3. 이진수binary number.

• 모든 정수는 짝수와 홀수로 나누어 생각할 수 있습니다. 짝수의 대표로 0, 홀수의 대표로 1을 뽑아 봅니다. 이 두 수 사이의 덧셈과 곱셈을 생각하면 다음 표와 같이 나타납니다.

+	0	1
0	0	1
1	1	2

×	0	1
0	0	0
1	0	1

• 위의 표를 보면 1+1=2인 경우만을 제외하고 계산 결과는 모두 0 아니면 1입니다.

+	0	1
0	0	1
1	1	2

×	0	1
0	0	0
1	0	1

- 2는 짝수이기 때문에 이것을 짝수의 대표 0으로 바꾸어 놓아 봅시다.

$$1+1=0$$

- 그러면 이 덧셈표는 아래 표와 같이 됩니다.

+	0	1
0	0	1
1	1	0

보충해 주는 수: 보수(complement of a number)

• r진수 m자리수 n의 **보수**는 아래와 같습니다.

$$\mathbf{r^m-n}$$

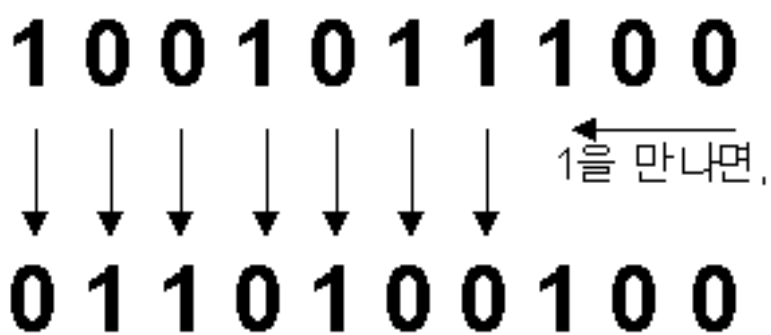
• 10진수에 대해서 생각해 보면, 7의 보수는 10^1-7=3입니다. 77의 보수는 10^2-77=33입니다.

• 10이나 100을 만들기 위해 보충해야 하는 수가 각각 3과 33이라는 의미입니다.

• 보수는 뺄셈을 위해 사용할 수 있습니다. 예를 들어 10진수 8-3을 계산한다고 하면, 8+(3의 보수)=8+7=15 인데 1자리의 뺄셈이므로 십의 자리 1을 버리면, 8+7=5가 되어 원하는 결과를 얻습니다.

• 보수의 이러한 성질 때문에 컴퓨터는 음의 정수(Integer)를 나타내기 위해 보수를 사용합니다.

• 2진수 표현에서 **1의 보수(1's Complement)**와 **2의 보수(2's Complement)**를 구하는 것은 쉽습니다. 1의 보수는 단순히 1과 0을 토글(Toggle)시킴으로 구할 수 있고, 2의 보수는 1의 보수를 구한 다음, 결과에 1을 더하면 구할 수 있습니다.
• 예를 들어, 1001011100의 1의 보수는 0110100011이며, 2의 보수는 0110100011 + 1 = 0110100100입니다.



2의 보수를 구하는 법

- 2진수 n비트로 나타낼 수 있는 수의 범위는 다음과 같습니다.

$$0 \sim 2^n-1$$ ■

> 2개에서 n개를 뽑아내는 중복 순열의 수이므로 ${}_2\Pi_n=2^n$입니다. 숫자는 0부터 시작하므로 최고 큰 수는 1을 빼주어야 합니다.

- unsigned char형은 $0 \sim 2^8-1$, 즉 $0 \sim 255$를 나타낼 수 있고
- unsigned short int형은 2바이트이므로, $0 \sim 2^{16}-1$, 즉 $0 \sim 65535$를 나타낼 수 있습니다.
- 음수인 경우는 **가장 중요한 비트(Most Significant Bit: MSB)**를 부호 비트(Sign Bit)로 사용하여, MSB가 1인 경우는 이 수의 2의 보수를 취한 값을 음수로 취합니다.

• 부호 있는 3비트로 1의 보수와 2의 보수로 각각의 정수를 표현해 보면 아래 표와 같습니다.

1의 보수	정수값	2의 보수	정수값
000	+0	000	+0
001	+1	001	+1
010	+2	010	+2
011	+3	011	+3
100	-3	100	-4
101	-2	101	-3
110	-1	110	-2
111	-0	111	-1

1의 보수와 2의 보수의 부호 있는 정수값

- 101의 경우 1의 보수에 대해서,

MSB가 1이므로 음수
101의 1의 보수는 010=2

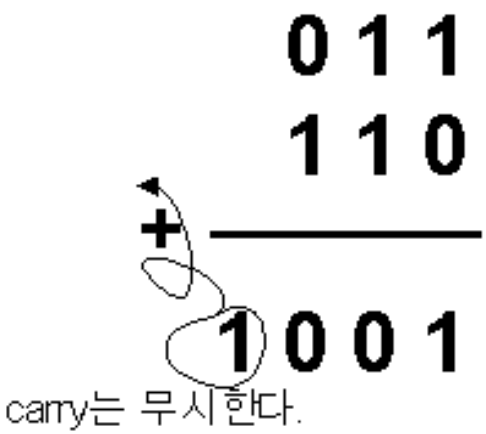
이므로 –2가 됩니다.

- 2의 보수에 대해서는,

MSB가 1이므로 음수
101의 2의 보수는 011=3

이므로 –3이 됩니다.

- 올림수 무시



2의 보수에서 올림수의 무시

- 2의 보수로 수를 표현했을 때, n비트의 부호있는 정수의 범위는 다음과 같습니다.

$$-2^{n-1} \sim 2^{n-1}-1$$

- signed short int인 경우 -32768(-2^{15}) ~ 32767($2^{15}-1$) 범위의 수를 표현할 수 있습니다.
- 아래 프로그램의 결과는 얼마가 인쇄될까요?

```
#include <stdio.h>
void main() {
    char c=129;
    printf("%d\n", c);
}
```

MSB가 1이므로 음수

1000 0001의 2의 보수는 0111 1111=127

그러므로 -127이 출력되는 것입니다.

진보된 주제: 비트 플래그(bit flag), 비트 마스크(bit mask)

• 구구단을 출력하는 프로그램인데 구구단을 출력하다가 사용자가 임의의 키를 누르면 종료하도록 만들려고 합니다.

```
#include <stdio.h>
#include <conio.h>

void main() {
    int i=1,j;

    //clrscr();
    while (i<100) {
        ++i;
        j=1;
        while (j<100) {
```

```
            printf("%5d*%5d=%8d",i,j,i*j);
            ++j;
        }//while
    }//while
}
```

• 두 번째 while문을 탈출하도록 논리를 구성해야 하는데, 단순하게 두 번째 while문안에서 break를 사용하는 것은 문제를 해결하지 못합니다.

if (kbhit()) break;

• break는 자기를 둘러싼 가장 가까운 반복 구조■의 블록을 탈출하기 때문입니다.
• 두 번째 while문에서 어떤 사건이 발생했다는 것을 기억시키기 위해서 특정한 변수를 사용해야 합니다. 변수가 이러한 목적으로 사용되었을 때 **깃발(Flag) 변수**라고 합니다.

```
#include <stdio.h>
#include <conio.h>

void main() {
    int i=1,j;
    int flag=0;//사건을 기록하기 위해서 사용합니다.

    //clrscr();
    while (i<100) {
        ++i;
        j=1;
        while (j<100) {
            printf("%5d*%5d=%8d",i,j,i*j);
            ++j;
            if (kbhit()) {
                flag=1;
                break;
            }//if
        }//while
```

```
        if (flag==1) break;//사건이 일어난 경우 while을 탈출한다
    }//while
}
```

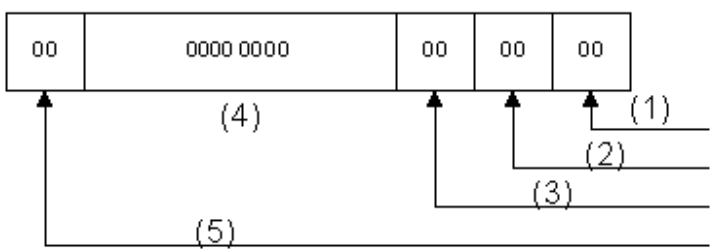
• 정수형 변수의 이진 비트열의 각각의 비트들이 이러한 깃발 변수로 사용되었을 때를 비트 플래그bit flag라고 합니다.

• 정수형 변수 short는 16비트이므로 16가지의 on/off 상태를 기록할 수 있습니다. 만약 상태의 개수가 4가지라면, 상태를 나타내는데 2비트가 필요하므로 하나의 short정수를 8가지의 깃발 변수로 사용할 수 있습니다.

• 컴퓨터 바둑 프로그램을 구현할 때 판(Board)의 각각의 위치에 대해, 아래의 정보를 유지한다고 합시다.

(1) 판의 현재 상태: 2비트(비었음, 검은돌, 흰돌, 가장자리)
(2) 이웃한 검은색 돌의 수: 2비트(Neumann이웃이므로 최대 4까지 가능)
(3) 이웃한 흰색 돌의 수: 2비트
(4) 덩어리 번호: 8비트
(5) 마킹을 위한 임시 비트: 2비트

• 정수형 배열 board[19+2][19+2]을 선언해서 각각의 판 위치에 대해 아래 그림처럼 정보를 유지할 수 있습니다.

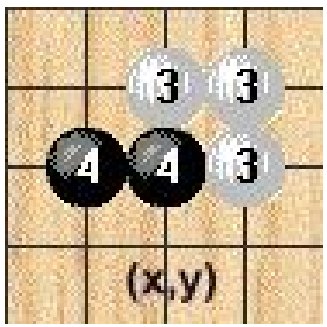


- 구조체structure 작성

```
struct BOARD {
    char nState, nBlack, nWhite;
    unsigned char nGroup;
    char nMarking;
} board[21][21];
```

- 각각의 자리마다, 5바이트를 사용하므로, 21×21×5=2205 바이트를 사용해야 합니다.

- **'어떻게 각각의 비트 플래그를 갱신하는가?'**



board[y][x]의 상태

- 위 그림과 같은 상태의 board[y][x]의 값은, 다음과 같습니다.

00 00000100 10 01 01

- (x,y)에 검은 돌이 있으며, 4-연결 이웃에 각각 검은돌, 흰돌이 1개, 2개 있음을 나타내며, 덩어리 번호는 4임을 나타냅니다.

• 4의 마지막 남은 이웃 자리에 검은 돌을 놓았다고 합시다. 그러면, 다음과 같이 갱신해야 합니다.

00 00000100 10 10 01

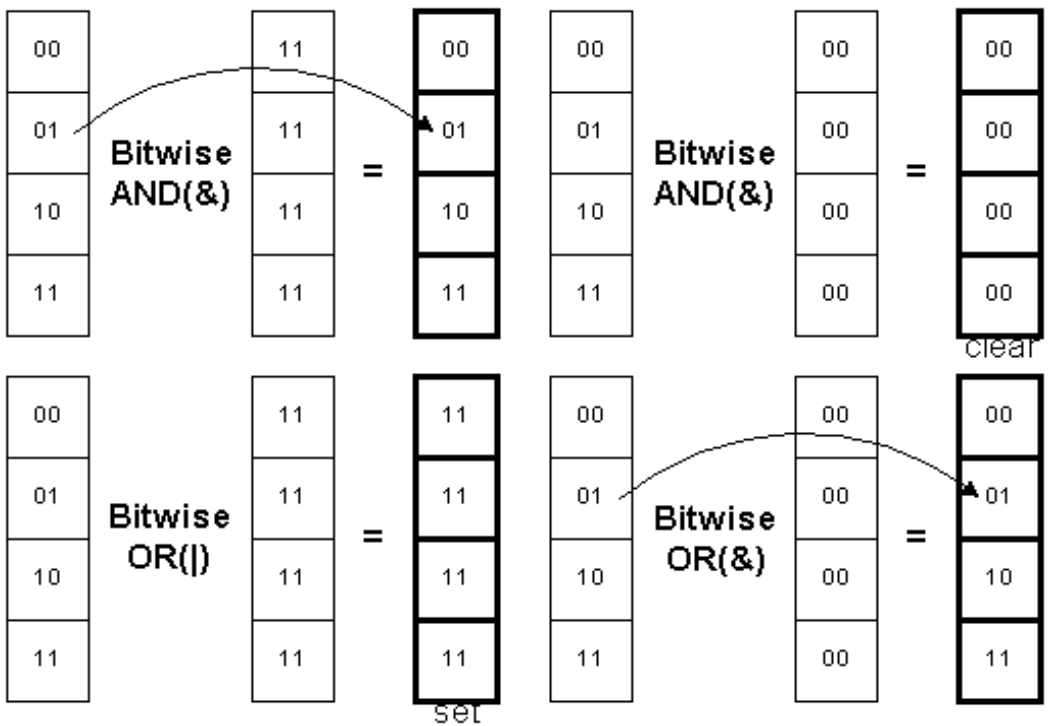
• 어떻게 밑줄 친 비트의 일부만을 갱신할 수 있을까요?

(1) 갱신하고자 하는 비트열만을 일반 정수 형태로 변환합니다.
(2) 변환된 정수를 갱신합니다.
(3) 갱신된 정수 비트열을 원래 자리에 다시 가져다 놓습니다.

```
unsigned int i=board[y][x];//i= 00 00000100 10 10 01
unsigned int t;

t=(i & 0x000c)>>2;//0x000c는 이진수로 0000 0000 0000 1100입니다.
++t;//t가 1에서 2가 됩니다.
i=(i & 0xfff3) | (t<<2);//0xfff3은 이진수로 1111 1111 1111 0011입니다.
board[y][x]=i;
```

- & 와 | 연산은 아래와 같은 특징을 가집니다.



비트 마스크의 원리

실습문제

1. 아래의 출력 결과는 얼마인가요?

```
#include <stdio.h>

void main() {
    int i=7;

    printf("%d\n",~i+1);
}
```

2. 아래의 출력 결과는 얼마인가요?

```
#include <stdio.h>

void main() {
    int i=0x5555, j=0x1234, k;

    k=i^j;
    printf("%x\n",k);
    printf("%x\n",k^j);
}
```

3. 아래의 프로그램에서 Product(a,b) 함수는 a*b를 쉬프트(shift) 연산과 모듈로(modulo) 연산만을 사용하여 계산합니다. Product() 함수를 자세히 설명하세요(힌트: Booth 알고리즘).

```
#include <stdio.h>

long Product(long a,long b) {
    long p=0;
    while (a>0) {
        if (a%2==1) p+=b;
        a>>=1;
        b<<=1;
    }//while
    return p;
}//Product()

void main() {
    printf("%ld\n",Product(3,7));//결과는 3*7=21입니다.
}
@
```