Data Representation

컴퓨터개론

(Introduction to Computer Systems)

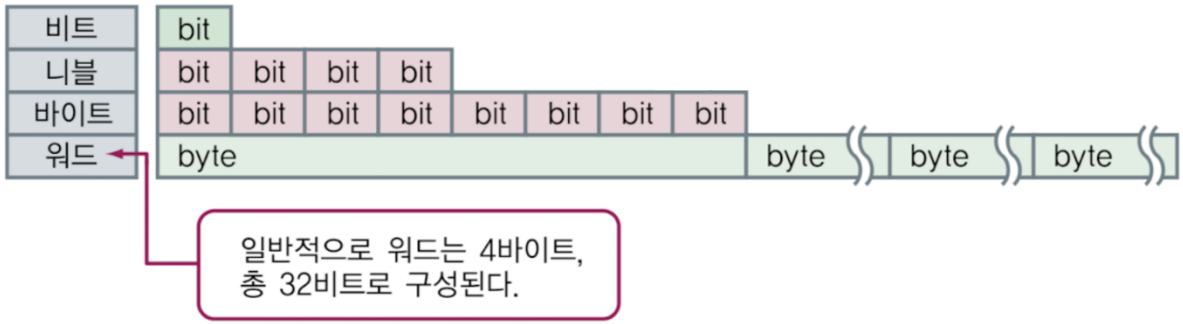
GEN1030

(Recap) 자료 표현 원리

- 컴퓨터에게 가장 적합한 수의 표현은?
- 컴퓨터 내부에는 전기가 흐르거나(**On**) 흐르지 않는(**Off**) 두 가지 전기 신호만을 표현 가능
- 2진수(Binary)
  - 숫자 0과 1만 사용
  - 컴퓨터는 논리의 조합이 간단하고, 사용되는 소자(트랜지스터) 특성상 이진법을 사용하는 것이 효율적
  - 전기가 흐를 경우(On) '참(True)' 의미하는 '1' , 흐르지 않을 경우(Off) '거짓(False)' 의미하는 '0'

(Recap) 정보의 표현

- **Bit** (**Bi**nary digit): 컴퓨터 메모리의 저장 단위 또는 정보 처리 단위 중에서 가장 작은 단위
  - 두 가지 정보만 표현 - 전기의 흐름 상태인 On, Off 표현하는 단위가 Bit
- **Nibble**: 4개의 bit 묶은 단위 (4 bits)
- **Byte**: 8개의 bit 묶은 단위 (8 bits)
- **Word**: 4개 혹은 8개의 byte 묶은 단위
  - 4개는 32 bits, 8개는 64 bits
  - CPU 아키텍쳐(architecture)에 따라 달라짐
  - 32-bit CPU (32 bit를 한번에 처리할 수 있는 CPU)기준 4 byte



(Recap) 8진수, 16진수

- 8진수
  - 0부터 7까지(8개) 이용하여 숫자 표현
- 16진수
  - 16개의 **숫자나 문자**를 이용하여 표현
  - 숫자: 0부터 9까지
  - 문자: A부터 F까지, 각 알파벳은 10~15에 대응
    - 소문자 a~f로도 표현 가능

*컴퓨터 내부에서는 2진수를 사용하지만, 실제로는 16진수로 표현하는 경우 많음

$$
\begin{aligned}
&1\mathrm{AF}_{16} \\
&= 1 \times 16^2 + \mathrm{A} \times 16^1 + \mathrm{F} \times 16^0 \\
&= 256 + 160 + 15 \\
&= 431
\end{aligned}
$$

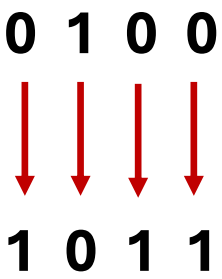
A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

(Recap) 2진수의 음수 표현

- 음수 표현을 위해서는 먼저 Bit 크기를 정하고 수를 표현

- **보수**
  - 쉽게 말하면 보충하는 수
  - 각 자릿수의 수와 보수를 더하면 진수의 자리올림이 발생하고 해당 자릿수는 0이 되도록 하는 수

- **1의 보수(1's complement)**
  - 주어진 이진수의 bit를 0은 1로, 1은 0으로 각각 변환하는 방법
  - 단점: 0이 +0과 -0으로 각각 다르다
  - +0 = 0000 / -0 = 1111



(Recap) 2진수의 음수 표현

- **2의 보수(2's complement)**
- 변환 방법 1

N bit에서 –a의 2의 보수 계산 방법:
2^N - a

예시) 4 bit 사용

-4 = ?

$2^4 - 4$

= 12

= 1100_2

(Recap) 2진수의 음수 표현

- **2의 보수(2's complement)**
- 변환 방법 2
  - 음수의 2의 보수는 1의 보수 값보다 1이 크다는 것을 이용

1단계: 음수의 절대값인 양의 정수의 이진수를 N bit에서 구한다
2단계: 1단계에서 얻은 이진수의 1의 보수를 N bit에서 구한다
3단계: 2단계에서 얻은 이진수에 1을 더한 N bit만을 취한다

예시) 4 bit 사용,
-4 = ?

1단계: 0100

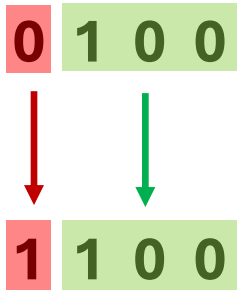
2단계: 1011

3단계: 1011 + 1 = 1100

(Recap) 2진수의 음수 표현

- **2의 보수(2's complement)**
- 변환 방법 3
  - 양수의 N bit 2진수에서 가장 오른쪽의 0에서 처음으로 나오는 1까지 그대로 두고 나머지 왼쪽 bit를 모두 1의 보수로

예시) 4 bit 사용,
-4 = ?



(Recap) Signed, Unsigned 정수

- Signed 정수
  - 양수와 음수를 표현하는데 주로 2진수 및 2의 보수 방법을 이용
  - n개 bit로 정수의 양수, 음수 표현 할 때
    - 수의 범위: $-(2^{n-1})$ ~ $+(2^{n-1}-1)$

- Unsigned 정수
  - 0과 양수만을 다루는 정수 표현
  - n개 bit 일때 수의 범위: ~ $+(2^{n}-1)$
  - ex) 8 bit: 2^{8}(=256)개 정보 표현 가능

컴퓨터의 정보 종류

실수의 표현

- 부동소수점 수(Floating point number)
  - 수의 소수점 위치를 움직일 수 있게 한다는 의미
  - 정수의 표현 방식? 고정소수점 수(Fixed point number)
- 정규화(Normalization)
  - 실수의 표현을 **표준화** ex) $352.45 = 3.5245 \times 10^2$
  - 방법
    - 실수의 소수점을 이동
    - 소수점 왼쪽에 단 하나의 자릿수가 오도록
    - 소수점의 원래 위치는 진수의 지수로 표현
  - 정규화된 실수 표현 → 부동소수

1234.5432	→	1.2345432×10^3
24.345078	→	2.4345078×10^1
0.003045	→	3.045×10^{-3}
-134.784556	→	-1.34784556×10^2

2진수와 부동소수 표현

- 부동소수의 표현: 부호부, 소수부 , 지수부로 구성
  - 부호부(Sign): 0(+) 또는 1(-)
  - 가수(Mantissa): 수의 정밀도(precision) 표현
  - 지수(Exponent): 수의 크기(magnitude) 표현

$$26.625 \rightarrow 11010.101_2 \rightarrow +1.1010101 \times 2^4$$

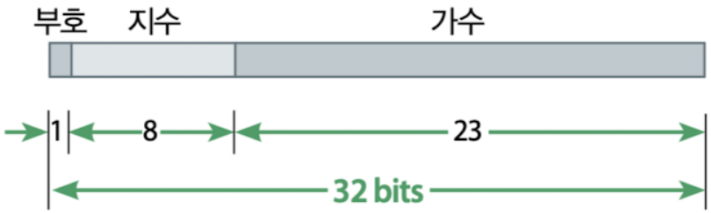
2진수 실수	정규화	부호	지수	가수
101.11	1.0111×2^2	0(+)	2	0111
10011.101	1.0011101×2^4	0(+)	4	0011101
0.001011	1.011×2^{-3}	0(+)	-3	011
-0.00000111	-1.11×2^{-6}	1(-)	-6	11

- **한정된 Bit 수를 이용하여 정밀도를 높게 표시할 수 있음**
- 고정소수점보다 매우 큰 정수, 매우 작은 소수 표현 가능

부동소수의 저장 표현

- 전기전자기술협회(IEEE)에서 부동소수 저장을 위한 표준 정의
  - 단정도, 배정도
  - 부호부, 지수부, 가수부

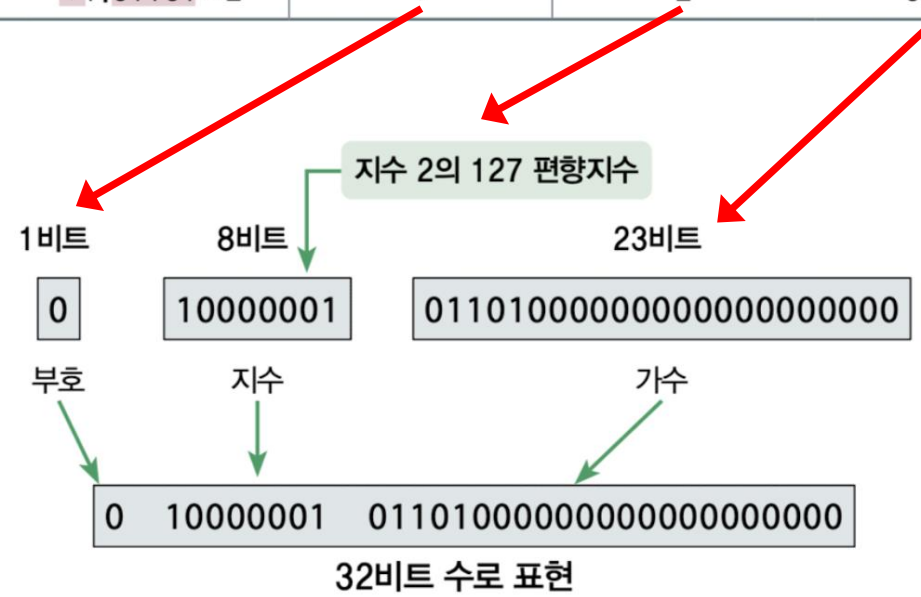
- **단정도 형식(Single precision format)**
  - 32 bit 크기
  - 지수부: 원래의 수에 127 (0111 1111) 더한 2진수 (*127 편향지수*)
    - -126 부터 +127까지
  - 가수부: 부동소수의 가수를 왼쪽부터 저장, 오른쪽 나머지 bit는 0으로 채운 2진수



단정도 예시

- 5.625가 메모리에 저장되는 단정도 형식

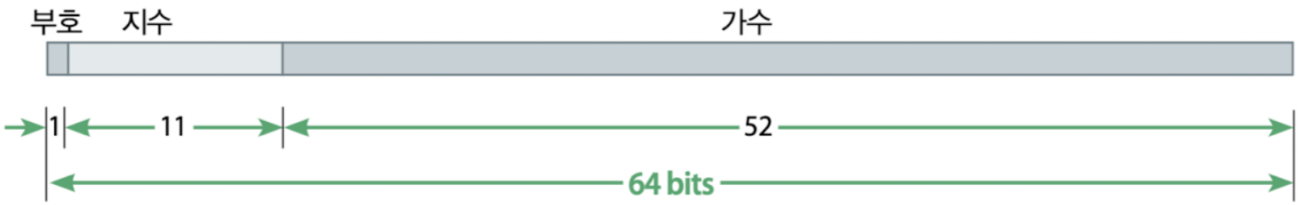
원 실수	정규화	부호	지수	가수
101.101	$+1.01101\times2^{2}$	+	2	01101



부동소수의 저장 표현

- 전기전자기술협회(IEEE)에서 부동소수 저장을 위한 표준 정의
  - 단정도, 배정도
  - 부호부, 지수부, 가수부

- **배정도 형식(Double precision format)**
  - 64 bit 크기 (단정도보다 더 높은 정밀도 제공)
    - 단정도에 비해 지수와 가수의 표현 크기가 큼
  - 지수부: 원래의 수에 1023 (011 1111 1111) 더한 2진수 (*1023 편향지수*)
    - -1022 부터 +1023까지
  - 가수부: 부동소수의 가수를 왼쪽부터 저장, 오른쪽 나머지 bit는 0으로 채운 2진수



127 편향지수, 1023 편향지수

- 편향지수를 왜 사용할까?
- 부동소수 표현에서 지수는 음수(-)와 양수(+) 모두 필요
  - 1.101×2^{4} 또는 1.101×2^{-2}
  - 컴퓨터 하드웨어는 양수만 있을 시 더 빠르게 계산 가능

- **편향지수(Biased Exponent)**
  - 저장하려는 지수 값에 "Bias"를 더하여 저장
  - 즉, 음수 지수를 모두 양수로 변환하여 저장
  - 숫자 표현 시 필요한 정확도와 범위가 다름

$$\text{Bias} = 2^{k-1} - 1 \quad (k\text{는 지수 bit 수})$$

  - 단정도 지수 = 8 bit → Bias = $2^{8-1} - 1 = 127$
  - 배정도 지수 = 11 bit → Bias = $2^{11-1} - 1 = 1023$

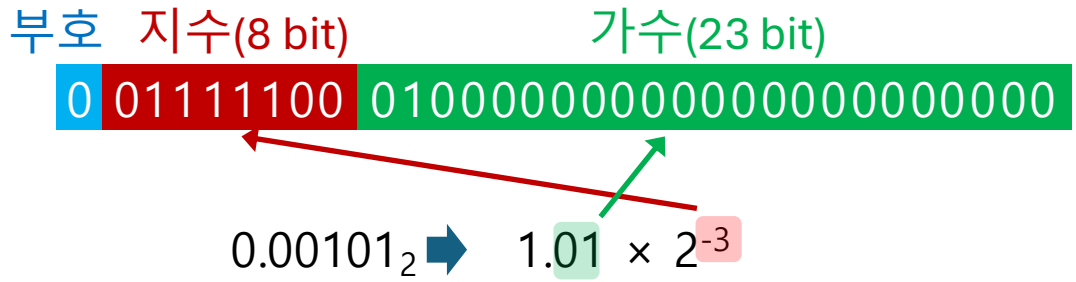
127 편향지수, 1023 편향지수

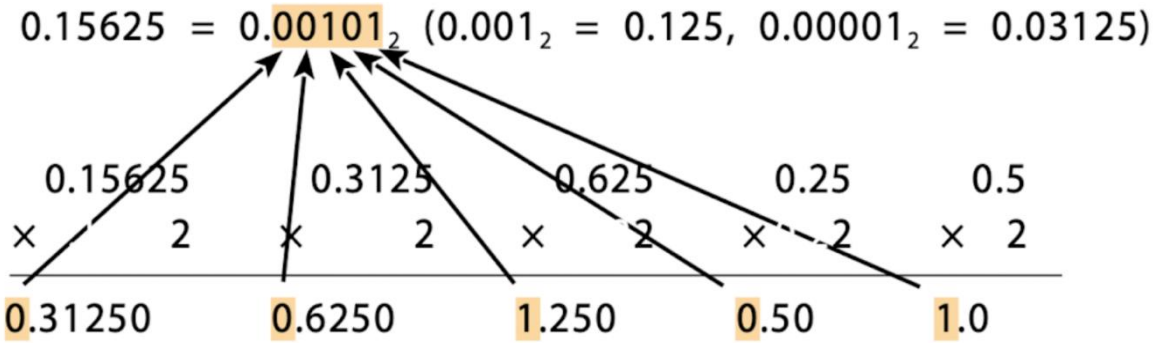
- 지수부에 이용되는 127 편향지수와 1023 편향지수

원래의 수	127 편향지수		1023 편향지수	
	10진수	2진수(8비트)	10진수	2진수(11비트)
−1022	−		1	000 0000 0001
…	−		…	
−127	−		896	011 0000 0001
−126	1	0000 0001	897	011 0000 0010
…	…		…	
−2	125	0111 1101	1021	011 1111 1101
−1	126	0111 1110	1022	011 1111 1110
0	127	0111 1111	1023	011 1111 1111
1	128	1000 0000	1024	100 0000 0000
2	129	1000 0001	1025	100 0000 0001
…	…		…	
127	254	1111 1110	1150	100 0111 1110
128	−		1151	100 0111 1111
…	−		…	
1023	−		2046	111 1111 1110

부동소수 저장 표현 예시

- 10진수 0.15625 → 2진수 단정도 형식으로



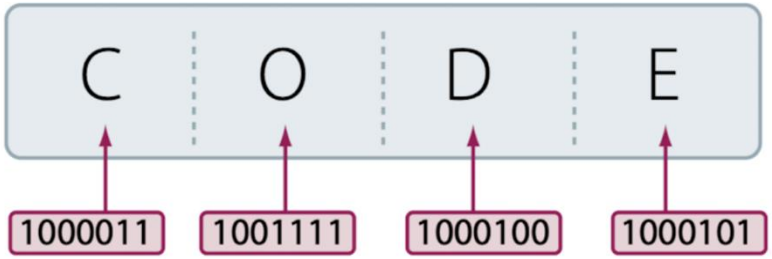


문자, 논리 표현

문자와 코드표

- 컴퓨터에서 문자는 하나의 정해진 수로 표현

- 영문자
  - 7 bit 조합



  - n비트를 사용하면 총 2^n개의 서로 다른 문자 표현이 가능
  - 각각의 조합에 일정한 문자를 할당하여 지정한 것을 문자 코드(Code)

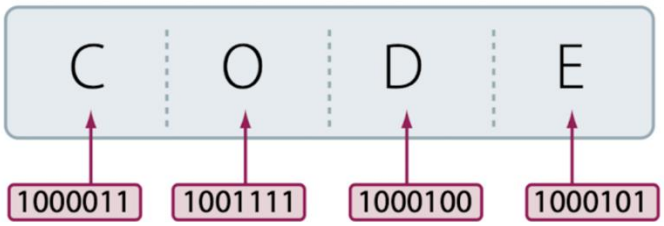
- 국제 표준 문자 코드
  - 아스키(ASCII)
  - 유니코드(Unicode)

아스키 코드

- ASCII (American Standard Code for Information Interchange)
  - 미국 표준협회에서 국제적인 표준으로 정한 문자 코드 체계
  - 7 bit 사용해 128개의 문자, 숫자, 특수문자 코드 규정

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
010	Space	!	"	#	$	%	&	'	(	)	*	1	,	–	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	5	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

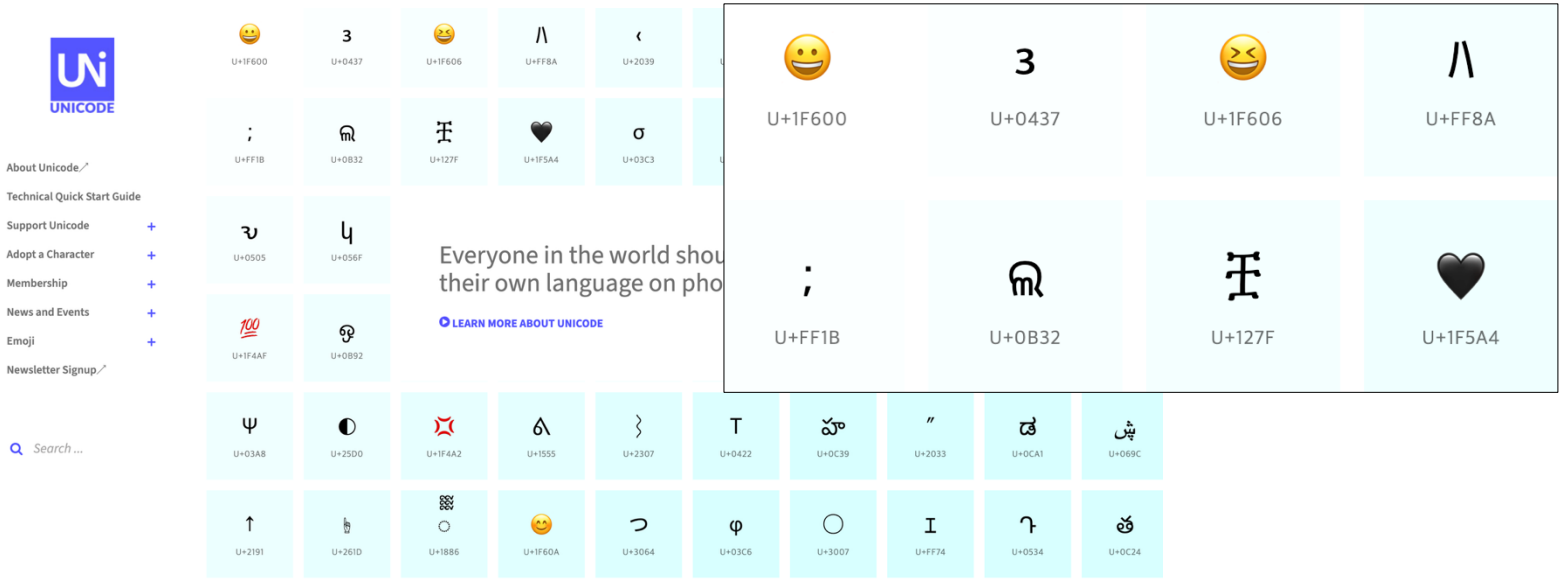
- 실제로 메모리에 저장될 때 한 문자는 8 bit (1 byte)에 저장



유니코드

- 아스키 문제?
  - 128개의 문자만 표현 가능 (7 bit)
  - 영어 알파벳, 숫자, 기본 기호 정도 표현

- 유니코드(Unicode)
  - 전 세계 모든 언어를 하나의 코드 체계로 통합하기 위해 제안
  - 문자에 번호(code point) 부여



유니코드와 UTF-8

- UTF-8
  - Unicode Transformation Format
  - 유니코드를 위한 가변길이 인코딩(variable-length encoding) 방식
  - 문자마다 사용하는 byte 수가 다름 (1~4 byte 사용)
    - 영어 - 1 byte, 한글 - 3 byte
  - 1 byte 영역은 ASCII와 호환
  - 웹에서 많이 사용됨

Unicode	Character	UTF-8 (hex)
U+0041	A	41
U+0042	B	42
U+0043	C	43

논리 표현 및 연산

- 논리값
  - 참(True)과 거짓(False) 의미하는 두 가지 정보
  - 하나의 Bit 정보도 1과 0 → 각각 True, False에 대응

- 논리연산
  - 논리값을 가지고 계산하는 연산
  - 단항연산자
    - 하나의 항이 연산에 참여
    - ex) YES (혹은 BUFFER), NOT
  - 이항연산자
    - 두 개의 항이 연산에 참여
    - ex) AND, OR

논리 연산 종류

- 단항연산자
  - **YES**(or BUFFER): 입력을 그대로 전달
  - **NOT**: 입력값의 반대로 출력

진리표

YES

Input A	Out
0	0
1	1

NOT

Input A	Out
0	1
1	0

논리 연산 종류

- 이항연산자
  - **AND**: 입력이 둘 다 1일 때만 1 출력
  - **OR**: 입력 둘 중 하나라도 1이면 1 출력
  - **NAND**: AND의 반대
  - **NOR**: OR의 반대
  - **XOR**: 두 입력이 서로 다를 때 1 출력
  - **XNOR**: XOR의 반대

AND

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

NAND

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

XOR

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

논리 연산

- 논리연산 법칙들
  - NOT(논리부정): ′로 표시
  - AND(논리곱): · 로 표시
  - OR(논리합): + 로 표시

① 항등 법칙	$0 + x = x,\ \ 1 + x = 1$	$0 \cdot x = 0,\ \ 1 \cdot x = x$
② 동일 법칙	$x + x = x$	$x \cdot x = x$
③ 보수 법칙	$x + x' = 1$	$x \cdot x' = 0$
④ 부정의 부정	$(x')' = x$	
⑤ 교환 법칙	$x \cdot y = y \cdot x$	$x + y = y + x$
⑥ 결합 법칙	$x + (y + z) = (x + y) + z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
⑦ 분배 법칙	$x \cdot (y + z) = x \cdot y + x \cdot z$	$x + y \cdot z = (x + y) \cdot (x + z)$
⑧ 드모르강 법칙	$(x + y)' = x' \cdot y'$	$(x \cdot y)' = x' + y'$
⑨ 흡수 법칙	$x + (x \cdot y) = x$	$x \cdot (x + y) = x$
흡수 법칙 증명	$x + (x \cdot y) = x \cdot 1 + (x \cdot y)$ $= x \cdot (1 + y)$ $= x \cdot 1$ $= x$	$x \cdot (x + y) = (x + 0) \cdot (x + y)$ $= x + (0 \cdot y)$ $= x + 0$ $= x$

논리 함수

- 논리함수(Logical function)
  - 입력변수와 결과값이 모두 0 또는 1만 가짐

- 함수 y = f(x1, x2, x3, ...)
  - 입력변수인 xi가 모두 0 또는 1의 값 만을 허용
  - 결과값인 y 역시 0 또는 1만을 가짐

논리 함수

- 논리 함수 진리표 사용한 논리 함수 표현/간소화 예시

행	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
1	0	0	0	1
2	0	0	1	1
3	0	1	0	0
4	0	1	1	0
5	1	0	0	0
6	1	0	1	1
7	1	1	0	0
8	1	1	1	0

1행: $x_1' \cdot x_2' \cdot x_3'$

2행: $x_1' \cdot x_2' \cdot x_3$

6행: $x_1 \cdot x_2' \cdot x_3$

$$f(x_1, x_2, x_3) = x_1' \cdot x_2' \cdot x_3' + x_1' \cdot x_2' \cdot x_3 + x_1 \cdot x_2' \cdot x_3$$

$$= x_1' \cdot x_2' \cdot (x_3' + x_3) + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{분배법칙}$$

$$= x_1' \cdot x_2' \cdot 1 + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{보수법칙}$$

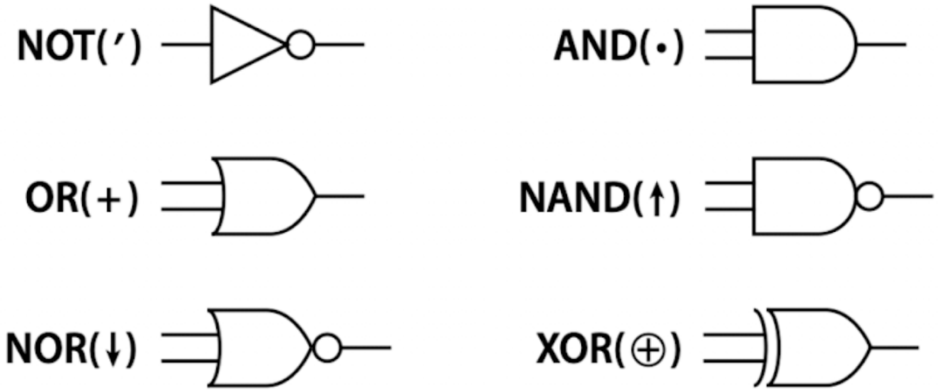
$$= x_1' \cdot x_2' + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{항등법칙}$$

$$= x_2' \cdot (x_1' + x_1 \cdot x_3) \quad \leftarrow \text{분배법칙}$$

논리 회로

- 논리 연산자들은 논리 게이트(Logic gate)라는 물리적인 장치로 구현될 수 있음

- 논리 게이트 기호



논리 회로

- 논리 회로(Logic circuit)
  - 논리 함수 결과 얻기 위해 논리 연산 결과를 수행하여 출력값을 내보내는 물리적 장치

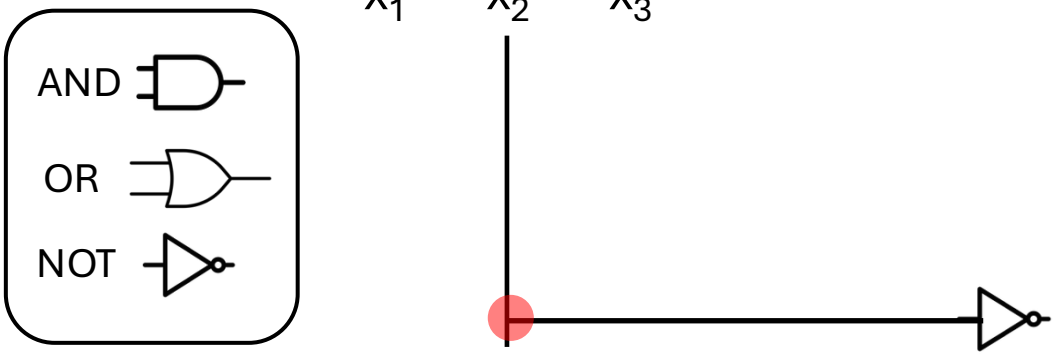
$$f(x_1, x_2, x_3) = x_1' \cdot x_2' \cdot x_3' + x_1' \cdot x_2' \cdot x_3 + x_1 \cdot x_2' \cdot x_3$$

$$= x_1' \cdot x_2' \cdot (x_3' + x_3) + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{분배법칙}$$

$$= x_1' \cdot x_2' \cdot 1 + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{보수법칙}$$

$$= x_1' \cdot x_2' + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{항등법칙}$$

$$= x_2' \cdot (x_1' + x_1 \cdot x_3) \quad \leftarrow \text{분배법칙}$$



논리 회로

- 논리 회로(Logic circuit)
  - 논리 함수 결과 얻기 위해 논리 연산 결과를 수행하여 출력값을 내보내는 물리적 장치

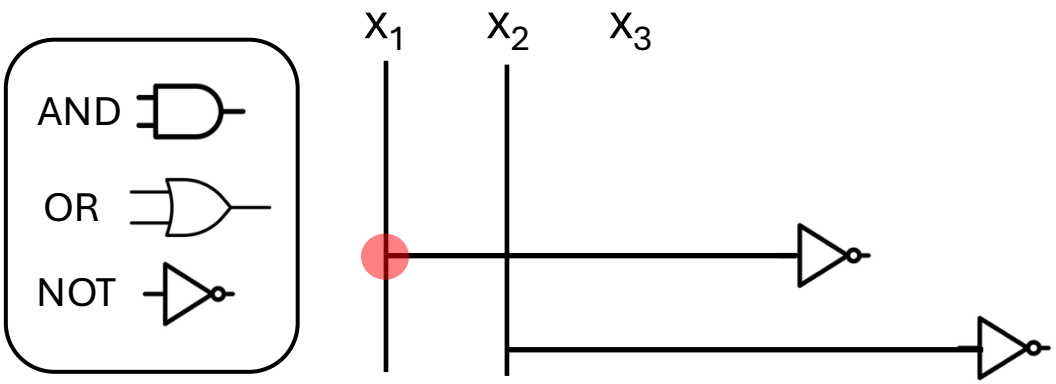
$$f(x_1, x_2, x_3) = x_1' \cdot x_2' \cdot x_3' + x_1' \cdot x_2' \cdot x_3 + x_1 \cdot x_2' \cdot x_3$$

$$= x_1' \cdot x_2' \cdot (x_3' + x_3) + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{분배법칙}$$

$$= x_1' \cdot x_2' \cdot 1 + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{보수법칙}$$

$$= x_1' \cdot x_2' + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{항등법칙}$$

$$= x_2' \cdot (x_1' + x_1 \cdot x_3) \quad \leftarrow \text{분배법칙}$$



논리 회로

- 논리 회로(Logic circuit)
  - 논리 함수 결과 얻기 위해 논리 연산 결과를 수행하여 출력값을 내보내는 물리적 장치

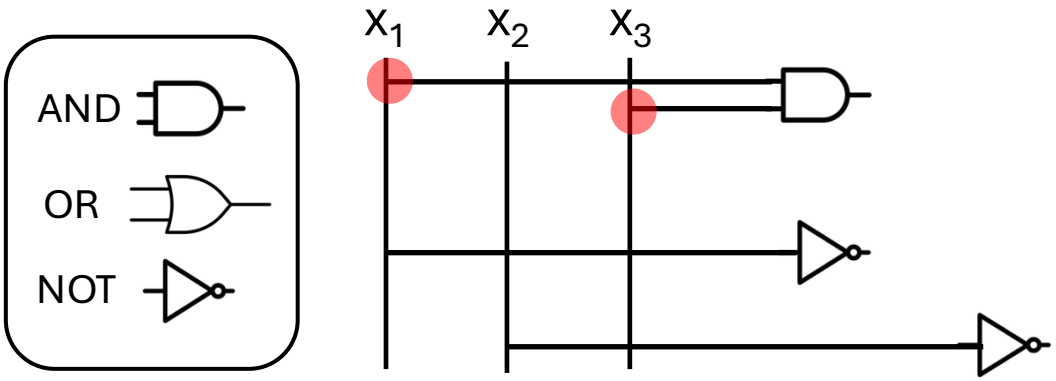
$$f(x_1, x_2, x_3) = x_1' \cdot x_2' \cdot x_3' + x_1' \cdot x_2' \cdot x_3 + x_1 \cdot x_2' \cdot x_3$$

$$= x_1' \cdot x_2' \cdot (x_3' + x_3) + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{분배법칙}$$

$$= x_1' \cdot x_2' \cdot 1 + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{보수법칙}$$

$$= x_1' \cdot x_2' + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{항등법칙}$$

$$= x_2' \cdot (x_1' + x_1 \cdot x_3) \quad \leftarrow \text{분배법칙}$$



논리 회로

- 논리 회로(Logic circuit)
  - 논리 함수 결과 얻기 위해 논리 연산 결과를 수행하여 출력값을 내보내는 물리적 장치

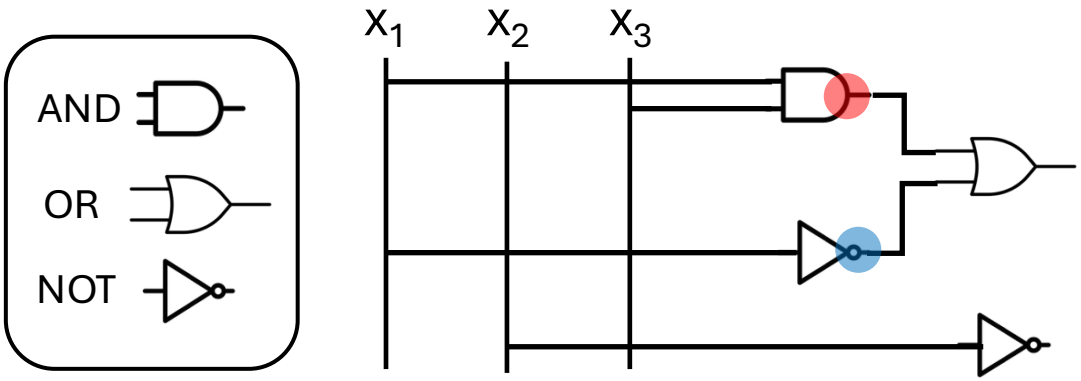
$$f(x_1, x_2, x_3) = x_1' \cdot x_2' \cdot x_3' + x_1' \cdot x_2' \cdot x_3 + x_1 \cdot x_2' \cdot x_3$$

$$= x_1' \cdot x_2' \cdot (x_3' + x_3) + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{분배법칙}$$

$$= x_1' \cdot x_2' \cdot 1 + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{보수법칙}$$

$$= x_1' \cdot x_2' + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{항등법칙}$$

$$= x_2' \cdot (x_1' + x_1 \cdot x_3) \quad \leftarrow \text{분배법칙}$$



논리 회로

- 논리 회로(Logic circuit)
  - 논리 함수 결과 얻기 위해 논리 연산 결과를 수행하여 출력값을 내보내는 물리적 장치

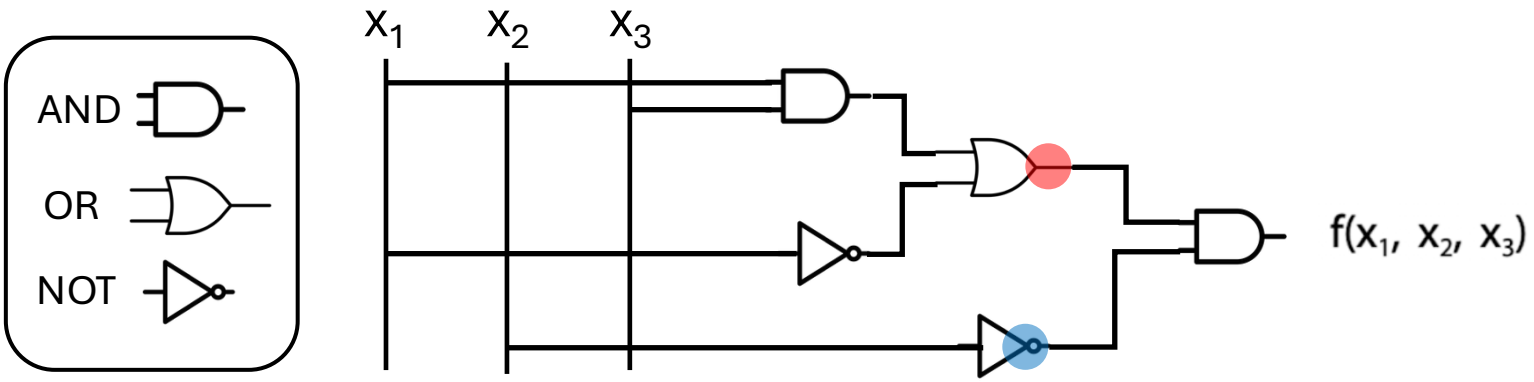
$$f(x_1, x_2, x_3) = x_1' \cdot x_2' \cdot x_3' + x_1' \cdot x_2' \cdot x_3 + x_1 \cdot x_2' \cdot x_3$$

$$= x_1' \cdot x_2' \cdot (x_3' + x_3) + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{분배법칙}$$

$$= x_1' \cdot x_2' \cdot 1 + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{보수법칙}$$

$$= x_1' \cdot x_2' + x_1 \cdot x_2' \cdot x_3 \quad \leftarrow \text{항등법칙}$$

$$= x_2' \cdot (x_1' + x_1 \cdot x_3) \quad \leftarrow \text{분배법칙}$$



Summary

- 컴퓨터에서 수 표현
  - 2진법(binary)
  - 16진법(hexadecimal)
  - 실수의 표현 – 부동소수점 수
- 컴퓨터에서 문자 표현
  - ASCII
  - Unicode
- 논리 연산, 논리 회로