# 電子計算機 **2E**

2E　2003.6.11

## 本日の授業のテーマ

先週の練習問題の解説と、これまでの学習をまとめます。

(1)　デジタルコンピューター
(2)　ビットと情報
(3)　基数の変換(2, 8, 10, 16進数)
(4)　小数の表現
(5)　負の数の表現
(6)　浮動小数点表示

## 1 デジタルコンピューター

- 世の中で使われているコンピューターのほとんどは、デジタルコンピュータです。

- デジタルコンピューターの内部では、2 進数が使われています。理由は以下の通りです。

  (1) ノイズに強い
  (2) ハードウェアーを実現するのが容易
  (3) 演算が簡単
  (4) ブール代数が使えて、論理演算が容易

- デジタルコンピューターは、整数の 0 と 1 で全ての処理を行います。データや命令が全て、0 と 1 で表現されるということです。

- 一方、アナログコンピューターが処理するデータは、連続的に変化する量です。命令 (デジタルコンピューターのプログラムに相当) は、ハードウェアーで実現されます。

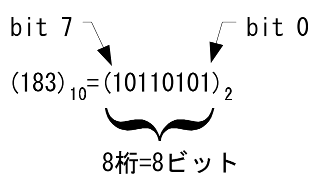
- デジタルコンピューターとアナログコンピュータの比較

| | デジタル | アナログ |
|---|---|---|
| 扱うデータ | 2 値 (0,1) | 連続値 |
| ノイズ | ノイズに強い | ノイズに弱い |
| 同一プログラムの実行結果 | 計算結果は同じ | 微妙に異なる (ノイズの影響) |
| プログラム変更 | 容易 | 困難 |

## 2 ビットと情報

- ビットとは情報の単位です。1 ビットで 2 つの事象が表現できます。2 ビットで 4 事象、3 ビットで 8 事象が表現できます。N ビットで、$2^N$ 個の事象が表現できます。

- 2 進数の各桁は各ビット (bit 0, bit 1, bit 2, ・・・) を表し、桁数は情報の量 (単位: ビット) を表します。




- 情報科学の世界では、ビット(bit)以外に、バイト(Byte)という単位も使われます。

  1 Byte(バイト) = 8 bits (ビット)

## 3 基数の変換(2, 8, 10, 16 進数)

・いろいろな数の表記方法があります。N 進数の場合、N 個の底で数を表現します。

| | |
|---|---|
| 2 進数 | 0, 1 |
| 8 進数 | 0, 1, 2, 3, 4, 5, 6, 7 |
| 10 進数 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| 16 進数 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F |

・桁上がりは、2 進数の場合 1 の次で 10 に、8 進数の場合 7 の次で 10 に、10 進数の場合 9 の次で 10 に、16 進数の場合 F の次で 10 になります。

・我々が通常用いていうる、数の表現の意味は、次の通りです。数字の並ぶ順序が重要です。これを「位取り記数法」と言います。

$$(1905)_{10}=(1\times10^3+9\times10^2+0\times10^1+5\times10^0)_{10}$$

・2 進数から、10 進数への変換は、通常の位取り記数法を考えれば、簡単です。

$$\begin{aligned}(1101)_2 &= (1\times10^{11}+1\times10^{10}+0\times10^1+1\times10^0)_2\\ &= (1\times2^3+1\times2^2+0\times2^1+1\times2^0)_{10} \quad \leftarrow \text{普通はここから計算}\\ &= (8+4+0+1)_{10}\\ &= (13)_{10}\end{aligned}$$

・2 進数の各桁の 10 進数の値を覚えておくと便利です。

1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096

・逆に、10 進数から 2 進数への変換は、2 で割った余りを並べる。

```
2 ) 19  ― 1        2 ) 2003 ― 1
2 )  9  ― 1        2 ) 1001 ― 1
2 )  4  ― 0        2 )  500 ― 0
2 )  2  ― 0        2 )  250 ― 0
     1             2 )  125 ― 1
                   2 )   62 ― 0
                   2 )   31 ― 1
                   2 )   15 ― 1
                   2 )    7 ― 1
                   2 )    3 ― 1
                          1
```

矢印の順に 0 と 1 を並べると 2 進数になる

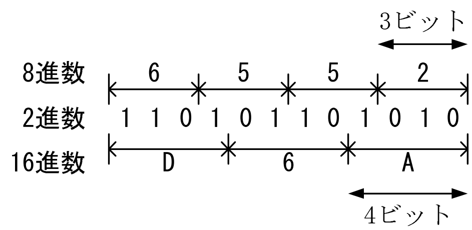
それぞれ、$(19)_{10}=(10011)_2$、$(2003)_{10}=(11111010011)_2$ です。

・基数の変換(8, 16→10 進数)。これも、2 進数と同じです。

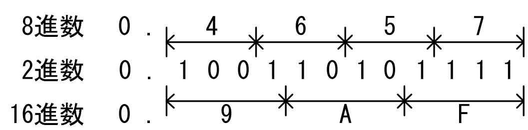
$$(376)_8 = (3\times10^2+7\times10^1+6\times10^0)_8$$
$$= (3\times8^2+7\times8^1+6\times8^0)_{10}$$
$$= (3\times64+7\times8+6\times1)_{10}$$
$$= (254)_{10}$$

$$(376)_{16} = (3\times10^2+7\times10^1+6\times10^0)_{16}$$
$$= (3\times16^2+7\times16^1+6\times16^0)_{10}$$
$$= (3\times256+7\times16+6\times1)_{10}$$
$$= (886)_{10}$$

・整数の基数の変換(2→8, 16 進数)。8 進数への変換は第 0 ビット(最小桁)から 3 桁ずつ区切り、16 進数への変換は 4 桁ずつ区切り計算します。

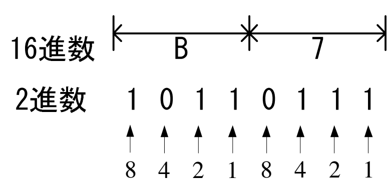



・小数の基数の変換(2→8, 16 進数)。8 進数への変換は小数点の次の桁から 3 桁ずつ区切り、16 進数への変換は 4 桁ずつ区切り計算します。




・桁数が合わない場合は、整数部は先頭に、小数部は最後尾に必要なだけゼロを書き足して考えます。例えば、16 進数へ変換する場合、$(101100)_2 = (00101100)_2$ や $(0.111111)_2 = (0.11111100)_2$ として計算します。

・小数、整数の基数の変換(8, 16→2 進数)。16 進数から 2 進数への変換は、16 進数の各桁を(1, 2, 4, 8)の和に分けて、それぞれのビットに対応させます。8 進数の場合は、(1, 2, 4)の和に分けます。




$(B)_{16}=(11)_{10}=(8+2+1)_{10}$ $(7)_{16}=(7)_{10}=(4+2+1)_{10}$

・基数の変換(10→進数 8, 16)。2 つの方法が有ります。

①一旦、2 進数へ変換した後、8 及び 16 進数へ変換する。 ←おすすめ

②整数の場合、8 又は 16 で割って、その余りが各桁になる。小数の場合は、8 又は 16 を乗じて、整数部が各桁になる。

## 4 小数の表現

・10 進数の小数の表現は、次のようになります。これも位取り記数法です。

$$(0.1325)_{10}=(1\times10^{-1}+3\times10^{-2}+2\times10^{-3}+5\times10^{-4})_{10}$$

・2 進数の小数の表現も同じです。したがって、2 進数小数を 10 進数小数への変換は、簡単です。

$$\begin{aligned}(0.10101)_2 &= (1\times10^{-1}+0\times10^{-10}+1\times10^{-11}+0\times10^{-100}+1\times10^{-101})_2\\ &= (1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+0\times2^{-4}+1\times2^{-5})_{10}\\ &= (0.5+0.125+0.03125)_{10}\\ &= (0.65625)_{10}\end{aligned}$$

←普通はここから計算

・10 進数小数から 2 進数小数への変換は少し難しくて、2 倍して、その整数部分を書き出します(右の筆算)。場合によっては、循環小数になる場合もあります。

$(0.65625)_{10}=(0.10101)_2$

$(0.1)_{10}=(0.00011001100110\cdots)_2$

| | |
|---|---|
| 0.65625 × 2 = 1.3125 | 0.1 × 2 = 0.2 |
| 0.3125 × 2 = 0.625 | 0.2 × 2 = 0.4 |
| 0.625 × 2 = 1.25 | 0.4 × 2 = 0.8 |
| 0.25 × 2 = 0.5 | 0.8 × 2 = 1.6 |
| 0.5 × 2 = 1.0 | 0.6 × 2 = 1.2 |
| 0.0 | 0.2 × 2 = 0.4 |
| | 0.4 × 2 = 0.8 |
| | 0.8 × 2 = 1.6 |
| | 0.9 × 2 |
| | ⋮ |

繰り返し

・2 進数小数から 16 進数小数への変換は小数点以下を 4 桁ずつ区切って、16 進数に変換します。桁が合わないときには最後尾にゼロを加えます。整数と同じ。

$(0.1010111010)_2$
$= (0.1010111010)_2$
$= (0.1010\ 1110\ 1000)_2$
$= (0.ae8)_{16}$

・16 進数小数から 2 進数小数への変換は小数点以下を 16 進数を順番に 2 進数へ変換すればよい。整数と同じ。

$(0.bf5)_{16}$
$= (0.1011\ 1111\ 0101)_2$
$= (0.101111110101)_2$

## 5 負の数の表現

・負の数の代表的な表現方法には、①絶対値表現　②1 の補数による表現 ③2 の補数による表現があります。

・これらのうち、現在のコンピューター内部での負の数の表現は、2 の補数が使われています。 たんに、補数と言えば、2 の補数のことです。

・コンピューター内部では、負の整数は 2 の補数による表現を行います。その手順は、以下の通りです。

① 負の数の絶対値を 2 進数で表現して、ビット反転する。
② +1 加算

［例］ $(-18)_{10}$ は、コンピューター内部、8 ビット表現では、$(11101110)_2$ と表されます(メモリーへの格納状態)。

| | | | |
|---|---|---|---|
| $(-18)_{10}$ | 00010010 | ← | 18 の 2 進数表現 |
| | 11101101 | ← | ビット反転 |
| | 11101110 | ← | +1 加算 |

・2 の補数を使うと、以下の有利な点があります。

・負の数の加算が通常の加算器で出来る。
・正の数の減算をする場合、① 2 の補数に変換して、② 加算器による加算で可能となる。減算器を作るより、この方が回路が簡単になる。

［例］ $(21-14)_{10}$ と $(14-21)_{10}$ を加算器(8 ビット)を使って計算する。

```
  00010101  ←  21            00001110  ←  14
+ 11110010  ← -14          + 11101011  ← -21
 100000111  ←  7(8 ビット)   11111001
   8ビット
第8ビットは無視            第7ビットが1
                           なので負の数
                             00000110  ←  ビット反転
                             00000111  ←  +1 加算
                              4+2+1=7  ←  10 進数に変換
```

・補数を求める手順 (①ビット反転 ②+1 加算) は、コンピューター内部表現では、×(−1)と同じです。

・コンピューター内部では、正の数は 2 進数でそのままの表現です。一方、負の数は 2 の補数を使います。正か負かの判断は、最上位のビットで判断します。最上位のビットが 0 ならば正、1 であれば負です。

・したがって、最上位のビットが符号を表すため、絶対値は残りのビットで表すことになります。8 ビットの場合、以下の範囲で整数が表現できます。整数の範囲は、−128～127 になります。

| | |
|---|---|
| 正の整数の最大値 | $(01111111)_2=(2^7-1)_{10}=(127)_{10}$ |
| 負の整数の最大絶対値 | $(10000000)_2=(2^7)_{10}=(128)_{10}$ |

## 6 浮動小数点表示

- 浮動小数点表示とは、指数化（例えば、$-0.123\ \sim 1^{-2}$）して数値を表現します。この指数部 (-0.123)と仮数部(-2)をコンピューターのメモリに記憶させます。

- 数値データが整数と指定されない限りこの浮動小数点が用いられます。

- この方法の長所と短所は、以下の通りです。

| | |
|---|---|
| 長所 | 決められたビット数内で、非常に小さな数値から大きな数値まで表現可能になる。 |
| 短所 | 桁落ち誤差が発生する場合がある。 |