プログラミング基礎
第7回

藤江 真也

2021年6月4日

今回の講義の進め方

- このスライドと，「第7回 ビット演算 補足資料と課題」を適宜参照しながら，一部手書きの説明をあわせて進めていきます
- 課題は「補足資料と課題」中に含まれる課題を手書きで作成してもらいます
  - 解答用紙も用意してありますが，同レイアウトであれば指定の用紙でなくても構いません

準　備

- ファイル（0604.tgz）をダウンロード

```
$ wget http://sites.fujielab.org/ip/files/0604.tgz
```

- ダウンロードしたファイルを展開

```
$ tar zxvf 0604.tgz
```

- 展開されたディレクトリに移動

```
$ cd 0604
```

- dec2bin.cなどのファイルがあることを確認

```
$ ls
```


n進数とビット

10進数

- 桁(ケタ)について考えよう

9	3	7	6	0	2	4
百万	十万	万	千	百	十	一

  - 各ケタは 0 ～ 9 の10通りの数を持つ
  - あるケタの数が9より一つ大きくなるとき
    そのケタの数は0になり
    次のケタ(1つ上のケタ)の数が一つ大きくなる

    ⟶ 10で一つケタが進む数

2進数

- 最もシンプルに各ケタに0か1しかもたない数を考えよう

10進数	2進数	10進数	2進数	10進数	2進数
0:	0 0 0 0 0	6:	0 0 1 1 0	12:	0 1 1 0 0
1:	0 0 0 0 1	7:	0 0 1 1 1	13:	0 1 1 0 1
2:	0 0 0 1 0	8:	0 1 0 0 0	14:	0 1 1 1 0
3:	0 0 0 1 1	9:	0 1 0 0 1	15:	0 1 1 1 1
4:	0 0 1 0 0	10:	0 1 0 1 0	16:	1 0 0 0 0
5:	0 0 1 0 1	11:	0 1 0 1 1	17:	1 0 0 0 1

1ケタの数字が2通りでも意外とちゃんと数字を表現できる

ビット

- ビット(bit)とは(例えば 0 か 1 という)
  2つの状態だけを表すことのできる単位
- 1bit で表せる数字は 0 か 1 だけ
- では 2bit では？ 3bitでは？

ビット数が増えると…

2bitの場合は 0～3 (4通り)

0:	0 0 0 0 0	6:	0 0 1 1 0	12:	0 1 1 0 0
1:	0 0 0 0 1	7:	0 0 1 1 1	13:	0 1 1 0 1
2:	0 0 0 1 0	8:	0 1 0 0 0	14:	0 1 1 1 0
3:	0 0 0 1 1	9:	0 1 0 0 1	15:	0 1 1 1 1
4:	0 0 1 0 0	10:	0 1 0 1 0	16:	1 0 0 0 0
5:	0 0 1 0 1	11:	0 1 0 1 1	17:	1 0 0 0 1

3bitの場合は 0～7 (8通り)

では4bitの場合は…？

2進数

- n番目のケタ（nは0から始める）の1は，2^nだけの大きさを持つ

1	1	0	1	1	0	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

$128 + 64 + 0 + 16 + 8 + 0 + 2 + 0 = 218$

8bit（8ケタ）の2進数は
0～255（256通り＝2^8通りの数）を表すことができる

16進数

- 各ケタが 0 ～ 15 の数字を持つ（4bit分）
- 二桁になると都合が悪いので
  10～15はA～F（a～f）で表す

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
(10) (11) (12) (13) (14) (15)

16進数→10進数

3	D
	(13)
×	×
16^1	16^0
(16)	(1)
‖	‖
48	+ 13 = 61

C	2	F	8
(12)		(15)	
×	×	×	×
16^3	16^2	16^1	16^0
(4096)	(256)	(16)	(1)
‖	‖	‖	‖
49152	+ 512 +	240	+ 8 = 49912

10進数 ←→ 2進数，16進数

- 手書きで説明

8進数

- 各ケタが 0～7 の数字を持つ（3bit分）

2進数，8進数，16進数の関係

- 2進数との対応を考える場合，8進数や16進数の方がケタが進むタイミングが同じ分，10進数より相性がよい（のでよく使う．特に16進数）

10進数	2進数	8進数	16進数
0	00000	00	00
1	00001	01	01
2	00010	02	02
3	00011	03	03
4	00100	04	04
5	00101	05	05
6	00110	06	06
7	00111	07	07
8	01000	10	08

10進数	2進数	8進数	16進数
9	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10
17	10001	21	11

コンピュータの世界は基本的に2進数

- 各数値型は特定のbit数で表される
- 8bit は 1Byte
- 整数型
  - char ... 8bit = 1Byte
  - short ... 16bit = 2Byte
  - int ... 32bit = 4Byte
  - long ... 64bit = 8Byte
- 浮動小数点型
  - float ... 32bit = 4Byte
  - double ...64bit = 8Byte

（longは処理系によって32bitの場合もある）

C言語での10進数，8進数，16進数

- 単に 10 と書いても何進数かわからないと本当の数値がわからない
- C言語では以下のような決まりになっている

10進数	8進数	16進数
a = 5;	a = 05;	a = 0x5;
a = 10;	a = 012;	a = 0xA;
a = 15;	a = 017;	a = 0xF;
a = 16;	a = 020;	a = 0x10;
	頭に0を付ける	頭に0xを付ける

10進数, 2進数, 16進数変換

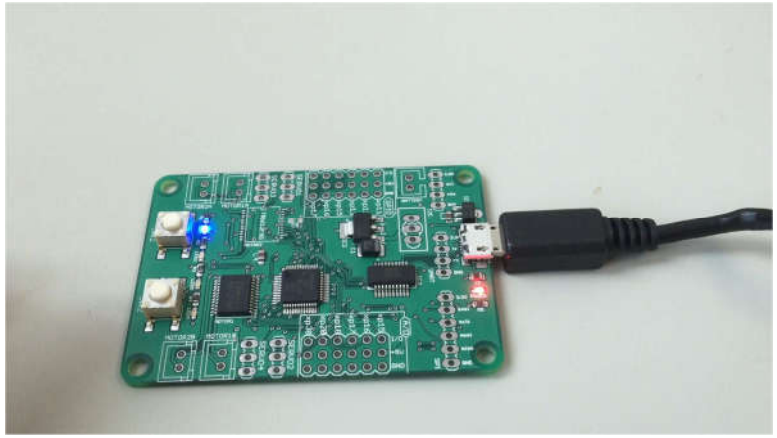
- 10進数から2進数(16ケタ), 16進数(4ケタ)への変換
  - 10進数で16や25, 57などは2進数, 16進数でどうなるか？
  - 手で計算した結果が正しいかプログラムで確認しよう

dec2bin.c

- 2進数(16ケタ)から10進数, 16進数(4ケタ)への変換
  - 2進数で 0000 0010 0011 1100 などは10進数, 16進数でどうなるか？
  - 手で計算した結果が正しいかプログラムで確認しよう

bin2dec.c

ビット単位の演算



ディジタル・スイッチ
LED(発光ダイオード)

ONとOFFの2状態しか持たない

⇨ 1つのスイッチやセンサの状態管理には1ビットの情報があれば十分

8bitの数が持つ情報

SW4	SW3	SW2	SW1	LED4	LED3	LED2	LED1
1	1	0	1	1	0	1	0

- 8bitの情報があれば, 合計8個のスイッチ／LEDの状態を独立に管理することができる
- short 型の整数は 16bit だから,
  なんと16個も独立に管理することができる！

例えば…

SW4	SW3	SW2	SW1	LED4	LED3	LED2	LED1
1	1	0	1	1	0	1	0

```
if ( SW1が押されている ) {
    LED1を点灯させる;
} else {
    LED1を消灯させる;
}
```


例えば…

SW4	SW3	SW2	SW1	LED4	LED3	LED2	LED1
1	1	0	1	1	0	1	0

このケタだけが1か0かを
どうやって判断するか？

このケタだけを1や0に
どうやって変更するか？

ビット演算

ビット演算

- 論理積（AND）
- 論理和（OR）
- 排他的論理和（XOR）
- 補数（NOT）

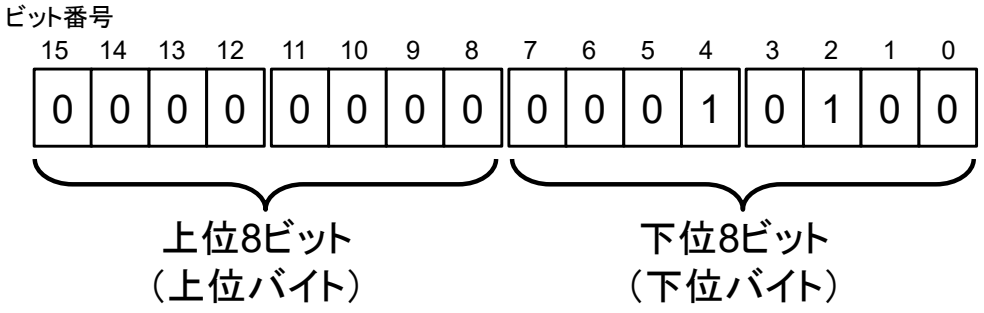
→ 補足資料「2. ビット演算」

→ 具体例 bitop.c

変数の2進数表現（ビット表現）

- short型の場合 … 16bit

short a = 20;　と宣言＆初期化すると…



ビット演算の主な利用例

- マスク処理（AND演算）…
  特定のビットのみを取り出すことができる

例）1011 1001 1011 1011 の下位8bitのみが欲しい
0000 0000 1111 1111 とANDを取ればよい
結果 0000 0000 1011 1011

- マスク処理（OR演算）…
  特定のビットを1にすることができる

例）1011 1001 1001 1011 の4～7ビットを1にしたい
0000 0000 1111 0000 とORを取ればよい
結果 1011 1001 1111 1011

符号付き整数／符号無し整数

- 例えば4bitあれば，2^4=16通りの数字を表すことができる
- 0と正の数だけを考えれば 0～15 が表せるが，負の数まで考えると符号（+または−）のために1bit分情報が必要になる
- C言語では，どちらにするかを選べる
  - unsigned（符号なし）→ 0～15
  - signed（符号あり）→ −8～7

2の補数

bitcomp.c

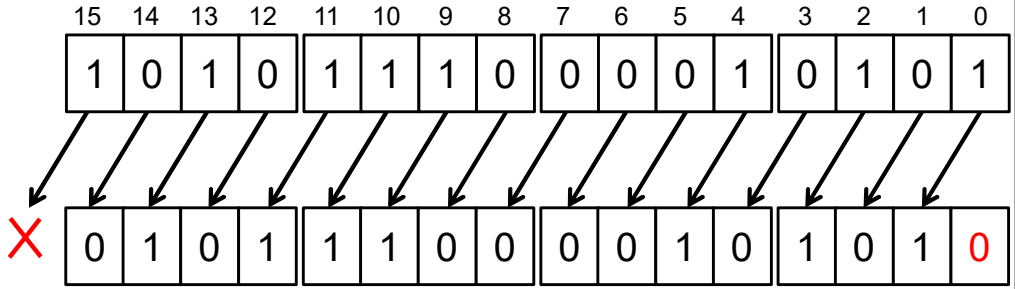
- 例えば4bitの整数で，0011（3）と足して，ケタが溢れて 10000 になる数字はどうやったら得られるか?
  01111 に 1 を足すと 10000 になる
  0011 と足して 1111 になる数は 1100
  元々の数字の0/1を反転させたもの
  1100 に 1 を足した 1101 が正解
- ある2進数の各ケタの0/1を反転させて1を足した数を2の補数といい，C言語では補数の関係が符号（正負）の反転の関係になる
  - 足して0になるので都合がよい

シフト演算

bitshift.c

- 左シフト (<<)
  - ビットを左にずらす
  - 左側のビットは消える，右側のビットは0になる

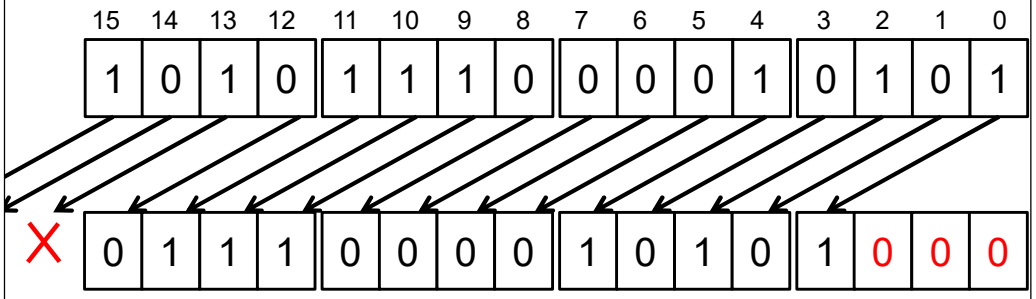
unsigned short型の変数aが以下の内容のときの a << 1 の結果



シフト演算

シフトするビット数を与える数値で調整できる

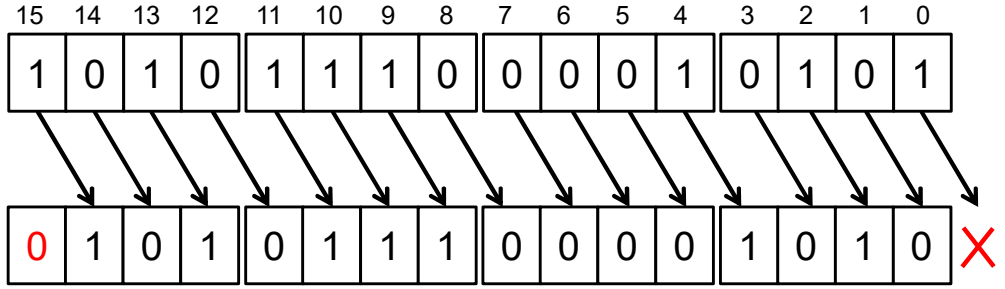
例えば a << 3の場合



シフト演算

- 右シフト（>>）
  - ビットを右にずらす
  - 右側のビットは消える，
    左側のビットには0が入る（unsignedの場合）

unsigned short型の変数aが以下の内容のときの a >> 1 の結果



シフト演算の算術的意味

- 左シフト（<<）
  - 桁あふれさえしなければ a << n は，
    a を 2^n 倍しているのと同じ
- 右シフト（>>）
  - a >> n は，
    a を 2^n で割っているのと同じ

ビット演算の代入演算子

- 各種ビット演算にもそれに対応する代入演算子がある
  - x &= y ⟺ x = x & y
  - x |= y ⟺ x = x | y
  - x ^= y ⟺ x = x ^ y
  - x <<= y ⟺ x = x << y
  - x >>= y ⟺ x = x >> y

課題

- 「ビット演算 補足資料と課題」の中の課題1～課題4について解答を作成し，提出してください
- 解答用紙があります．印刷ができる人は印刷して手書きで解答した上で写真に撮りmanabaにアップロードしてください．
  - 印刷ができない人は，どの課題の答えかがわかるようにすれば適当なレポート用紙などで結構ですので同様にアップロードしてください
- 計算過程，導出過程が求められている課題は，答えだけでなく過程も示してください．