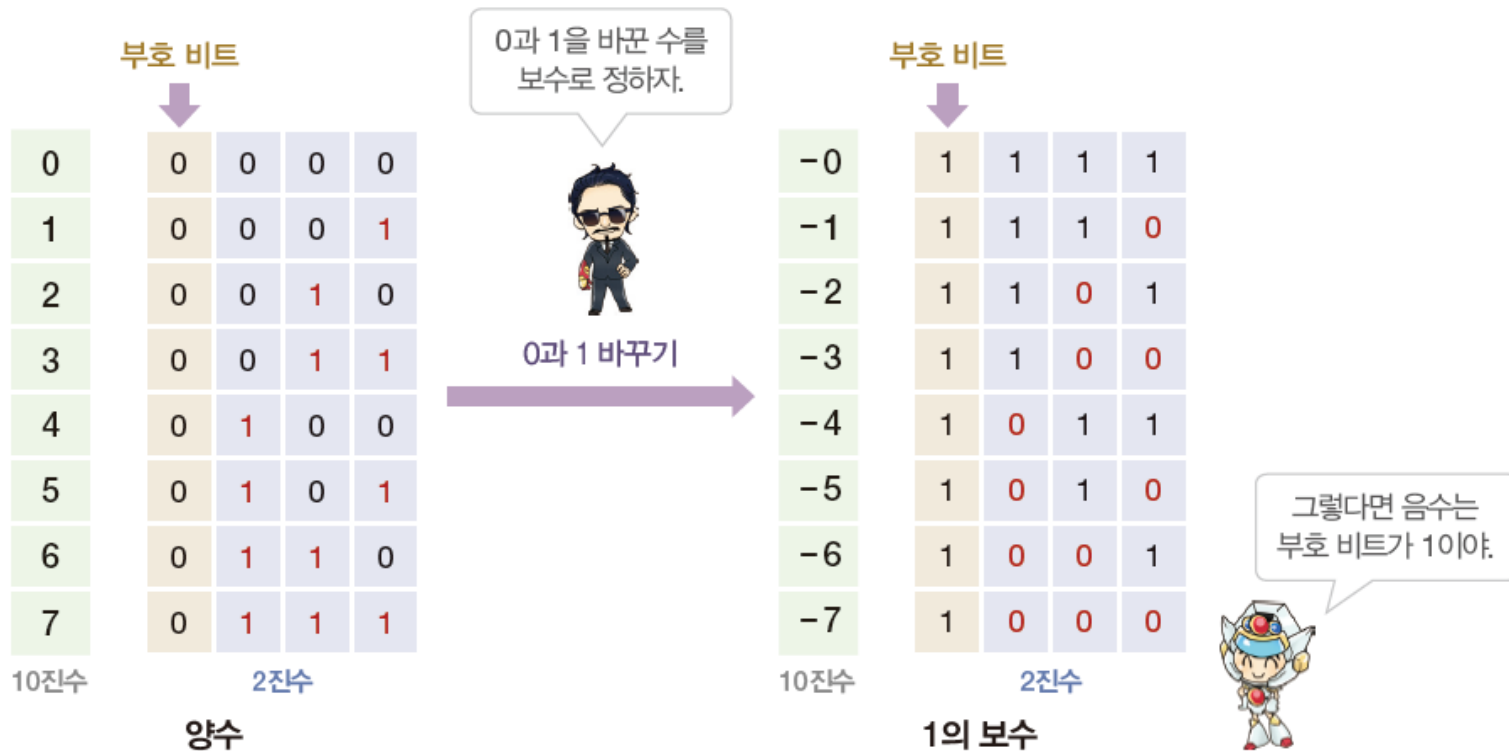


그림 3-19 1의 보수

# 03. 컴퓨터의 데이터 표현

## 1. 숫자 표현

### ■ 정수

- 2의 보수 two's complement

- 필요 없는 -0을 없애기 위해 1의 보수에 1씩 더함
- 표현할 수 있는 음수가 양수보다 1개 더 많음



## 03. 컴퓨터의 데이터 표현

### ■ 컴퓨터의 정수 표현

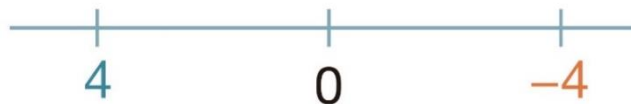


그림 3-18 양수 4와 음수 -4

#### ■ 음수는 1의 보수 또는 2의 보수로 표현

- 1의 보수 : 0은 1로, 1은 0으로 바꿈
- 2의 보수 : 1의 보수+1

[예] -4 표기 : 1의 보수 1011, 2의 보수 1100

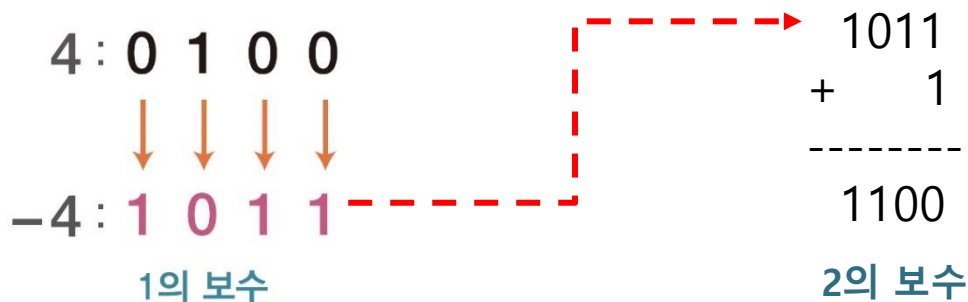


그림 3-19 2진수 0100을 음수로 만드는 과정

# 03. 컴퓨터의 데이터 표현

## ■ 컴퓨터의 정수 표현

- 1의 보수
  - 숫자 0이 +0과 -0으로 2개가 됨
  - 정수 값의 범위 : 7 ~ -7
- 2의 보수
  - -0을 없애고 그 밑의 음수가 한 칸씩 올라옴
  - 정수 값의 범위 : 7 ~ -8
- n비트의 2진수에서 최상위 비트(MSB; Most Significant Bit)는 부호 비트(Signed Bit)

0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	+0
1	1	1	1	-0
1	1	1	0	-1
1	1	0	1	-2
1	1	0	0	-3
1	0	1	1	-4
1	0	1	0	-5
1	0	0	1	-6
1	0	0	0	-7

(a) 1의 보수

0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	+0
1	1	1	1	-1
1	1	1	0	-2
1	1	0	1	-3
1	1	0	0	-4
1	0	1	1	-5
1	0	1	0	-6
1	0	0	1	-7
1	0	0	0	-8

(b) 2의 보수



1을 더해 한 칸씩 위로

## 03. 컴퓨터의 데이터 표현

### ■ 2진 정수 연산

- 뺄셈의 원리를 보면,  $A-B$  대신  $A+(B \text{의 } 2\text{의 보수})$ 를 계산하면 됨
- 뺄셈에서 2의 보수 방식을 사용하는 이유는 뺄셈을 가산기를 사용하여 수행할 수 있음

$$7-3 = 7+(-3)$$

$$\begin{aligned} 7 &= 0111 \\ 3 &= 0011 \\ &= (1\text{의 보수}) \quad 1100 \\ &= (2\text{의 보수}) \quad 1101 \end{aligned}$$

$$\begin{array}{r} 7 = 0111 \\ +(-3) = 1101 \\ \hline 4 = 10100 \end{array}$$

$$3-5 = 3+(-5)$$

$$\begin{aligned} 3 &= 0011 \\ 5 &= 0101 \\ &= (1\text{의 보수}) \quad 1010 \\ &= (2\text{의 보수}) \quad 1011 \end{aligned}$$

$$\begin{array}{r} 3 = 0011 \\ +(-5) = 1011 \\ \hline -2 = 1110 \end{array}$$

\* 최상위비트(MSB)가 1이므로 결과값이 음수임을 확인  
(2의 보수)  $0010 = 2$



## 03. 컴퓨터의 데이터 표현

### 1. 숫자 표현

#### ■ 정수

- 부호 없는 정수 unsigned integer
  - 모든 수가 양수라고 가정하는 표현법

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1

10진수

2진수

맨 앞의 비트는  
부호가 아니야.



8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

10진수

2진수

그림 3-21 부호 없는 정수

## 03. 컴퓨터의 데이터 표현

### 1. 숫자 표현

#### ■ 실수

- 프로그래밍 언어에서 실수는 부동 소수점 방식인 float로 저장
- 정규화 normalization
  - 숫자를 일정한 단위로 맞춤

2800000000		28억
390000000		3.9억
90000000		0.9억
2500000000		25억
1800000000		18억
2300000000		23억

정규화하면  
읽기 편해.




그림 3-22 실수의 정규화

## 03. 컴퓨터의 데이터 표현

### 1. 숫자 표현

#### ■ 실수

- 실수의 정규화는 모든 수를 한자리수.XXXX로 표현
- 컴퓨터에 저장할 때 가수(멘티사)와 지수를 보관

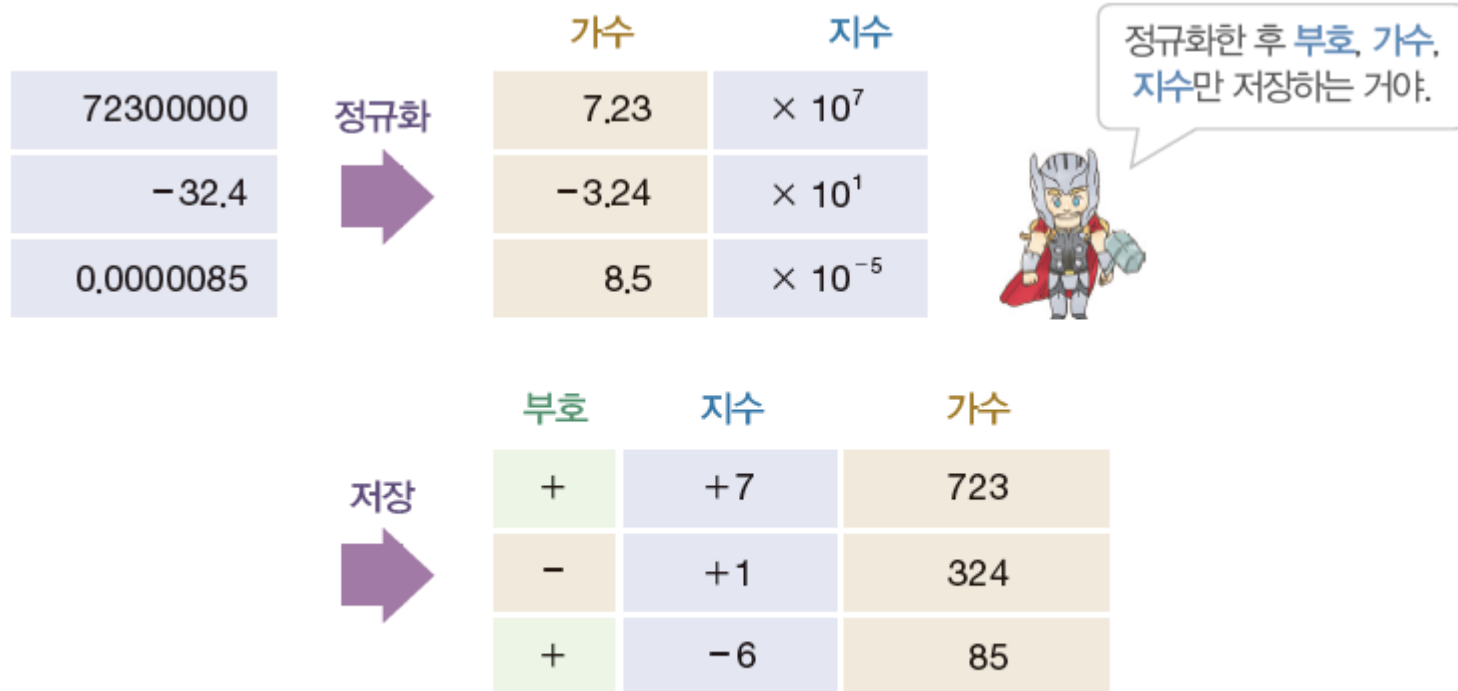


그림 3-23 실수의 정규화

## 03. 컴퓨터의 데이터 표현

### 1. 숫자 표현

#### ■ 정수 표현법과 실수 표현법의 한계

- 정수 표현법은 수의 크기에 한계가 있음

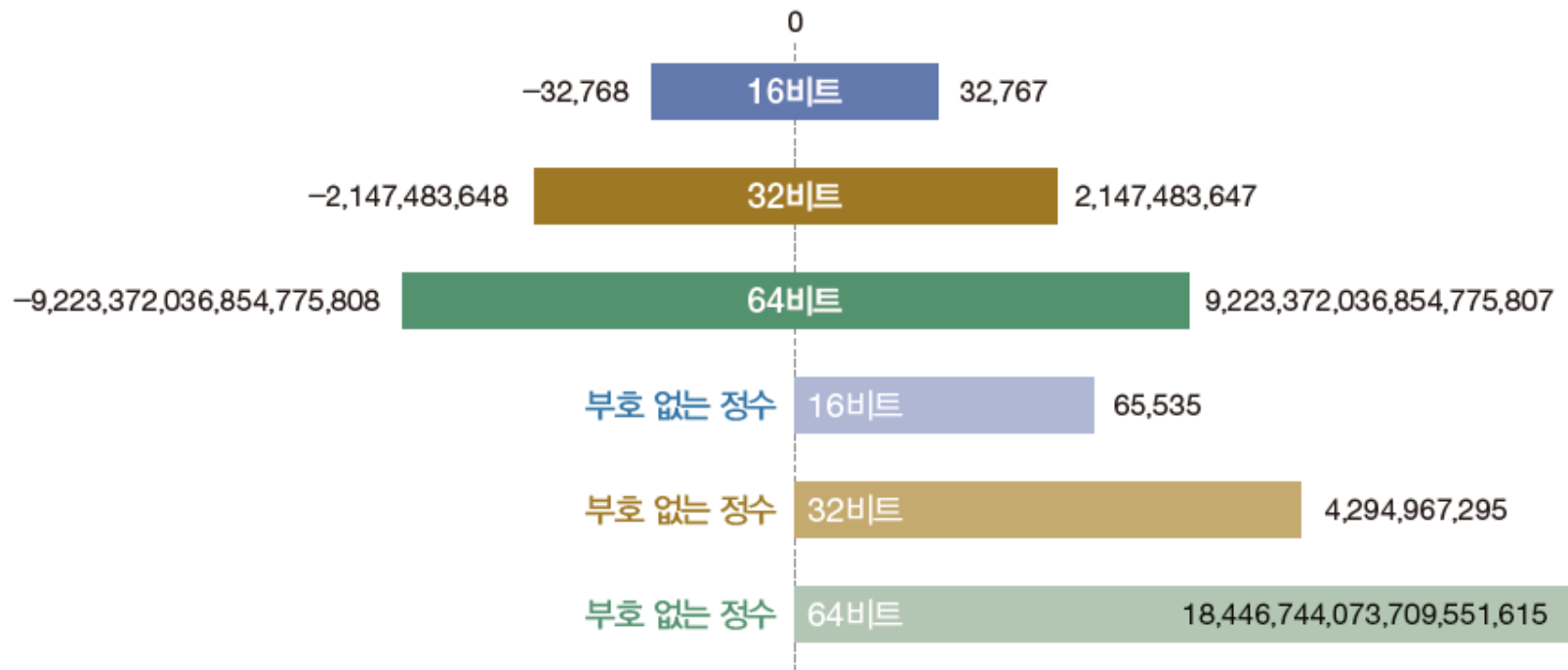


그림 3-24 비트와 정수의 최댓값

# 03. 컴퓨터의 데이터 표현

## 1. 숫자 표현

### ■ 정수 표현법과 실수 표현법의 한계

- 오버플로 overflow

- 정수가 저장할 수 있는 수보다 더 큰 수를 저장할 때 발생하는 오류

- 언더플로 underflow

- 정수가 저장할 수 있는 수보다 더 작은 수를 저장할 때 발생하는 오류



그림 3-25 오버플로와 언더플로

## 03. 컴퓨터의 데이터 표현

### 2. 불린(Boolean)

- 값이 참 또는 거짓인 데이터 형식
- 프로그래밍에서 반복문이나 분기에 많이 사용
  - if X then A, else B
  - while(X)
- C, 파이썬 등 불린 자료형 bool은 True or False
  - C의 경우 'stdbool.h' 추가

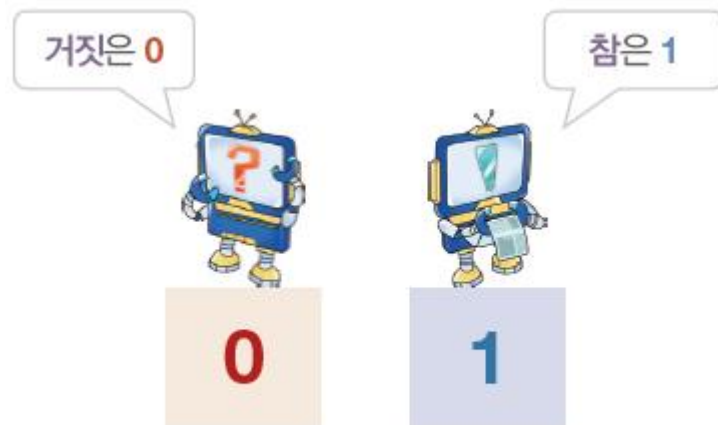


그림 3-27 불린

## 03. 컴퓨터의 데이터 표현

### 3. 문자 표현

#### ■ 아스키코드

- 미국국립표준협회(America National Standards Institute, ANSI)가 제정한 컴퓨터 간 정보 교환용 미국 표준 코드
- 컴퓨터는 문자를 처리하려고 숫자와 문자를 하나씩 대응하는 코드를 사용, 아스키코드(ASCII code)가 대표적
- 7비트로 구성되기 때문에 아스키코드로 표현할 수 있는 문자는 0~127, 총 128( $2^7$ )개



그림 3-28 문자 저장 방식

### 03. 컴퓨터의 데이터 표현

표 3-3 아스키코드

제어 문자			출력 가능한 문자								
10진수	16진수	부호	10진수	16진수	부호	10진수	16진수	부호	10진수	16진수	부호
000	00	NULL	032	20	SP	064	40	@	096	60	`
001	01	SOH	033	21	!	065	41	A	097	61	a
002	02	STX	034	22	"	066	42	B	098	62	b
003	03	ETX	035	23	#	067	43	C	099	63	c
004	04	EOL	036	24	\$	068	44	D	100	64	d
005	05	ENQ	037	25	%	069	45	E	101	65	e
006	06	ACK	038	26	&	070	46	F	102	66	f
007	07	BEL	039	27	'	071	47	G	103	67	g
008	08	BS	040	28	(	072	48	H	104	68	h
009	09	HT	041	29	)	073	49	I	105	69	i
010	0A	LF	042	2A	*	074	4A	J	106	6A	j



### 03. 컴퓨터의 데이터 표현

제어 문자			출력 가능한 문자								
10진수	16진수	부호	10진수	16진수	부호	10진수	16진수	부호	10진수	16진수	부호
011	0B	VT	043	2B	+	075	4B	K	107	6B	k
012	0C	FF	044	2C	,	076	4C	L	108	6C	l
013	0D	CR	045	2D	-	077	4D	M	109	6D	m
014	0E	SO	046	2E	.	078	4E	N	110	6E	n
015	0F	SI	047	2F	/	079	4F	O	111	6F	o
016	10	DLE	048	30	0	080	50	P	112	70	p
017	11	DC1	049	31	1	081	51	Q	113	71	q
018	12	DC2	050	32	2	082	52	R	114	72	r
019	13	DC3	051	33	3	083	53	S	115	73	s
020	14	DC4	052	34	4	084	54	T	116	74	t
021	15	NAK	053	35	5	085	55	U	117	75	u
022	16	SYN	054	36	6	086	56	V	118	76	v

### 03. 컴퓨터의 데이터 표현

제어 문자			출력 가능한 문자								
10진수	16진수	부호	10진수	16진수	부호	10진수	16진수	부호	10진수	16진수	부호
023	17	ETB	055	37	7	087	57	W	119	77	w
024	18	CAN	056	38	8	088	58	X	120	78	x
025	19	EM	057	39	9	089	59	Y	121	79	y
026	1A	SUB	058	3A	:	090	5A	Z	122	7A	z
027	1B	ESC	059	3B	;	091	5B	[	123	7B	{
028	1C	FS	060	3C	<	092	5C	₩	124	7C	
029	1D	GS	061	3D	=	093	5D	]	125	7D	}
030	1E	RS	062	3E	>	094	5E	^	126	7E	~
031	1F	US	063	3F	?	095	5F	-			



그림 3-29 아스키코드의 저장과 변환

## 03. 컴퓨터의 데이터 표현

### 3. 문자 표현

#### ■ 유니코드 unicode

- 컴퓨터에서 세계 각국의 문자를 통일되게 표현하고 다룰 수 있도록 만든 국제적인 코드 규약
- 2바이트로 구성됨
  - UTF는 부호 비트의 길이에 따라 UTF-8, UTF-16, UTF-32 등이 있으며, 이는 가변 길이 인코딩을 수행하여 ASCII 코드의 길이에 맞게 호환
- 10만 개가 넘는 문자를 표현하여 거의 모든 언어를 표현
- 한글 '가'의 유니코드 값은 AC00,  
컴퓨터 내부에는 2진수 1010110000000000으로 저장

	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	갬 AC60	거 AC70	검 AC80	겐 AC90	갯 ACA0	결 ACB0	격 ACC0	겟 ACD0	고 ACE0	곰 ACF0
1	각 AC01	갑 AC11	갯 AC21	갱 AC31	갯 AC41	갯 AC51	갯 AC61	걱 AC71	겁 AC81	갯 AC91	격 ACA1	겹 ACB1	겹 ACC1	겹 ACD1	곡 ACE1	곱 ACF1
2	각 AC02	갯 AC12	갯 AC22	갯 AC32	갯 AC42	갯 AC52	갯 AC62	긔 AC72	긔 AC82	갯 AC92	겹 ACA2	겹 ACB2	겹 ACC2	겹 ACD2	긔 ACE2	긔 ACF2

그림 3-30 한글 유니코드

**04**

**연산과 논리 게이트**

## 03. 컴퓨터의 데이터 표현

### 1. 컴퓨터의 연산

- 연산 operation : 일정한 규칙에 따라 계산
- 연산을 언어로 표현하면 이해하기 어려울 수 있어 기호 사용



그림 3-31 2배를 표현하는 방법

- 연산자 operator : 연산에 사용되는 기호
- 피연산자 operand : 연산의 대상이 되는 기호나 숫자



그림 3-32 연산자와 피연산자

## 03. 컴퓨터의 데이터 표현

### 1. 컴퓨터의 연산

- 대입 연산자 ==

대입 연산자는 연산자  
오른쪽에 있는 것을 연산자  
왼쪽에 대입해야 해.



피연산자

A

연산자

=

피연산자

8

같다는 의미가  
아니야.



같다는 의미의  
연산자는 ==야.



그림 3-33 대입 연산자

### ■ 연산자의 우선순위

표 3-4 연산자 우선순위

순위	연산자	연산 기호	연산 방향
1순위	부호 연산자	-	오른쪽부터 차례로
2순위	곱셈, 나눗셈 연산자	*, /	왼쪽부터 차례로
3순위	덧셈, 뺄셈 연산자	+, -	왼쪽부터 차례로

## 03. 컴퓨터의 데이터 표현

### 1. 컴퓨터의 연산

#### ■ 컴퓨터의 사칙연산

- 덧셈 하나로 사칙연산을 모두 처리함
- 뺄셈은 보수(음수)를 더하여 처리

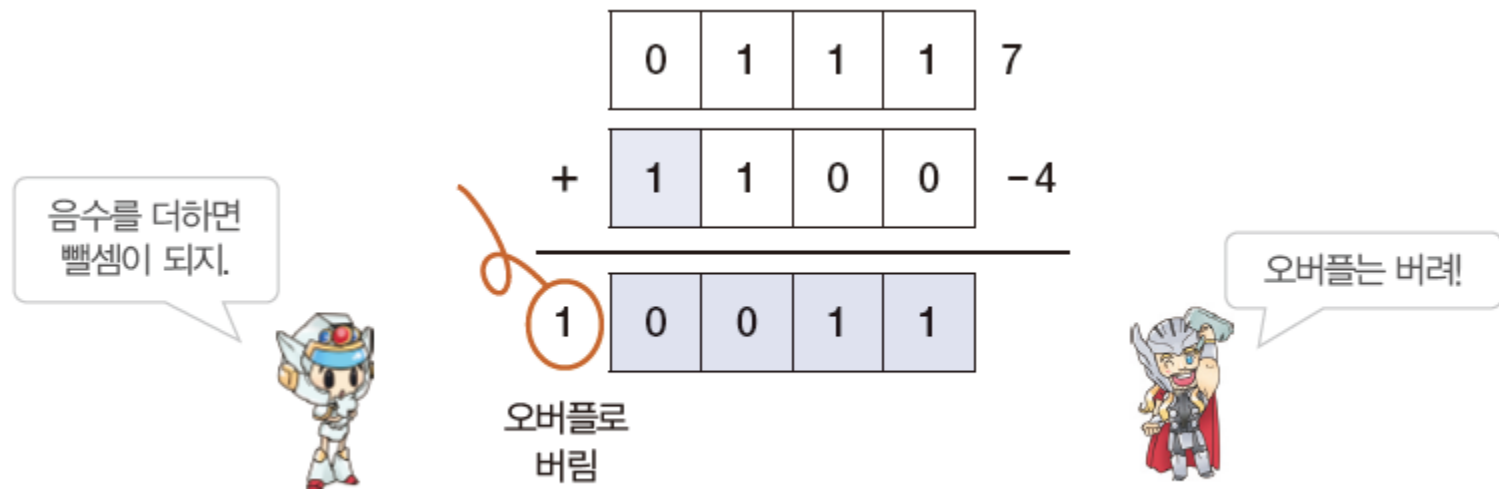


그림 3-34 컴퓨터의 뺄셈 연산

## 03. 컴퓨터의 데이터 표현

### 1. 컴퓨터의 연산

#### ■ 컴퓨터의 사칙연산

- 곱셈은 덧셈을 반복하여 계산
- 나눗셈은 뺄셈을 반복하여 계산

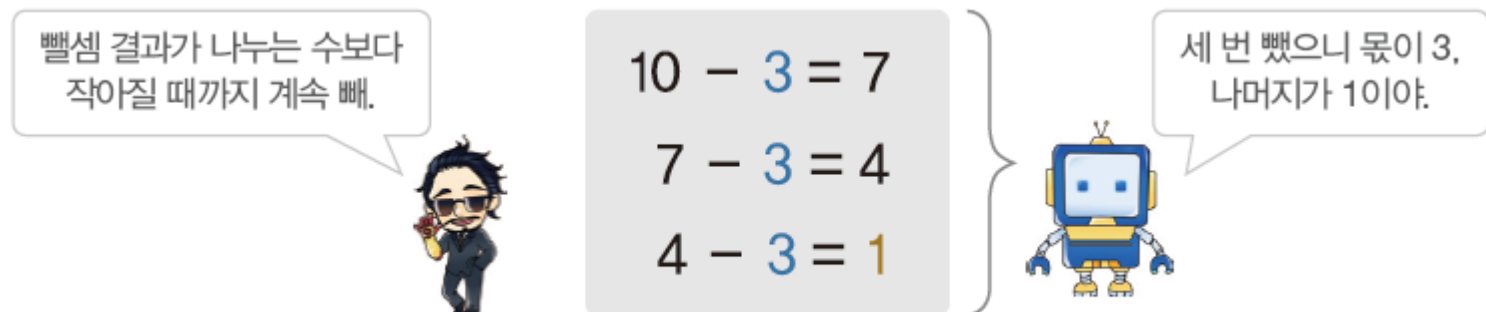


그림 3-35 컴퓨터의 나눗셈 연산



## 03. 컴퓨터의 데이터 표현

### 1. 컴퓨터의 연산

#### ■ 컴퓨터의 사칙연산

- 자리 이동(시프트)
  - 2의 거듭제곱인 수를 곱하거나 2의 거듭제곱으로 나눌 때 쉽게 처리
  - 2의 지수만큼 모든 자릿수를 이동시킴

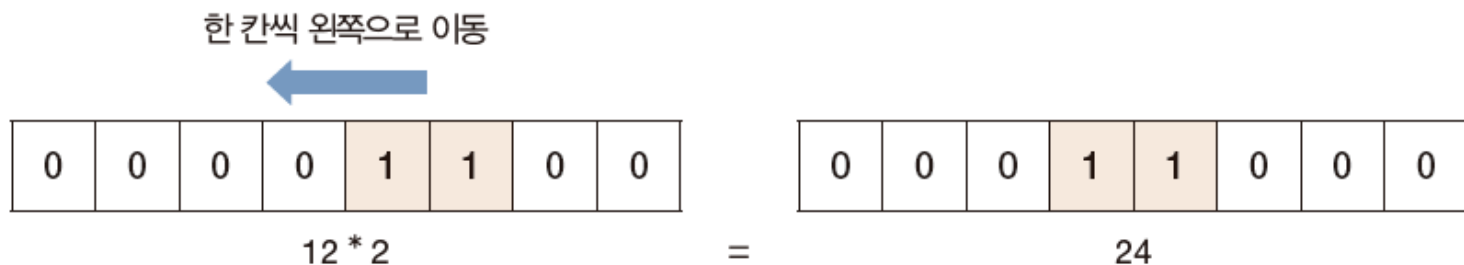


그림 3-36 자리 이동을 이용하는 곱셈 연산

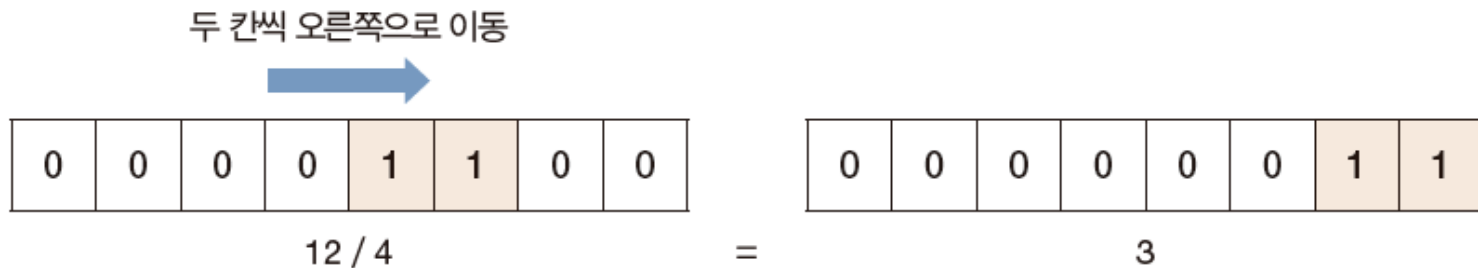
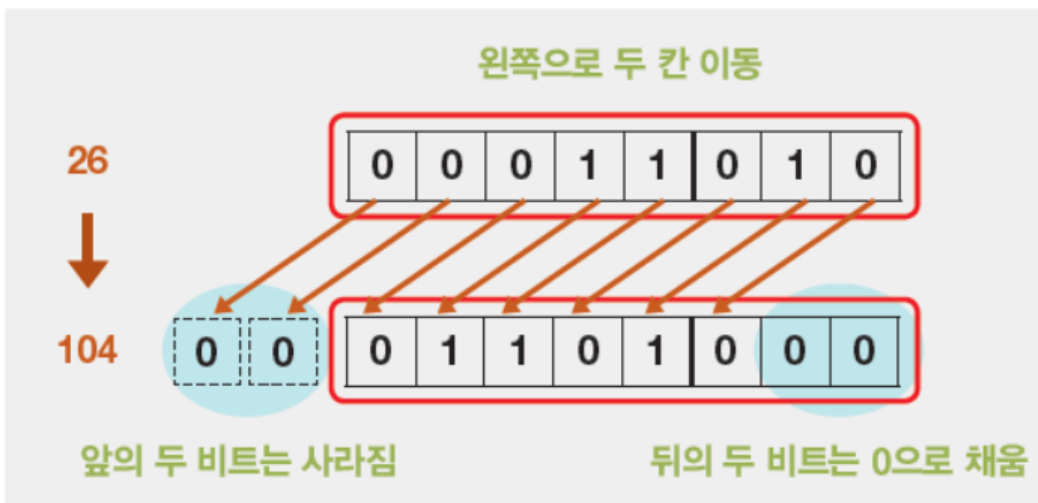


그림 3-37 자리 이동을 이용하는 나눗셈 연산

### 03. 컴퓨터의 데이터 표현

#### ■ 왼쪽 시프트(<<) 연산자

- 나열된 비트를 왼쪽으로 시프트(Shift)해주는 연산자



- 앞의 두 00은 떨어져나가고, 뒤에 빈 두 칸에는 00이 채움. 결과는 26에서 104로 바뀌었는데, 이는 왼쪽으로 시프트 할 때마다  $2^n$ 을 곱한 효과가 나기 때문임 ( $26 \times 2^2 = 104$ ). 즉 왼쪽으로 1회 시프트 할 때는  $2^1$ 을, 2회에는  $2^2$ 을, 3회에는  $2^3$ 을 곱한 효과가 남

### 03. 컴퓨터의 데이터 표현

- 왼쪽 시프트(<<) 연산자 예

```
a = 10  
a << 1 ; a << 2 ; a << 3 ; a << 4
```

출력 결과

```
20 40 80 160
```

- 시프트 할 때마다  $10 \times 2^1 = 20$ ,  $10 \times 2^2 = 40$ ,  $10 \times 2^3 = 80$ ,  $10 \times 2^4 = 160$ 의 결과가 나옴

### 03. 컴퓨터의 데이터 표현

#### ■ 오른쪽 시프트(>>) 연산자

- 나열된 비트를 오른쪽으로 시프트(Shift)해주는 연산자



- 오른쪽의 두 비트는 떨어져나가고 왼쪽의 두 비트에는 부호 비트(양수는 0이, 음수는 1)가 채워짐. 이는  $2^n$ 으로 나눈 효과.
- 또한 시프트 연산은 정수만 연산하므로 몫만 남음( $26 / 2^2 = 6$ ). 즉 오른쪽으로 1회 시프트 할 때는  $2^1$ , 2회에는  $2^2$ , 3회에는  $2^3$ 으로 나누는 효과가 남

### 03. 컴퓨터의 데이터 표현

- 오른쪽 시프트(>>) 연산자 예

```
a = 10  
a >> 1 ; a >> 2 ; a >> 3 ; a >> 4
```

출력 결과

```
5 2 1 0
```

- 시프트 할 때마다  $10/2^1=5$ ,  $10/2^2=2$ ,  $10/2^3=1$ ,  $10/2^4=0$ 의 결과가 나옴

## 03. 컴퓨터의 데이터 표현

### 2. 논리 연산자

- 논리 연산 logical operation : 불린 자료형인 참과 거짓의 연산

#### ■ AND 연산자

- 두 조건이 모두 참(True)일 때만 결과가 참



영자	미숙	부산
NO	NO	못 간다
NO	OK	못 간다
OK	NO	못 간다
OK	OK	간다

영자 그리고(AND) 미숙이 OK 하면 부산에 간다.

그림 3-38 AND 논리 연산

## 03. 컴퓨터의 데이터 표현

### 2. 논리 연산자

#### ■ OR 연산자

- 두 조건이 모두 거짓(False)일 때만 결과가 거짓



영자	미숙	부산
NO	NO	못 간다
NO	OK	간다
OK	NO	간다
OK	OK	간다

영자 또는(OR) 미숙이 OK 하면, 부산에 간다.

그림 3-39 OR 논리 연산

## 03. 컴퓨터의 데이터 표현

### 2. 논리 연산자

#### ■ XOR 연산자

- 두 조건이 서로 다를 때만 결과가 참



영자	미숙	부산
NO	NO	못 간다
NO	OK	간다
OK	NO	간다
OK	OK	못 간다

영자와 미숙 중 한 사람과(XOR) 부산에 가거나, 아니면 못 간다.

그림 3-40 XOR 논리 연산



## 03. 컴퓨터의 데이터 표현

### 2. 논리 연산자

#### ■ NOT 연산자

- 참과 거짓을 바꾸는 연산자



미숙	부산
OK	못 간다
NO	간다

영자는 미숙과 반대로(NOT) 한다.

그림 3-41 NOT 논리 연산

## 03. 컴퓨터의 데이터 표현

### 2. 논리 연산자

- AND 연산 : 값 2개가 모두 참일 때만 참이 되는 연산
- OR 연산 : 값 2개 중 하나라도 참이면 참이 되는 연산
- XOR 연산 : 값 2개 중 하나라도 다르면 참이 되며, 값 2개가 같으면 거짓이 되는 연산
- NOT 연산 : 반대 값을 만드는 연산

입력		$X \text{ AND } Y$	$X \text{ OR } Y$	$X \text{ XOR } Y$	NOT $X$	NOT $Y$
$X$	$Y$					
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

### 03. 컴퓨터의 데이터 표현

표 2-18 기본 논리 연산의 진리표

입력		X AND Y	X OR Y	X XOR Y
X	Y			
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

#### ■ 응용 논리 연산

- 선택적 세트(Selective Set)
- 특정 비트를 세트하는, 즉 1로 만드는 연산이다.

A=1 0 0 1 0 0 1 0 연산 전

B=0 0 0 0 1 1 1 1 선택적 세트(OR) 연산

---

A=1 0 0 1 1 1 1 1 연산 후

- 선택적 보수(Selective Complement)
- 특정 비트를 1의 보수로 만드는, 즉 반전하는 연산이다.

A=1 0 0 1 0 0 1 0 연산 전

B=0 0 0 0 1 1 1 1 선택적 보수(XOR) 연산

---

A=1 0 0 1 1 1 0 1 연산 후

# 03. 컴퓨터의 데이터 표현

표 2-18 기본 논리 연산의 진리표

입력		X AND Y	X OR Y	X XOR Y
X	Y			
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

## ■ 논리 연산

### ■ 응용 논리 연산

#### - 마스크(Mask)

- 특정 비트만 선택적으로 리셋하는, 즉 0으로 만드는 연산이다.

A=1 0 1 1 0 1 0 1 연산 전

B=0 0 0 0 1 1 1 1 마스크(AND) 연산

A=0 0 0 0 0 1 0 1 연산 후

#### - 삽입(Insert)

- 특정 위치에 새로운 비트 값을 추가하는 연산으로, 마스크 연산과 선택적 세트 연산을 차례로 수행하면 된다.

A=1 0 1 1 1 0 1 0 연산 전

B=1 1 1 1 0 0 0 0 마스크(AND) 연산

A=1 0 1 1 0 0 0 0 마스크 결과

B=0 0 0 0 1 1 0 0 삽입(OR) 연산

A=1 0 1 1 1 1 0 0 최종 삽입 결과

## 03. 컴퓨터의 데이터 표현

### 3. 논리 게이트

- 논리 게이트 logic gate : 논리 연산을 사용하여 만든 컴퓨터의 논리회로
- 컴퓨터의 모든 작업은 논리 게이트에 의해 이루어짐

#### ■ 논리 게이트의 종류

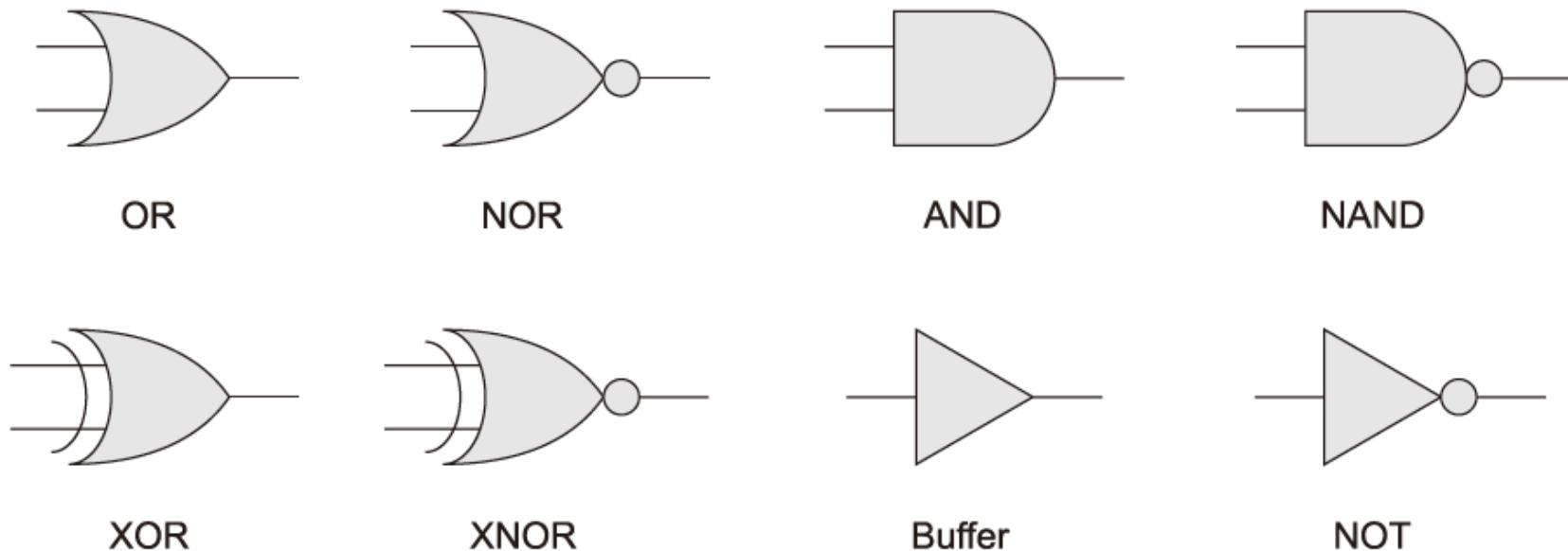


그림 3-42 논리 게이트 심벌

## 03. 컴퓨터의 데이터 표현

### 3. 논리 게이트

- 논리 게이트의 종류

표 3-6 논리 연산 진리표 2

NAND 연산			NOR 연산			XNOR 연산		
입력		출력	입력		출력	입력		출력
0	0	1	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0
1	0	1	1	0	0	1	0	0
1	1	0	1	1	0	1	1	1

## 03. 컴퓨터의 데이터 표현

### 3. 논리 게이트

#### ■ 논리 게이트의 종류

- (1) 자동차 문이 열려 있을 때 문 열림 경고등을 켜는 회로
  - 어느 문이든지 하나 이상 열려있으면 경고등이 작동해야 함
- (2) 자동차가 후진할 때 후방 경고등이 켜지는 회로
  - 엔진이 작동 중이어야 하고 동시에 후진 기어가 작동해야 함

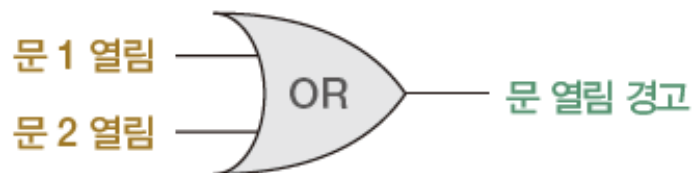


그림 3-43 논리 게이트 사용 예

입력(A)	입력(B)	출력(X)
0	0	0
0	1	1
1	0	1
1	1	1

입력(A)	입력(B)	출력(X)
0	0	0
0	1	0
1	0	0
1	1	1

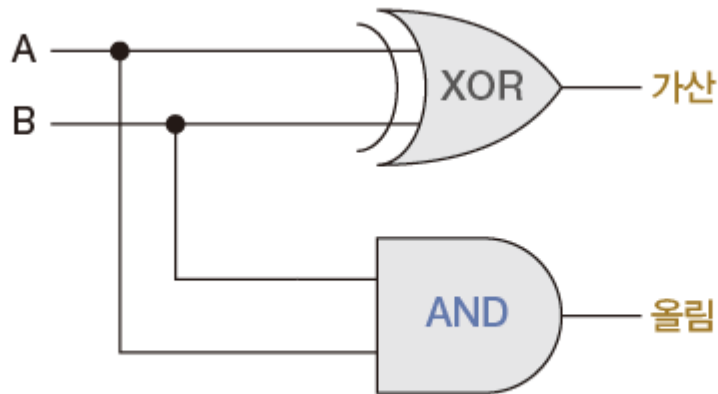
## 03. 컴퓨터의 데이터 표현

### 3. 논리 게이트

#### ■ 가산기

- 한 자리 2진수 A와 B의 덧셈 연산

가산기에서는  
XOR는 더하고  
AND는 올리지.



A	B	가산	올림
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

그림 3-44 가산기



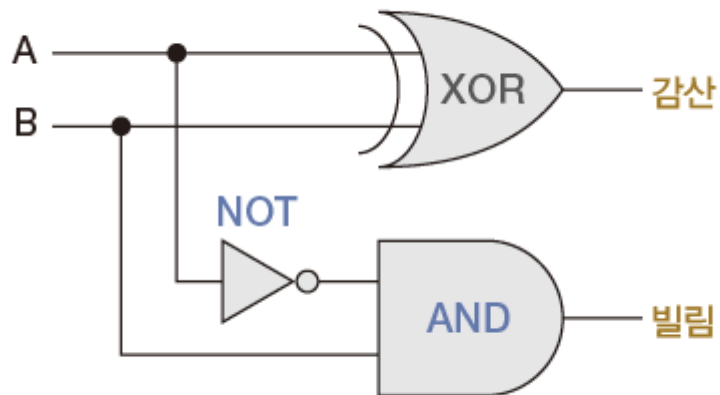
## 03. 컴퓨터의 데이터 표현

### 3. 논리 게이트

#### ■ 감산기

- 2진수 A에서 B를 빼는 연산

감산기에서는  
XOR는 빼고  
NOT A AND B는  
빌려오지.



A	B	감산	빌림
0	0	0	0
1	0	1	0
0	1	1	1
1	1	0	0

그림 3-45 감산기