

# 컴퓨터과학전공 전공체험

## [ 2주차 : 데이터처리 ]

상명대 컴퓨터과학전공  
신경섭

# Contents

- 비트 저장
  - 1.1 비트, 연산, 게이트
  - 1.2 주기억장치
  - 1.3 대용량 기억장치
- 2진 체계
  - 2.1 비트 패턴을 이용한 정보 표현
  - 2.2 2진법
  - 2.3 정수의 저장
  - 2.4 분수의 저장
- 비트의 처리/활용
  - 3.1 데이터 압축
  - 3.2 통신 오류

# 1.1 비트와 비트 패턴

- 비트(bit): 2진 숫자(binary digit) - **0 또는 1**
- 비트 패턴은 정보 표현에 사용된다.
  - 숫자
  - 텍스트 문자
  - 이미지
  - 사운드
  - 기타 등등

예) 비트는 언제 사용?

32비트 컴퓨터, 64비트 컴퓨터,

32비트 운영체제,

인터넷 속도 100메가: 100Mbps

## 1.1 부울(Boolean) 연산

- 부울 연산:  
한 개 이상의 참(true)/거짓(false) 값을 다루는 연산
- 부울 연산자
  - AND: &
  - OR: |
  - XOR (eXclusive OR): ^
  - NOT: ~

# 1.1 부울 연산 AND, OR, XOR, NOT

## AND 연산

$$\begin{array}{r} 0 \\ \text{AND } 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0 \\ \text{AND } 1 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ \text{AND } 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ \text{AND } 1 \\ \hline 1 \end{array}$$

## OR 연산

$$\begin{array}{r} 0 \\ \text{OR } 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0 \\ \text{OR } 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ \text{OR } 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ \text{OR } 1 \\ \hline 1 \end{array}$$

## XOR 연산

$$\begin{array}{r} 0 \\ \text{XOR } 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0 \\ \text{XOR } 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ \text{XOR } 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ \text{XOR } 1 \\ \hline 0 \end{array}$$

# 1.1 게이트(gate)

- 게이트: 부울 연산을 계산하는 장치
  - 흔히 (소형) 전자 회로로 구현된다
  - 컴퓨터 제작에 사용되는 구성 요소

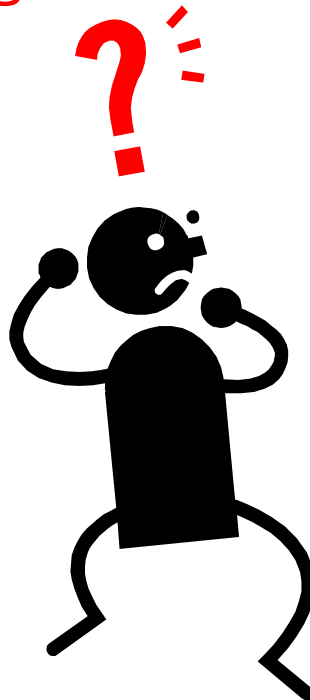
예)

AND Gate

OR Gate

XOR Gate

Not Gate



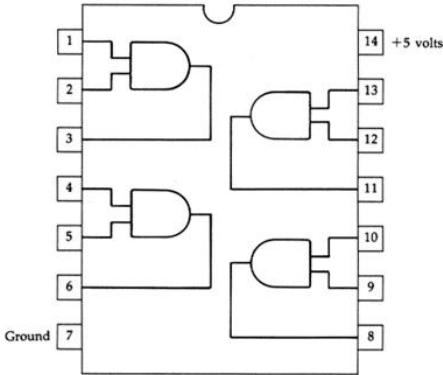
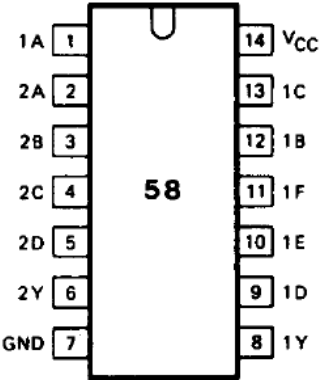
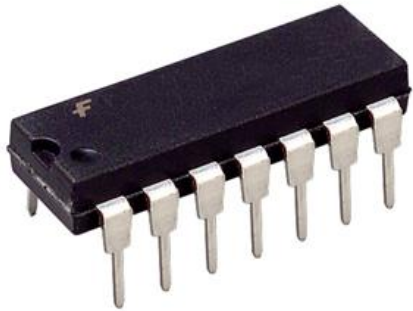
# 1.1 AND, OR, XOR, NOT 게이트 기호와 입출력 값

AND Gate		
입력1	입력2	출력
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate		
입력1	입력2	출력
0	0	0
0	1	1
1	0	1
1	1	1

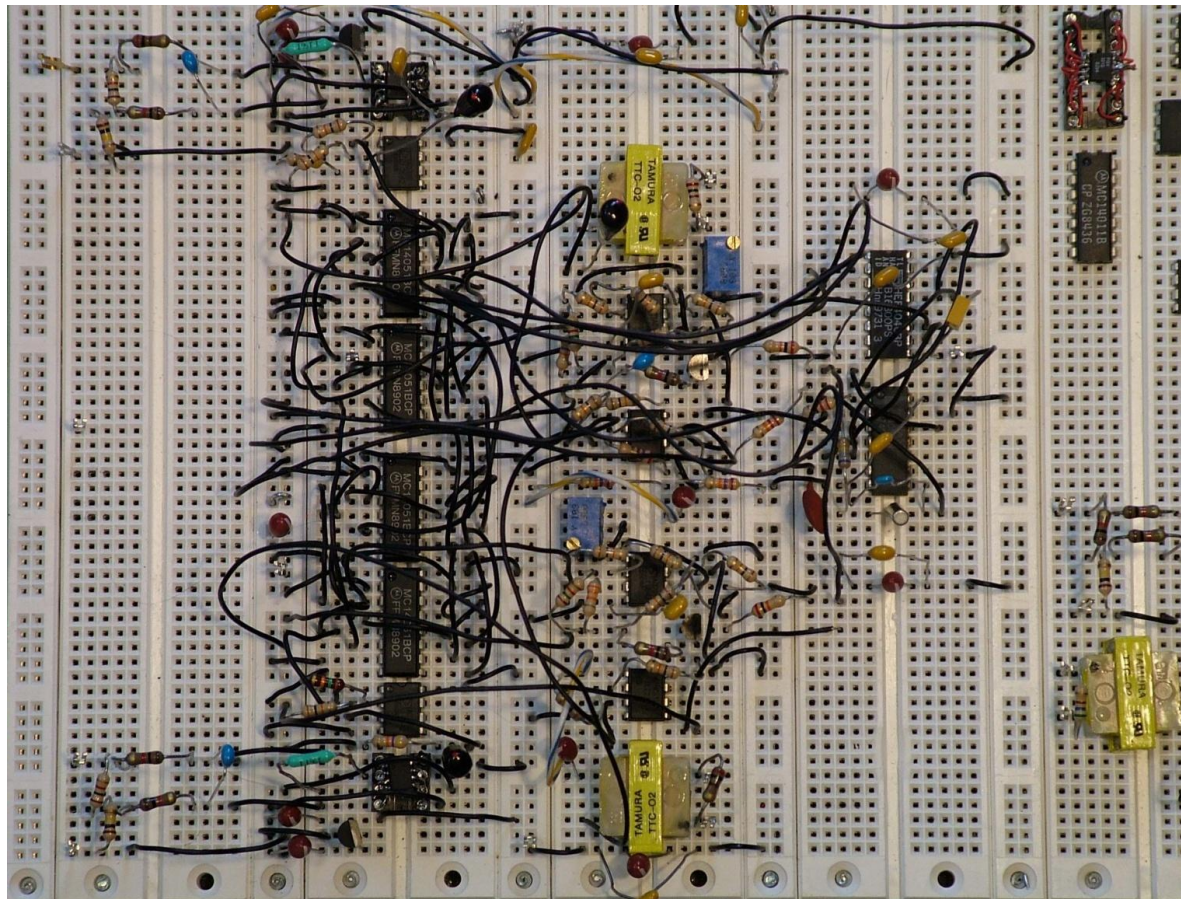
XOR Gate		
입력1	입력2	출력
0	0	0
0	1	1
1	0	1
1	1	0

NOT Gate	
입력	출력
0	1
1	0



# 1.1 AND, OR, XOR, NOT 게이트 사용

- 빵판(Bread-board) : 논리회로

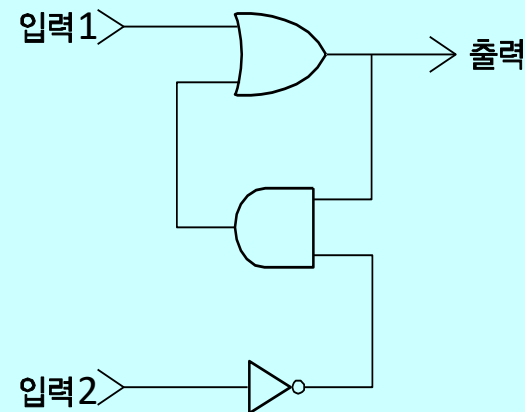




# 1.1 플립플롭(flip-flop)

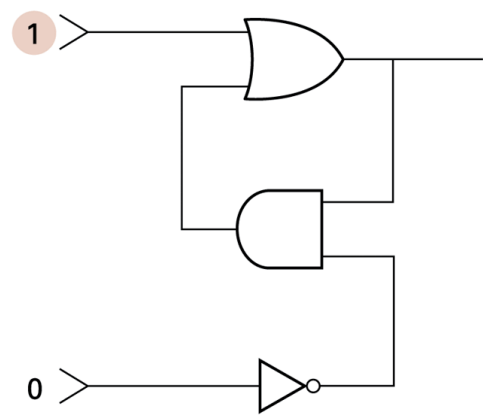
- 플립플롭: 게이트로 만든 회로, 1 비트를 저장할 수 있음
  - 입력신호에 의해서 상태를 바꾸도록 지시할 때까지 현재의 2진상태 유지
  - 저장 값을 1로 설정하는 입력 라인을 가짐 (입력1)
  - 저장 값을 0로 설정하는 입력 라인을 가짐 (입력2)
  - 두 입력 라인 모두 0일 때, 가장 최근의 저장 값을 유지함  
(플립플롭 디자인에 따라서 다름)

- 플립플롭 예:



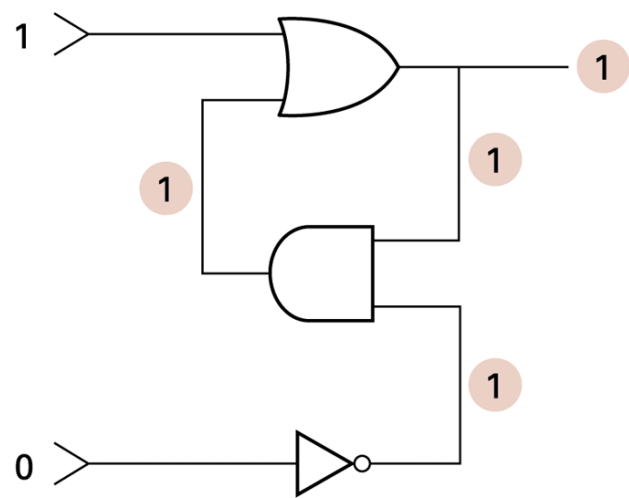
# 1.1 플립플롭의 출력을 1로 설정하기

a. 1 is placed on the upper input.



위쪽 입력에 1을 건다

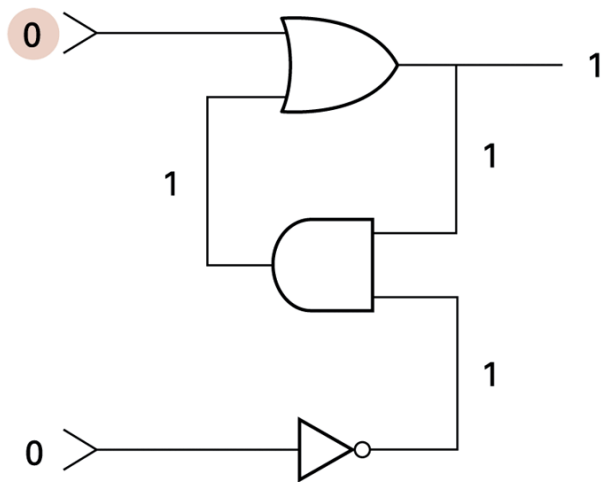
b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.



이 결과로 OR 게이트의 출력이 1이 되며, 이는 다시 AND 게이트의 출력을 1로 만든다.

# 1.1 플립플롭의 출력을 1로 설정하기 (계속)

c. The 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.

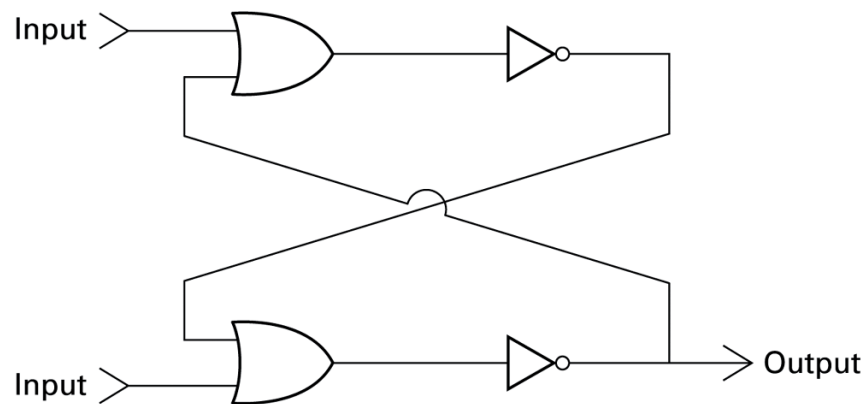


AND 게이트의 출력 1은  
위쪽 입력이 0으로 돌아간 후에도  
OR 게이트의 출력이 변하지 않게 만든다.

# 1.1 플립플롭을 구성하는 또 다른 방법

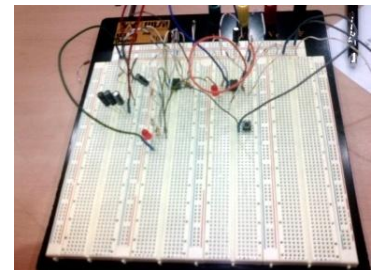
## 1. 플립플롭의 예

RS FlipFlop, D FlipFlop, JK FlipFlop 등



## 2. 메모리, 레지스터 등을 구성하는데 활용

## 3. 자세한 이론, 활용 및 실험은 논리회로 수업에서 학습



## 1.1 16진법(Hexadecimal Notation)

- 16진법: 긴 비트 패턴을 위한 간이 표기법
  - 비트 패턴을 4 비트 그룹들로 분할
  - 각 그룹을 한 개의 기호로 표현

예) 8비트: 10100011 → 1010 0011 → A3

참고) 10진법, 16진법, 8진법, 2진법

# 1.1 16진 인코딩 체계

비트 패턴      16진법 표현

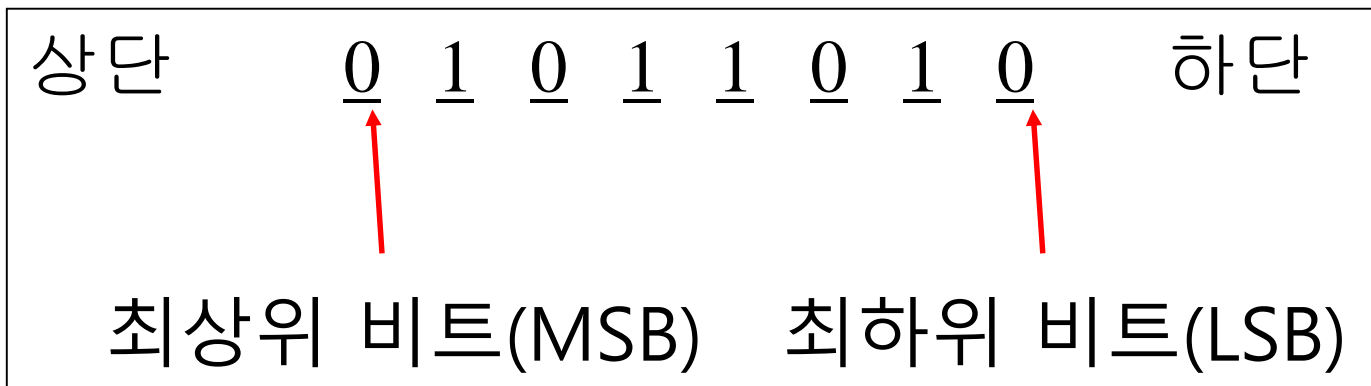
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

비트 패턴      16진법 표현

1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

## 1.2 주기억장치 셀

- 셀(cell): 주기억장치의 단위
  - 대개 한 바이트(byte)에 해당하는 **8 비트**
  - 최상위 비트(Most significant bit): 메모리 셀의 가장 왼쪽(상단) 비트
  - 최하위 비트(Least significant bit): 메모리 셀의 가장 오른쪽(하단) 비트
- 바이트 크기 메모리 셀의 구성

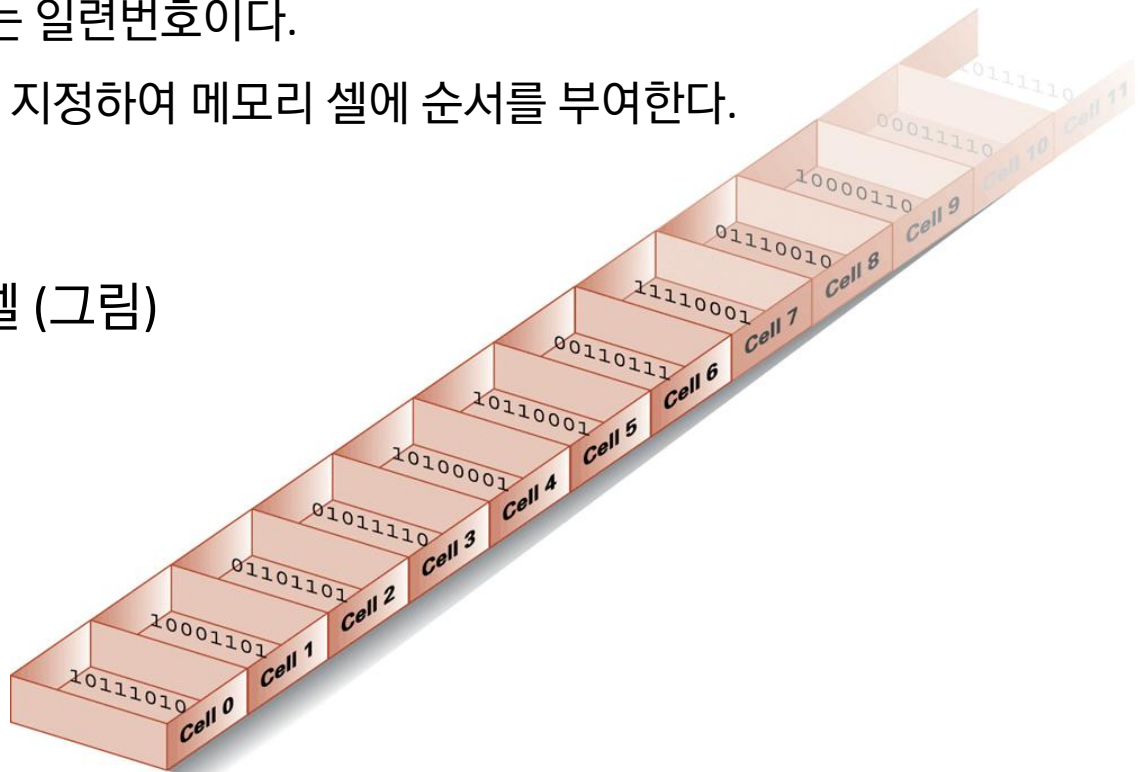


## 1.2 주기억장치 주소 (Address)

- **주소:** 컴퓨터 주기억장치 안의 셀을 식별하는 고유한 이름
  - 이 이름은 실제로는 숫자이다.
  - 이 숫자들은 0에서 시작하는 일련번호이다.
  - 이 방식으로 셀들에 번호를 지정하여 메모리 셀에 순서를 부여한다.

### 2. 주소에 따라 배열된 메모리 셀 (그림)

참고) 8비트 컴퓨터,  
16비트 컴퓨터,  
32비트 컴퓨터의 주소범위





## 1.2 메모리 용어

- RAM(Random Access Memory, 임의 접근 메모리):  
임의의 순서로 셀들에 접근할 수 있는 메모리
- DRAM (Dynamic RAM, 동적 메모리):  
휘발성 (volatile) 메모리로 이루어진 RAM
- ROM(Read Only Memory):  
비휘발성 (nonvolatile) 메모리로 임의 Writing이 불가능  
접근 속도가 느림  
예) ROM, PROM, EPROM – CMOS(BIOS)에 활용

## 1.2 메모리 용량의 측정 단위

- Kilobyte:  $2^{10}$  바이트 = 1024 바이트
  - Example: 3 KB =  $3 \times 1024$  바이트
  - “kilo” 대신 “kibi”를 사용하기도 함
- Megabyte:  $2^{20}$  바이트 = 1,048,576 바이트
  - Example: 3 MB =  $3 \times 1,048,576$  바이트
  - “mega” 대신 “mebi”를 사용하기도 함
- Gigabyte:  $2^{30}$  바이트 = 1,073,741,824 바이트
  - Example: 3 GB =  $3 \times 1,073,741,824$  바이트
  - “giga” 대신 “gibi”를 사용하기도 함
- Terabyte:  $2^{40}$  바이트 = ? 바이트
  - Example: 1 TB =  $1,024 \times 1$  Gigabyte

## 1.3 대용량 저장장치(Mass Storage)

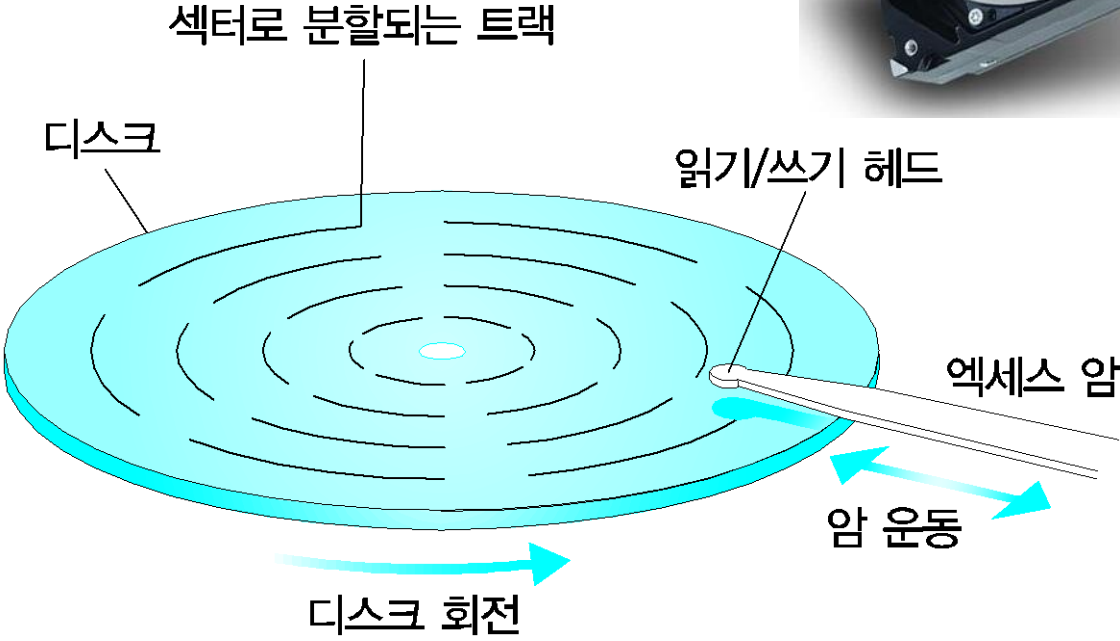
- 온라인 저장장치와 오프라인 저장장치
- 대개 주기억장치보다 크다
- 대개 주기억장치에 비해 휘발성이 낮다
- 대개 주기억장치보다 느리다

## 1.3 대용량 저장장치 종류

- **자기** 저장장치 (Magnetic)
  - 디스크: 3.5inch, 2.5inch 등
  - 테이프: DAT(4mm, 8mm), DLT
- **광학** 장치 (Optical)
  - CD
  - DVD
- **플래시** 드라이브(Flash Drive)
  - MicroSD, Compact Flash (CF), SD, Memory Stick, ...

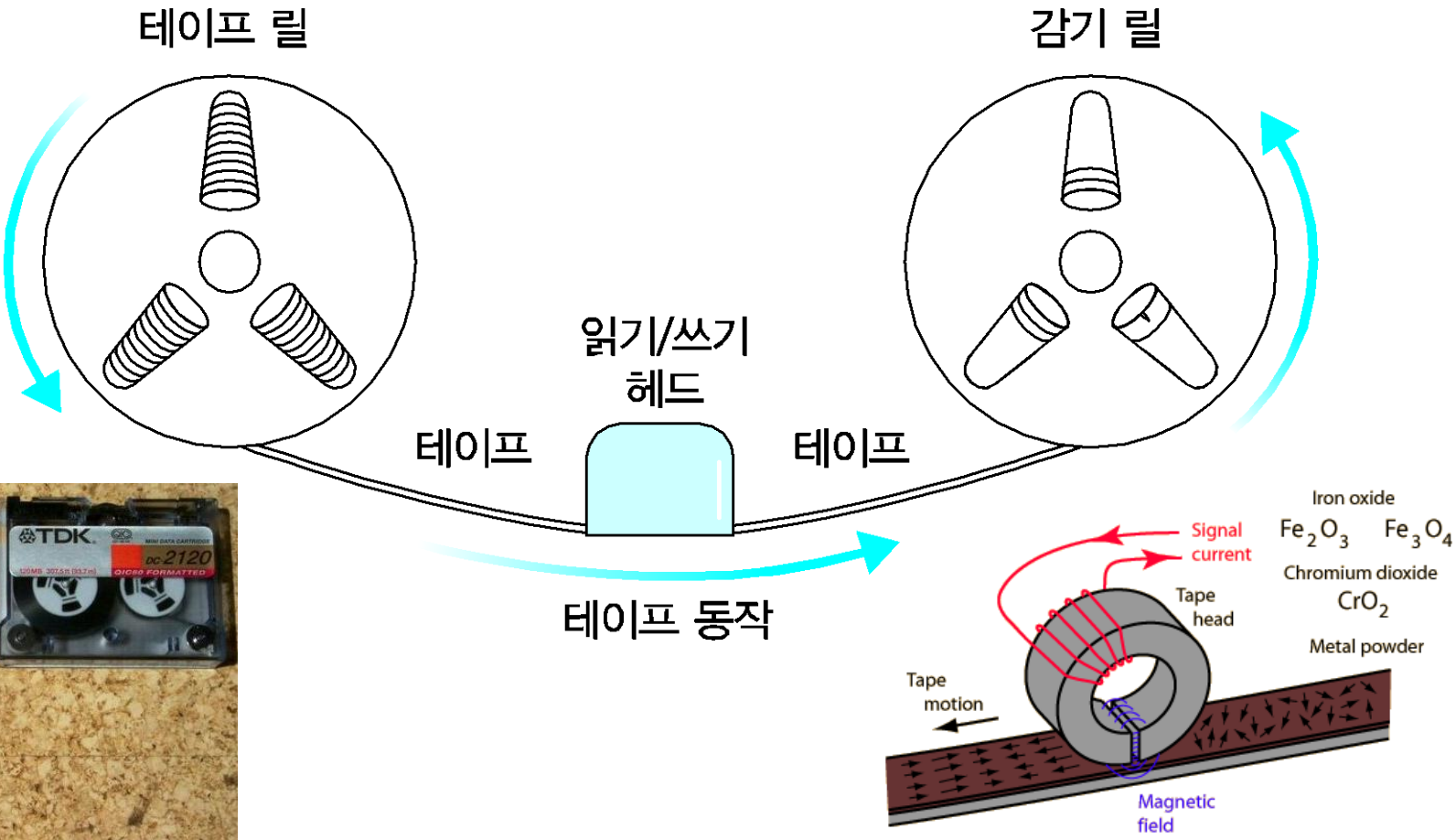
# 1.3 디스크 저장장치

- 디스크 실린더의 구조
  - HDD의 용량 = 실린더 수 \* 섹터수 \* 트랙수
  - HDD 대표 브랜드?



# 1.3 자기테이프 저장장치

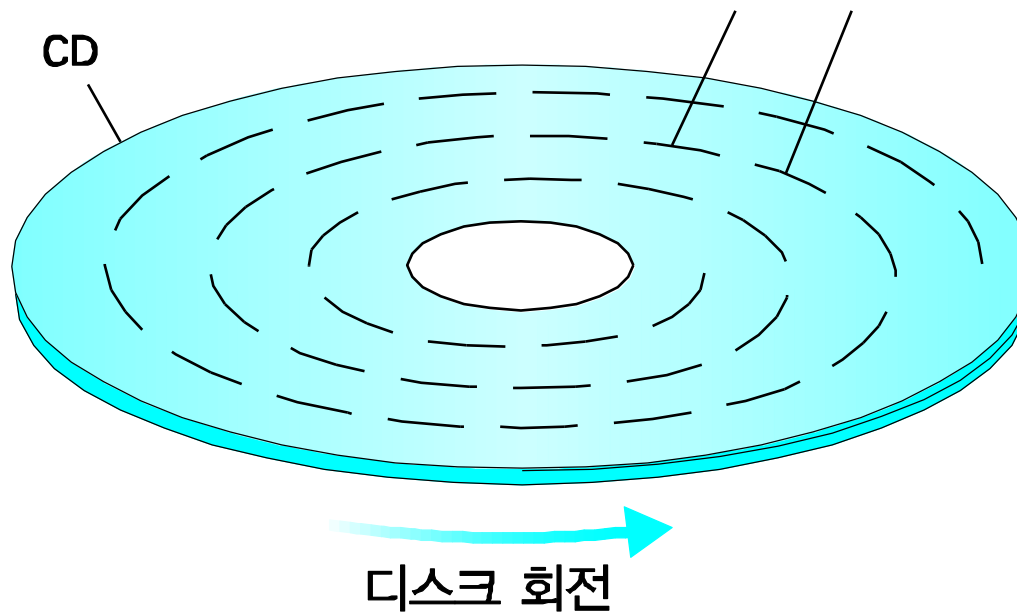
- Magnetic Tape: 대용량 등으로 방송 및 백업 등에 이용



## 1.3 CD 저장장치

- CD-DA: Compact Disk for Digital Audio


바깥쪽으로 향하는 나선형 트랙 하나에  
데이터가 기록되며, 트랙은 섹터들로 이루어진다



참고) CD-R, CD-RW, DVD-R, DVD-RW

# 1.3 파일(file)

- 파일: **대용량 저장장치에서의 데이터 저장 단위**
- 물리적 레코드와 논리적 레코드
- **버퍼(buffer):** (하나의 장치서 다른 장치로 전송 과정에서) 임시로 데이터 보관을 위해 사용되는 메모리 영역
  - 버퍼를 이용하는 과정: 버퍼링(buffering)



Intel Celeron Dual-Core Processor  
**E1200** Specification

제조사	Intel
제품명	Celeron Dual-Core Processor E1200
코어구조	Intel Core-Micro Architecture
소켓방식	LGA775
제품형번	E1200
작동속도	1.6GHz
FSB속도	FSB 800MHz
코어형태	Dual-Core / Conroe
캐시용량	32KB capacity L1 Cache
	Shared 512KB L2 Cache (2-way)
코어배수	x8 Bus/Core ratio
전압범위	Unknown
TDP	65W
온도제원	73.3℃
제품특징	MMX, SSE3, xD, C1E, EIST, Intel Thermal Monitor 2



## 2.1 텍스트의 표현

- (문자, 구두점 등등의)  
각 글자에는 고유한 비트 패턴이 할당
  - **ASCII (American Standard Code for Information Interchange)**: 영문 텍스트에서 사용되는 기호를 표현하기 위한 **7 비트 패턴**
  - **Unicode**: 세계 각국 언어에서 사용되는 주요 기호들을 표현하기 위한 **16 비트 패턴 (2바이트)**
  - **ISO (International Standard Organization) 표준**: 세계 각국 언어에서 사용되는 대부분의 기호들을 표현하기 위한 **32 비트 패턴**

# 2.1 ASCII 코드표

- 128개의 비트 패턴과 문자들을 매핑함

제어 문자		공백 문자		구두점		숫자		알파벳			
10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	\$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(	72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29	)	73	0x49	I	105	0x69	i
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[	123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	\	124	0x7C	
29	0x1D	GS	61	0x3D	=	93	0x5D	]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

## 2.1 메시지 “Hello.”의 ASCII 표현

01001000 01100101 01101100 01101100 01101111 00101110

H e l l o .

## 2.1 숫자의 표현

- 2진법: 기수 2로 숫자를 표현하기 위해 비트들을 사용함
- 컴퓨터에서 숫자 표현의 한계
  - 오버플로우(overflow): 너무 큰 값을 표현하려 할 때 발생
  - 언더플로우(underflow): 너무 작은 값을 표현하려 할 때 발생
  - 절삭(truncation): 표현 가능한 두 값 사이에 존재하는 값에 발생

예) 8비트 정수형: 0 ~ 255 정수 범위만 표현

16비트 정수형: 0 ~ 65535 정수 범위만 표현

## 2.1 이미지의 표현 – 이미지 chapter에서

- 비트맵(bitmap) 기법
  - 픽셀(pixel): “picture element”의 줄임말 (화소)
  - RGB (Red, Green, Blue)
  - 휘도(luminance)와 색도 차이(chrominance)
- 벡터 기법 (벡터이미지)
  - 크기 변경이 자유롭다
  - TrueType와 PostScript

예) 네비게이션 또는 구글 지도에서의 경계선 영상

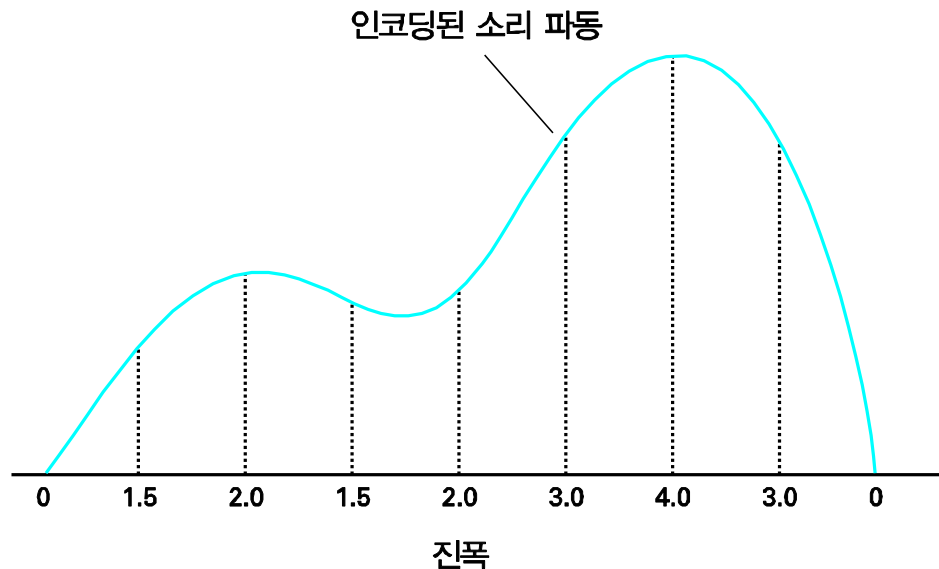
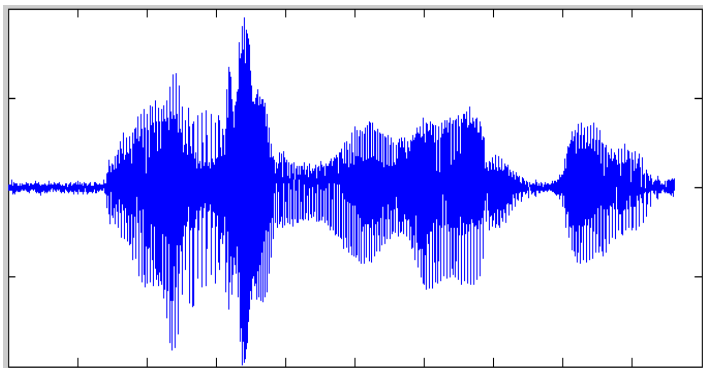


## 2.1 소리의 표현 – 사운드 chapter에서

### ■ 샘플링 기법

- 고품질 리코딩에 사용됨
- 실제 오디오를 기록하기 위하여 사용
- **아날로그 신호의 디지털화** (LP → Wav, CD, MP3 등) !

예) 연속 값 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0 등이 표현하는 소리 파동

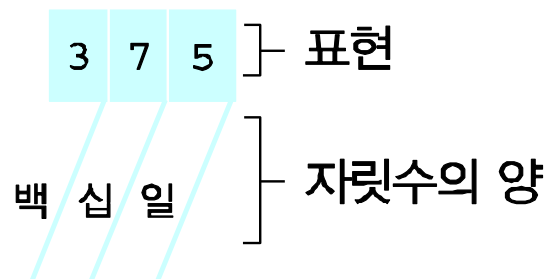


## 2.2 2진 체계

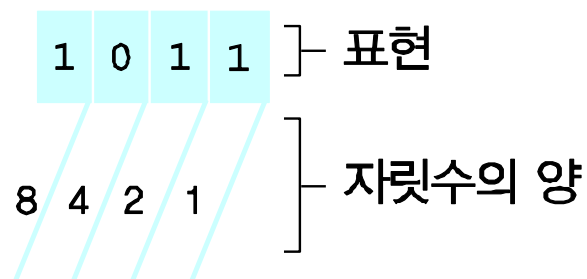
- 전통적인 10진 체계는 10의 멍승에 기초하고 있다.
- 2진 체계는 2의 멍승에 기초하고 있다.

예) 10진수 체계와 2진수 체계

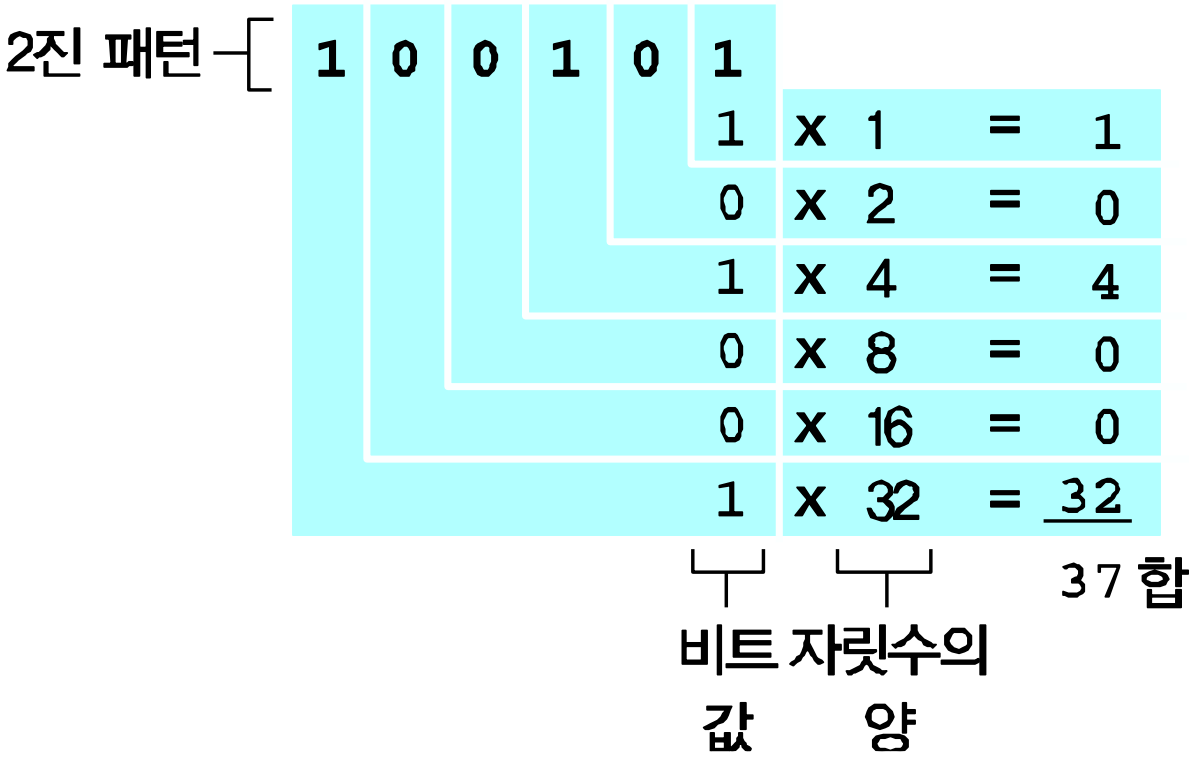
a. 10진수 체계



b. 2진수 체계



# 2.2 2진법 표현 100101의 해석





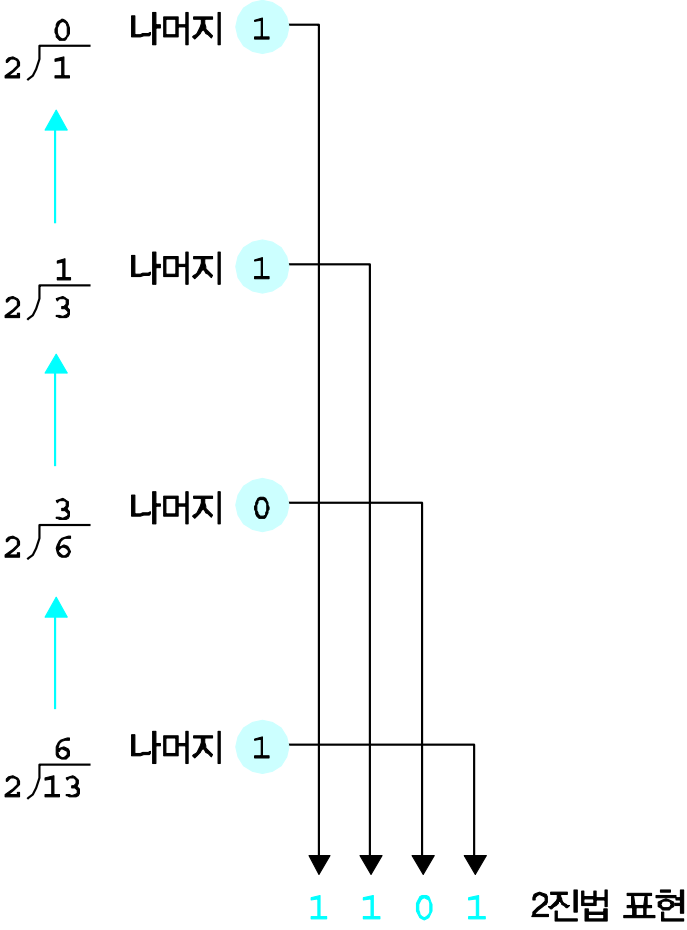
# 2.2 자연수의 2진법 표현을 구하는 알고리즘

단계 1. 주어진 값을 2로 나누고 나머지를 기록한다.

단계 2. 단계 1의 몫이 0이 아니면, 이 몫을 2로 나누고 그 나머지를 기록하는 작업을 계속한다.

단계 3. 이제 몫은 0이 되어있을 것이며, 나머지들을 기록된 순서대로 오른쪽에서 왼쪽으로 나열하면 원래 값의 2진법 표현을 얻는다.

예) 13: 1101, 27 : 11011



## 2.2 2진 덧셈 법칙

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$



# 2.2 소수 포함 10진수의 2진수 변환

- 소수를 포함하는 10진수의 2진수 표현
  - 2진수 변환을 위해 2의 지수 승으로 표현해야 함

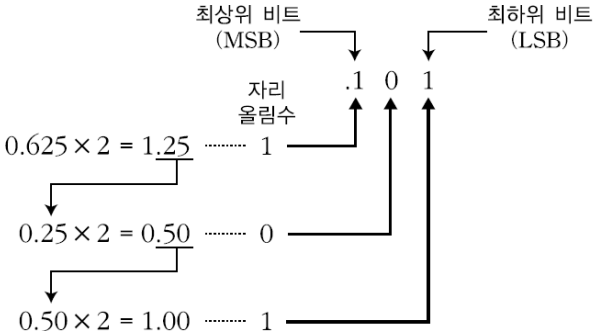
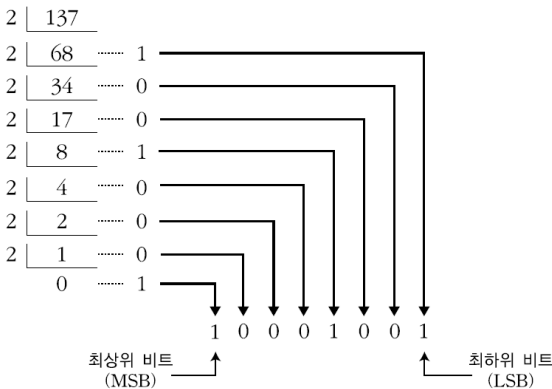
$$(137.625)_{10} = 1 \times 10^2 + 3 \times 10^1 + 7 \times 10^0 + 6 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$
$$= A_m \times 2^m + \dots + A_1 \times 2^1 + A_0 \times 2^0 + A_{-1} \times 2^{-1} + A_{-2} \times 2^{-2} + \dots + A_{-m} \times 2^{-m}$$

- 정수부분은 2로 연속적인 나눗셈을  
소수부분은 2로 연속적인 곱셈을 수행한다.

- 1단계 : 정수부분과 소수부분을 분리한다.
- 2단계 : 정수부분의 10진수를 2진수로 변환한다.

$$(137)_{10} = (10001001)_{10}$$

- 3단계 : 소수부분의 10진수를 2진수로 변환한다.  
 $(0.625)_{10} = (0.101)_2$
- 4단계 : 얻어진 정수와 소수의 2진수를 합한다.  
 $(137.625)_{10} = (10001001)_2 + (0.101)_2 = (10001001.101)_2$



## 2.2 정수의 저장

- 2의 보수(Two's complement) 표기법:  
가장 널리 사용되는 정수 표현 체계
- 초과(excess) 표기법:  
또 다른 정수 표현 체계로 실수 표현에 사용

→ 두 표기법 모두에서 **오버플로우** 문제가 발생할 수 있다.

# 2.2 2의 보수 표기 체계

a. 길이가 3인 패턴을 사용할 경우

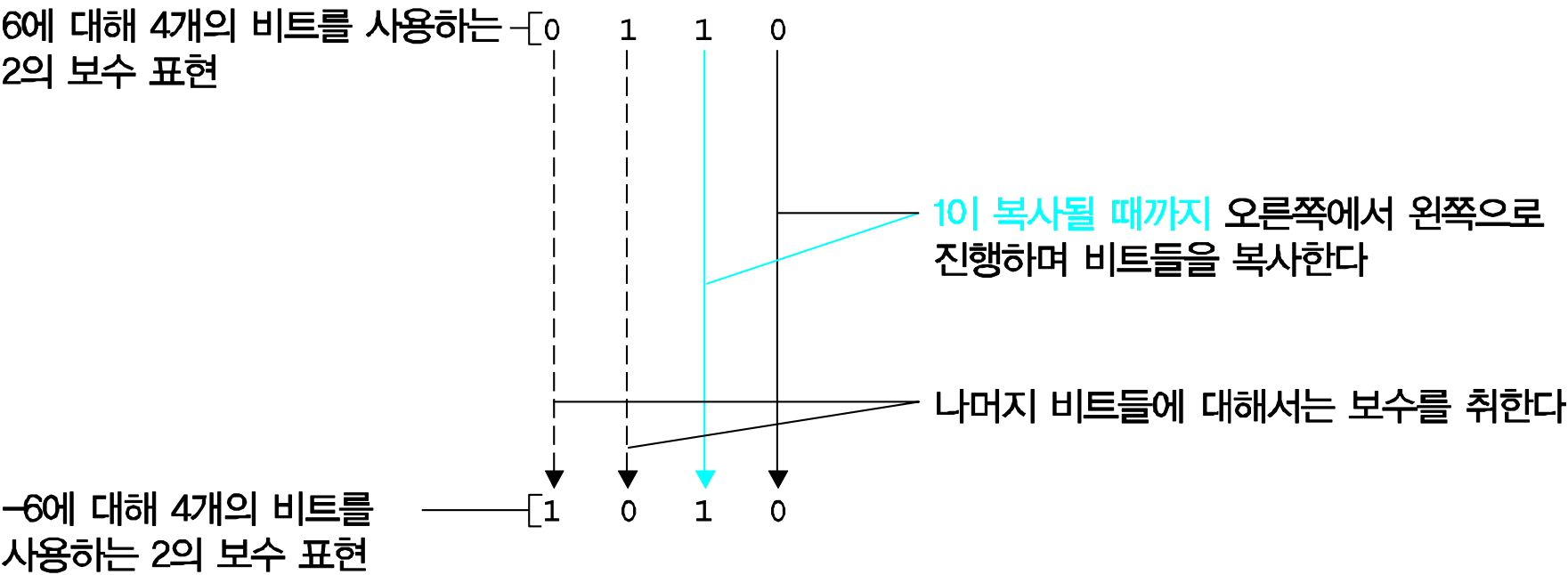
비트 패턴	표현 값
011	3
010	2
001	1
000	0
111	-1
110	-2
101	-3
100	-4

b. 길이가 4인 패턴을 사용할 경우

비트 패턴	표현 값
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

# 2.2 2의 보수 계산 방법

- 4개 비트를 사용해 -6을 2의 보수 표기법으로 인코딩



- 모든 비트를 1 ↔ 0 으로 변환후에 +1 을 수행

# 2.2 2의 보수 표기법으로 변환된 덧셈 문제

10진법을 사용한 문제		2의 보수 표기법으로 표현된 문제		10진법을 사용한 결과 값
$\begin{array}{r} 3 \\ + 2 \\ \hline \end{array}$	→	$\begin{array}{r} 0011 \\ + 0010 \\ \hline 0101 \end{array}$	→	5
$\begin{array}{r} -3 \\ + -2 \\ \hline \end{array}$	→	$\begin{array}{r} 1101 \\ + 1110 \\ \hline 1011 \end{array}$	→	-5
$\begin{array}{r} 7 \\ + -5 \\ \hline \end{array}$	→	$\begin{array}{r} 0111 \\ + 1011 \\ \hline 0010 \end{array}$	→	2



## 2.2 2의 보수 표기법을 통한 수의 표현 범위

- 2진수의 표현 범위

- 2의 보수를 사용한 3비트 이진수 표현의 예

$$+3 = (011)_2$$

$$+2 = (010)_2$$

$$+1 = (001)_2$$

$$+0 = (000)_2$$

$$-1 = (111)_2$$

$$-2 = (110)_2$$

$$-3 = (101)_2$$

$$-4 = (100)_2$$

- 표현할 수 있는 수의 범위는 -4 ~ 3이 된다. 이것은  $-2^{3-1} \sim 2^{3-1}-1$ 로 표현된다.

- n비트 데이터 경우로 일반화할 경우 수의 범위

- $-2^{n-1} \leq N \leq 2^{n-1}-1$

# 3.1 데이터의 압축 – 기본 용어 및 원리만 파악

- 손실 압축(lossy compression)과 무손실 압축(lossless compression)

- RLE(Run-Length Encoding)

예) 10011100 00110000 → 1(1),0(2),1(3),0(4),1(2),0(4)

- 빈도 종속 인코딩(Huffman 코드)

예) abacccddeaaccc ...

(부호)	(빈도)	(비트)
a	5	10
b	2	1111
c	9	0
d	3	110
e	1	1110

- 상대적 인코딩(Relative encoding): 차이값을 인코딩

예) 4 5 6 8 9 → 4 1 1 2 1

- 사전(dictionary) 인코딩 (LZW 인코딩 같은 적응적 사전 인코딩 포함)

## 3.1 이미지의 압축 – 기본 용어 및 원리만 파악

- GIF: 만화에 유리
- JPEG: 사진에 유리
- TIFF: 이미지 보관에 유리

## 3.1 오디오 및 비디오의 압축 – 기본 용어만 파악

- MPEG (Motion Picture Expert Group)
  - 고화질 TV 방송
  - 화상 회의
- MP3 (MPEG Layer 3)
  - 시간적 차폐
  - 주파수 차폐

## 3.2 통신 오류

- 패리티 비트 (짝수 패리티와 홀수 패리티)
  - 검사 바이트 (CRC)
  - 오류 정정 코드
  - 기본적인 아이디어는 비트를 추가로 사용하여 오류를 검출함!
- 구체적인 예제는 **컴퓨터네트워크 / 데이터통신** 교과목에서 학습

# 3.2 문자 A와 F의 홀수 패리티로 조정된 ASCII 코드

