



# 목차

**01**

컴퓨터 단위

**02**

진법과 진법 변환

**03**

컴퓨터 내부의 데이터 표현

**04**

논리 연산자

# 학습목표

- 컴퓨터의 저장 용량 단위와 처리 속도 단위를 알아본다.
- 2진법과 16진법을 이해하고, 진법 변환 방법을 살펴본다.
- 컴퓨터 내부의 데이터 표현 방법을 살펴본다.
- 논리 연산자(AND, OR, XOR, NOT)의 연산 방법을 알아본다.

## ■ 비트(bit)

- 컴퓨터에서 데이터를 표시할 때 사용하는 최소 단위
- 2진법을 사용하기 때문에 1비트로 표현할 수 있는 수는 0과 1

## ■ 바이트(byte)

- 비트 8개를 묶어 사용하는 단위
- 1바이트 = 8비트



그림 3-1 비트와 바이트

## ■ CPU 데이터 처리 속도

- 초기 : 한 번에 8비트를 처리하는 CPU 사용 (예: 개인용 컴퓨터 애플 II)
- 이후 : 32비트 CPU 등장
- 현재 : 64비트 CPU 대중화



그림 3-2 32비트와 64비트 차이

## ■ 워드(word)

- 컴퓨터가 한 번에 처리할 수 있는 데이터 크기를 나타내는 단위
- 32비트 CPU : 한 번에 32비트 데이터 처리, 1워드=32비트
- 64비트 CPU : 한 번에 64비트 데이터 처리, 1워드=64비트

## ■ 요약

표 3-1 컴퓨터 용량 단위

용량 단위	설명
비트(bit)	데이터를 표현하는 최소 단위
바이트(byte)	8비트를 하나로 모은 것
워드(word)	컴퓨터가 한 번에 처리할 수 있는 데이터 단위 <b>예</b> 32비트 CPU의 1워드는 32비트

표 3-2 큰 용량을 표현하는 단위

용량 단위	표기	2진 크기	10진 크기	바이트 대비 크기	10진 단위
바이트(Byte)	B	1	1	1B	일
킬로바이트(Kilo Byte)	KB	$2^{10}$	$10^3$	1,000B	일천
메가바이트(Mega Byte)	MB	$2^{20}$	$10^6$	1,000,000B	일백만
기가바이트(Giga Byte)	GB	$2^{30}$	$10^9$	1,000,000,000B	일십억
테라바이트(Tera Byte)	TB	$2^{40}$	$10^{12}$	1,000,000,000,000B	일조
페타바이트(Peta Byte)	PB	$2^{50}$	$10^{15}$	1,000,000,000,000,000B	일천조

- 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



## ■ CPU 속도

### ■ 클럭(clock)

- 컴퓨터에서 일정한 박자를 만들어 내는 장치
- 컴퓨터 내 모든 부품은 클럭이 일정한 간격으로 틱을 만들면 거기에 맞추어 작업

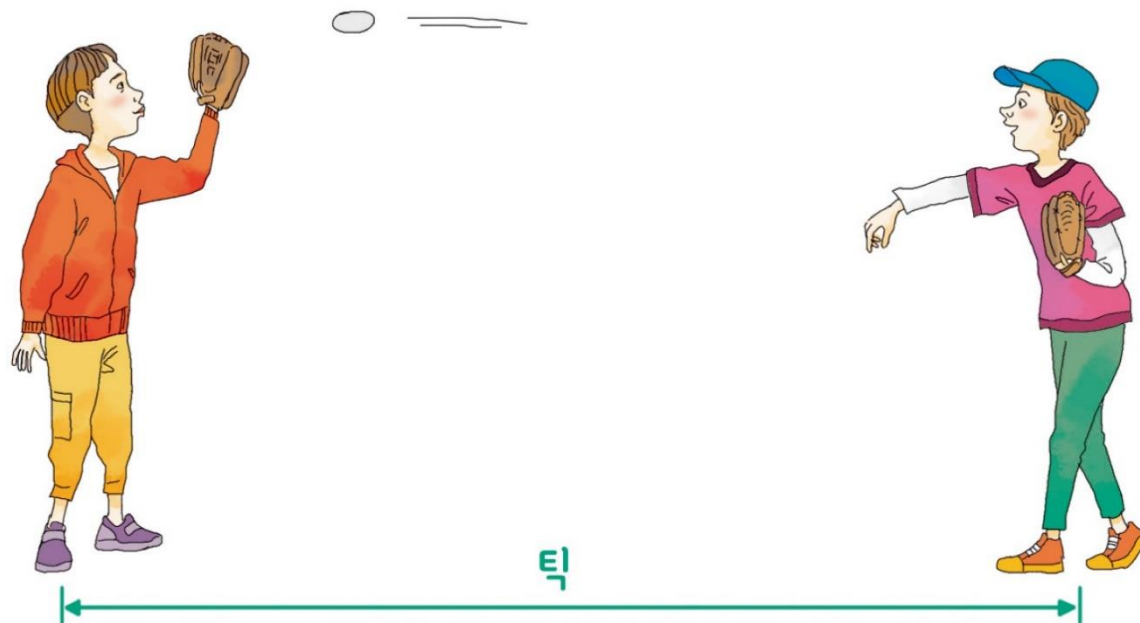


그림 3-3 틱의 개념

## ■ CPU 속도

- 헤르츠(hertz)
  - CPU 성능을 나타내는 단위(표시 단위 : Hz)
  - 1초 동안 클럭 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

## ■ 1헤르츠와 2헤르츠

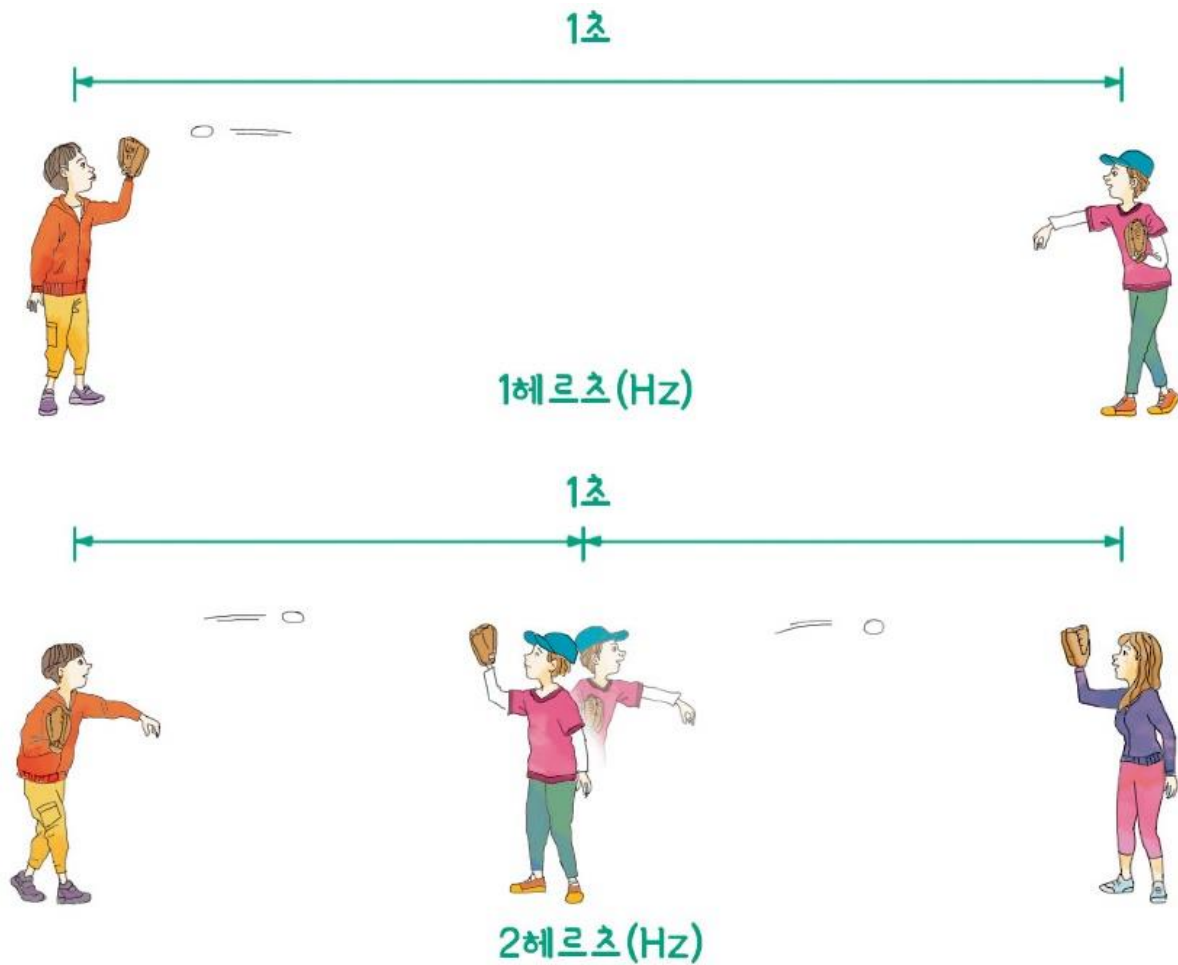


그림 3-4 헤르츠

## ■ 하드디스크 속도 (rpm)

- 디스크 원반이 1분 동안 회전하는 수
- 7,200rpm → 디스크 원반이 1분에 7,200번 회전
- 숫자가 클수록 데이터를 저장하거나 읽는 속도가 빠름  
(5,400rpm < 7,200rpm)

## ■ 네트워크 전송량 (bps)

- ‘bit per second’의 약어로 네트워크상에서 1초 동안 보내는 데이터의 양
  - 파일 용량 표기 vs 네트워크 전송량 표기
    - 파일 용량 표기 : 단위가 바이트(byte)며 대문자 B로 표기
    - 네트워크 전송량 표기 : 단위가 비트(bit)며 소문자 b로 표기
- \*\* 1바이트는 8비트이므로 10MB는 기본적으로 10Mbps보다 8배 큼**

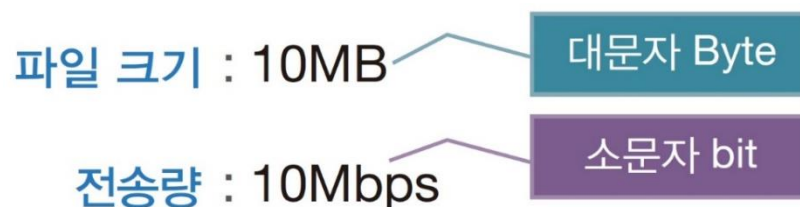


그림 3-5 파일 크기와 네트워크 전송량 표기 차이

## ■ 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

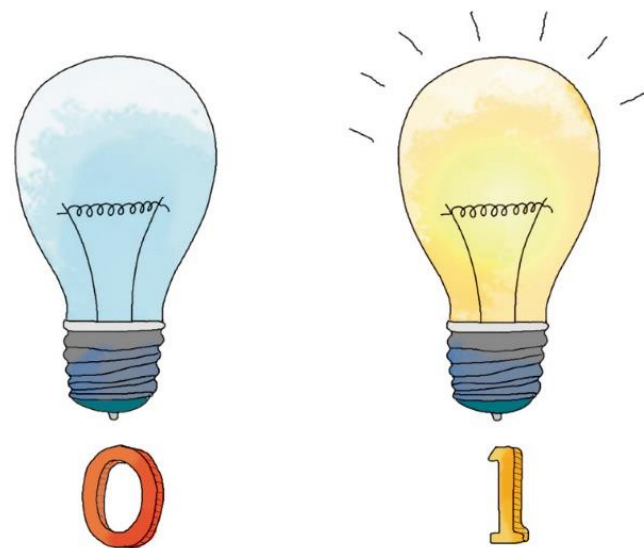


그림 3-6 2진법 개념

## ■ 2진법의 식당 메뉴 비유

- 두 가지 음식만 팔면 음식 메뉴를 만드는 속도를 올릴 수 있고 테이블 회전율도 높일 수 있음

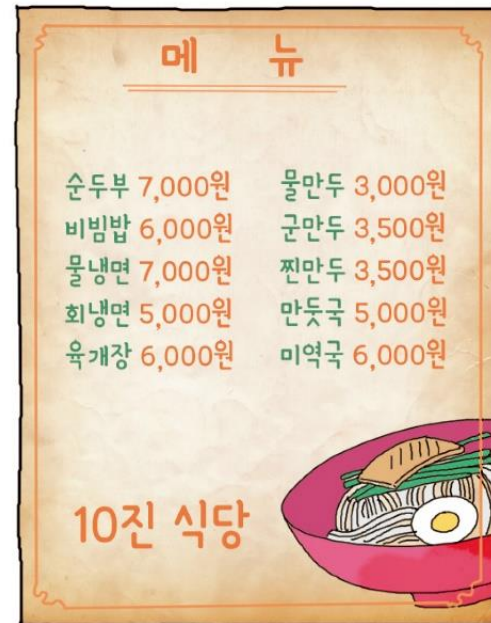
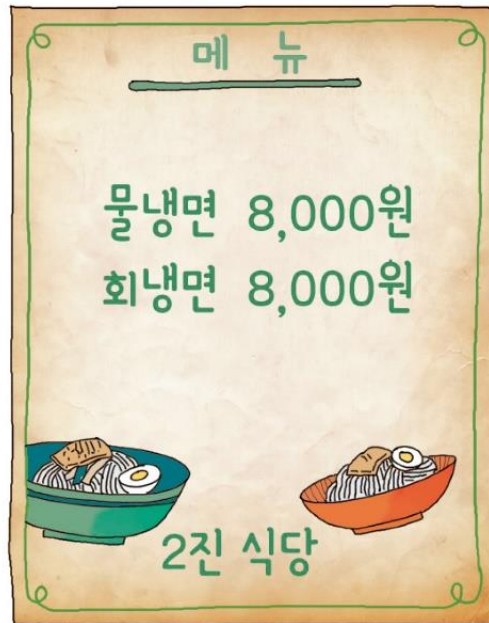


그림 3-7 2진 식당과 10진 식당

## ■ 10진수 표현의 원리

$$\begin{aligned} 237 &= 2 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 \\ &= 2 \times 100 + 3 \times 10 + 7 \times 1 \\ &= 237 \end{aligned}$$

그림 3-8 10진수 표현

## ■ 2진수 → 10진수 변환

- 2진수의 각 자릿수를 곱한 후 모두 더함

**\*\* 숫자 아래의 밑첨자는 진법을 나타냄, 10진수는 표기 생략**

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

그림 3-9 2진수의 10진수 변환



## ■ 10진수 → 2진수 변환

- 10진수를 계속 2로 나누면서 몫은 아래에, 나머지는 오른쪽에 기록한 후 더 이상 나누어지지 않을 때 나머지를 거꾸로 읽음

$$\begin{array}{r} 2 \overline{) 237} \\ 2 \overline{) 118} \text{ ..... } 1 \\ 2 \overline{) 59} \text{ ..... } 0 \\ 2 \overline{) 29} \text{ ..... } 1 \\ 2 \overline{) 14} \text{ ..... } 1 \\ 2 \overline{) 7} \text{ ..... } 0 \\ 2 \overline{) 3} \text{ ..... } 1 \\ 1 \text{ ..... } 1 \end{array}$$

$237 = 11101101_2$

그림 3-10 10진수의 2진수 변환

## ■ 16진법

- 0~F까지 숫자 16개를 사용
- 1~9는 10진수와 같고 이후 숫자 6개는 알파벳 사용  
(10은 A, 11는 B, 12는 C, 13은 D, 14는 E, 15는 F로 표기)

## ■ 16진법을 사용하는 이유

- 컴퓨터에서는 16진법은 바이트를 좀 더 적은 숫자로 표현하는 데 사용
- 2진수 11111111을 표현하려면 8자리가 필요하지만,  
16진수로 표현하면 FF의 2자리로 표현 가능

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

그림 3-11 4자리 2진수에 대응하는 1자리 16진수

## ■ 문제

- 2진수 11101101을 16진수로 변환하면?

정답 : ED

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 → 10진수 변환

- 각 자리의 숫자와 해당 위치의 단위 값을 곱한 후 모두 더하면 됨

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

**그림 3-12** 16진수의 10진수 변환

## ■ 10진수 → 16진수 변환

- 10진수를 계속 16로 나누면서 몫은 아래에, 나머지는 오른쪽에 기록한 후 더 이상 나누어지지 않을 때 나머지를 거꾸로 읽음

$$16 \overline{) 237}$$

$$14 \cdots 13$$

$$237 = \text{ED}_{16}$$

**그림 3-13** 10진수의 16진수 변환

## ■ 16진수 사용 예

- RGB는 컴퓨터에서 이미지의 점 색상 하나를 표현할 때 사용
  - \*\* RGB는 빛의 삼원색인 빨간색(Red), 녹색(Green), 파란색(Blue)을 뜻함
- RGB는 각각 1바이트 크기를 가지므로 1바이트가 표현할 수 있는 값은 0에서 255까지 총 256( $2^8$ )단계
- 컴퓨터에서 하나의 색상을 표현할 때는 빨간색(R), 녹색(G), 파란색(B)을 256단계로 섞어 사용
  - RGB(0, 0, 0) : 검은색
  - RGB(255, 255, 255) : 흰색
  - RGB(255, 0, 0) : 빨간색
  - RGB(0, 255, 0) : 녹색
  - RGB(0, 0, 255) : 파란색

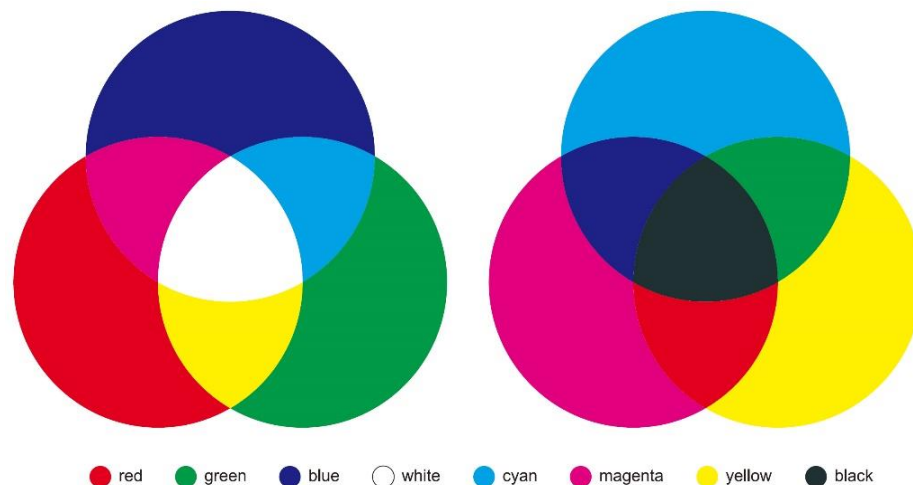


그림 3-14 RGB와 CMYK

## ■ 포토샵의 컬러 피커

- 빨간색(R), 녹색(G), 파란색(B) 입력란에 0부터 255까지 숫자를 입력
- 아래쪽 입력란에 16진수를 넣어도 됨(녹색은 #00ff00이라고 작성)

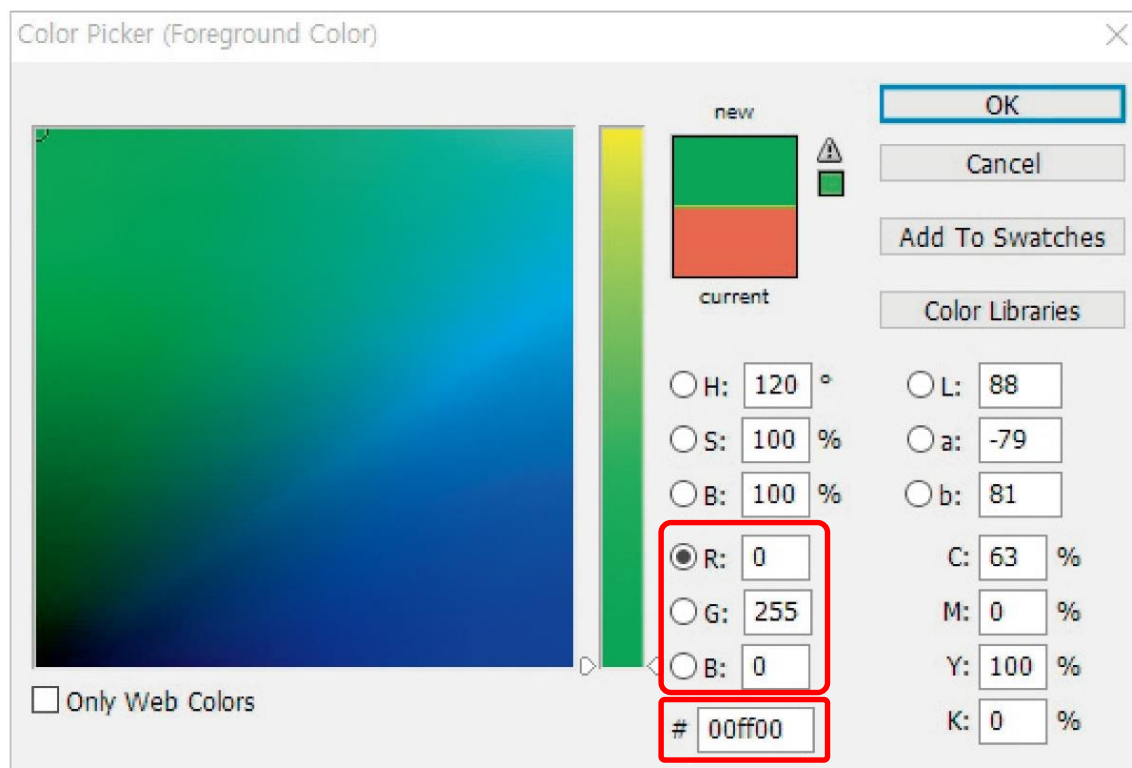


그림 3-15 포토샵에서 녹색 RGB 값을 10진수와 16진수로 설정하는 모습



## ■ CMYK

- 이미지를 종이에 출력하거나 인쇄할 때는 RGB 대신 CMYK를 사용
- CMYK : 시안(Cyan), 마젠타(Magenta), 옐로(Yellow), 블랙(Black=key)

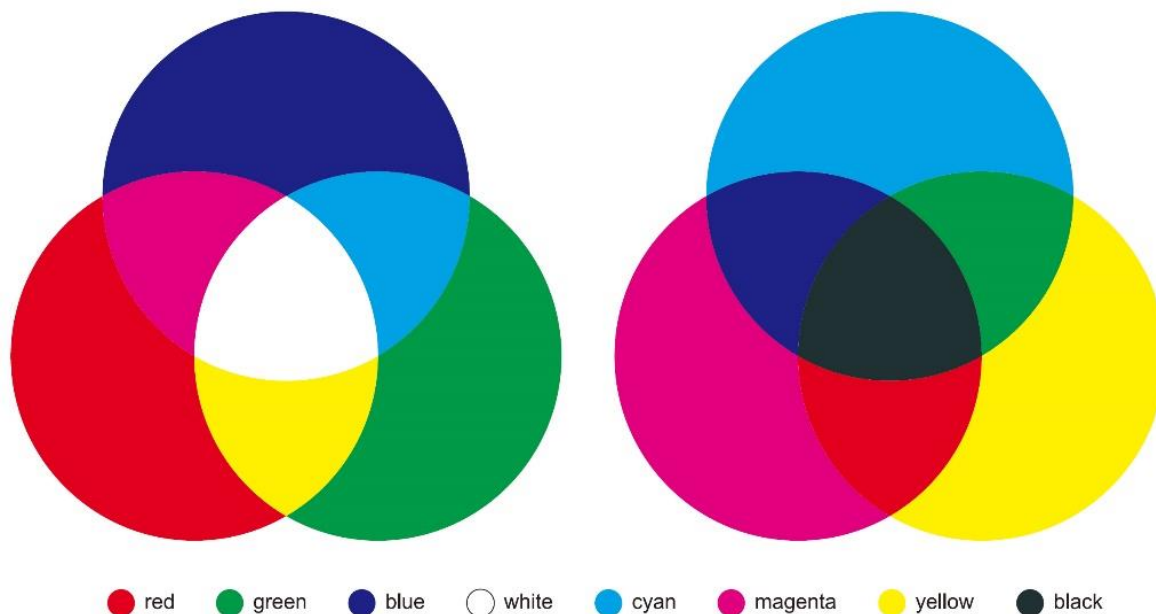


그림 3-14 RGB와 CMYK

## ■ 아스키코드

- 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
- 52개의 영문 알파벳(a~Z), 10개의 숫자(0~9), 32개의 특수문자(@, # 등) 그리고 하나의 공백문자로 구성

0x30 = 0 (숫자 0)

0x40 = A (대문자 알파벳)

0x61 = a (소문자 알파벳)

표 3-3 아스키코드 일부

10 진수	16 진수	부호	10 진수	16 진수	부호	10 진수	16 진수	부호	10 진수	16 진수	부호
032	20	SP	056	38	8	080	50	P	104	68	h
033	21	!	057	39	9	081	51	Q	105	69	i
034	22	"	058	3A	:	082	52	R	106	6A	j
035	23	#	059	3B	;	083	53	S	107	6B	k
036	24	\$	060	3C	<	084	54	T	108	6C	l
037	25	%	061	3D	=	085	55	U	109	6D	m
038	26	&	062	3E	>	086	56	V	110	6E	n
039	27	'	063	3F	?	087	57	W	111	6F	o
040	28	(	064	40	@	088	58	X	112	70	p

표 3-3 아스키코드 일부(계속)

10 진수	16 진수	부호	10 진수	16 진수	부호	10 진수	16 진수	부호	10 진수	16 진수	부호
041	29	)	065	41	A	089	59	Y	113	71	q
042	2A	*	066	42	B	090	5A	Z	114	72	r
043	2B	+	067	43	C	091	5B	[	115	73	s
044	2C	,	068	44	D	092	5C	\	116	74	t
045	2D	-	069	45	E	093	5D	]	117	75	u
046	2E	.	070	46	F	094	5E	^	118	76	v
047	2F	/	071	47	G	095	5F	_	119	77	w
048	30	0	072	48	H	096	60	`	120	78	x
049	31	1	073	49	I	097	61	a	121	79	y
050	32	2	074	4A	J	098	62	b	122	7A	z
051	33	3	075	4B	K	099	63	c	123	7B	{
052	34	4	076	4C	L	100	64	d	124	7C	
053	35	5	077	4D	M	101	65	e	125	7D	}
054	36	6	078	4E	N	102	66	f	126	7E	~
055	37	7	079	4F	O	103	67	g	127	7F	DEL

## ■ 아스키코드의 저장과 변환

- 사람이 'YOU'라고 입력하면 컴퓨터는 '89, 79, 85'의 2진수 값 저장
- 저장된 내용을 사용자에게 보여 줄 때는 숫자를 다시 문자로 바꾸어 보여줌

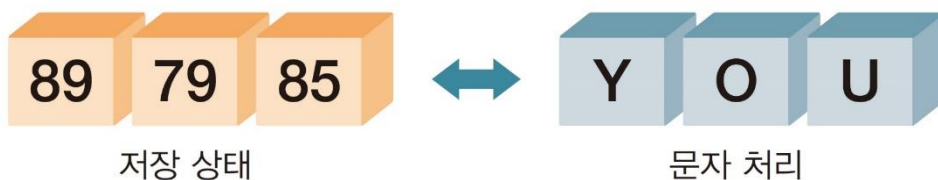


그림 3-16 아스키코드의 저장과 변환

## ■ 유니코드

- 전 세계 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰 수 있도록 설계된 산업 표준 문자코드
- 10만 개가 넘는 문자를 표현할 수 있기 때문에 대부분의 언어 표현이 가능
- 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	갸 AC60	거 AC70	검 AC80	겐 AC90	갠 ACA0	겉 ACB0	겉 ACC0	겉 ACD0	고 ACE0	곰 ACF0
1	각 AC01	갑 AC11	갸 AC21	갹 AC31	갹 AC41	갹 AC51	갹 AC61	걱 AC71	겁 AC81	갸 AC91	겉 ACA1	겉 ACB1	겉 ACC1	겉 ACD1	곡 ACE1	굽 ACF1
2	각 AC02	갸 AC12	갹 AC22	갹 AC32	갹 AC42	갹 AC52	갹 AC62	긔 AC72	긔 AC82	겉 AC92	겉 ACA2	겉 ACB2	겉 ACC2	겉 ACD2	곡 ACE2	긔 ACF2

그림 3-17 한글 유니코드

## ■ UTF-8

- 아스키 코드와 호환 가능
- 유니코드를 인코딩(Encoding)하는 방법
  - 인코딩(Encoding)이란 컴퓨터가 이해할 수 있는 형태로 바꾸어주는 것
- 가변형 인코딩 방식(1byte ~ 4byte)
- 영어 1바이트, 한글은 초성, 중성, 종성이 있기에 보통 3바이트로 표현
- 가변을 구분하기 위해 첫 바이트에 표식을 넣었는데 2 byte는 110으로 시작하고, 3바이트는 1110으로 시작한다.
- 나머지는 10으로 시작

## ■ UTF-16

- 16비트 기반으로 문자 인코딩, 한글 또한 2바이트로 저장

## ■ EUC-KR(MS949, CP949)

- 한국에서 독자적으로 사용하는 인코딩 방식
- 완성된 상태의 문자가 2바이트 값으로 정해져서 표현하는 방식
- 영어 같은 1바이트 문자를 사용하는 곳이 아닌 한글, 일본어, 중국어 같은 2바이트 문자를 사용하는 곳에서 많이 사용된다.



## ■ ?

- 대체 이런 상황은 왜 발생하는 것일까요? 그 이유는 컴퓨터에서 한글을 표현하는 다양한 문자열 셋과 인코딩 방식들이 맞지 않기 때문입니다.

痠⑥뽕怨듬켄 ?뽕씩?쌔켄 愿???뽕뽕, 媛겅닉?ㄴ쫘 ?댄븍?쌔린

愿쉴연 ?v샐?쌔췌 媛 ?뽕켄?맏뽕 ?듬뽕留??깅뽕 ?긏린

1. ?ㄴ뽕以??겅뽕 ?뽕쫘?얏뽕

a. 由ㄴ님?ㄴ? ?댕췌??紐⑤겅???뽕쫘?얏뽕

3?슌꺽 ?ㄴ뽕?뽕췌 紐⑤겅???댄뽕由ㄴ??댕꺽 怨심ㅇ??? ?뽕쫘?얏뽕濡? 由ㄴ

님??而ㄴ쫘 2.6.18 媛겅켄??而ㄴ뽕?겅쫘?댕쫘 而ㄴ쫘怨?xxx?ㄴ뽕???쌔뽕

h1211 蹂대뽕瑜??댕췌?쌔뽕 吏뽕뽕??

理쉴꺽 紐(?)뽕??紐⑤겅???뽕씩 ?쉴뽕???쉴역?댕뽕?심꺽, ?뽕췌?쌔린 ?액꺽

吏紐삼뽕怨?뽕꺽꺽??

## ■ 정수란?

- 컴퓨터가 사용하는 숫자 중 가장 기본 값으로 1, 2, 3, 4, ...처럼 셀 수 있는 수(int)

## ■ 컴퓨터의 정수 표현

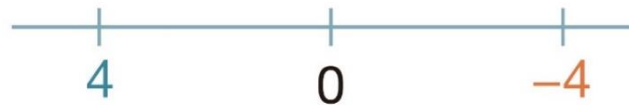


그림 3-18 양수 4와 음수 -4

- 양수는 2진법을 그대로 사용

[예] 4 표기 : 0100

## ■ 컴퓨터의 정수 표현

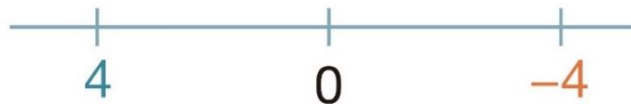


그림 3-18 양수 4와 음수 -4

### ■ 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

## ■ 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

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

그림 3-20 4비트로 표현할 수 있는 수의 범위

## ■ 실수란?

- 소수점 이하의 자리가 존재하는 숫자(float)

## ■ 컴퓨터의 실수 표현

- 아주 작은 수부터 매우 큰 수까지 표현할 수 있기 때문에 다루기가 까다로워 정규화하여 표현
- 정규화(normalization)
  - 숫자를 일정한 단위로 맞추는 작업

11,200,000,000	→	112억
300,000,000		3억
2,700,000,000		27억

그림 3-21 정수의 정규화

## ■ 컴퓨터의 실수 표현

### ■ 실수의 정규화

- 모든 수를 0.XXXX로 표현 (예:  $72300000 \rightarrow 0.723 \times 10^8$ )
- 정규화된 수에서 0 이하 숫자와 지수만 저장하면 아무리 큰 수라도 컴퓨터에 간단히 저장할 수 있음
- 정규화된 표현에서 소수점 아래 숫자를 가수 혹은 멘티사(mentisa)라 함

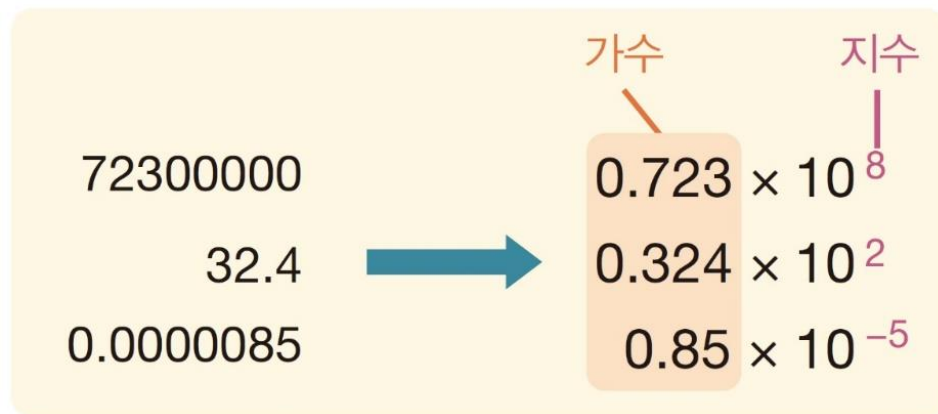


그림 3-22 실수의 정규화

## ■ 정규화된 실수가 컴퓨터에 저장되는 과정 - 사례 1

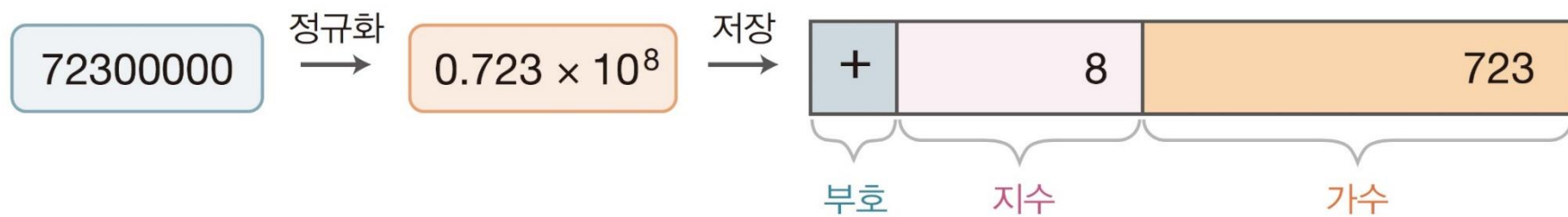
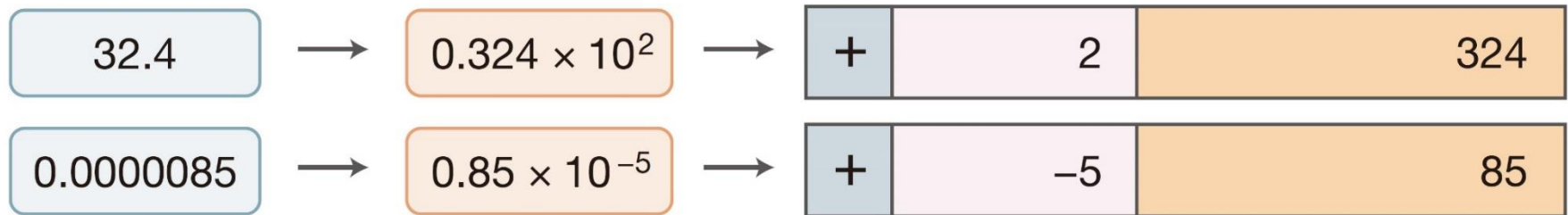


그림 3-23 정규화된 실수  $0.723 \times 10^8$  저장

## ■ 정규화된 실수가 컴퓨터에 저장되는 과정 - 사례 2, 사례 3



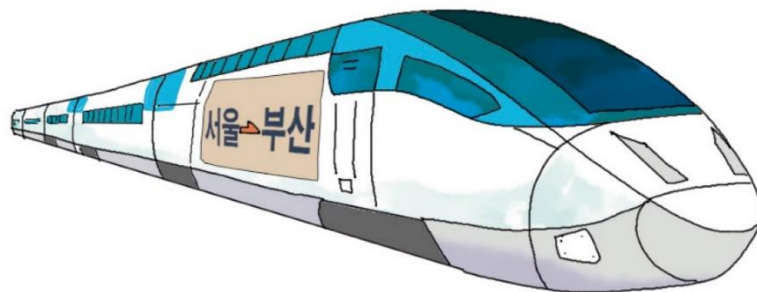
**그림 3-24** 정규화된 실수  $0.324 \times 10^2$ 과  $0.85 \times 10^{-5}$  저장



## ■ AND 연산자

- 두 조건이 모두 참(true)일 때만 결과가 참

영자 그리고(AND) 미숙이 가면, 부산 간다.



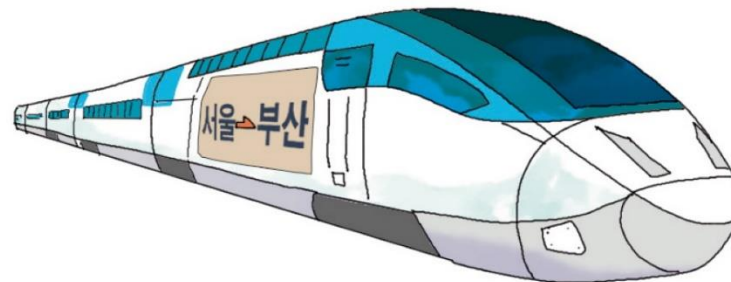
영자	미숙	부산
OK	OK	간다
OK	NO	못 간다
NO	OK	못 간다
NO	NO	못 간다

그림 3-25 AND 논리 연산

## ■ OR 연산자

- 두 조건이 모두 거짓(false)일 때만 결과가 거짓

영자 혹은(OR) 미숙이 가면, 부산 간다.



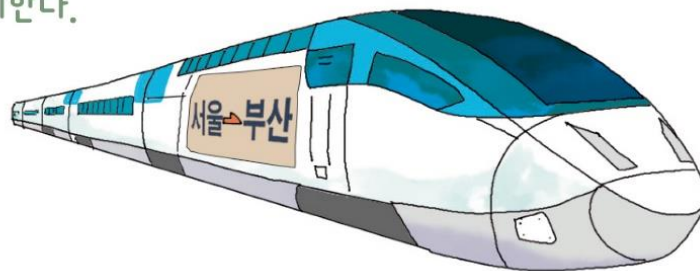
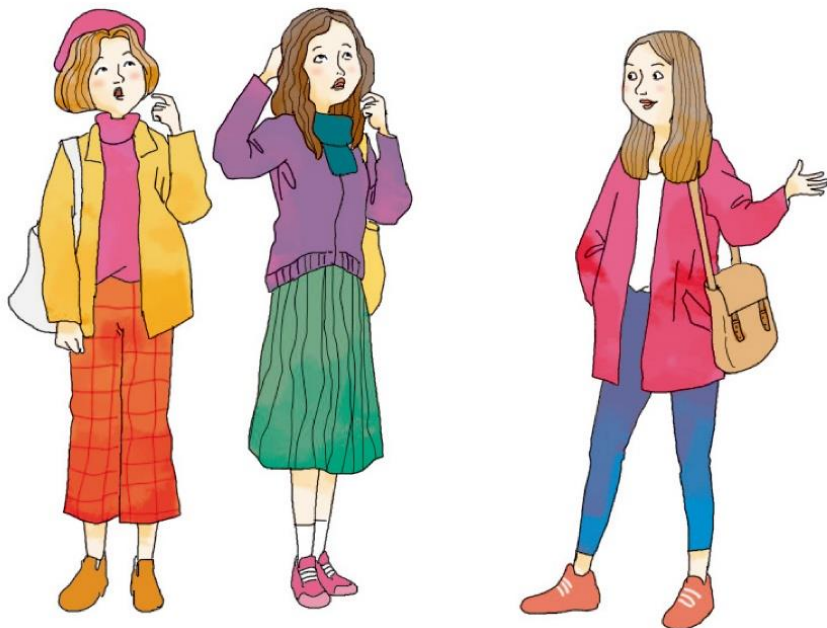
영자	미숙	부산
OK	OK	간다
OK	NO	간다
NO	OK	간다
NO	NO	못 간다

그림 3-26 OR 논리 연산

## ■ XOR 연산자

- 두 조건이 서로 다를 때만 결과가 참

영자나 미숙이 둘 중 하나 하고만(XOR) 부산 가든가 아니면 포기한다.



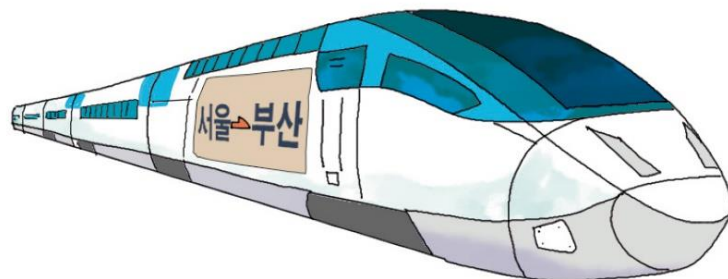
영자	미숙	부산
OK	OK	못 간다
OK	NO	간다
NO	OK	간다
NO	NO	못 간다

그림 3-27 XOR 논리 연산 결과

## ■ NOT 연산자

- 참과 거짓을 바꾸어 주는 연산

미숙이와 반대로 (NOT) 한다.



미숙	부산
OK	못 간다
NO	간다

그림 3-28 NOT 논리 연산 결과

## 정리

- 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

표 3-4 논리 연산의 진리표

AND 연산			OR 연산			XOR 연산			NOT 연산	
입력		출력	입력		출력	입력		출력	입력	출력
T	T	T	T	T	T	T	T	F	T	F
T	F	F	T	F	T	T	F	T	F	T
F	T	F	F	T	T	F	T	T		
F	F	F	F	F	F	F	F	F		

**Thank you!**