

Microprocessor (W2)

- Data Representation -

Dong Min Kim
Department of IoT
Soonchunhyang University
dmk@sch.ac.kr

Contents

- 01 진법과 진법 변환
- 02 정수 표현
- 03 실수 표현
- 04 디지털 코드
- 05 에러 검출 코드

01 진법과 진법 변환

1 디지털 정보의 단위

- 1nibble = 4bit
- 1byte = 8bit
- 1byte = 1문자(character)
- 영어는 1byte로 1 문자 표현, 한글은 2byte가 필요
- 1워드 : 특정 CPU에서 취급하는 명령어나 데이터의 길이에 해당하는 비트 수
- 워드 길이는 8·16·32·64비트 등 8의 배수가 가능하다.

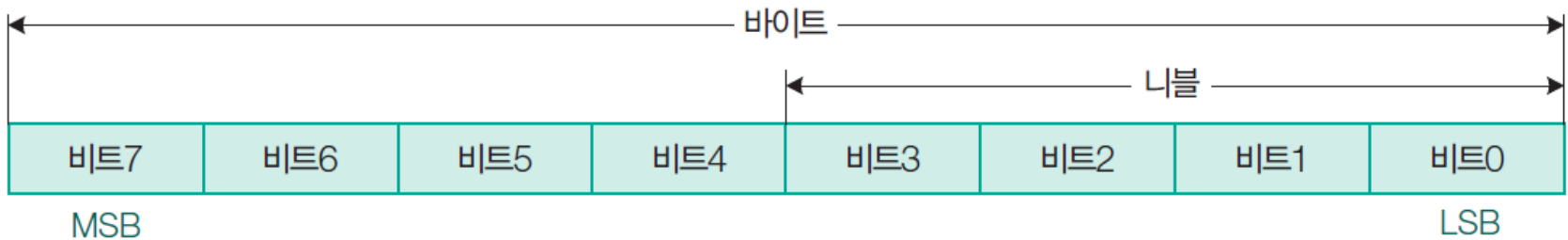


그림 2-1 비트, 니블, 바이트의 관계

MSB Most Significant Bit: 최상위 비트

LSB Least Significant Bit: 최하위 비트

01 진법과 진법 변환

❖ SI 단위와 IEC 단위 비교

표 2-1 SI 단위와 IEC 단위 비교

SI(10진수 단위)			IEC(2진수 단위)			
값	기호	이름	값	기호	이름	10진수 변환 크기
$(10^3)^1 = 10^3$	k, K	kilo-	$(2^{10})^1 = 2^{10} \cong 10^{3.01}$	Ki	kibi-	1,024
$(10^3)^2 = 10^6$	M	mega-	$(2^{10})^2 = 2^{20} \cong 10^{6.02}$	Mi	mebi-	1,048,576
$(10^3)^3 = 10^9$	G	giga-	$(2^{10})^3 = 2^{30} \cong 10^{9.03}$	Gi	gibi-	1,073,741,824
$(10^3)^4 = 10^{12}$	T	tera-	$(2^{10})^4 = 2^{40} \cong 10^{12.04}$	Ti	tebi-	1,099,511,627,776
$(10^3)^5 = 10^{15}$	P	peta-	$(2^{10})^5 = 2^{50} \cong 10^{15.05}$	Pi	pebi-	1,125,899,906,842,624
$(10^3)^6 = 10^{18}$	E	exa-	$(2^{10})^6 = 2^{60} \cong 10^{18.06}$	Ei	exbi-	1,152,921,504,606,846,976
$(10^3)^7 = 10^{21}$	Z	zetta-	$(2^{10})^7 = 2^{70} \cong 10^{21.07}$	Zi	zebi-	1,180,591,620,717,411,303,424
$(10^3)^8 = 10^{24}$	Y	yotta-	$(2^{10})^8 = 2^{80} \cong 10^{24.08}$	Yi	yobi-	1,208,925,819,614,629,174,706,176

kibi-: kilobinary, mebi-: megabinary, gibi-: gigabinary,
tebi-: terabinary, pebi-: petabinary, exbi-: exabinary,
zebi-: zettabinary, yobi-: yottabinary

01 진법과 진법 변환

2 진법

❖ 10진법

- 10진수: 기수가 10인 수
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9의 10개 수로 표현

$$\begin{aligned} 9345.35 &= 9 \times 1000 + 3 \times 100 + 4 \times 10 + 5 \times 1 + 3 \times 0.1 + 5 \times 0.01 \\ &= 9 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 + 3 \times 10^{-1} + 5 \times 10^{-2} \end{aligned}$$

❖ 2진법

- 기수가 2인 수
- 0, 1 두 개의 수로 표현

$$\begin{aligned} 1010.1011_{(2)} &= 1 \times 1000_{(2)} + 0 \times 100_{(2)} + 1 \times 10_{(2)} + 0 \times 1_{(2)} \\ &\quad + 1 \times 0.1_{(2)} + 0 \times 0.01_{(2)} + 1 \times 0.001_{(2)} + 1 \times 0.0001_{(2)} \\ &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} \end{aligned}$$

01 진법과 진법 변환

❖ 8진법

- 0~7까지 8개의 수로 표현

$$\begin{aligned} 607.36_{(8)} &= 6 \times 100_{(8)} + 0 \times 10_{(8)} + 7 \times 1_{(8)} + 3 \times 0.1_{(8)} + 6 \times 0.01_{(8)} \\ &= 6 \times 8^2 + 0 \times 8^1 + 7 \times 8^0 + 3 \times 8^{-1} + 6 \times 8^{-2} \end{aligned}$$

❖ 16진법

- 0~9, A~F까지 16개의 기호로 표현

$$\begin{aligned} 6C7.3A_{(16)} &= 6 \times 100_{(16)} + C \times 10_{(16)} + 7 \times 1_{(16)} + 3 \times 0.1_{(16)} + A \times 0.01_{(16)} \\ &= 6 \times 16^2 + C \times 16^1 + 7 \times 16^0 + 3 \times 16^{-1} + A \times 16^{-2} \end{aligned}$$

- 8진수보다는 16진수를 사용하는 경우가 더 많은데 실제로 컴퓨터 구조나 어셈블리어에서는 16진수를 많이 쓴다. 자릿수를 더 짧게 표현할 수 있기 때문이다.

01 진법과 진법 변환

❖ 2진수에 해당하는 8진수, 16진수, 10진수 표현

표 2-2 2진수에 해당하는 8진수, 16진수, 10진수 표현

2진수	8진수	10진수	2진수	16진수	10진수	2진수	16진수	10진수
000	0	0	0000	0	0	1000	8	8
001	1	1	0001	1	1	1001	9	9
010	2	2	0010	2	2	1010	A	10
011	3	3	0011	3	3	1011	B	11
100	4	4	0100	4	4	1100	C	12
101	5	5	0101	5	5	1101	D	13
110	6	6	0110	6	6	1110	E	14
111	7	7	0111	7	7	1111	F	15

3 진법 변환

□ 2진수, 8진수, 16진수를 10진수로 변환

① 2진수를 10진수로 변환한 예

$$\begin{aligned}101101.101_{(2)} &= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\&= 32 + 0 + 8 + 4 + 0 + 1 + 0.5 + 0 + 0.125 \\&= 45.625_{(10)}\end{aligned}$$

② 8진수를 10진수로 변환한 예

$$\begin{aligned}364.35_{(8)} &= 3 \times 8^2 + 6 \times 8^1 + 4 \times 8^0 + 3 \times 8^{-1} + 5 \times 8^{-2} \\&= 3 \times 64 + 6 \times 8 + 4 \times 1 + 3 \times 0.125 + 5 \times 0.015625 \\&= 192 + 48 + 4 + 0.375 + 0.078125 \\&= 244.453125_{(10)}\end{aligned}$$

③ 16진수를 10진수로 변환한 예

$$\begin{aligned}A3.D2_{(16)} &= 10 \times 16^1 + 3 \times 16^0 + 13 \times 16^{-1} + 2 \times 16^{-2} \\&= 10 \times 16 + 3 \times 1 + 13 \times 0.0625 + 2 \times 0.00390625 \\&= 160 + 3 + 0.8125 + 0.0078125 \\&= 163.8203125_{(10)}\end{aligned}$$

01 진법과 진법 변환

□ 10진수 - 2진수 변환

- 정수부분과 소수부분으로 나누어 변환
- 정수부분은 2로 나누고, 소수부분은 2를 곱한다.
- 10진수 75.6875를 2진수로 변환한 예

2	75	나머지	→	2진수
2	37	...	1	→ 1
2	18	...	1	→ 11
2	9	...	0	→ 011
2	4	...	1	→ 1011
2	2	...	0	→ 01011
2	1	...	0	→ 001011
	0	...	1	→ 1001011
				몫

2진수	←	정수	소수
		0.	6875
		×	2
0.1	←	1.	3750
		×	2
0.10	←	0.	7500
		×	2
0.101	←	1.	5000
		×	2
0.1011	←	1.	0

곱셈 결과 정수를 적는다.

소수 부분이 0이 될 때까지 계산한다.

$$75.6875_{(10)} = 1001011.1011_{(2)}$$

01 진법과 진법 변환

□ 10진수 - 8진수 변환

- 10진수 75.6875를 8진수로 변환
- 8로 나누고, 곱한다.

8		75		나머지	→	8진수
8		9	...	3	→	3
8		1	...	1	→	13
		0	...	1	→	113
				몫		

8진수	←	정수	소수	
		0.	6875	
		×	8	
0.5	←	5.	5000	곱셈 결과 정수를 적는다.
		×	8	
0.54	←	4.	0	소수 부분이 0이 될 때까지 계산한다.

$$75.6875_{(10)} = 113.54_{(8)}$$

01 진법과 진법 변환

□ 10진수 - 16진수 변환

- 10진수 75.6875를 16진수로 변환
- 16으로 나누고, 곱한다.

16		75		나머지	→	16진수
16		4	...	11	→	B
		0	...	4	→	4B
		몫				

16진수	←	정수	소수
		0.	6875
		×	16
0.B	←	11.	0000

곱셈 결과 정수를 적는다.
소수 부분이 0이 될 때까지 계산한다.

$$75.6875_{(10)} = 4B.B_{(16)}$$

01 진법과 진법 변환

□ 2진수 - 8진수 - 10진수 - 16진수 상호 변환

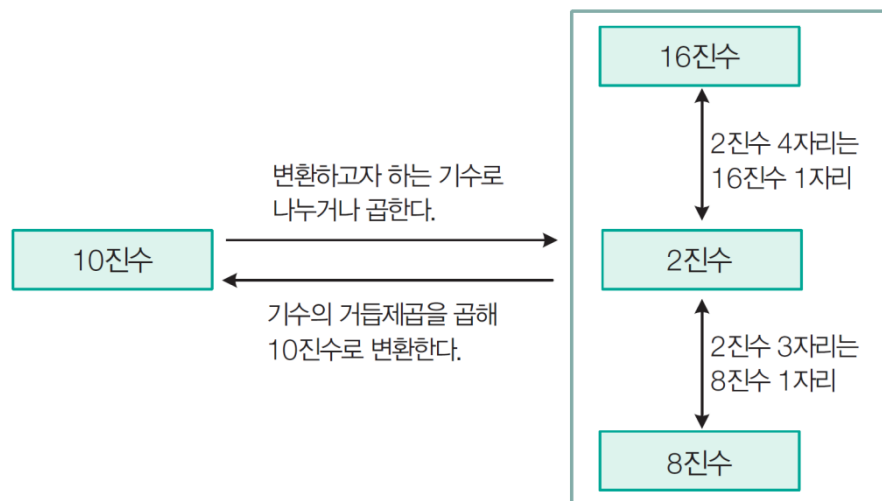


그림 2-2 2진수, 8진수, 10진수, 16진수 상호 변환 개념도

10진수	2진수	8진수	16진수
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

01 진법과 진법 변환

- 10진수를 8진수, 16진수로 변환한 예

$$\begin{aligned} 75.6875 &= 1001011.1011_{(2)} \\ &= \textcolor{teal}{001} \ 001 \ 011.101 \ \textcolor{teal}{100}_{(2)} \\ &= \quad 1 \quad 1 \quad 3. \ 5 \quad 4_{(8)} \end{aligned}$$

$$\begin{aligned} 75.6875 &= 1001011.1011_{(2)} \\ &= \textcolor{teal}{0100} \ 1011.1011_{(2)} \\ &= \quad 4 \quad \text{B.} \quad \text{B}_{(16)} \end{aligned}$$

- 8진수와 16진수를 2진수로 변환한 예

$$\begin{aligned} \quad 3 \quad 6 \quad 7. \quad 7 \quad 5_{(8)} \\ = 011 \ 110 \ 111. \ 111 \ 101_{(2)} \end{aligned}$$

$$\begin{aligned} \quad 9 \quad \text{A} \quad 3. \quad 5 \quad 0 \quad \text{F} \quad 3_{(16)} \\ = 1001 \ 1010 \ 0011. \ 0101 \ 0000 \ 1111 \ 0011_{(2)} \end{aligned}$$

01 진법과 진법 변환

예제 2-1 10진수 48.8125를 2진수, 8진수, 16진수로 변환하여라.

풀이

2	48	나머지	→	2진수
2	24	...	0	→ 0
2	12	...	0	→ 00
2	6	...	0	→ 000
2	3	...	0	→ 0000
2	1	...	1	→ 10000
	0	...	1	→ 110000
	몫			

2진수	←	정수	소수
		0.	8125
		×	2
0.1	←	1.	6250
		×	2
0.11	←	1.	2500
		×	2
0.110	←	0.	5000
		×	2
0.1101	←	1.	0

$$110000.1101_{(2)} = 110\ 000.\ 110\ 100_{(2)} = 60.64_{(8)}$$

$$= 0011\ 0000.1101_{(2)} = 30.D_{(16)}$$

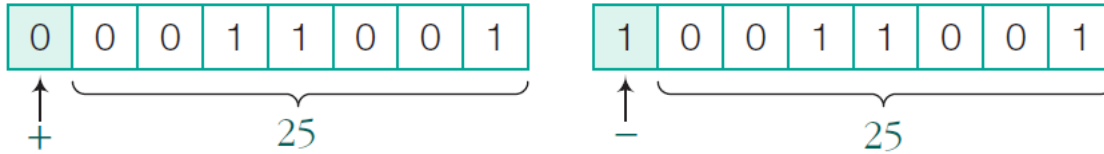
End of Example

1 보수의 개념과 음수

- ❖ 최상위비트(MSB)를 부호비트로 사용
 - 양수(+): 0 음수(-): 1
- ❖ 2진수 음수를 표시하는 방법
 - 부호와 절댓값(sign-magnitude)
 - 1의 보수(1's complement)
 - 2의 보수(2's complement)

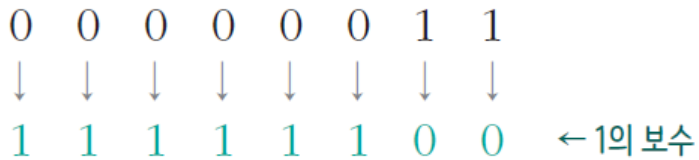
❖ 부호와 절댓값

- 부호비트만 양수와 음수를 나타내고 나머지 비트들은 같다.



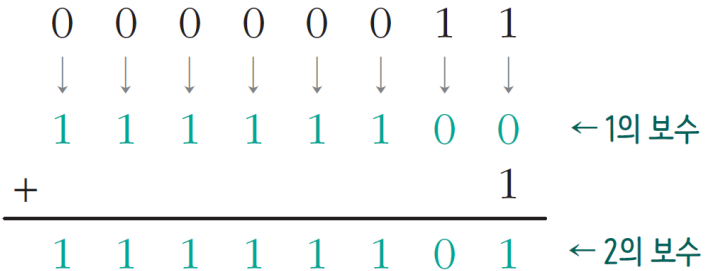
❖ 1의 보수 방식

- $0 \rightarrow 1, 1 \rightarrow 0$ 으로 변환



❖ 2의 보수 방식

- 1의 보수 + 1 = 2의 보수



02 정수 표현

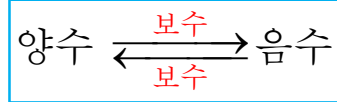
❖ 수의 표현 방법에 따른 10진수 대응 값

표 2-3 수의 표현 방법에 따른 10진수 대응 값

4비트 2진수	부호 없는 수	부호와 절댓값	1의 보수	2의 보수
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	-0	-7	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

02 정수 표현

- ❖ 양수를 보수로 바꾸면 음수
- ❖ 음수를 보수로 바꾸면 양수



- ❖ 2진수와 그 수의 1의 보수와의 합은 모든 bit가 1이 된다.
- ❖ 2진수와 그 수의 2 보수와의 합은 모든 bit가 0이 된다.
(자릿수를 벗어나는 비트는 제외)

❖ 2의 보수에 대한 10진수의 표현 범위

표 2-4 n 비트 2의 보수에 대한 10진수의 표현 범위

비트 수	2의 보수를 사용한 2진 정수의 표현 범위
n 비트	$-2^{n-1} \sim +2^{n-1}-1$
4비트	$-2^{4-1}(-8) \sim +2^{4-1}-1(+7)$
8비트	$-2^{8-1}(-128) \sim +2^{8-1}-1(+127)$
16비트	$-2^{16-1}(-32,768) \sim +2^{16-1}-1(+32767)$
32비트	$-2^{32-1}(-2,147,483,648) \sim +2^{32-1}-1(+2,147,483,647)$
64비트	$-2^{64-1}(-9,223,372,036,854,775,808) \sim +2^{64-1}-1(+9,223,372,036,854,775,807)$

□ 2의 보수를 10진수로 변환

❖ 2의 보수로 표현된 수 $00101101_{(2)}$ 을 10진수로 변환하는 방법

- 최상위 비트가 0이므로 양수. 따라서 $00101101_{(2)}$ 의 값을 10진수로 변환하여 +부호를 붙인다.

$$\begin{aligned}00101101_{(2)} &= 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\&= 0 + 0 + 32 + 0 + 8 + 4 + 0 + 1 \\&= 45\end{aligned}$$

❖ 2의 보수로 표현된 수 $10101100_{(2)}$ 을 10진수로 변환하는 방법

- 최상위 비트가 1이므로 음수다.
- 2의 보수로 바꾼 후 10진수로 변환하여 -부호를 붙이면 -84가 된다.

$$10101100_{(2)} \xrightarrow[\text{음수} \rightarrow \text{양수}]{\text{1의 보수} + 1 = 01010011 + 1 = \text{2의 보수}} 01010100_{(2)}$$

$$\begin{aligned}01010100_{(2)} &= 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \\&= 0 + 64 + 0 + 16 + 0 + 4 + 0 + 0 \\&= 84\end{aligned}$$

2 부호 확장

- 부호 확장이란 늘어난 비트 수 만큼 부호를 늘려주는 방법

표 2-5 2진수 표현 방법에 따른 부호 확장

2진수 표현 방식	부호 확장 방법	구분	8비트	16비트 확장
부호와 절댓값	부호만 MSB에 복사하고, 나머지는 0으로 채운다.	양수	00101010	00000000 00101010
		음수	10010111	10000000 00010111
1의 보수	늘어난 길이만큼 부호와 같은 값으로 모두 채운다.	양수	00101010	00000000 00101010
		음수	10010111	11111111 10010111
2의 보수	늘어난 길이만큼 부호와 같은 값으로 모두 채운다.	양수	00101010	00000000 00101010
		음수	10010111	11111111 10010111

3 2진 정수 연산

- 뺄셈의 원리를 보면, A-B 대신에 A+(B의 2의 보수)를 계산하면 된다.
- 뺄셈에서 2의 보수 방식을 사용하는 이유는 뺄셈을 가산기를 사용하여 수행할 수 있다.

① 자리올림수 → 0110000

양수	49 =	00110001
+ 양수	+ 58 = +	00111010
양수	107 =	0 01101011

② 자리올림수 → 1111110

양수	58 =	00111010
- 양수	- 49 = -	00110001
양수		00111010
+ 음수		+ 11001111
양수	9 =	1 00001001

2의 보수

③ 자리올림수 → 0000000

양수	49 =	00110001
- 양수	- 58 = -	00111010
양수		00110001
+ 음수		+ 11000110
음수	- 9 =	0 11110111

2의 보수

④ 자리올림수 → 1001110

- 양수	- 49 = -	00110001
- 양수	- 58 = -	00111010
음수		11001111
+ 음수		+ 11000110
음수	- 107 =	1 10010101

2의 보수

2의 보수

⑤ 자리올림수 → 1000010

양수	98 =	01100010
+ 양수	+ 74 = +	01001010
음수	- 84 =	0 10101100

서로
다름

overflow

⑥ 자리올림수 → 0111110

- 양수	- 98 = -	01100010
- 양수	- 74 = -	01001010
음수		10011110
+ 음수		+ 10110110
양수	+ 84 =	1 01010100

2의 보수

2의 보수

서로
다름

underflow

예제 2-2

8비트 연산 $98+74$ 는 오버플로가 발생하여 잘못 계산된 결과를 얻었다. 부호 확장으로 올바른 결과를 얻을 수 있음을 설명하여라.

풀이

부호를 확장하여 연산하는 과정은 다음과 같다.

$$\begin{array}{rcl} & \text{carry} & \underline{0}0000000 \ 1000010 \\ 98 & = & 00000000 \ 01100010 \\ + \ 74 & = & + \ 00000000 \ 01001010 \\ \hline 172 & = & \underline{0} \ 00000000 \ 10101100 \end{array}$$

직전 캐리와 최종 캐리(밑줄 친 부분)가 같으므로 정상적으로 계산된 것임을 알 수 있고, 그 결과는 $10101100_{(2)} = 128+32+8+4 = 172_{(10)}$ 이다.

End of Example

03 실수 표현

- 컴퓨터의 부동소수점수는 **IEEE 754표준**을 따른다.
- **부호**(sign), **지수**(exponent), **가수**(mantissa)의 세 영역으로 표시
- 부호(S)가 0일 때는 양수를 나타내고, 1일 때는 음수를 나타낸다.
- 단정도(single precision) 부동소수점수와 배정도(double precision) 부동소수점수의 두 가지 표현 방법이 있다.

구분	IEEE 754 표준 부동 소수점 수의 비트 할당	바이트스
단정도 부동 소수점 수	<p>31 30 29 ... 24 23 22 21 ... 1 0</p> <p>S 지수 가수</p>	127
배정도 부동 소수점 수	<p>63 62 61 ... 53 52 51 50 ... 1 0</p> <p>S 지수 가수</p>	1023

그림 2-5 단정도 및 배정도 부동 소수점 수에 할당된 비트 수

단정도(단일정밀도): 32비트, 배정도(2배정밀도): 64비트,
4배정도(4배정밀도): 128비트

03 실수 표현

□ 정규화(normalization) : 과학적 표기방법

❖ 2진수의 정규화

$$\begin{aligned} 75.6875 &= 1001011.1011_{(2)} \\ &= 1.0010111011_{(2)} \times 2^6 \\ &= 1.0010111011_{(2)} \times 2^{10_{(2)}} \end{aligned}$$

❖ 바이어스(bias) : 지수의 양수, 음수를 나타내기 위한 방법

- IEEE 754 표준에서는 바이어스 127(단정도) 또는 1023(배정도)을 사용
- 표현 지수 = 바이어스 + 2진 지수 값

부호	지수(바이어스 127)	가수(1.xxx ₍₂₎)
양수	01111111(127) + 110(6)	1.을 생략한 가수
0	10000101	001011101100000000000000

03 실수 표현

❖ 10진수 -0.2를 단정도 부동소수점으로 표현

- 2진수로 변환하고 정규화한다.

$$\begin{aligned} -0.2 &= -0.00110011001100110011001\dots_{(2)} \\ &= -1.100110011001100110011001\dots_{(2)} \times 2^{-3} \\ &= -1.100110011001100110011001\dots \times 2^{-11_{(2)}} \end{aligned}$$

부호	지수(바이어스 127)	가수($1.xxx_{(2)}$)
음수	$01111111(127) - 11(3)$	1.을 생략한 가수
1	01111100	10011001100110011001100

□ 컴퓨터에서의 부동소수점수의 표현 범위

표 2-6 단정도 및 배정도 부동 소수점 수의 표현 범위

구분	단정도 부동 소수점 수	배정도 부동 소수점 수
비정규화된 2진수	$\sim \pm 2^{-149}$ 에서 $\pm (1-2^{-23}) \times 2^{126}$ 까지	$\sim \pm 2^{-1074}$ 에서 $\pm (1-2^{-52}) \times 2^{1022}$ 까지
정규화된 2진수	$\sim \pm 2^{-126}$ 에서 $\pm (2-2^{-23}) \times 2^{127}$ 까지	$\sim \pm 2^{-1022}$ 에서 $\pm (2-2^{-52}) \times 2^{1023}$ 까지
10진수	$\sim \pm 1.40 \times 10^{-45}$ 에서 $\pm 3.40 \times 10^{38}$ 까지	$\sim \pm 4.94 \times 10^{-324}$ 에서 $\pm 1.798 \times 10^{308}$ 까지

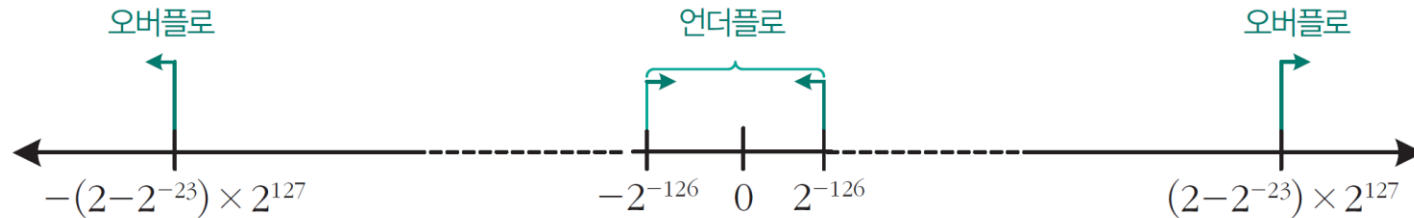


그림 2-6 단정도 부동 소수점 수의 표현 범위

예제 2-3

$\frac{1}{256}$ 을 단정도 부동소수점 방식으로 표현하여라.

풀이

$\frac{1}{256}$ 을 정규화된 방법으로 표현하면 $\frac{1}{256} = \frac{1}{2^8} = 2^{-8} = 1.0 \times 2^{-8}$ 이다.

그러므로 $1.0_{(2)} \times 2^{-1000_{(2)}}$ 를 단정도 부동소수점수로 표현하면 다음과 같다.

부호	지수(bias 127)	가수($1.\times\times\times_{(2)}$)
양수	$01111111(127) - 1000(8)$	1.을 생략한 가수
0	01110111	0000000000000000000000000000

End of Example

1 BCD 코드(Binary Coded Decimal Code : 2진화 10진 코드, 8421 코드)

- BCD코드는 10진수 0(0000)부터 9(1001)까지를 2진화한 코드
- 표기는 2진수이지만 의미는 10진수
- 1010부터 1111까지 6개는 사용하지 않음

표 2-7 BCD(8421) 코드

10진수	BCD 코드	10진수	BCD 코드	10진수	BCD 코드
0	0000	10	0001 0000	20	0010 0000
1	0001	11	0001 0001	31	0011 0001
2	0010	12	0001 0010	42	0100 0010
3	0011	13	0001 0011	53	0101 0011
4	0100	14	0001 0100	64	0110 0100
5	0101	15	0001 0101	75	0111 0101
6	0110	16	0001 0110	86	1000 0110
7	0111	17	0001 0111	97	1001 0111
8	1000	18	0001 1000	196	0001 1001 0110
9	1001	19	0001 1001	237	0010 0011 0111

04 디지털 코드

- 예를 들어 10진수 237을 BCD 코드로 변환하면

10진수	2	3	7
	↓	↓	↓
BCD 코드	0010	0011	0111

- BCD 코드의 연산은 다음 예에서 ❶ ~ ❷와 같이 10진수처럼 연산한다.
- 계산 결과가 ❸과 같이 9를 초과하여 BCD 코드를 벗어나는 경우에는 결과에 6(0110)을 더한다.

❶ 10진수 덧셈	BCD 덧셈
6	0110 _(BCD)
+ 3	0011 _(BCD)
<hr/>	<hr/>
9	1001 _(BCD)

❷ 10진수 덧셈	BCD 덧셈
42	0100 0010 _(BCD)
+ 37	0011 0111 _(BCD)
<hr/>	<hr/>
79	0111 1001 _(BCD)

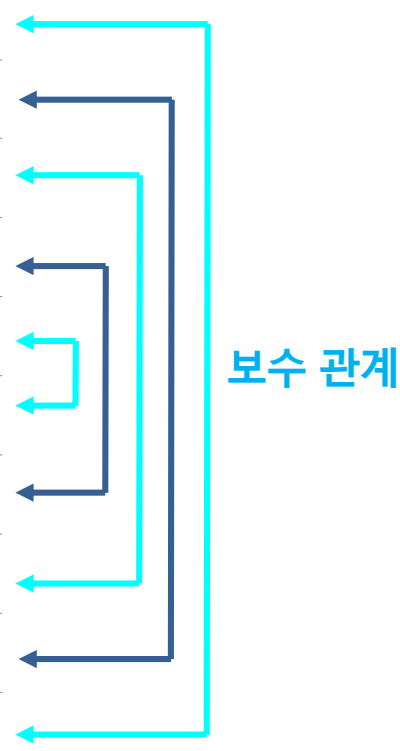
❸ 10진수 덧셈	BCD 덧셈
8	1000 _(BCD)
+ 7	+ 0111 _(BCD)
<hr/>	<hr/>
	1111 _(BCD)
	+ 0110 _(BCD)
	<hr/>
15	0001 0101 _(BCD)

2 3초과 코드(excess-3 code)

- BCD코드(8421코드)로 표현된 값에 3을 더해 준 값으로 나타내는 코드
- 자기 보수의 성질

표 2-8 3초과 코드

10진수	BCD 코드	3초과 코드
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100



보수 관계

3 그레이 코드

- 가중치가 없는 코드이기 때문에 연산에는 부적당하지만, 아날로그-디지털 변환기나 입출력 장치 코드로 주로 쓰인다.
- 연속되는 코드들 간에 하나의 비트만 변화하여 새로운 코드가 된다.

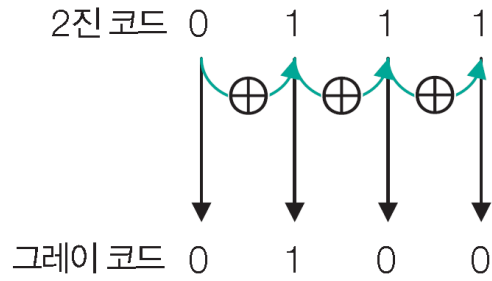
표 2-9 4비트 그레이 코드

10진 코드	2진 코드	그레이 코드	10진 코드	2진 코드	그레이 코드
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

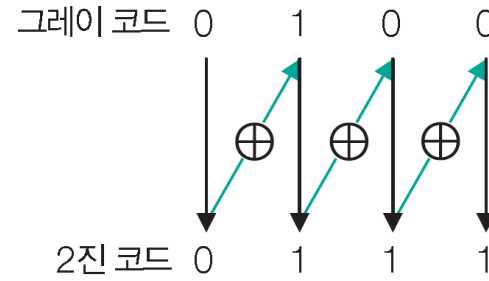
```

0 1 1 0
0 1 1 1
0 1 0 1
    
```


❖ 2진 코드와 그레이 코드의 상호 변환 방법



(a) 2진 코드를 그레이 코드로 변환하는 방법



(b) 그레이 코드를 2진 코드로 변환하는 방법

XOR의 진리표

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

$$F = A \oplus B$$

그림 2-7 2진 코드와 그레이 코드의 상호 변환 방법

4 다양한 2진 코드

- 가중치 코드 : 각 비트의 위치에 따라 값이 정해진 코드

표 2-10 가중치 코드

84-2-1은 8421로도 표현한다.

10진수	8421 코드 (BCD)	2421 코드	5421 코드	84-2-1 코드	51111 코드	바이퀴너리 코드 (biquinary code) 5043210	링 카운터 (ring counter) 9876543210
0	0000	0000	0000	0000	00000	0100001	0000000001
1	0001	0001	0001	0111	00001	0100010	0000000010
2	0010	0010	0010	0110	00011	0100100	0000000100
3	0011	0011	0011	0101	00111	0101000	0000001000
4	0100	0100	0100	0100	01111	0110000	0000010000
5	0101	1011	1000	1011	10000	1000001	0000100000
6	0110	1100	1001	1010	11000	1000010	0001000000
7	0111	1101	1010	1001	11100	1000100	0010000000
8	1000	1110	1011	1000	11110	1001000	0100000000
9	1001	1111	1100	1111	11111	1010000	1000000000

04 디지털 코드

- **비가중치 코드** : 각 위치에 해당하는 값이 없는 코드이며, 이러한 코드들은 데이터 변환 같은 특수한 용도로 사용

표 2-11 비가중치 코드

10진수	3초과 코드	5중 2코드	시프트 카운터	그레이 코드
0	0011	11000	00000	0000
1	0100	00011	00001	0001
2	0101	00101	00011	0011
3	0110	00110	00111	0010
4	0111	01001	01111	0110
5	1000	01010	11111	0111
6	1001	01100	11110	0101
7	1010	10001	11100	0100
8	1011	10010	11000	1100
9	1100	10100	10000	1101

예제 2-4 10진수 3864를 2421코드로 변환하여라.

풀이

각 자리 별로 변환하면 다음과 같다. 여기서 3, 6, 4는 각각 2가지 경우가 존재한다.

3	8	6	4
0011 or 1001	1110	1100 or 0110	0100 or 1010

End of Example

예제 2-5

74-2-1코드로 표현한 아래의 수를 10진수로 나타내어라.

① 0110 ② 1100 ③ 1001 ④ 1011

풀이

$$\textcircled{1} \ 0110 = 7 \times 0 + 4 \times 1 + (-2) \times 1 + (-1) \times 0 = 4 - 2 = 2$$

$$\textcircled{2} \ 1100 = 7 \times 1 + 4 \times 1 + (-2) \times 0 + (-1) \times 0 = 7 + 4 = 11$$

$$\textcircled{3} \ 1001 = 7 \times 1 + 4 \times 0 + (-2) \times 0 + (-1) \times 1 = 7 - 1 = 6$$

$$\textcircled{4} \ 1011 = 7 \times 1 + 4 \times 0 + (-2) \times 1 + (-1) \times 1 = 7 - 2 - 1 = 4$$

End of Example

□ ASCII 코드

- 미국 국립 표준 연구소(ANSI)가 제정한 정보 교환용 미국 표준 코드
- 128가지의 문자를 표현 가능

표 2-13 ASCII 코드

(a) 코드 구성

b ₈	b ₇ b ₆ b ₅	b ₄ b ₃ b ₂ b ₁
패리티	존 비트	디지털 비트
1	3	4

(b) 표준 ASCII 코드표

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[₩]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

□ 유니코드

- ASCII 코드의 한계성을 극복하기 위하여 개발된 인터넷 시대의 표준
- 다양한 운영체제와 최신의 모든 웹 브라우저 및 기타 많은 제품에서 유니코드를 지원한다.
- 미국, 유럽, 동아시아, 아프리카, 아시아 태평양 지역 등의 주요 언어들에 적용될 수 있다.
- 유니코드는 유럽, 중동, 아시아 등 거의 대부분의 문자를 포함하고 있으며, 10만개 이상의 문자로 구성되어 있다.
- 구두표시, 수학기호, 전문기호, 기하학적 모양, 덩벙 기호 등을 포함
- 앞으로도 계속해서 산업계의 요구나 새로운 문자들을 추가하여 나갈 것이다.

1 패리티 비트

- 짝수패리티(even parity) : 데이터에서 1의 개수를 짝수 개로 맞춤
- 홀수패리티(odd parity) : 1의 개수를 홀수 개로 맞춤
- 패리티 비트는 데이터 전송과정에서 에러 검사를 위한 추가비트
- 패리티는 단지 에러 검출만 가능하며, 여러 비트에 에러가 발생할 경우에는 검출이 안될 수도 있음

표 2-15 7비트 ASCII 코드에 패리티 비트를 추가한 코드

데이터	짝수 패리티	홀수 패리티
⋮	⋮	⋮
A	0 1000001	1 1000001
B	0 1000010	1 1000010
C	1 1000011	0 1000011
D	0 1000100	1 1000100
⋮	⋮	⋮

예제 2-6

짝수 패리티 시스템에서 코드 그룹 10110, 11010, 110011, 10101110100, 1100010101011을 수신하였다. 에러가 발생한 그룹을 찾아보아라.

풀이

- 짝수 패리티가 필요하므로 1이 홀수 개인 그룹은 에러 발생
- 10110, 11010, 1100010101011에 비트 에러가 발생하였음

End of Example

2 해밍 코드

- 에러를 정정할 수 있는 코드
- 추가적으로 많은 비트가 필요하므로 많은 양의 데이터 전달이 필요
- 데이터 비트와 패리티 비트와의 관계

$$2^p \geq d + p + 1$$

- 예를 들어 $d = 8$ 이면 $2^p \geq 8 + p + 1$ 을 만족하는 p 를 계산하면 4가 된다.
- 해밍코드에서는 짝수 패리티를 사용

표 2-16 해밍 코드에서 패리티 비트의 위치와 패리티 생성 영역

비트 위치	1	2	3	4	5	6	7	8	9	10	11	12
기호	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
P_1 영역	✓		✓		✓		✓		✓		✓	
P_2 영역		✓	✓			✓	✓			✓	✓	
P_4 영역				✓	✓	✓	✓					✓
P_8 영역								✓	✓	✓	✓	✓

□ 8비트 데이터의 에러 정정 코드

$$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11}$$

$$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11}$$

$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12}$$

$$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12}$$

for example

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
		0		0	1	0		1	1	1	0

$$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

❖ 해밍 코드에서 패리티 비트 생성 과정

표 2-17 원본 데이터가 00101110일 경우 해밍 코드 생성 예

비트 위치	1	2	3	4	5	6	7	8	9	10	11	12
기호	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
원본 데이터			0		0	1	0		1	1	1	0
생성된 코드	0	1	0	1	0	1	0	1	1	1	1	0

□ 해밍 코드에서 패리티 비트 검사 과정

전송된 데이터 : 010111011110

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
0	1	0	1	1	1	0	1	1	1	1	0

☞ 패리티들을 포함하여 검사

$$P_1 = P_1 \oplus D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$P_2 = P_2 \oplus D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$P_4 = P_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$P_8 = P_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

- 검사된 패리티를 $P_8 P_4 P_2 P_1$ 순서대로 정렬
- 모든 패리티가 0이면 에러 없음
- 하나라도 1이 있으면 에러 발생: 결과가 0101이므로 에러 있음
- 0101을 10진수로 바꾸면 5이며, 수신된 데이터에서 앞에서 5번째 비트 0101**1**1011110에 에러가 발생한 것이므로 0101**0**1011110으로 바꾸어 주면 에러가 정정된다.

05 에러 검출 코드

표 2-18 해밍 코드에서 에러가 발생한 경우 정정

비트 위치	1	2	3	4	5	6	7	8	9	10	11	12
기호	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
해밍 코드	0	1	0	1	1	1	0	1	1	1	1	0
패리티 검사	1	0		1				0				

$P_8P_4P_2P_1 = 0101_{(2)} = 5_{(10)}$: 5번째 비트에 에러가 발생. $1 \rightarrow 0$ 으로 정정

해밍 코드 수정	0	1	0	1	0	1	0	1	1	1	1	0
----------	---	---	---	---	---	---	---	---	---	---	---	---

예제 2-7

다음 해밍 코드 중 에러가 있는지 검사하여라.

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

풀이

비트 위치		1	2	3	4	5	6	7	8	9	10	11	12
기호		P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
해밍 코드		1	1	1	1	0	1	0	0	1	0	1	0
P_1 계산	0	1		1		0		0		1		1	
P_2 계산	0		1	1			1	0			0	1	
P_4 계산	0				1	0	1	0					0
P_8 계산	0								0	1	0	1	0

$P_8 P_4 P_2 P_1 = 0000$ 이므로 에러 없음

End of Example

3 순환 중복 검사(CRC)

- 높은 신뢰도를 확보하며 에러 검출을 위한 오버헤드가 적고, 랜덤 에러나 버스트 에러를 포함한 에러 검출에 매우 좋은 성능을 갖는다.

❖ CRC 발생기 및 검출기

- 수신 측에서는 수신된 $d + k$ 비트의 데이터를 키 값으로 나누었을 때 나머지가 0이면 에러가 없는 것이지만, 0이 아니면 에러가 발생한 것으로 판단한다.

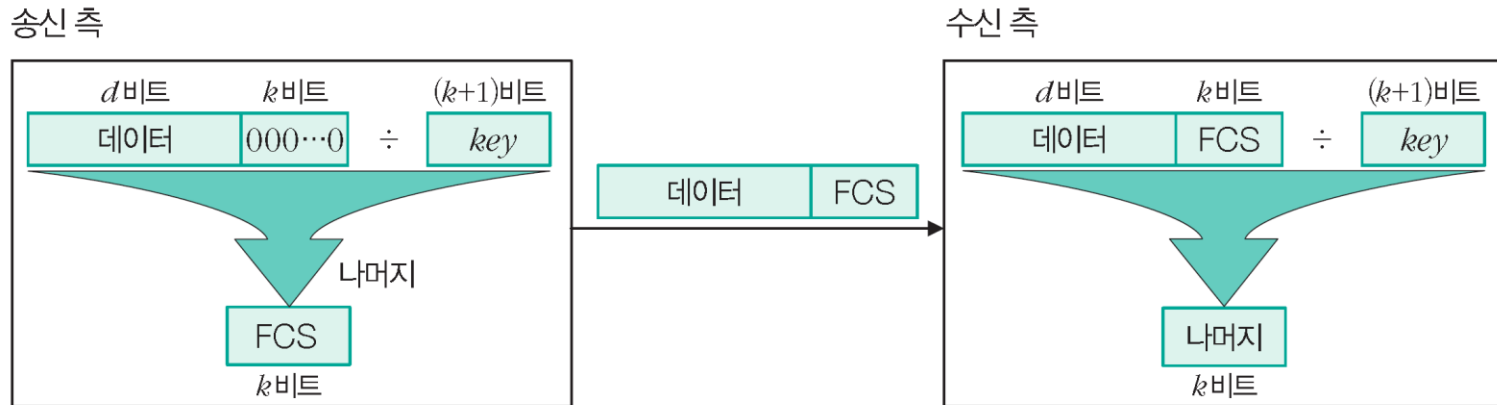


그림 2-8 CRC 발생기와 검사기

cf. FCS: Frame Check Sequence

05 에러 검출 코드

❖ CRC 계산에 사용되는 모듈로-2 연산

- 사칙 연산에서 캐리는 고려하지 않는다.
- 덧셈 연산은 뺄셈 연산과 결과가 같으며 XOR 연산과도 같다.

• 덧셈 연산: $0+0=0$, $0+1=1$, $1+0=1$, $1+1=0$

• 뺄셈 연산: $0-0=0$, $0-1=1$, $1-0=1$, $1-1=0$

- 데이터가 100100이고, 키 값이 1101인 경우 FCS를 계산하는 예

$$\begin{array}{r} 111101 \\ 1101 \overline{) 100100000} \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 1000 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 1010 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 1110 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 0110 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{0000} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 1100 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 001 \leftarrow \text{FCS} \end{array}$$

(a) CRC 발생기에서 계산 과정

$$\begin{array}{r} 111101 \\ 1101 \overline{) 100100001} \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 1000 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 1010 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 1110 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 0110 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{0000} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 1101 \downarrow \downarrow \downarrow \downarrow \downarrow \\ \underline{1101} \downarrow \downarrow \downarrow \downarrow \downarrow \\ 000 \leftarrow \text{나머지} \end{array}$$

(b) 검사기에서 계산 과정

그림 2-9 CRC 발생기와 검사기에서 2진 나눗셈

❖ CRC 하드웨어

- 시프트 레지스터와 XOR 게이트(\oplus)를 사용하여 구성할 수 있다.

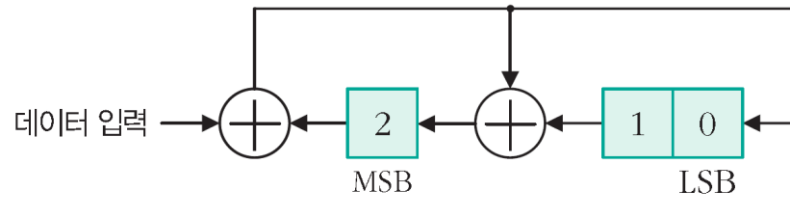


그림 2-10 FCS 생성 회로의 예

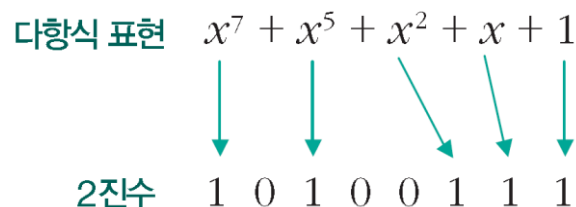
- 데이터 100100을 MSB부터 차례로 입력하면, FCS = 001이 된다.

데이터 입력	2	1	0
1 0 0 1 0 0	0	0	0
1 0 0 1 0 0	1	0	1
1 0 0 1 0 0	1	1	1
1 0 0 1 0 0	0	1	1
1 0 0 1 0 0	0	1	1
1 0 0 1 0 0	1	1	0
1 0 0 1 0 0	0	0	1

← 초깃값

❖ 생성 다항식

- Key 값을 생성하는 비트를 결정하는 다항식



- 생성 다항식은 에러 검출 능력을 고려하여 결정되는데, 현재 다음 유형이 사용된다
 - CRC-8: $x^8 + x^2 + x + 1$
 - CRC-16: $x^{16} + x^{12} + x^5 + 1$
 - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

❖ CRC 테이블 기법

- [그림 2-10]과 같이 비트 단위로 CRC를 계산하면 회로는 간단하지만 처리 속도에 문제가 있다.
- 이를 위해 미리 CRC를 계산하여 메모리에 저장해 놓고 사용한다.

Summary

- 2·8·10·16진수 등의 표현 방법, 상호 변환 방법
- 2진수의 연산과 2진수 음수의 표현 방법
- 부동 소수점 2진수를 IEEE 754 표준 방식으로 표현
- 가중치 코드와 비가중치 코드 이해
- 에러 검출 코드 이해