

## 2 과목 | 전자계산기 구조

19.3, 18.3, 17.5, 15.5, 14.3, 13.6, 12.5, 11.3, 10.9, 07.3, 06.5, 05.5, 04.9, ...

### 핵심

#### 055 불대수의 기본 공식



040122

- 교환법칙 :  $A + B = B + A$ ,  $A \cdot B = B \cdot A$
- 결합법칙 :  $A + (B + C) = (A + B) + C$ ,  $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
- 분배법칙 :  $A \cdot (B + C) = A \cdot B + A \cdot C$ ,  $A + B \cdot C = (A + B) \cdot (A + C)$
- 멱등법칙 :  $A + A = A$ ,  $A \cdot A = A$
- 보수법칙 :  $A + A^- = 1$ ,  $A \cdot A^- = 0$
- 항등법칙 :  $A + 0 = A$ ,  $A + 1 = 1$ ,  $A \cdot 0 = 0$ ,  $A \cdot 1 = A$
- 드모르강 :

$$\overline{A+B} = \overline{A} \cdot \overline{B}, \quad \overline{A \cdot B} = (\overline{A} + \overline{B})$$

$$\overline{\overline{A}} = A$$

- 복원법칙 :  $\overline{\overline{A}} = A$

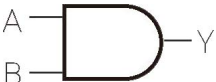

20.6, 18.8, 18.4, 17.3, 14.5, 13.8, 12.8, 07.9, 06.5, 05.5, 04.9, 03.8, 02.5, ...







### 핵심

#### 056 논리 게이트



040124

게이트	기호	의미	진리표	논리식															
AND		입력신호가 모두 1일 때 1 출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	$Y = A \cdot B$ $Y = AB$
A	B	Y																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OR		입력신호 중 1개만 1 이어도 1 출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	$Y = A + B$
A	B	Y																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
NOT		입력된 정보를 반대		$Y = A'$															

		로 변환하여 출력	<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Y	0	1	1	0	$Y = A_{\neg}$									
A	Y																		
0	1																		
1	0																		
BUFFER		입력된 정보를 그대로 출력	<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	Y	0	0	1	1	$Y = A$									
A	Y																		
0	0																		
1	1																		
NAND		NOT + AND, 즉 AND의 부정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	$Y = A \cdot B$ $Y = AB$
A	B	Y																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOR		NOT + OR, 즉 OR의 부정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	$Y = A + B$
A	B	Y																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	
XOR		입력신호가 모두 같으면 0, 한 개라도 틀리면 1출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	$Y = A \oplus B$ $Y = A \bar{B} + \bar{A} B$
A	B	Y																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
XNOR		NOT + XOR, 즉 XOR의 부정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1	$Y = A \odot B$ $Y = \overline{A \oplus B}$ $Y = \overline{AB} + \overline{\overline{AB}}$
A	B	Y																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

19.8, 16.3, 13.8, 12.8, 12.3, 08.9, 05.3, 02.3

**핵심****057 조합논리회로와 순서논리회로**

- 조합논리회로는 임의의 시간에서의 출력이 이전의 입력에는 관계없이 현재의 입력 조합(0 또는 1)으로부터 직접 결정되는 논리회로이다. 이해 반해 순서논리회로는 외부로부터의 입력과 현재 상태에 따라 출력이 결정된다.
- **조합논리회로의 종류** : 반가산기, 전가산기, 병렬가산기, 반감산기, 전감산기, 디코더, 인코더, 멀티플렉서, 디멀티플렉서, 다수결회로, 비교기 등
- **순서논리회로의 종류** : 플립플롭, 레지스터, 카운터, RAM, CPU 등

15.3, 12.3, 11.6, 10.5, 08.9, 07.5, 05.9, 04.5, 02.9, 01.6

**핵심****058 반가산기(HA; Half Adder)**

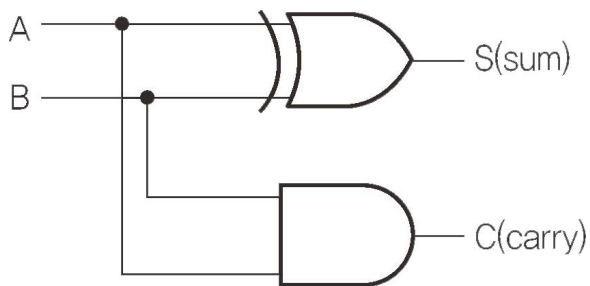
040126

1Bit짜리 2진수 2개를 덧셈한 합(S)과 자리올림 수(C)를 구하는 회로이다.

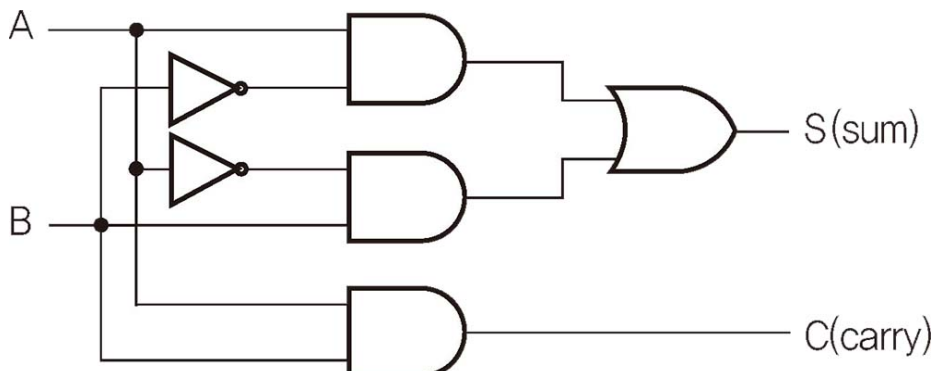
**진리표**

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

**논리식** :  $\text{Carry} = A \cdot B$   $\text{Sum} = A_{\neg} \cdot B + A \cdot B_{\neg} = A \oplus B$

**논리회로**

**Sum에 해당하는 X-OR 게이트를 풀어서 그린 반가산기**



20.6, 19.4, 18.3, 17.5, 15.8, 15.3, 14.5, 11.8, 11.6, 10.5, 08.5, 06.3, 99.6

핵심

059 전가산기(FA; Full Adder)



040127

자리올림 수( $C_i$ )를 포함하여 1Bit 크기의 2진수 3자리를 더하여 합(Sum)과 자리올림 수(Carry)를 구하는 회로이다.

진리표

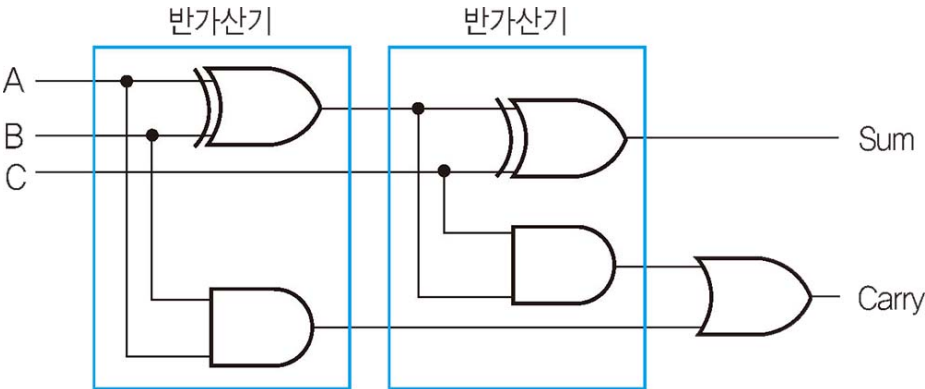
A	B	$C_i$	Sum	$C_{i+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

논리식

- 합계(Sum) =  $(A \oplus B) \oplus C_i$
- 자리올림(Carry) =  $(A \oplus B)C_i + AB$

회로

전가산기는 2개의 반가산기(HA)와 1개의 OR Gate로 구성된다.



17.3, 16.8, 09.8, 07.9, 07.5, 06.3

핵심

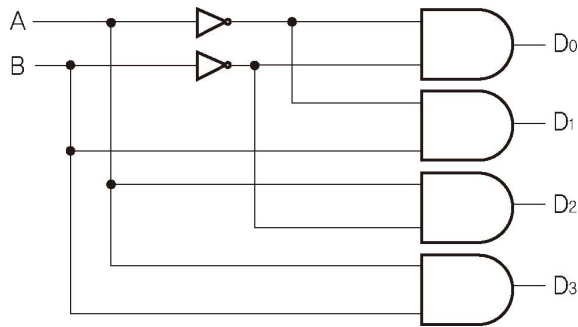
060 디코더(Decoder)



040128

- n Bit의 Code화된 정보를 그 Code의 각 Bit 조합에 따라  $2^n$ 개의 출력으로 번역하는 회로이다.
- 명령어의 명령부나 번지를 해독할 때 사용하며, 주로 AND 게이트로 구성된다.

#### • 회로



19.3, 17.8, 16.8, 16.5, 13.6, 13.3, 10.9, 08.9, 08.3, 06.5, 06.3, 05.3, 02.9

### 핵심

#### 061 플립플롭



040132

- 플립플롭은 전원이 공급되고 있는 한, 상태의 변화를 위한 신호가 발생할 때까지 현재의 상태를 그대로 유지하는 대표적인 순서 논리회로이다.
- 플립플롭 1개가 1Bit를 구성하는 2진 셀(Binary Cell)이 된다.
- 반도체 기억장치에서 2진수 1자리값을 기억하는 메모리 소자이다.
- 플립플롭은 레지스터를 구성하는 기본 소자이다.
- 기본적인 플립플롭은 2개의 NAND 또는 NOR 게이트를 이용하여 구성한다.

플립플롭	특징
RS	플립플롭의 기본으로, S와 R선의 입력을 조절하여 임의의 Bit 값을 그대로 유지시키거나, 무조건 0 또는 1의 값을 기억시키기 위해서 사용됨
JK	<ul style="list-style-type: none"> <li>• RS FF에서 S = R = 1일 때 동작되지 않는 결점을 보완한 플립플롭</li> <li>• RS FF의 입력선 S와 R에 AND 게이트 2개를 추가하여 JK FF의 입력선 J와 K로 사용함</li> <li>• 모든 플립플롭의 기능을 포함함</li> </ul>
D	<ul style="list-style-type: none"> <li>• RS FF의 R선에 인버터(Inverter)를 추가하여 S선과 하나로 묶어서 입력선을 하나만 구성한 플립플롭</li> <li>• 입력하는 값을 그대로 저장하는 기능을 수행함</li> </ul>
T	<ul style="list-style-type: none"> <li>• JK FF의 두 입력선을 묶어서 한 개의 입력선으로 구성한 플립플롭</li> <li>• T = 0인 경우는 변화가 없고, T = 1인 경우에 현재의 상태를 토글(Toggle)시킴. 즉 원 상태와 보수 상태의 2가지 상태로만 서로 전환됨</li> </ul>
마스터-슬레이브(M/S)	<ul style="list-style-type: none"> <li>• 출력측의 일부가 입력측에 궤환(FeedBack)되어 유발되는 레이스 현상을 없애기 위해 고안된 플립플롭</li> <li>• 2개의 플립플롭으로 구성되는 데, 한쪽 회로가 마스터이고 다른 한쪽이 슬레이브의 위치에 있어 마스터-슬레이브 플립플롭이라 함</li> </ul>

특성 표

〈RS 플립플롭〉

S	R	$Q_{(T+1)}$	암기
0	0	$Q_{(T)}$	무(상태 변화 없음)
0	1	0	공(항상 0)
1	0	1	일(항상 1)
1	1	동작 안 됨	불(불가)

〈JK 플립플롭〉

J	K	$Q_{(T+1)}$	암기
0	0	$Q_{(T)}$	무(상태 변화 없음)
0	1	0	공(항상 0)
1	0	1	일(항상 1)
1	1	보수	보(보수)

잠깐만요!

JK 플립플롭은 J와 K에 모두 1이 입력될 때 보수가 출력되는 것이 RS 플립플롭과 다릅니다.

16.8, 07.5, 07.3, 06.5, 01.9, 01.3, 00.10, 00.3, 99.10, 99.8, 99.4

핵심

062 자료 구성의 단위

비트(Bit, Binary Digit)	<ul style="list-style-type: none"><li>• 자료(정보) 표현의 최소 단위</li><li>• 2가지 상태(0과 1)를 표시하는 2진수 1자리</li></ul>
니블(Nibble)	<ul style="list-style-type: none"><li>• 4개의 비트(Bit)가 모여 1개의 Nibble을 구성함</li><li>• 4비트로 구성되며 16진수 1자리를 표현하기에 적합함</li></ul>
바이트(Byte)	<ul style="list-style-type: none"><li>• 문자를 표현하는 최소 단위</li><li>• 8개의 비트(Bit)가 모여 1Byte를 구성함</li><li>• 1Byte는 256(<math>2^8</math>)가지의 정보를 표현할 수 있음</li><li>• 주소 지정의 단위로 사용됨</li></ul>
워드(Word)	<ul style="list-style-type: none"><li>• CPU가 한 번에 처리할 수 있는 명령 단위</li><li>• 반워드(Half Word) : 2Byte</li><li>• 풀워드(Full Word) : 4Byte</li><li>• 더블워드(Double Word) : 8Byte</li></ul>
필드(Field)	<ul style="list-style-type: none"><li>• 파일 구성의 최소 단위</li><li>• 의미 있는 정보를 표현하는 최소 단위</li></ul>
레코드(Record)	<ul style="list-style-type: none"><li>• 하나 이상의 관련된 필드가 모여서 구성</li></ul>

	<ul style="list-style-type: none"> <li>컴퓨터 내부의 자료 처리 단위로서, 일반적으로 레코드는 논리 레코드(Logical Record)를 의미함</li> <li>데이터베이스에서 개체(Entity)에 해당됨</li> </ul>
<b>블록(Block) 물리 레코드(Physical Record)</b>	<ul style="list-style-type: none"> <li>하나 이상의 논리 레코드가 모여서 구성</li> <li>각종 저장 매체와의 입·출력 단위를 의미하며, 일반적으로 물리 레코드(Physical Record)라고 함</li> </ul>
<b>파일(File)</b>	프로그램 구성의 기본 단위로, 여러 레코드가 모여서 구성됨
<b>데이터베이스(Database)</b>	여러 개의 관련된 파일(File)의 집합

20.6, 19.4, 17.3, 16.8, 15.5, 15.3, 14.8, 14.5, 14.3, 12.3, 10.3, 09.8, 09.5, ...

**핵심****063 진법 변환**

040133

**10진수를 2진수, 8진수, 16진수로 변환**

- 정수 부분** : 10진수의 값을 변환할 진수로 나누어 더 이상 나뉘지지 않을 때까지 나누고, 나머지를 역순으로 표시함
  - 소수 부분** : 10진수의 값에 변환할 진수를 곱한 후 결과의 정수 부분만을 차례대로 표기하되, 소수 부분이 0 또는 반복되는 수가 나올 때까지 곱하기를 반복함
- 예 (47.625)<sub>10</sub>를 2진수, 8진수, 16진수로 변환하기

**〈정수 부분〉**

2진수	8진수	16진수
$  \begin{array}{r}  2 \overline{)47} \\  2 \overline{)23} \dots 1 \\  2 \overline{)11} \dots 1 \\  2 \overline{)5} \dots 1 \\  2 \overline{)2} \dots 1 \\  \quad 1 \dots 0  \end{array}  $	$  \begin{array}{r}  8 \overline{)47} \\  \quad 5 \dots 7  \end{array}  $	$  \begin{array}{r}  16 \overline{)47} \\  \quad 2 \dots 15(F)  \end{array}  $
$(47)_{10} = (101111)_2$	$(47)_{10} = (57)_8$	$(47)_{10} = (2F)_{16}$

**〈소수 부분〉**

2진수	8진수	16진수
$  \begin{array}{r}  0.625 \\  \times 2 \\  \hline  1.250 \\  \times 2 \\  \hline  0.5 \\  \times 2 \\  \hline  1.0  \end{array}  $	$  \begin{array}{r}  0.625 \\  \times 8 \\  \hline  5.000  \end{array}  $	$  \begin{array}{r}  0.625 \\  \times 16 \\  \hline  10.000  \end{array}  $
$(0.625)_{10} = (0.101)_2$	$(0.625)_{10} = (0.5)_8$	$(0.625)_{10} = (0.A)_{16}$

$$(47.625)_{10} \rightarrow (101111.101)_2$$

$$(47.625)_{10} \rightarrow (57.5)_8$$

$$(47.625)_{10} \rightarrow (2F.A)_{16}$$

## 2진수, 8진수, 16진수를 10진수로 변환

정수 부분과 소수 부분의 각 자리를 분리하여 변환하려는 각 진수의 자리값과 자리의 지수승을 곱한 결과값을 모두 더하여 계산한다.

예  $(101111.101)_2$ 를 10진수로 변환하기

$$\begin{aligned} & (1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad . \quad 1 \quad 0 \quad 1)_2 \\ & \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \quad \times \quad \times \quad \times \\ & = 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \quad . \quad 2^{-1} \quad 2^{-2} \quad 2^{-3} \\ & = 32 + 0 + 8 + 4 + 2 + 1 \quad . \quad 0.5 + 0 + 0.125 \\ & = 47.625 \end{aligned}$$

예  $(57.5)_8$ 를 10진수로 변환하기

$$\begin{aligned} & (5 \quad 7 \quad . \quad 5)_8 \\ & \quad \times \quad \times \quad \quad \times \\ & = 8^1 \quad 8^0 \quad . \quad 8^{-1} \\ & = 40 + 7 \quad . \quad 0.625 \\ & = 47.625 \end{aligned}$$

예  $(4F.2)_{16}$ 를 10진수로 변환하기

$$\begin{aligned} & (4 \quad F \quad . \quad 2)_{16} \\ & \quad \times \quad \times \quad \quad \times \\ & = 16^1 + 16^0 \quad . \quad 16^{-1} \\ & = 64 + 15 \quad . \quad 0.125 \\ & = 79.125 \end{aligned}$$

## 2진수, 8진수, 16진수 상호 변환

- **2진수를 8진수로** : 정수 부분은 소수점을 기준으로 왼쪽방향으로 3자리씩, 소수 부분은 소수점을 기준으로 오른쪽 방향으로 3자리씩 묶어서 변환함
- **2진수를 16진수로** : 정수 부분은 소수점을 기준으로 왼쪽방향으로 4자리씩, 소수 부분은 소수점을 기준으로 오른쪽 방향으로 4자리씩 묶어서 변환함



- **8진수, 16진수를 2진수로** : 8진수 1자리는 2진수 3비트로, 16진수 1자리는 2진수 4비트로 풀어서 변환함
- **8진수를 16진수로** : 8진수를 2진수로 변환한 뒤 2진수를 16진수로 변환함
- **16진수를 8진수로** : 16진수를 2진수로 변환한 뒤 2진수를 8진수로 변환함

18.4, 16.5, 16.3, 05.9, 05.3, 04.3, 03.3, 02.9, 01.9, 99.10, 99.8

**핵심****064 보수**

040134

컴퓨터가 기본적으로 수행하는 덧셈 회로를 이용하여 뺄셈을 수행하기 위해 사용한다.

<b>r의 보수</b>	<ul style="list-style-type: none"> <li>• 10진법에는 10의 보수가 있고, 2진법에는 2의 보수가 있음</li> <li>• 보수를 구할 숫자의 자릿수만큼 0을 채우고 가장 왼쪽에 1을 추가하여 기준을 만들 예 33의 10의 보수는? <math>33 + X = 100 \rightarrow X = 100 - 33 \rightarrow X = 67</math> 예 10101의 2의 보수는? <math>10101 + X = 100000 \rightarrow X = 100000 - 10101 \rightarrow X = 01011</math></li> </ul>
<b>r-1의 보수</b>	<ul style="list-style-type: none"> <li>• 10진법에는 9의 보수가 있고, 2진법에는 1의 보수가 있음</li> <li>• 10진수 N에 대한 9의 보수는 주어진 숫자의 자릿수 만큼 9를 채워 기준을 만들 예 33의 9의 보수는? <math>33 + X = 99 \rightarrow X = 99 - 33 \rightarrow X = 66</math></li> <li>• 2진수 N에 대한 1의 보수는 주어진 숫자의 자릿수 만큼 1을 채워 기준을 만들 예 10101의 1의 보수는? <math>10101 + X = 11111 \rightarrow X = 11111 - 10101 \rightarrow X = 01010</math></li> </ul>

19.4, 18.8, 18.3, 17.5, 15.3, 14.5, 13.8, 12.8, 11.8, 11.6, 11.3, 09.5, 08.9, 07.5, 07.3, 06.9, 06.3, 05.5, ...

**핵심****065 2진 연산**

- 정수값을 2진수로 변환하여 표현하는 방식이다.
- 표현할 수 있는 범위는 작지만 연산 속도가 빠르다.
- n Bit 크기의 워드가 있을 때 맨 처음 1Bit는 부호(Sign) 비트로 사용되고 나머지 n-1 Bit에 2진수로 표현된 정수값이 저장된다.
- 컴퓨터에서 정수를 표기할 때 크기에 제한을 받는 가장 큰 이유는 워드의 비트 수 때문이다.
- **양수** : 부호 비트에 0을 넣고, 변환된 2진수 값을 Data Bit의 오른쪽에서 왼쪽 순으로 차례로 채우고 남은 자리에 0을 채움
- **음수** : 음수를 표현할 때는 다음과 같은 3가지 방법을 사용함

종류	표현 방법	비고
부호화 절대치법(Signed Magnitude)	양수 표현에 대하여 부호 Bit의 값만 0을 1로 바꿈	2가지 형태의 0 존재(+0, -0)
부호화 1의 보수법(Signed 1's Complement)	양수 표현에 대하여 1의 보수를 취	

plement)	함	
부호화 2의 보수법(Signed 2's Complement)	양수 표현에 대하여 2의 보수를 취함	한 가지 형태의 0만 존재(+0)

## 표현 범위

종류	범위	n = 8	n = 16	n = 32
부호화 절대치법	$-2^{n-1}+1 \sim +2^{n-1}-1$	$-127 \sim +127$	$-32767 \sim +32767$	$-2^{31}+1 \sim +2^{31}-1$
부호화 1의 보수법	1			
부호화 2의 보수법	$-2^{n-1} \sim +2^{n-1}-1$	$-128 \sim +127$	$-32768 \sim +32767$	$-2^{31} \sim +2^{31}-1$

18.4, 17.8, 16.3, 15.3, 12.8, 11.6, 10.9, 09.8, 09.5, 08.5, 05.9, 05.4, ...

## 핵심

### 066 10진 연산



040137

10진수 1자리를 2진수 4자리로 표현하는 방식으로, 언팩(Unpack) 연산과 팩(Pack) 연산이 있다.

### 언팩(Unpack) 연산

- 존(Zone)형 10진 연산이라고도 한다.
- 연산이 불가능하고, 데이터의 입·출력에 사용된다.
- 1Byte로 10진수 1자리를 표현한다.
- 4개의 존(Zone) 비트와 4개의 숫자(Digit) 비트를 사용한다.
- 최하위(가장 오른쪽) 바이트의 존(Zone) 부분을 부호로 사용한다.

Zone	Digit	Zone	Digit	Zone	Digit	...	Sign	Digit
------	-------	------	-------	------	-------	-----	------	-------

- **Zone 부분** : 무조건 1111을 넣음
- **Digit 부분** : 10진수 1자리를 4Bit 2진수로 표현함
- **Sign 부분** : 양수는 C(1100<sub>2</sub>), 음수는 D(1101<sub>2</sub>), 부호 없는 양수는 F(1111<sub>2</sub>)로 표현함

### 팩(Pack) 연산

- 연산이 가능하고, 데이터의 입·출력이 불가능하다.
- 1Byte로 10진수 2자리를 표현한다.
- 최하위(가장 오른쪽) 바이트의 4Bit 부분을 부호로 사용한다.

Digit	Digit	Digit	Digit	Digit	Digit	...	Digit	Sign
-------	-------	-------	-------	-------	-------	-----	-------	------

- **Digit 부분** : 10진수 1자리를 4Bit 2진수로 표현함

- **Sign 부분** : 양수는 C(1100<sub>2</sub>), 음수는 D(1101<sub>2</sub>), 부호 없는 양수는 F(1111<sub>2</sub>)로 표현함

20.8, 16.3, 14.3, 13.6, 11.8, 11.6, 09.8, 08.3, 06.9, 05.9, 05.5, 02.3, 01.6

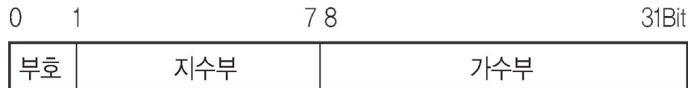
**핵심****067 부동 소수점 표현**

040138

부동 소수점 방식은 소수점이 포함된 실수 데이터의 표현과 연산에 사용하는 방식이다.

**부동 소수점 방식의 특징**

- 고정 소수점 방식으로 표현하는 것보다 매우 큰 수나 작은 수, 매우 정밀한 수를 적은 비트로 표현할 수 있다.
- 과학이나 공학 또는 수학적인 응용에 주로 사용된다.
- 고정 소수점 방식에 비해 연산 시간이 많이 걸린다.
- 지수부와 가수부를 분리하는 정규화 과정이 필요하다.
- 정규화의 목적은 유효 자릿수를 최대로 하여 수의 정밀도를 높이기 위한 것이다.
- 4Byte를 사용하는 단정도와 가수부를 4Byte 추가하여 좀더 정밀하게 표현할 수 있는 8Byte 배정도 표현법이 있다.

**부동 소수점 수의 연산 방법****• 덧셈, 뺄셈**

- ① 0인지의 여부를 조사한다.
- ② **가수의 위치 조정** : 두 자료의 지수를 비교한 후 소수점의 위치를 이동하여 지수가 큰 쪽에 맞춘다.
- ③ 가수부 값끼리 더하거나 뺀다.
- ④ 결과를 정규화한다.

**• 곱셈**

- ① 0인지의 여부를 조사한다.
- ② 지수를 더한다.
- ③ 가수를 곱한다.
- ④ 결과를 정규화한다.

**• 나눗셈**

- ① 0인지의 여부를 조사한다.

- ② 부호를 결정한다.
- ③ 피제수가 제수보다 작게 피제수의 위치를 조정한다.
- ④ 지수의 뺄셈을 한다.
- ⑤ 가수의 나눗셈을 한다.

18.8, 12.5, 08.5, 07.3, 06.5, 05.5, 05.3, 04.9, 03.8, 03.5, 02.9, 01.6, 99.10

## 핵심

### 068 자료의 외부적 표현

BCD(Binary Coded Decimal, 2진화 10진 코드)	<ul style="list-style-type: none"> <li>• 6Bit 코드로 IBM 사에서 개발</li> <li>• 1개의 문자를 2개의 Zone 비트와 4개의 Digit 비트로 표현함</li> <li>• 6Bit는 <math>2^6</math>개를 표현할 수 있으므로 64개의 문자를 표현할 수 있음</li> <li>• 1Bit의 Parity Bit를 추가하여 7Bit로 사용함</li> <li>• 영문 소문자를 표현하지 못함</li> </ul>
ASCII 코드(American Standard Code for Information Interchange)	<ul style="list-style-type: none"> <li>• 7Bit 코드로 미국 표준협회에서 개발</li> <li>• 1개의 문자를 3개의 Zone 비트와 4개의 Digit 비트로 표현함</li> <li>• <math>2^7 = 128</math>가지의 문자를 표현할 수 있음</li> <li>• 1Bit의 Parity Bit를 추가하여 8Bit로 사용함</li> <li>• 통신 제어용 및 마이크로 컴퓨터의 기본 코드임</li> </ul>
EBCDIC(Extended BCD Interchange Code, 확장 2진화 10진 코드)	<ul style="list-style-type: none"> <li>• 8Bit 코드로 IBM에서 개발</li> <li>• 1개의 문자를 4개의 Zone 비트와 4개의 Digit 비트로 표현함</li> <li>• <math>2^8 = 256</math>가지의 문자를 표현할 수 있음</li> <li>• 1Bit의 Parity Bit를 추가하여 9Bit로 사용함</li> <li>• 대형 기종의 컴퓨터에서 사용함</li> </ul>

16.8, 15.8, 13.8, 13.3, 11.6, 11.3, 10.3, 09.8, 09.3, 07.9, 07.5, 07.3, ...

## 핵심

### 069 기타 자료의 표현 방식



040141

BCD 코드	<ul style="list-style-type: none"> <li>• 10진수 1자리의 수를 2진수 4Bit로 표현함</li> <li>• 4Bit의 2진수 각 Bit가 <math>8(2^3)</math>, <math>4(2^2)</math>, <math>2(2^1)</math>, <math>1(2^0)</math>의 자리값을 가지므로 8421 코드라고도 함</li> <li>• 대표적인 가중치 코드</li> <li>• 문자 코드인 BCD에서 Zone 부분을 생략한 형태임</li> <li>• 10진수 입·출력이 간편함</li> </ul>
Excess-3 코드(3초과 코드)	<ul style="list-style-type: none"> <li>• BCD + 3, 즉 BCD 코드에 3을 더하여 만든 코드임</li> <li>• 대표적인 자기 보수 코드이며, 비가중치 코드임</li> </ul>
Gray 코드	<ul style="list-style-type: none"> <li>• BCD 코드의 인접하는 비트를 X-OR 연산하여 만든 코드</li> <li>• 입·출력장치, A/D 변환기, 주변장치 등에서 숫자를 표현할 때 사용</li> <li>• 1Bit만 변화시켜 다음 수치로 증가시키기 때문에 하드웨어적인 오류가 적음</li> </ul>
	<ul style="list-style-type: none"> <li>• 코드의 오류를 검사하기 위해서 데이터 비트 외에 1Bit의 패리티 체크 비트를 추가하는</li> </ul>

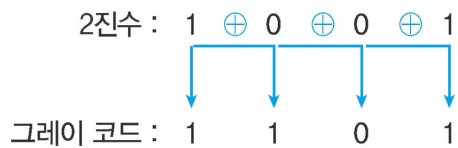
패리티 검사 코드	것으로 1Bit의 오류만 검출할 수 있음 • Odd(기수) Parity : 코드에서 1인 Bit의 수가 홀수가 되도록 0이나 1을 추가함 • Even(우수) Parity : 코드에서 1인 Bit의 수가 짝수가 되도록 0이나 1을 추가함
해밍 코드	• 오류를 스스로 검출하여 교정이 가능한 코드 • 1Bit의 오류만 교정할 수 있음 • 데이터 비트 외에 여러 검출 및 교정을 위한 잉여 비트가 많이 필요함 • 해밍 코드 중 1, 2, 4, 8, 16 …… $2^n$ 번째 비트는 오류 검출을 위한 패리티 비트임
허프만 코드	사용되는 문자의 빈도수에 따라 코드의 길이가 달라짐

20.8, 18.3, 16.8, 14.3, 10.3, 09.3, 06.3, 05.9, 04.9, 04.5, 03.3, 02.5, 01.6, 01.3

**핵심****070 그레이 코드(Gray Code) 변환****2진수를 Gray Code로 변환하는 방법**

- ① 첫 번째 그레이 비트는 2진수의 첫 번째 비트를 그대로 내려쓴다.
- ② 두 번째 그레이 비트부터는 변경할 2진수의 해당 번째 비트와 그 왼쪽의 비트를 XOR 연산하여 쓴다.

예 2진수 1001을 Gray Code로 변환하시오.

**Gray Code를 2진수로 변환하는 방법**

- ① 첫 번째 2진수 비트는 그레이 코드의 첫 번째 비트를 그대로 내려쓴다.
- ② 두 번째 2진수 비트부터는 왼쪽에 구해 놓은 2진수 비트와 변경할 해당 번째 그레이 비트를 XOR 연산하여 쓴다.

예 Gray Code 1001을 2진수로 변환하시오.

