情報処理工学 第２回

藤田 一寿

公立小松大学保健医療学部臨床工学科

2進数における四則演算

２進数の足し算，引き算

- 10進数の足し算，引き算と変わりはない.
- しかし，桁上り，桁下がりに注意する.

足し算の例

- 2進数11011と10101を足せ.

$$\begin{array}{r} 11011 \\ +\ 10101 \\ \hline \end{array}$$

$$\begin{array}{r} 1 \\ 11011 \\ +10101 \\ \hline 0 \end{array}$$

一番下の桁から足していく.　1+1=10なので桁上りがある.

足し算の例

$$\begin{array}{r} 1 \\ 11011 \\ +10101 \\ \hline 00 \end{array} \qquad \begin{array}{r} 1 \\ 11011 \\ +10101 \\ \hline 000 \end{array} \qquad \begin{array}{r} 1 \\ 11011 \\ +10101 \\ \hline 0000 \end{array} \qquad \begin{array}{r} 11011 \\ +\ 10101 \\ \hline 110000 \end{array}$$

引き算の例

- 2進数11011から2進数10101を引け.

桁下がり
10-1を計算

$$\begin{array}{r} 11011 \\ -10101 \\ \hline 0 \end{array} \qquad \begin{array}{r} 11011 \\ -10101 \\ \hline 10 \end{array} \qquad \begin{array}{r} 10\boxed{0}11 \\ -10\boxed{1}01 \\ \hline \boxed{1}10 \end{array}$$

別のやり方

- 2進数を10進数になおして計算し，その計算結果を2進数に変換する.

$$
\begin{aligned}
11011_2 + 10101_2 &= 27_{10} + 21_{10} = 48_{10} \\
&= 110000_2
\end{aligned}
$$

16進数の足し算・引き算

- 16進数同士の足し算・引き算は当然可能ですが，人間の頭が10進数や2進数に慣れているため，10進数か2進数に変換して計算したほうが楽でしょう．
- 特に16進数と2進数には便利な関係性があるので，その関係を知っていると計算が楽になります．

16進数の足し算を10進数に変換して行う.

- 16進数の1Aと27を足せ.

$$1\mathrm{A}_{16} + 27_{16} = 26_{10} + 39_{10} = 65_{10} = 41_{16}$$

10進数に変換する　　16進数に戻す

16進数の足し算を2進数に変換して行う.

- 16進数の1Aと27を足せ.

16進数の各桁を2進数に変換

$$
\begin{aligned}
1\mathrm{A}_{16} + 27_{16} &= (0001\ 1010)_2 + (0010\ 0111)_2 \\
&= (0100\ 0001)_2 = 41_{16}
\end{aligned}
$$

2進数4桁ごとに16進数に戻す

2進数の掛け算

- 掛け算も10進数と同じように計算できる.
- $1101_2 \times 101_2$は次のように計算できる.

$$
\begin{array}{r}
1101 \\
\times \quad\quad 101 \\
\hline
1101 \\
0000 \\
1101 \\
\hline
1000001
\end{array}
$$

2進数の割り算

- 割り算も10進数と同じように計算できる.
- $1000001_2 \div 101_2$は次のように計算できる.

$$
\begin{array}{r@{}l}
 & \;\;\;\; 1101 \\ \cline{2-2}
101 \big) & 1000001 \\
 & \;101 \\ \cline{2-2}
 & \;\;\;110 \\
 & \;\;\;101 \\ \cline{2-2}
 & \;\;\;\;\;\;101 \\
 & \;\;\;\;\;\;101 \\ \cline{2-2}
 & \;\;\;\;\;\;\;\;\;0
\end{array}
$$

演習

- 次の計算をせよ.
  - 1010_2+1110_2
  - $1010_2 \times 110_2$
  - $1111_2 \div 101_2$

２進数における負数表現

２進数における負値表現

- 最上位ビットを符号とし，残りのビットを絶対値とする.

- 例：４ビットつかって数値を表現する場合
  - 10進数の5は0101
  - 10進数の-5は1101

- マイナスの値のときは最上位ビットが符号ビットとなので，-5のときは最上位ビットは1となった.

符号ビットを使う問題点

- 計算がうまくいかない
- 10進数の5と-5の足し算は0なのに，単純に足したら0にならない.

$$5_{10} + (-5_{10}) \rightarrow 0101_2 + 1101_2 = 10010_2$$

- この計算をうまくやる方法が補数.

2の補数を使った負値表現

- 10進数-5を4桁の2の補数を用いて表す.

- まず，10進数の5を4桁の2進数に変換する.
  - 10進数の5は2進数では0101

- 得られた2進数を0と1を反転させて1を加える.
  - 0101を反転させると1010
  - これに1を加えると1011
- 以上で得られた1011が10進数-5の補数表現である.

10進数の5と-5の足し算が0になるか確認

- 2の補数を使った負値表現により10進数の5と-5の足し算がどうなるのか？

$$5_{10} + (-5_{10}) \rightarrow 0101_2 + (1011)_2 = 10000_2$$

これも0にならない！！

実は計算はできている

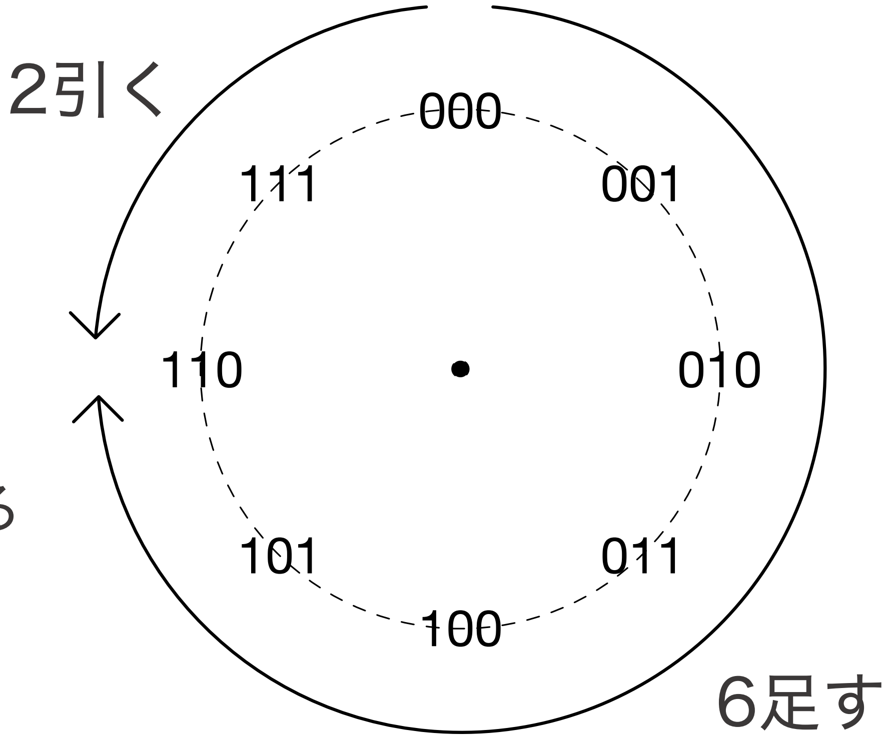
- ここで，数値は4桁の2進数で表しているので，最上位の桁（ビット）は捨てることになる．
- つまり

$$
\begin{aligned}
5_{10} + (-5_{10}) &\rightarrow 0101_2 + (1011)_2 \\
&= 10000_2 \rightarrow 0000_2
\end{aligned}
$$

ちゃんと０になっている！！

なぜ補数を使うとうまくいくのか？

- –2の補数を使い３桁の２進数で表すことを考える.
- 3桁の２進数なので数字は000から111まで表現できる.
- 3桁の２進数を円状に並べる.
- –2は0から2戻ると考えると, –２は6進むことと同じになる.
- すなわち, -2は110（2の補数）で表すことができる.
- 数式で考えると, 3ビットは8個の数字を表すことができるので, 2戻るは8-2=6進むことと同じになる.

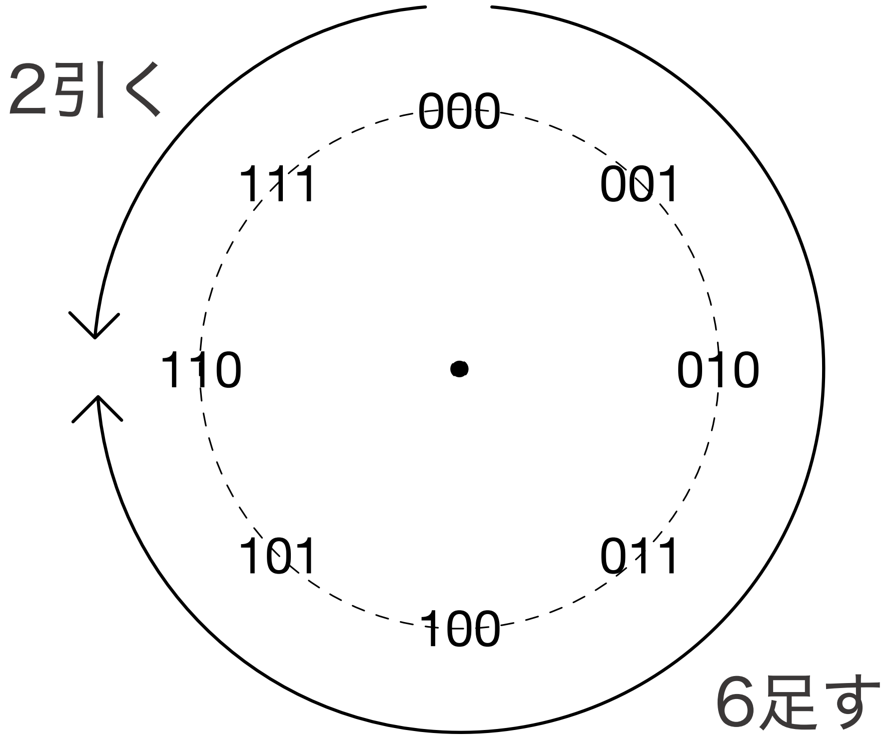


なぜ補数を使うとうまくいくのか？

8進んで２戻る→２戻ると同じ意味

$$
\begin{aligned}
8_{10} - 2_{10} &= 1000_2 - 010_2 \\
&= \boxed{111_2} + 001_2 \boxed{-\ 010_2} \\
&= \boxed{101_2} + 001_2 \\
&= 110_2
\end{aligned}
$$

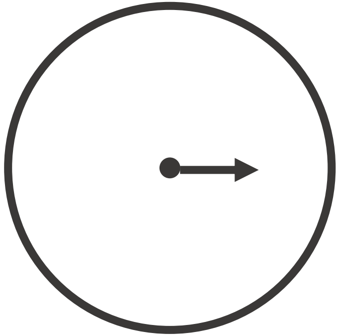
111−010は010のビット反転に相当する.
最終的にビット反転して１足したものになる.

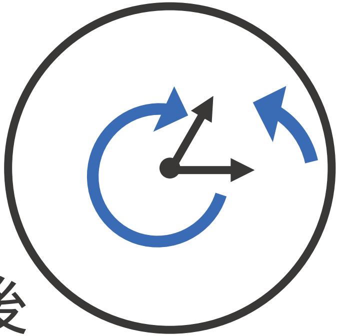


補足：時計を使った補数の直感的理解

- 現在アナログ時計が3時を示していた場合，2時間前は何時を示しているか.
  - 3-2=1時を示していた.
- しかし，時計においては2時間前は10 (12-2) 時間後とおなじになる.
  - 3+10=13時となるが，13時はアナログ時計では1時を示す.

- 2の補数を使った引き算もこれと同じ原理.





演習

- 2の補数を用い，次の数を4ビットの2進数で表わせ.

- -8_{10}

- -5_{10}

- -1_{10}

補数表現で正の数と負の数が混同しないのか？

- 4ビットの2進数で数を表現する場合.
  - –8から7までの数なら，正の数と負の数を混同することがない.
  - 負値の場合，最上位ビットが必ず1となる.

10進数	2の補数	1の補数
-8	1000	表現不可能
-7	1001	1000
-6	1010	1001
-5	1011	1010
-4	1100	1011
-3	1101	1100
-2	1110	1101
-1	1111	1110
-0	0000	1111

10進数	2の補数	1の補数
+0	0000	0000
+1	0001	0001
+2	0010	0010
+3	0011	0011
+4	0100	0100
+5	0101	0101
+6	0110	0110
+7	0111	0111
+8	表現不可能	表現不可能

浮動小数点数 (おまけ)

浮動小数点数

- コンピュータで実数を表現するときには浮動小数点数が用いられる.
- 浮動小数点数では次のような形で，実数aを表す.

$$a = m \times B^e$$

- m：仮数
- B：底または基数
- e：指数

10進数ならB=10，2進数ならB=2となる.

単精度と倍精度

- 仮数の符号，仮数，指数の合計が32ビットとなる浮動小数点数を単精度浮動小数点数という.
- 単精度浮動小数点数は10進数で約7桁の精度である.
- 64ビット用いた浮動小数点数を倍精度浮動小数点数という.
- 倍精度浮動小数点数は10進数で約16桁の精度である.

10進数における浮動小数点数

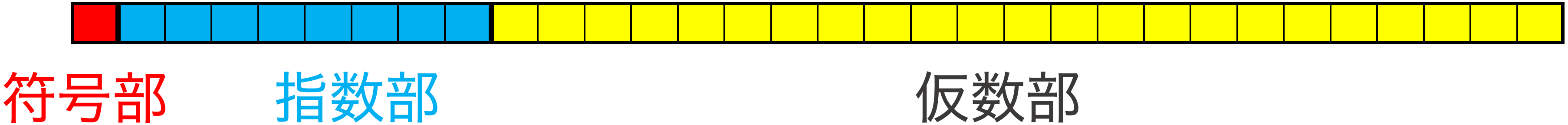
- 例えば1234.56を浮動小数点数で表すと

$$
\begin{aligned}
1234.56 &= 1.23456 \times 10^{3} \\
&= 123.456 \times 10^{1} \\
&= 123456.0 \times 10^{-2}
\end{aligned}
$$

- このように，浮動小数点数では指数部の値によって小数点の位置が変わる.

2進数を用いた浮動小数点数

- コンピュータでは多くの場合，2進数を用いた浮動小数点数(IEEE754)が採用されている．
- 単精度では，符号1ビット，指数部が8ビット，仮数部が23ビットで数値が表される．
- 指数部は8ビットなので，10進数でいえば-127から128まで使うことができる．そのために，実際の指数部に127を足したものを指数部として記憶する．補数を使わないことで大小比較が用意．
- 小数点をどこにするか決めなければ指数部は決まらない．仮数を最上位ビットは1であるように決める．そのため，仮数の最上位ビットは必ず1となるため省略することができる．



例

- 10進数の-27を基数が2の浮動小数点数で表してみる.
- まず10進数の27を2進数で表すと

$$27_{10} = 11011_2 = 1.1011_2 \times 2^4$$

- よって，仮数部は2進数で1.1011，指数部は10進数で4となることが分かる.
- 仮数部の最上位ビットは省略するので，実際に保存するのは1011となる.
- また，指数部は10進数で4であるが，127底上げするため，実際に保存されるのは10進数で131すなわち2進数の10000011である.
- 符号が－なので，符号部は1となる.

1	1	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

浮動小数点数からみる情報工学の考え方

- 浮動小数点数の作り方（規格）は様々に考えることができる.
  - 自分で規格を作ることも可能.
  - 例：指数部を補数にする.
- 規格は統一したほうがみんなが使える.
  - 例：浮動小数点数の規格が同じであれば，同じように四則演算が可能となる.
  - 例：通信規格を統一すれば，その規格に従って作れれた機器同士は通信可能となる.
- ルールの統一はどうする.
  - IEEEなどの業界団体が決める
  - 市場が決める（デファクト・スタンダード）
- 情報工学の世界では，自分で何でも作れる.
  - 自分のアイデアで世界を変えることが可能
  - しかし，自分で何でも作ることは非効率的．車輪の再開発を避ける．既存の規格，ライブラリなどを尊重.