



논리 설계 및 실험

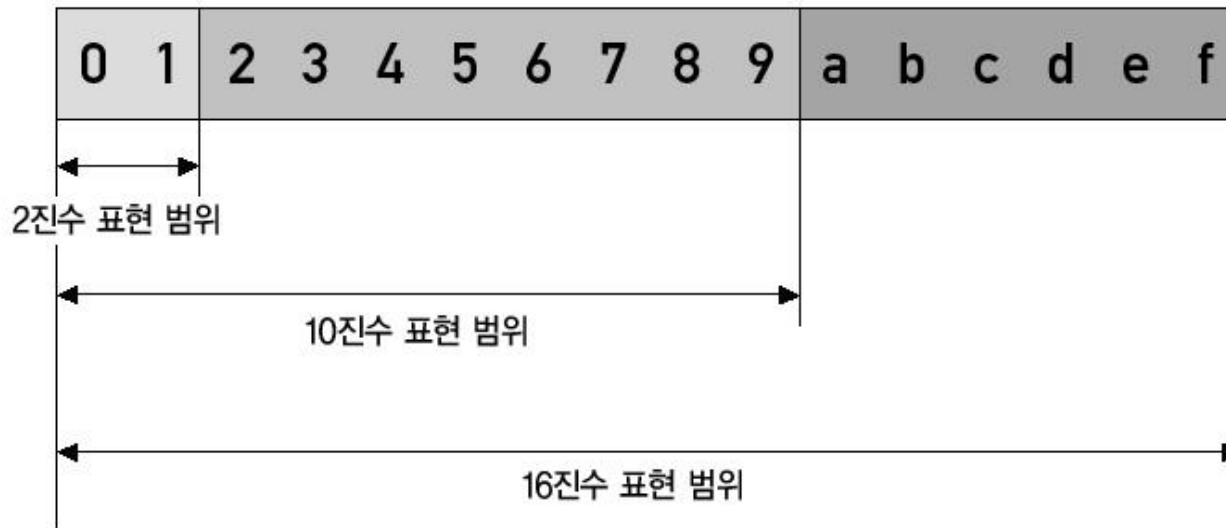
Date : 2016 . 03 . 25

김영대

4-1 컴퓨터의 데이터 표현

■ 진법에 대한 이해

- n 진수 표현 방식 : n 개의 문자를 이용해서 데이터를 표현



■ 2진수와 10진수

- 10진수 : 0~9를 이용한 데이터의 표현
- 2진수 : 0과 1을 이용한 데이터의 표현
- 컴퓨터는 내부적으로 모든 데이터 2진수로 처리

	2진수	10진수
	0	0
	1	1
자릿수 증가	1 0	2
	1 1	3
자릿수 증가	1 0 0	4
	1 0 1	5

■ 16진수와 10진수

- 16진수 : 0~9, a, b, c, d, e, f를 이용한 데이터의 표현

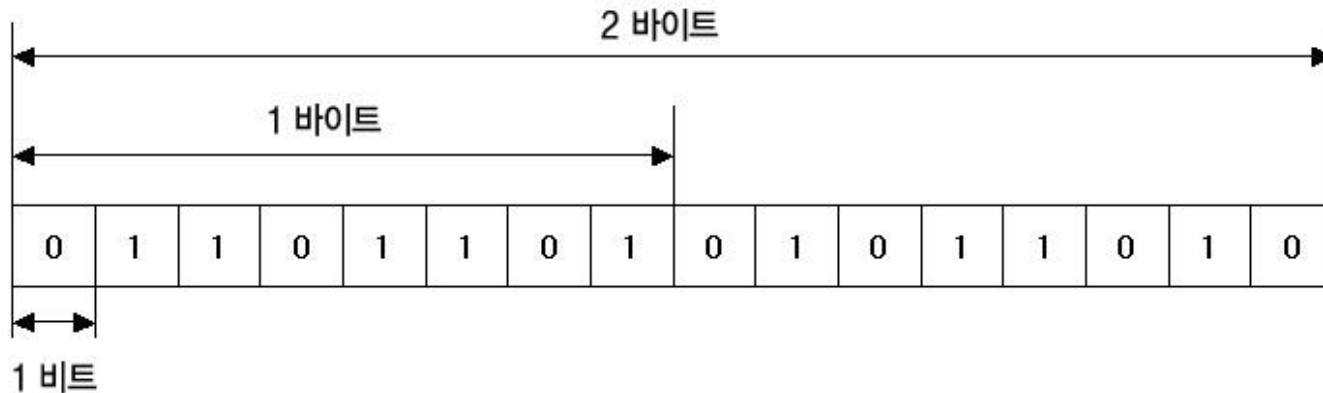
10진수	16진수
9	9
10	a
11	b
12	c
13	d
14	e
15	f
16	10
17	11

자릿수 증가

자릿수 증가

■ 데이터의 표현 단위인 비트(bit)와 바이트(byte)

- 비트 : 데이터 표현의 최소 단위, 2진수 값 하나 (0 or 1)을 저장
- 바이트 : 8비트 == 1 바이트



■ 연습문제

0	0	0	0	0	0	0	0	[]
0	0	0	0	0	0	0	1	[]
0	0	0	0	0	0	1	0	[]
0	0	0	0	0	1	0	0	[]
0	0	0	0	1	0	0	0	[]
0	0	0	1	0	0	0	0	[]
0	0	1	0	0	0	0	0	[]
0	1	0	0	0	0	0	0	[]
1	0	0	0	0	0	0	0	[]

■ 프로그램상에서의 8진수, 16진수 표현

- 8진수 : 0으로 시작
- 16진수 : 0x로 시작

```
int a = 10;           // 10진수. 아무런 표시도 없으므로...  
int b = 0xa;          // 16진수. 0x로 시작하므로...  
int c = 012;          // 8진수. 0으로 시작하므로...
```

```

#include <stdio.h>

int main(void)
{
    int a = 0xa7;
    int b = 0x43;

    int c = 032;
    int d = 024;

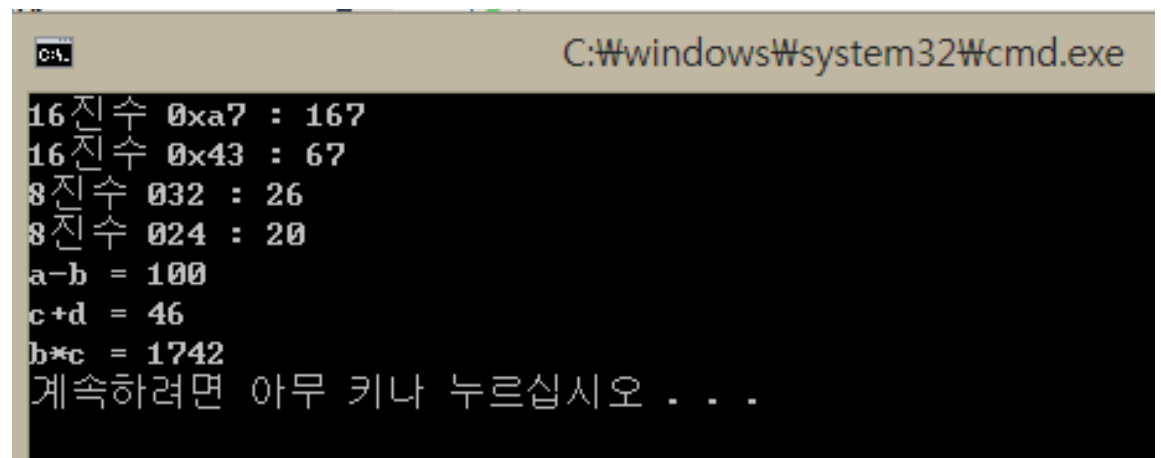
    printf("16진 수 0xa7 : %d\n", a);
    printf("16진 수 0x43 : %d\n", b);

    printf("8진 수 032 : %d\n", c);
    printf("8진 수 024 : %d\n", d);

    printf("a-b = %d\n", a-b);
    printf("c+d = %d\n", c+d);
    printf("b*c = %d\n", b*c);

    return 0;
} //end main

```



```

C:\windows\system32\cmd.exe
16진 수 0xa7 : 167
16진 수 0x43 : 67
8진 수 032 : 26
8진 수 024 : 20
a-b = 100
c+d = 46
b*c = 1742
계속하려면 아무 키나 누르십시오 . . .

```


4-2 정수의 표현 방식

■ 정수의 표현 방식

- MSB : 가장 왼쪽 비트, 부호를 표현
- MSB를 제외한 나머지 비트 : 데이터의 크기 표현



■ 잘못된 음의 정수 표현 방식

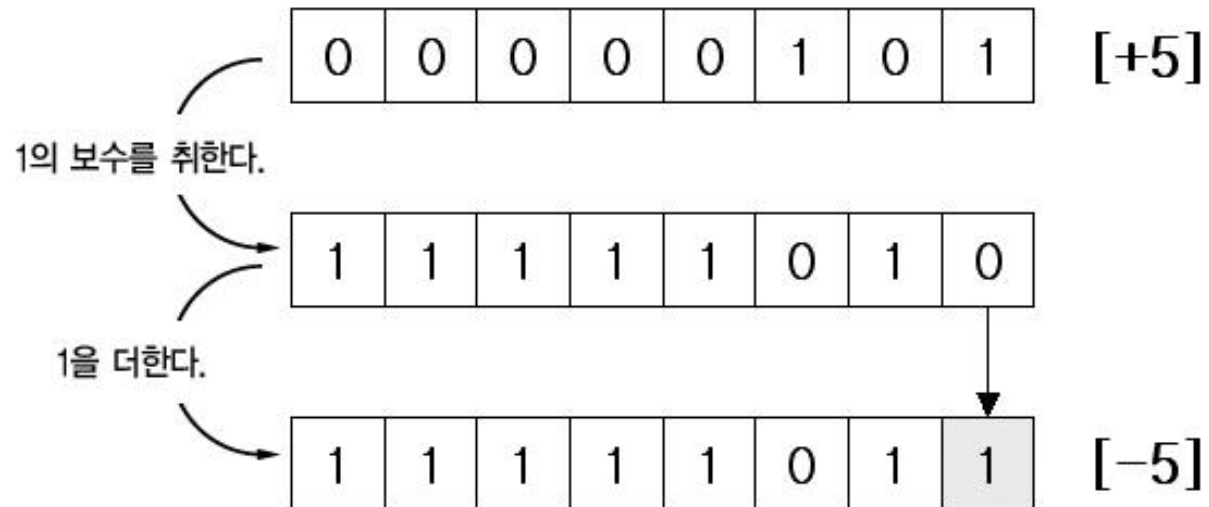
■ 양의 정수 표현 방식을 적용한 경우

$$\begin{array}{r} \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline \end{array} & [+5] \\ + & \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline \end{array} & [-5] \\ \hline \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ \hline \end{array} & [???] \end{array}$$

+5와 -5를
더하면???

■ 정확한 음의 정수 표현 방식

■ 2의 보수를 이용한 음의 정수 표현 방식



■ 음수 표현 방식의 증명

	0	0	0	0	0	1	0	1	[+5]
+	1	1	1	1	1	0	1	1	[-5]
<hr/>									
	1	0	0	0	0	0	0	0	[0]

올림 수(carry)는 버려진다.

4-3 비트 단위 연산

■ 비트 단위 연산자의 종류

연산자	연산자의 의미	결합성
&	비트 단위 AND ex) a & b	→
	비트 단위 OR ex) a b	
^	비트 단위 XOR ex) a ^ b	
~	비트 단위 NOT ex) ~a	
<<	왼쪽으로 이동 ex) a << 2	
>>	오른쪽으로 이동 ex) a >> 2	

■ & 연산자 : 비트 단위 AND

0 & 0 → 0을 반환

0 & 1 → 0을 반환

1 & 0 → 0을 반환

1 & 1 → 1을 반환

```
int main(void)
{
    int a=15;      // 00000000 00000000 00000000 00001111
    int b=20;      // 00000000 00000000 00000000 00010100
    int c = a&b;

    printf("AND 연산 결과 : %d", c); // 출력 결과 4
}
```

■ & 연산자 : 비트 단위 AND

	00000000	00000000	00000000	00001111
AND	00000000	00000000	00000000	00010100
<hr/>				
	00000000	00000000	00000000	00000100

■ | 연산자 : 비트 단위 OR

0 | 0 → 0을 반환

0 | 1 → 1을 반환

1 | 0 → 1을 반환

1 | 1 → 1을 반환

```
int main(void)
{
    int a=15; // 00000000 00000000 00000000 00001111
    int b=20; // 00000000 00000000 00000000 00010100
    int c = a|b;

    printf("OR 연산 결과 : %d", c);    // 출력 결과 31
}
```


■ | 연산자 : 비트 단위 OR

	00000000	00000000	00000000	00001111
OR	00000000	00000000	00000000	00010100
<hr/>				
	00000000	00000000	00000000	00011111

■ ^ 연산자 : 비트 단위 XOR

$0 \wedge 0 \rightarrow 0$ 을 반환

$0 \wedge 1 \rightarrow 1$ 을 반환

$1 \wedge 0 \rightarrow 1$ 을 반환

$1 \wedge 1 \rightarrow 0$ 을 반환

```
int main(void)
{
    int a=15; // 00000000 00000000 00000000 00001111
    int b=20; // 00000000 00000000 00000000 00010100
    int c = a^b;

    printf("XOR 연산 결과 : %d", c); // 출력 결과 27
}
```

■ ^ 연산자 : 비트 단위 XOR

	00000000	00000000	00000000	00001111
XOR	00000000	00000000	00000000	00010100
<hr/>				
	00000000	00000000	00000000	00011011

■ ~ 연산자 : 비트 단위 NOT

~ 0 → 1을 반환

~ 1 → 0을 반환

```
int main(void)
{
    int a=15;
    int b=~a;

    printf("NOT 연산 결과 : %d", b); // 출력 결과 -16
}
```

■ ~ 연산자 : 비트 단위 NOT

NOT	00000000	00000000	00000000	00001111
	11111111	11111111	11111111	11110000

부호 비트도 반전

■ << 연산자 : 왼쪽 쉬프트(shift) 연산

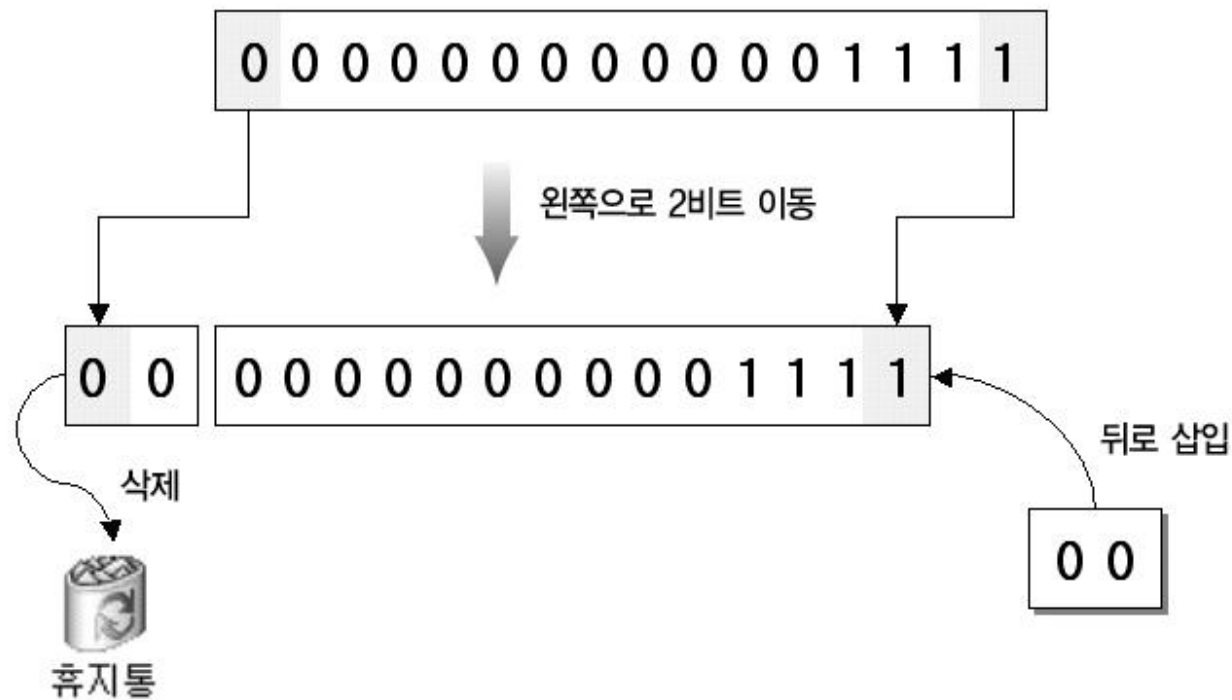
$a \ll b \rightarrow$ a의 비트들을 b칸씩 왼쪽으로 이동한 값을 반환

$8 \ll 2 \rightarrow$ 8의 비트들을 왼쪽으로 2칸씩 이동한 값을 반환

```
int main(void)
{
    int a=15; // 00000000 00000000 00000000 00001111
    int b= a<<2; // a의 비트들을 왼쪽으로 2칸씩 이동

    printf("<<2 연산 결과 : %d", b); // 출력 결과 60
}
```

■ << 연산자 : 왼쪽 쉬프트(shift) 연산



■ >> 연산자 : 오른쪽 쉬프트(shift) 연산

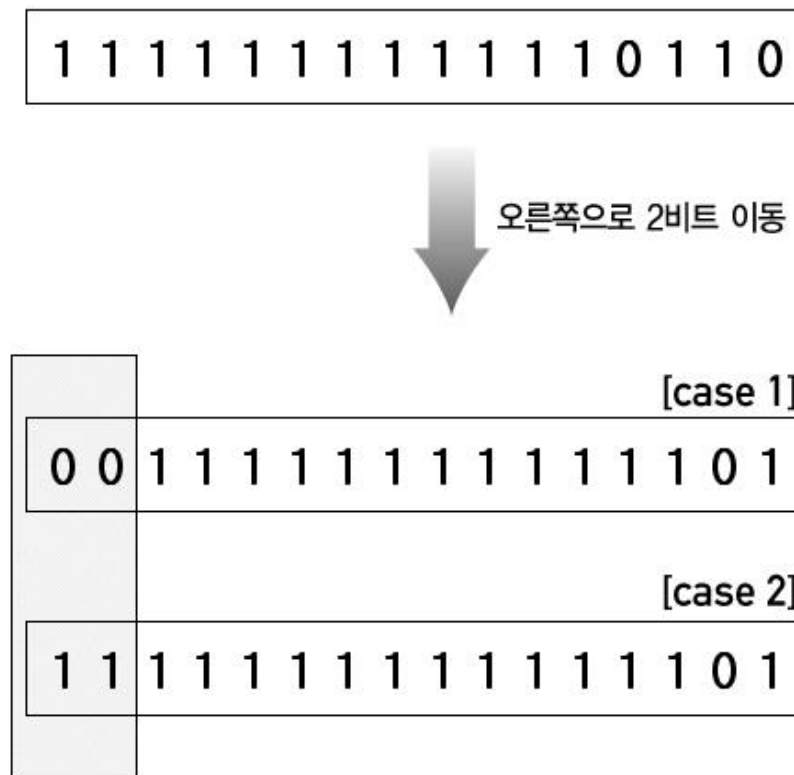
$a \gg b \rightarrow$ a의 비트들을 b칸씩 오른쪽으로 이동한 값을 반환

$8 \gg 2 \rightarrow$ 8의 비트를 왼쪽으로 2칸씩 이동한 값을 반환

```
a=-10;
```

```
b=a>>2; // a의 비트들을 2칸씩 오른쪽으로 이동한 값을 b에 저장
```


- >> 연산자 : 오른쪽 쉬프트(shift) 연산



예제

```
#include <stdio.h>

int main(void)
{
    int a = 0xa7;
    int b = 0x43;

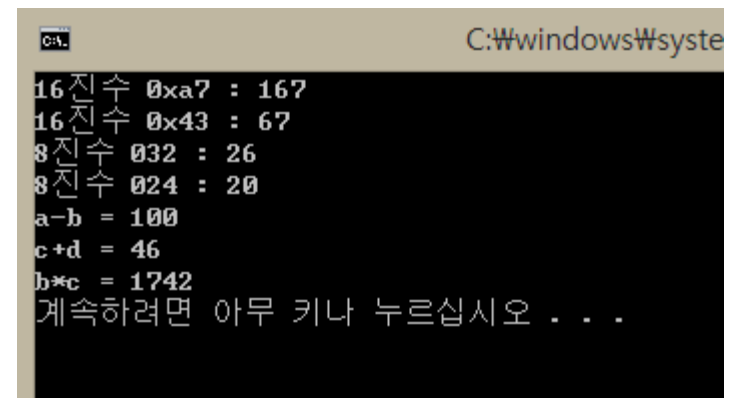
    int c = 032;
    int d = 024;

    printf("16진수 0xa7 : %d \n", a);
    printf("16진수 0x43 : %d \n", b);

    printf("8진수 032 : %d \n", c);
    printf("8진수 024 : %d \n", d);

    printf("a-b = %d \n", a-b);
    printf("c+d = %d \n", c+d);
    printf("b*c = %d \n", b*c);

    return 0;
} //end main
```



```
C:\windows\system32\cmd.exe
16진수 0xa7 : 167
16진수 0x43 : 67
8진수 032 : 26
8진수 024 : 20
a-b = 100
c+d = 46
b*c = 1742
계속하려면 아무 키나 누르십시오 . . .
```

예제

```
#include <stdio.h>

int main(void)
{
    int a = 15;
    int b = 20;
    int c = a&b;

    printf("AND 연산 결과 : %d \n", c);

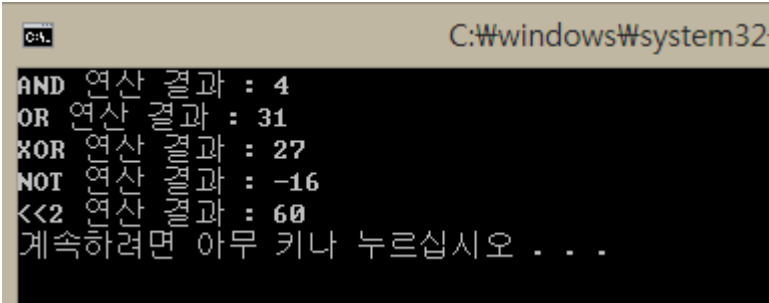
    c = a|b;
    printf("OR 연산 결과 : %d \n", c);

    c = a^b;
    printf("XOR 연산 결과 : %d \n", c);

    b = 0;
    b = ~a;
    printf("NOT 연산 결과 : %d \n", b);

    b = 0;
    b = a<<2; //shift 연산
    printf("<<2 연산 결과 : %d \n", b);

    return 0;
} //end main
```



```
C:\Windows\system32
AND 연산 결과 : 4
OR 연산 결과 : 31
XOR 연산 결과 : 27
NOT 연산 결과 : -16
<<2 연산 결과 : 60
계속하려면 아무 키나 누르십시오 . . .
```