04

디지털공학개론

수의 연산 및 수치 데이터

04

수의 연산 및 수치 데이터

1.수의 연산

2.2진수의 정수 표현법(고정 소수점 표현)

3.2진수의 실수 표현법(부동 소수점 표현)

1. 2진수의 사칙 연산

- 디지털 시스템에서의 수의 연산 방법은 10진수와 같이 사칙연산 방법 사용
- 연산 방법 10진 연산 과정과 같으나 뺄셈 연산은 직접 하지 못함
- 보수 개념을 이용하여 음수를 표현하고 덧셈 과정으로 처리 한다는 차이점이 있음
- 곱셈과 나눗셈은 하드웨어적 처리보다는 소프트웨어 적으로 처리하는 경우가 많음

1. 2진수의 사칙 연산

덧셈(Addition)

- 2진수의 덧셈은 모든 사칙 연산의 기본
- 덧셈의 결과로서 합(Sum)과 올림수(Carry) 출력

$$\begin{array}{r}0\\+\ 0\\\hline \boxed{0}\end{array}\quad \begin{array}{r}1\ 0\\+\ 1\\\hline \boxed{1}\end{array}\quad \begin{array}{r}1\\+\ 0\\\hline \boxed{1}\end{array}\quad \begin{array}{r}1\\+\ 1\\\hline 1\ \boxed{0}\\ \text{carry}\end{array}\qquad \begin{array}{r}13_{(10)}\\+\ 6_{(10)}\\\hline 19_{(10)}\end{array}\qquad \begin{array}{r}1\ 1\ 0\ 1\\+\ \ \ 1\ 1\ 0\\\hline 1\ 0\ 0\ 1\ 1\end{array}$$

뺄셈(Subtraction)

- 2진수의 뺄셈은 차(Difference)와 빌림수(Borrow) 출력

$$\begin{array}{r}0\\-\ 0\\\hline \boxed{0}\end{array}\quad \begin{array}{r}\text{borrow}\\1\ 0\\-\ 1\\\hline \boxed{1}\end{array}\quad \begin{array}{r}1\\-\ 0\\\hline \boxed{1}\end{array}\quad \begin{array}{r}1\\-\ 1\\\hline \boxed{0}\end{array}\qquad \begin{array}{r}13_{(10)}\\-\ 6_{(10)}\\\hline 7_{(10)}\end{array}\qquad \begin{array}{r}1\ 1\ 0\ 1\\-\ \ \ 1\ 1\ 0\\\hline 0\ 1\ 1\ 1\end{array}$$

◆ R진법의 덧셈

10진수

```
Carry →  11
          49
    +     58
    ---------
         107
```

2진수

```
Carry →  0110000
         00110001
     +   00111010
     ------------
         01101011
```

8진수

```
Carry →  10
          61
    +     72
    ---------
         153
```

16진수

```
Carry →   0
          31
    +     3A
    ---------
          6B
```

1. 2진수의 사칙 연산

곱셈(하드웨어적 처리)

- 피승수 값과 승수 값의 부분적과 Shift(자리이동) 기능을 수행하여 계산
- 가장 하위 자리부터 검사하여 승수 값이 1이면 부분적(Partial product)으로 피승수를 그대로 쓰고, 그 값이 0이면 비트 수만큼의 0을 채움
- 두 번째 비트에 대해서도 동일한 동작을 수행하지만 이 때는 비트 좌측으로 이동한 자리에 쓴다는 것이 다름

$$
\begin{array}{r}
0 \\
\times\ 0 \\
\hline
\boxed{0}
\end{array}
\qquad
\begin{array}{r}
0 \\
\times\ 1 \\
\hline
\boxed{0}
\end{array}
\qquad
\begin{array}{r}
1 \\
\times\ 0 \\
\hline
\boxed{0}
\end{array}
\qquad
\begin{array}{r}
1 \\
\times\ 1 \\
\hline
\boxed{1}
\end{array}
$$

$$
\begin{array}{r}
9_{(10)} \\
\times\ 6_{(10)} \\
\hline
54_{(10)}
\end{array}
\qquad
\begin{array}{rrrrrrr}
 & & & 1 & 0 & 0 & 1 \\
 & + & & & 1 & 1 & 0 \\
\hline
 & & & 0 & 0 & 0 & 0 \\
 & & 1 & 0 & 0 & 1 & \\
 & 1 & 0 & 0 & 1 & & \\
\hline
 & 1 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

1. 2진수의 사칙 연산

나눗셈(하드웨어적 처리)

- D ÷ V = Q .. R
- D = 피제수
- V = 제수

$$
\begin{array}{r}
12_{(10)} \\
\div\ 2_{(10)} \\
\hline
6_{(10)}
\end{array}
\qquad
\begin{array}{l r}
Q \rightarrow & 1\ 1\ 0 \\
\hline
D \rightarrow & 1\ 1\ 0\ 1 \\
V \rightarrow & 1\ 0\ \\
\hline
 & 1\ 0\ \\
 & 1\ 0\ \\
\hline
R \rightarrow & 0
\end{array}
$$

2. 보수(Complement)에 의한 2진수의 감산

- 보수(Complement Number) 음수를 표현하는 방법
- 디지털 시스템에서 뺄셈 연산은 보수를 사용한 덧셈 과정
- M - S = M + (-S) 여기서, M: 감수, S: 피감수
  ex) 10 - 5 = 10 + (-5) = 5
- 양수에서 양수를 빼는 개념이 아니라, 음수를 취해서 양수와 더해지는 개념을 사용

2. 보수(Complement)에 의한 2진수의 감산

(1) 보수 (Complement)

R의 보수와 R-1의 보수

R의 보수	R-1의 보수
$R^m - K$	$(R^m - 1) - K$

2. 보수(Complement)에 의한 2진수의 감산

(1) 보수 (Complement)

ex) 10진수 456의 10의 보수와 9의 보수를 구하시오.

- 10의 보수: $R^m - K = 10^3 - 456 = 544$
- 9의 보수: $(R^m - 1) - K = (10^3 - 1) - 456 = 999 - 456 = 543$

ex) 2진수 10101의 2의 보수와 1의 보수를 구하시오.

- 2의 보수: $R^m - K = 2^5 - 10101 = 100000 - 10101 = 01011$
- 1의 보수: $(R^m - 1) - K = (2^5 - 1) - 10101 = 11111 - 10101 = 01010$

1. 2진수의 사칙 연산

(2) 보수(Complement)에 의한 2진수의 감산

1의 보수에 의한 뺄셈

- 피감수(Minuend) M에 감수(Subtrahend) S의 1의 보수를 더함
- 최상위 비트에서의 자리올림이 발생하면 최하위 숫자에 1을 더함,
  만약 자리 올림이 발생하지 않으면 (-) 부호를 붙이고 다시 1의 보수를 취함

2의 보수에 의한 뺄셈

- 피감수(Minuend) M에 감수(Subtrahend) S의 2의 보수를 더함
- 최상위 비트에서의 자리올림이 발생하면 그 것을 무시하고 계산 종료되며,
  자리올림이 (-) 부호를 붙이고 다시 2의 보수를 취함

2. 보수(Complement)에 의한 2진수의 감산

(3) 보수(Complement)에 의한 2진수의 감산의 예

- 1의 보수를 이용한 25-19와 19-25 계산

+25	$11001_{(2)}$			
+19	$10011_{(2)}$	−19	$01101_{(2)}$	(2의 보수)
	$11001_{(2)}$		①$00110_{(2)}$	
+	$01101_{(2)}$		(무시)	
	$100110_{(2)}$		$00110_{(2)}$	(계산 결과)

+19	$10011_{(2)}$			
+25	$11001_{(2)}$	−25	$00110_{(2)}$	(1의 보수)
	$10011_{(2)}$			
+	$00110_{(2)}$	−	$00110_{(2)}$	(계산 결과)
	$11001_{(2)}$		(1의 보수)	

2. 보수(Complement)에 의한 2진수의 감산

(3) 보수(Complement)에 의한 2진수의 감산의 예

- 2의 보수를 이용한 25-19와 19-25 계산

$+25 \quad 11001_{(2)}$

$+19 \quad 10011_{(2)} \qquad -19 \quad 01101_{(2)}$ (2의 보수)

$$\begin{array}{r} 11001_{(2)} \\ +\ 01101_{(2)} \\ \hline 100110_{(2)} \end{array}$$

$100110_{(2)}$ → (무시)

$00110_{(2)}$ (계산 결과)

$+19 \quad 10011_{(2)}$

$+25 \quad 11001_{(2)} \qquad -25 \quad 00111_{(2)}$ (2의 보수)

$$\begin{array}{r} 10011_{(2)} \\ +\ 00111_{(2)} \\ \hline 11010_{(2)} \end{array}$$

$- \ 00110_{(2)}$ (계산 결과)

(2의 보수)

04

수의 연산 및 수치 데이터

1.수의 연산

2.2진수의 정수 표현법(고정 소수점 표현)

3.2진수의 실수 표현법(부동 소수점 표현)

1. 2진 음의 정수 표현

(1) 최상위비트(MSB)를 부호비트로 사용

- 양수(+) : 0   음수(-) : 1

(2) 2진 음수를 표시하는 방법

- 부호와 절대치(sign- magnitude)
- 1의 보수(1's complement)
- 2의 보수(2’s complement)

Tip) 1의 보수로 변환하는 방법 : 0 → 1, 1 → 0으로 변환

2의 보수로 변환하는 방법 : 1의 보수 + 1 = 2의 보수

2. 2진 음의 정수의 3가지 표현 방법

$b_7b_6b_5b_4b_3b_2b_1b_0$	8bit 크기이며, MSB가 부호비트 임.		
	부호와 절대치	1의 보수	2의 보수
01111111	+127	+127	+127
01111110	+126	+126	+126
01111101	+125	+125	+125
…	…	…	…
00000010	+2	+2	+2
00000001	+1	+1	+1
00000000	+0	+0	+0
10000000	-0	-127	-128
10000001	-1	-126	-127
10000010	-2	-125	-126
…	…	…	…
11111101	-125	-2	-3
11111110	-126	-1	-2
11111111	-127	-0	-1

3. 2의 보수를 사용한 2진 정수의 표현 범위

bit 수	2의 보수를 사용한 2진 정수의 표현 범위
n bit	$-2^{n-1} \sim +2^{n-1}-1$
4 bit	$-2^{4-1} \sim +2^{4-1}-1$ (-8 ~ +7)
8 bit	$-2^{8-1} \sim +2^{8-1}-1$ (-128 ~ +127)
16 bit	$-2^{16-1} \sim +2^{16-1}-1$ (-32768 ~ +32767)
32 bit	$-2^{32-1} \sim +2^{32-1}-1$ (- 2147483648 ~ + 2147483647)

3. 2의 보수를 사용한 2진 정수의 표현 범위

- 부호 확장이란 늘어난 비트 수 만큼 부호를 늘려주는 방법

<table>
<tr><th rowspan="2">2진수 표현 방법</th><th rowspan="2">부호 확장 방법</th><th colspan="3">예</th></tr>
<tr><th></th><th>8bit</th><th>16bit</th></tr>
<tr><td rowspan="2">부호와 절대치</td><td rowspan="2">부호만 MSB에 복사하고, 나머지는 0으로 채움</td><td>+</td><td>00101010</td><td>00000000 00101010</td></tr>
<tr><td>-</td><td>10010111</td><td>10000000 00010111</td></tr>
<tr><td rowspan="2">1의 보수</td><td rowspan="2">늘어난 길이만큼 부호와 같은 값으로 모두 채움</td><td>+</td><td>00101010</td><td>00000000 00101010</td></tr>
<tr><td>-</td><td>10010111</td><td>11111111 10010111</td></tr>
<tr><td rowspan="2">2의 보수</td><td rowspan="2">늘어난 길이만큼 부호와 같은 값으로 모두 채움</td><td>+</td><td>00101010</td><td>00000000 00101010</td></tr>
<tr><td>-</td><td>10010111</td><td>11111111 10010111</td></tr>
</table>

04

수의 연산 및 수치 데이터

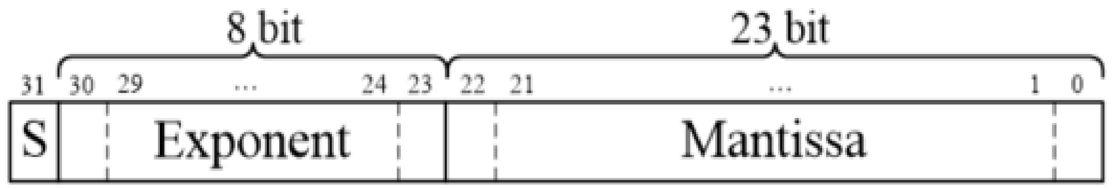
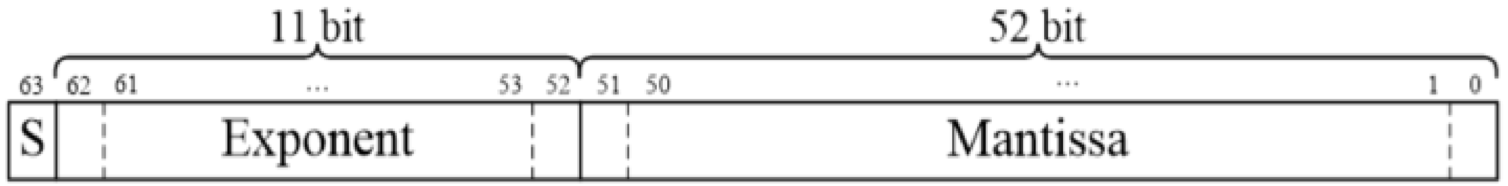
1.수의 연산

2.2진수의 정수 표현법(고정 소수점 표현)

3.2진수의 실수 표현법(부동 소수점 표현)

1. 2진 부동 소수점의 표현

- 컴퓨터의 부동소수점수는 IEEE 754표준을 따른다.
- 부호(sign), 지수(exponent), 가수(mantissa)의 세 영역으로 표시
- 단정도(single precision) 부동소수점수와 배정도(double precision) 부동소수점수의 두 가지 표현 방법이 있다.

구분	IEEE 754 표준 부동소수점수의 비트 할당	바이어스
단정도 부동소수점수		127
배정도 부동소수점수		1023

2. 정규화(Normalization)

(1) 2진수의 정규화

$$
\begin{aligned}
69.6875 &= 1000101.1011_{(2)} \\
&= 1.0001011011 \times 2^{6}{}_{(2)} \\
&= 1.0001011011 \times 2^{110_{(2)}}{}_{(2)}
\end{aligned}
$$

2. 정규화(Normalization)

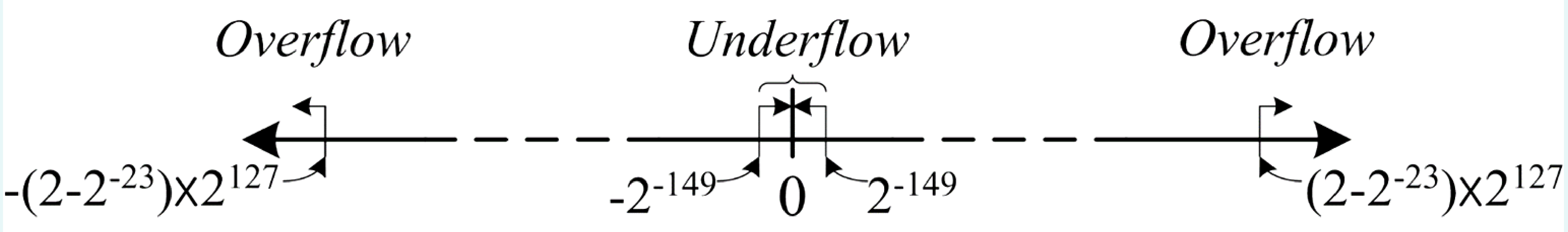
(2) 바이어스(bias) : 지수의 양수, 음수를 나타내기 위한 방법

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

부호 : 1Bit	지수(Bias 127) : 8비트	가수(1.xxx) : 23비트
양수	127 + 6 = 133 (01111111 + 00000110)	1.을 생략한 가수 (1.0001011011)
0	10000101	00010110110000000000000

3. 컴퓨터에서의 부동소수점수의 표현 범위

	단정도 부동소수점수	배정도 부동소수점수
비정규화된 2진수	~±2^{-149} to ±(1-2^{-23}) x 2^{126}	~±2^{-1074} to ±(1-2^{-52}) x 2^{1022}
정규화된 2진수	~±2^{-126} to ±(2-2^{-23}) x 2^{127}	~±2^{-1022} to ±(2-2^{-52}) x 2^{1023}
10진수	~±1.40 x 10^{45} to ±3.40 x 10^{38}	~±4.94 x 10^{-324} to ±1.798 x 10^{308}



< 단정도 부동소수점수의 표현 범위 >

04

디지털 정보의 표현

- 학습정리

● 2진수의 덧셈과 뺄셈

덧셈(Addition)

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline \boxed{0} \end{array} \quad \begin{array}{r} 0 \\ +\ 1 \\ \hline \boxed{1} \end{array} \quad \begin{array}{r} 1 \\ +\ 0 \\ \hline \boxed{1} \end{array} \quad \begin{array}{r} 1 \\ +\ 1 \\ \hline 1\,\boxed{0} \end{array} \qquad \begin{array}{r} 13_{(10)} \\ +\ 6_{(10)} \\ \hline 19_{(10)} \end{array} \qquad \begin{array}{r} 1\ 1\ 0\ 1 \\ +\ \ 1\ 1\ 0 \\ \hline 1\ 0\ 0\ 1\ 1 \end{array}$$

carry

뺄셈(Subtraction)

borrow

$$\begin{array}{r} 0 \\ -\ 0 \\ \hline \boxed{0} \end{array} \quad \begin{array}{r} 1\ 0 \\ -\ 1 \\ \hline \boxed{1} \end{array} \quad \begin{array}{r} 1 \\ -\ 0 \\ \hline \boxed{1} \end{array} \quad \begin{array}{r} 1 \\ -\ 1 \\ \hline \boxed{0} \end{array} \qquad \begin{array}{r} 13_{(10)} \\ -\ 6_{(10)} \\ \hline 7_{(10)} \end{array} \qquad \begin{array}{r} 1\ 1\ 0\ 1 \\ -\ \ 1\ 1\ 0 \\ \hline 0\ 1\ 1\ 1 \end{array}$$

● 2진수의 덧셈과 뺄셈

1의 보수에 의한 뺄셈

- 피감수(Minuend) M에 감수(Subtrahend) S의 1의 보수를 더함
- 최상위 비트에서의 자리올림이 발생하면 최하위 숫자에 1을 더함,
  만약 자리 올림이 발생하지 않으면 (-) 부호를 붙이고 다시 1의 보수를 취함

2의 보수에 의한 뺄셈

- 피감수(Minuend) M에 감수(Subtrahend) S의 2의 보수를 더함
- 최상위 비트에서의 자리올림이 발생하면 그 것을 무시하고 계산 종료되며,
  자리올림이 (-) 부호를 붙이고 다시 2의 보수를 취함

● 2진 음의 정수 표현

(1) 최상위비트(MSB)를 부호비트로 사용

- 양수(+) : 0   음수(-) : 1

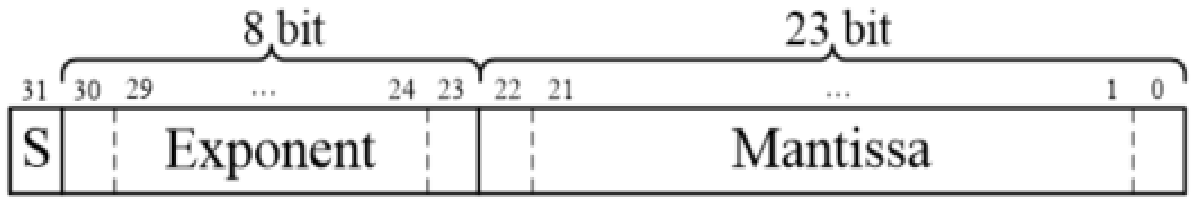
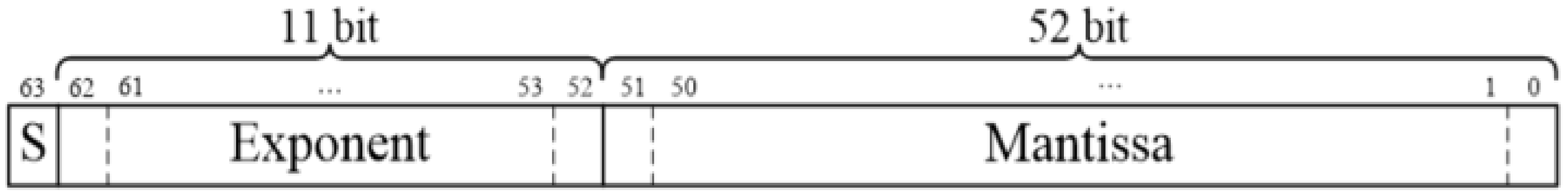
(2) 2진 음수를 표시하는 방법

- 부호와 절대치(sign- magnitude)
- 1의 보수(1's complement)
- 2의 보수(2’s complement)

$b_7b_6b_5b_4b_3b_2b_1b_0$	8bit 크기이며, MSB가 부호비트 임.		
	부호와 절대치	1의 보수	2의 보수
01111111	+127	+127	+127
01111110	+126	+126	+126
01111101	+125	+125	+125
···	···	···	···
00000010	+2	+2	+2
00000001	+1	+1	+1
00000000	+0	+0	+0
10000000	-0	-127	-128
10000001	-1	-126	-127
10000010	-2	-125	-126
···	···	···	···
11111101	-125	-2	-3
11111110	-126	-1	-2
11111111	-127	-0	-1

학습정리

- **2진 부동 소수점의 표현**

구분	IEEE 754 표준 부동소수점수의 비트 할당	바이어스
단정도 부동소수점수		127
배정도 부동소수점수		1023