

ビット演算

知識工学 I

東京工業高等専門学校 情報工学科 鈴木雅人

ビット演算の役割

char (1バイトデータ型)

文字 → ASCIIコード

符号あり → -128~127

符号なし → 0~255

short int (2バイトデータ型)

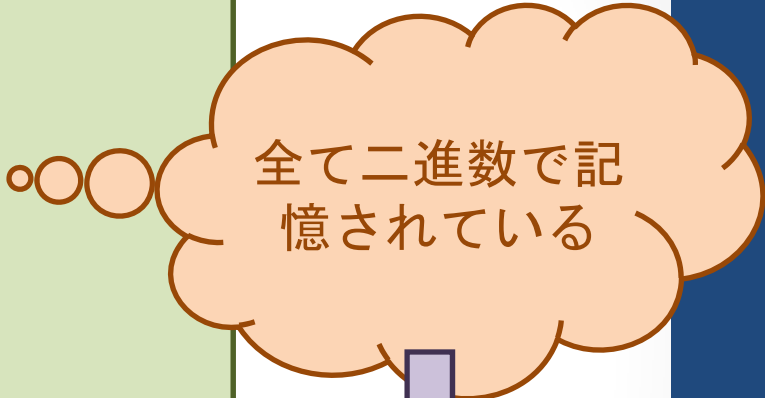
符号あり → -32768~32767

符号なし → 0~65535

long int (4バイトデータ型)

符号あり → -2147483648~2147483647

符号なし → 0~4294967295



全て二進数で記憶されている



変数に格納された二進数(本来の姿)にする演算: ビット演算子

画像処理・マイコン制御などで使われる

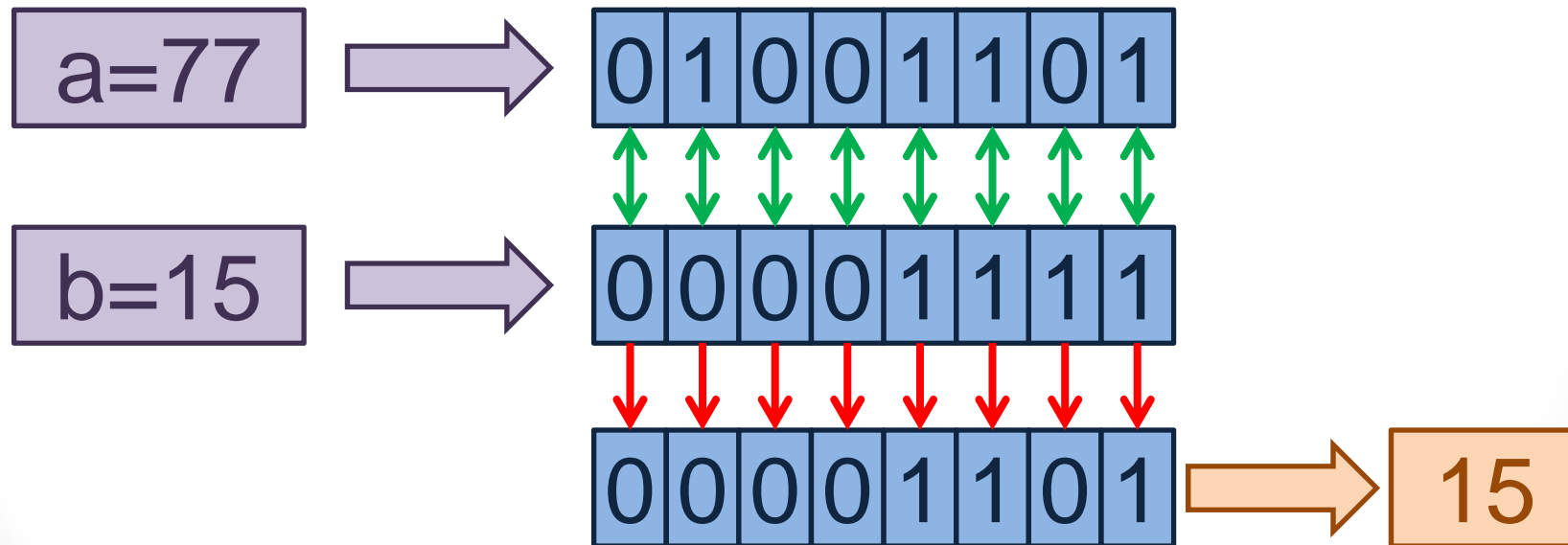
ビット演算の種類

左シフト演算子	<<	数字列を左にずらす
右シフト演算子	>>	数字列を右にずらす
ビットAND演算子	&	桁ごとに論理積を求める
ビットOR演算子		桁ごとに論理和を求める
ビットXOR演算子	^	桁ごとに排他的論理和を求める
ビットNOT演算子	~	全てのビットを反転させる

ビットAND演算子

```
char a = 77, b = 15 ;  
printf( "%d", a & b ) ;
```

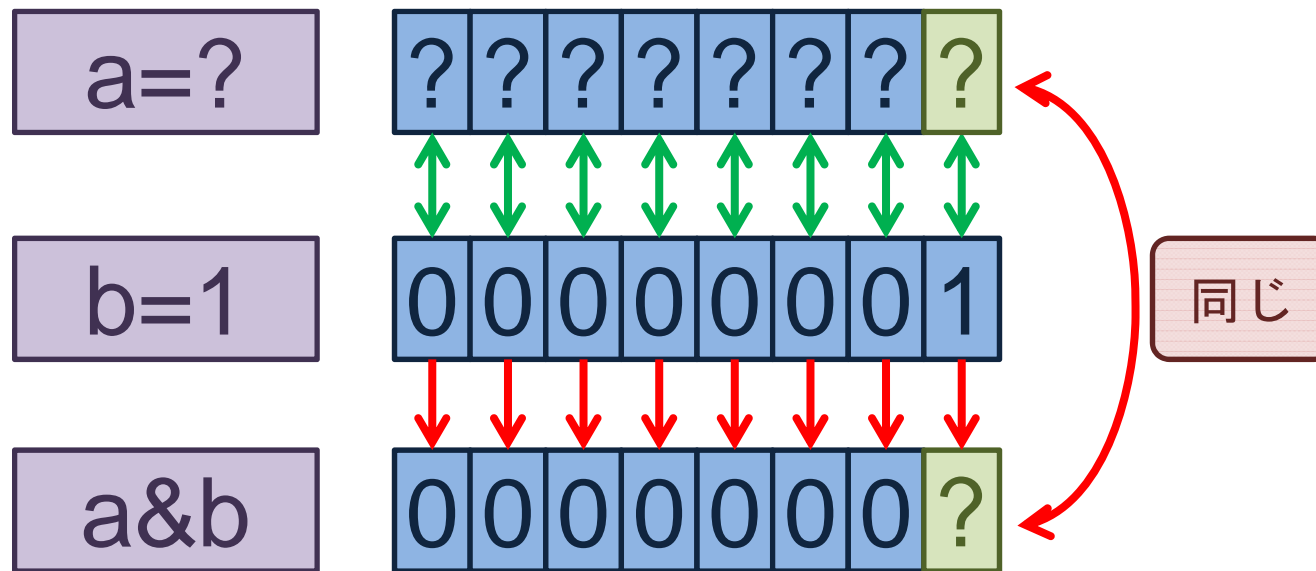
char型は1バイト
(8ビット)の数値
が表現できる



桁数が変わっても、負の数でも演算は全く同じ

ビットAND演算子の活用

特定のビットを調査できる



```
if ( a & b ) {  
    printf( "右端は1¥n" );  
} else {  
    printf( "右端は0¥n" );  
}
```

右シフト演算子

```
char a = 77 ;  
printf( "%d", a>>3 );
```

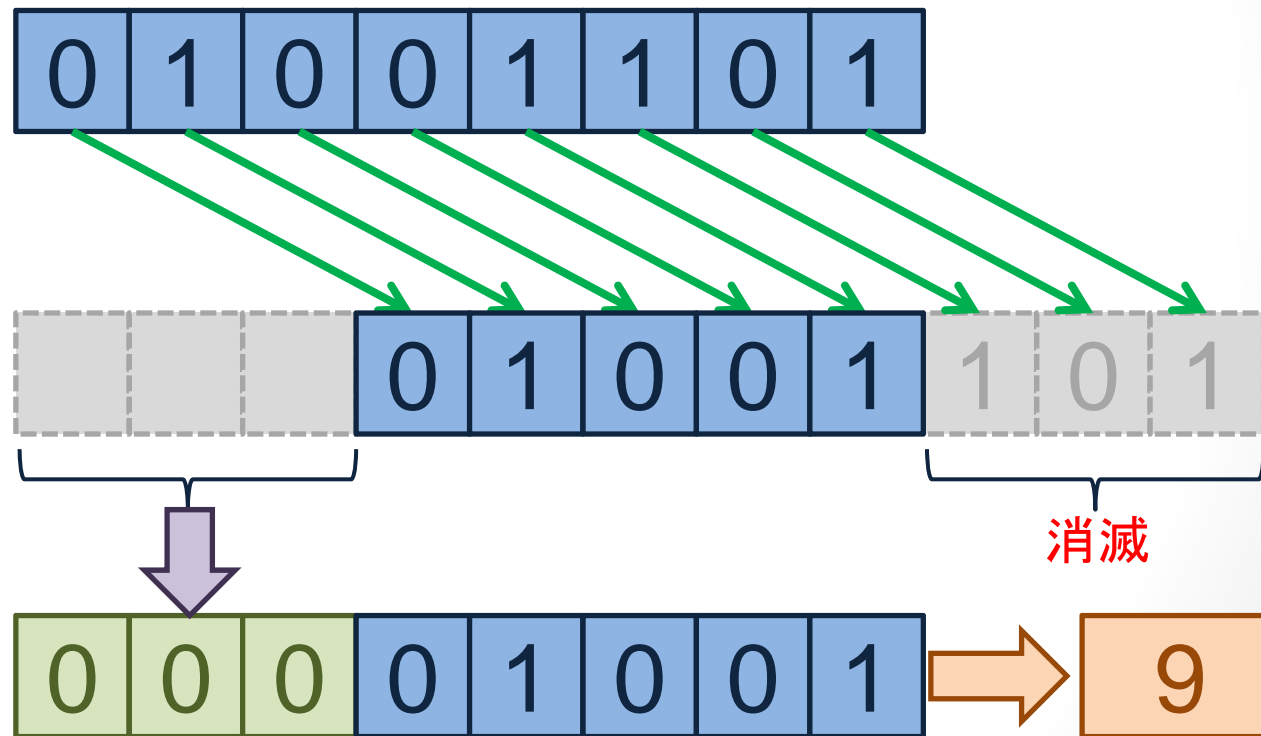
符号あり変数に正
の数が入っている
場合

3ビット右にずらす →

a=77

$\div 2^3$

a>>3



0で埋める

右シフト演算子

```
char a = -46 ;  
printf( "%d", a>>3 ) ;
```

符号あり変数に負
の数が入っている
場合

【復習】 2 の補数表現を用いて-46を二進数で表す

a=46

0	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---



ビット反転

1	1	0	1	0	0	0	1
---	---	---	---	---	---	---	---



1を加える

a=-46

1	1	0	1	0	0	1	0
---	---	---	---	---	---	---	---

右シフト演算子

```
char a = -46 ;  
printf( "%d", a>>3 );
```

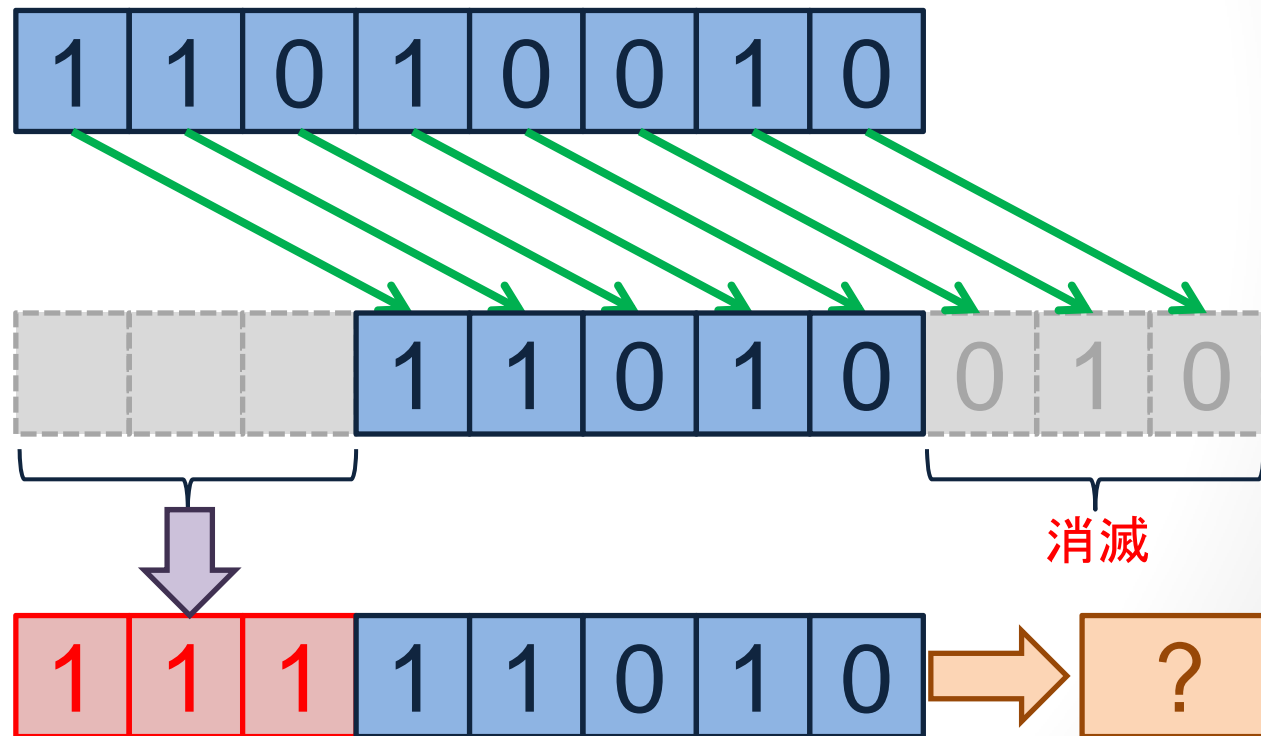
符号あり変数に負
の数が入っている
場合

3ビット右にずらす

a=-46

$\div 2^3$

a>>3



1で埋める

右シフト演算子

```
char a = -46 ;  
printf( "%d", a>>3 );
```

符号あり変数に負
の数が入っている
場合

【もう一度復習】 2 の補数表現を用いて答えを求める

1 1 1 1 1 0 1 0

ビット反転

0 0 0 0 0 1 0 1

1を加える

0 0 0 0 0 1 1 0

もとの数字は
「-6」でした

6

右シフト演算子

```
unsigned char a = 171 ;  
printf( "%d", a>>3 );
```

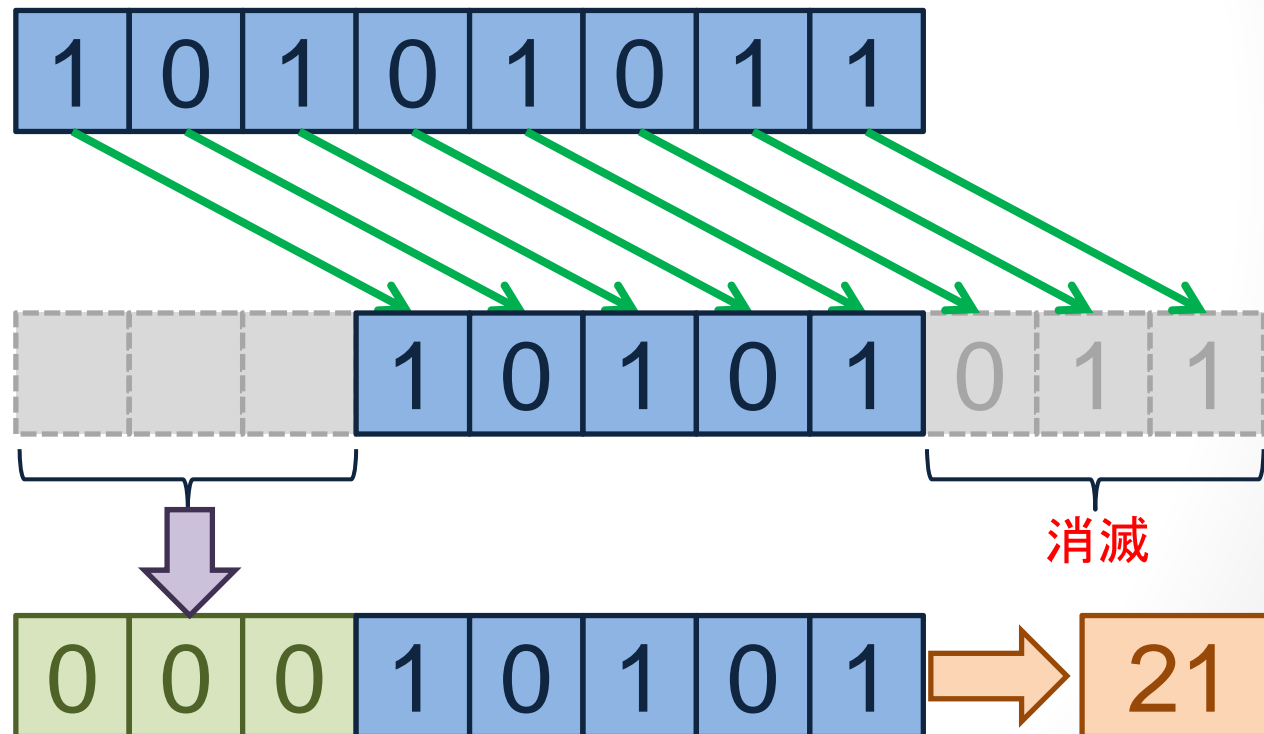
符号なし変数の場合

3ビット右にずらす

a=171

$\div 2^3$

a>>3



0で埋める

右シフト演算子の活用

特定のビットを調査できる

```
unsigned char a, b = 128 ;
```

a=?

b=128

b>>2

a&(b>>2)



```
if ( a & ( b >> 2 ) ) {  
    printf( "3ビット目は1¥n" );  
}
```