コンピュータの仕組み
(7)

柴山　潔

コンピュータの仕組み

1　コンピュータシステム
2　ハードウェア
3　内部装置
4　プロセッサ(4)
5　メモリ
6　外部装置
7　論理回路
8　オペレーティングシステム

4 プロセッサ(4)

4.1 プロセッサの動作とその制御

4.2 プロセッサの機能 ―演算と制御―

4.3 演算器

4.4 順序制御機構

10進数と2進数

- 10進数演算：人間が必要時に行う計算
  - 特に，算術演算
- 2進数演算：コンピュータが行う計算（処理）
  - 算術演算も2進数演算（1ビットの2進数演算は論理演算）で実現する必要

10進数とその数表現

- 10進数："0","1","2",…,"8","9" の10種類の数字を用いて表し，1桁を"10"の重みで表現
  - 人間が扱う数

- 数表現：数の示し方や書き方

代表的な10進数の数表現

- 整数(integer)：自然数（正整数）と“0”および負整数のいずれか
  - "＋"と"－"の正負を表す符号（正数の場合は省略可）と数字によって表現

  （例）4130　－1　＋674355　－2453
- 実数(real)：符号（正数の場合は省略可）と数字以外に小数点" . "を用いて表現
  - 小数点によって分けられる整数部と小数部との2個の部分で構成

  （例）403.12　－0.763289　＋3.14　1.170367

コンピュータでの数表現の制約(1)

- コンピュータのハードウェア機構には大きさの制約（例：ボードやICの大きさ）＝「ハードウェアは硬い」
  - 簡単に，演算器を増設することは不可
- オーバフロー(overflow)：演算対象のデータ（特に，演算結果）が用意しているハードウェア機構（＝演算器やレジスタなど）に納まらずあふれること
  - 「オーバフローの発生」は，実装規模が限られたハードウェア機構では避けられない事象
  - 算術演算だけではなく，シフト演算（循環シフト以外）などでも発生

コンピュータでの数表現の制約(2)

- 「ハードウェアは硬い（変更が不可能あるいは不自由）」が演算器の構成方法にも大きな影響
  - 「プロセッサ内に，どんな種類の，どれくらいの長さの演算器を，いくつ装備するのか？」をハードウェア機構の製作時に決定する必要あり

（再掲）

マシン語の実際

- コンピュータ（内部装置）では，情報（命令とデータ）はビット（列）か2進数かで表現
  - ビット（列）："0"か"1"の2種の記号（＝論理値）のいずれかを連結した記号（列）
    - ビット(bit)："0"か"1"の1個の記号，（参考）バイト(Byte) ＝ 8ビット
  - 2進数："0"か"1"の数字を1桁の重みを"2"とする規則に従って書き並べた数値
- マシン語 → ビット列か2進数（どちらも，"0"か"1"の列）で表現

2進数と2進数表現

- コンピュータ（ハードウェア）の言葉は"0"と"1"だけを並べるビット（列）か2進数
  - 演算器で行う算術演算の対象となる数（数値）は2進数で表わす
- 2進数表現：“0”,“1”の2種類の数字だけを用い，1桁（1ビット）の重みを“2”とする数表現（＝「数（値）」の示し方や書き方）
  ＝内部表現
  - 内部装置での数表現
  - 内部装置では，特に数値を2進数表現で示す

（再掲）

2進数1ビット（桁）の（筆算による）加算例

- （下位ビットからの）桁上げなし
- （下位ビットからの）桁上げあり

$(0+0_{+0}=0)_{10}$

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array}$$

$(0+1_{+0}=1)_{10}$

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

$(1+0_{+0}=1)_{10}$

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

$(1+1_{+0}=2)_{10}$

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

（上位ビットへの）桁上げ

$(0+0_{+1}=1)_{10}$

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

$(0+1_{+1}=2)_{10}$

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

（上位ビットへの）桁上げ

$(1+0_{+1}=2)_{10}$

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

（上位ビットへの）桁上げ

$(1+1_{+1}=3)_{10}$

$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 1 \end{array}$$

（上位ビットへの）桁上げ

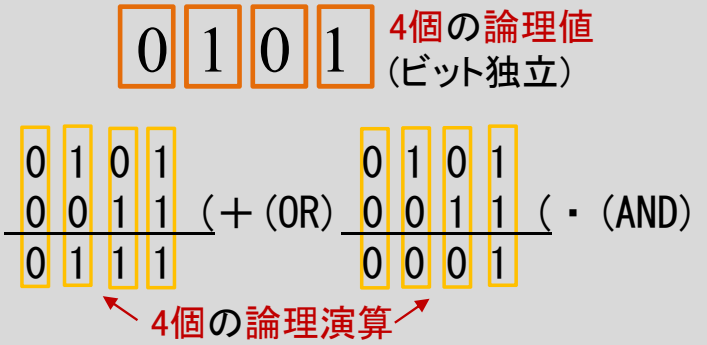
（重要）

（再掲）

ビット列と2進数値との比較

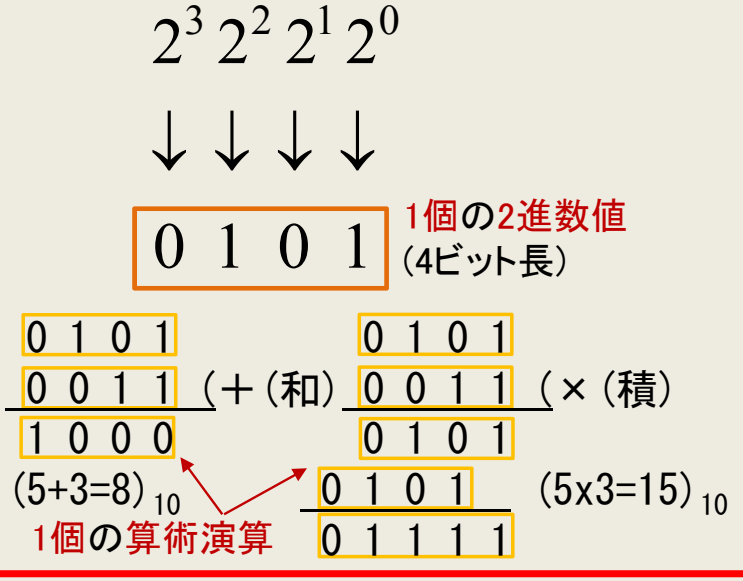
■ビット列：論理値の集まり

- 各ビットが独立した1個の論理値
  - 適用演算は論理演算（否定，論理積，論理和）
  - 1ビット（個）論理演算機構は対応論理素子1個で実現



■2進数値：2進数表現した数値

- ビット列全体で1個の数値として意味
- 各ビットに固有の相異なる2のべき乗の重み
- 並び順がその2進数の数値を決定
  - 適用演算は算術演算（四則演算ほか多数）
  - 算術演算では，隣接（上位と下位）ビット間に依存関係
  - 算術演算機能は組み合わせ論理回路で実現



2進数⇔10進数変換(1)

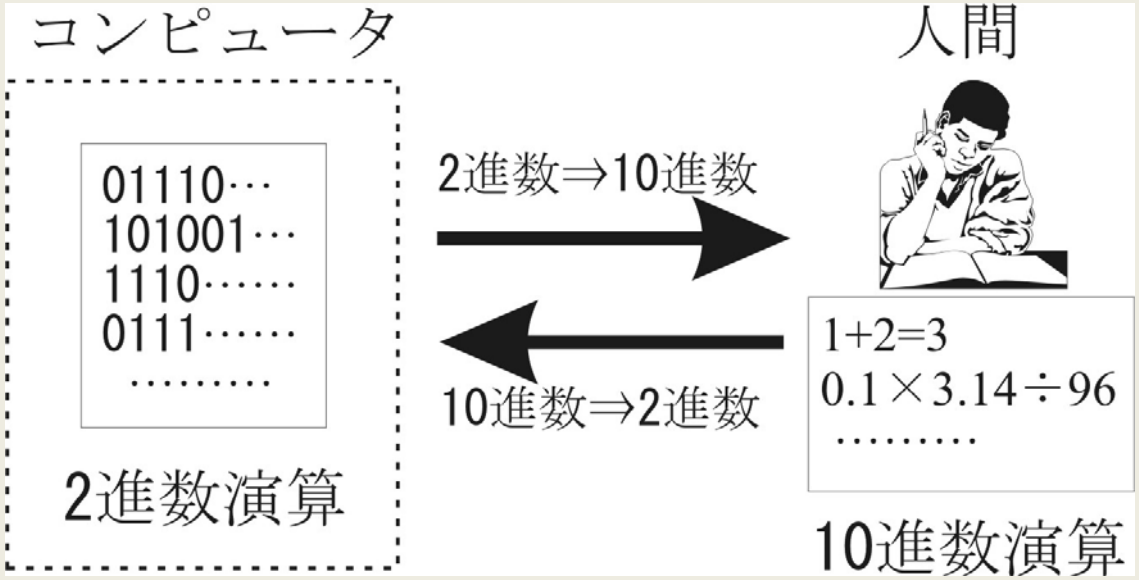
- 人間がコンピュータに計算を代行，あるいは人間－コンピュータ間の対話をする場合には，次の処理機能が必要

(1) 人間→コンピュータ：10進数表現の演算データを2進数表現に変換（10進数⇒2進数変換）

(2) コンピュータ→人間：2進数表現の計算結果を10進数表現に変換（2進数⇒10進数変換）

2進数⇔10進数変換(2)

- (1)(2)を行う2進数⇔10進数変換機構は，演算機構の1つとして，プロセッサ内に装備



10進数表現

$$D_M D_{M-1} \cdots D_2 D_1 \ . \ d_1 d_2 \cdots d_{m-1} d_m \qquad (4.1)$$

- ただし，いずれのD,dとも$0,1,2,\cdots,9$のどれか
- 特に断りがない場合と$(\cdots)_{10}$は10進数

2進数表現

$$B_N B_{N-1} \cdots B_2 B_1 \ . \ b_1 b_2 \cdots b_{n-1} b_n \qquad (4.2)$$

- ただし，いずれの B,b とも $0,1$ のどちらか
- $(\cdots)_2$は2進数（ビット列）

10進数の数値

$$\left(D_M \cdots D_2 D_1 \ . \ d_1 d_2 \cdots d_{m-1} d_m\right)_{10}$$

$$= D_M \times 10^{M-1} + \cdots + D_1 \times 10^0 + \frac{d_1}{10^1} + \cdots + \frac{d_m}{10^m} \qquad (4.3)$$

■ 式4.3は，10進数での数式で（数学的に）示した10進数値

2 進数の数値

$$\left(B_N B_{N-1} \cdots B_2 B_1 \ . \ b_1 b_2 \cdots b_{n-1} b_n\right)_2$$

$$= B_N \times 2^{N-1} + \cdots + B_1 \times 2^0 + \frac{b_1}{2^1} + \cdots + \frac{b_n}{2^n} \qquad (4.4)$$

■ 式4.4は，10進数での数式で（数学的に）示した2進数値

2進数⇒10進数変換の手順

$$\left(B_N B_{N-1} \cdots B_2 B_1 \ . \ b_1 b_2 \cdots b_{n-1} b_n\right)_2$$

$$= B_N \times 2^{N-1} + \cdots + B_1 \times 2^0 + \frac{b_1}{2^1} + \cdots + \frac{b_n}{2^n} \qquad (4.4)$$

■ 式4.4は10進数での数式

- 式4.4の右辺の $B_N, \cdots, B_1, b_1, \cdots, b_n$ にそれぞれ対応する"0"か"1"を代入・計算した値（10進数）が変換後の10進数表現

例題4.1：
2進数の$(1101.1011)_2$を10進数に変換

［解答］

$$(1101.1011)_2$$

$$= 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + \frac{1}{2^1} + \frac{0}{2^2} + \frac{1}{2^3} + \frac{1}{2^4}$$

$$= \underline{\underline{(13.6875)_{10}}}$$

10進数⇒2進数変換の手順

- 整数部: $D_M D_{M-1} \cdots D_2 D_1$

と

- 小数部: $d_1 d_2 \cdots d_{m-1} d_m$

とに分けて変換

$$\underline{D_M D_{M-1} \cdots D_2 D_1} \cdot \underline{d_1 d_2 \cdots d_{m-1} d_m}$$

10進数整数部⇒2進数整数部変換

- 10進数整数部を“2”で割り（÷2；除算），その商（10進数整数）を“2”で割る（÷2）ことを商が“0”になるまでくり返し
  - その一連の割算（除算）での余り（剰余）を，順に，2進数整数部として，最下位から上位へと並べる

例題4.2:
10進整数の$(13)_{10}$を2進整数に変換

［解答］

$$13 \div 2 = 6 \ldots 1$$

$$6 \div 2 = 3 \ldots 0$$

$$3 \div 2 = 1 \ldots 1$$

$$1 \div 2 = 0 \ldots 1$$

$$(13)_{10} = \underline{\underline{(1101)_2}}$$

10進数小数部⇒2進数小数部変換

- 10進数小数部に“2”をかけ（×2；乗算），その積の小数部だけに“2”をかける（×2）ことを，積の小数部が“0”になるまでくり返し
  - その一連のかけ算（乗算）の積の整数部（“0”か“1”）を，順に，2進数小数部として，小数点以下から書き並べる

例題4.3:
10進小数の$(0.6875)_{10}$を2進小数に変換

［解答］

$$0.6875\times 2 = 1.\underline{375}$$

$$0.375\times 2 = 0.\underline{75}$$

$$0.75\times 2 = 1.\underline{5}$$

$$0.5\times 2 = 1.\underline{0}$$

$$(0.6875)_{10} = \underline{\underline{(0.1011)_2}}$$

10進数小数⇒2進数小数変換での丸めと誤差

- 10進数小数⇒2進数小数変換において，「有限小数の10進数を2進数に変換すると無限小数になる」場合がある
  - 無限小数を有限のハードウェア機構で処理したり格納したりするには，丸め（＝その数をハードウェア機構の大きさに合わせたビット数で表現）という操作が必要
    - 丸められた数は近似値で，丸める前の数と等しくなく，誤差が発生
  - 丸めの機能や誤差への対策が10進数小数⇒2進数小数変換機構では必須

例題4.4：
「10進数の有限小数（実数）$(0.3)_{10}$を2進数に変換→無限循環小数」を確認

［解答］

$$0.3\times 2=0.\underline{6}\quad 0.6\times 2=1.\underline{2}\quad 0.2\times 2=0.\underline{4}\quad 0.4\times 2=0.\underline{8}$$

$$0.8\times 2=1.\underline{6}\quad 0.6\times 2=1.\underline{2}\quad \cdots$$

$$(0.3)_{10}=\underline{\underline{(0.0100110011001100 1\cdots)_2}}$$

$$\approx(0.010011001)_2=(0.298828125)_{10}$$

例題4.4(補遺):
「10進数の有限小数(実数)$(0.1)_{10}$を2進数に変換→無限循環小数」を確認

[解答]

$$0.1\times 2=0.\underline{2}\quad 0.2\times 2=0.\underline{4}\quad 0.4\times 2=0.\underline{8}\quad 0.8\times 2=1.\underline{6}$$

$$0.6\times 2=1.\underline{2}\quad 0.2\times 2=0.\underline{4}\quad \cdots$$

$$(0.1)_{10}=\underline{\underline{(0.0001100110011001100\,11\cdots)_2}}$$

$$\approx(0.000110011)_2=(0.099609375)_{10}$$

整数と実数

- 人間が行う計算の対象となる数（10進数）としては，整数と実数とが代表的
  - ➢ この両者を扱うだけで算術演算に関しては十分

↓

➢ コンピュータにおいても，整数と実数とを計算対象の数とする

整数演算と実数演算

■ コンピュータの内部装置で整数演算と実数演算とを区別する理由

➢ 四則演算を代表とする種々の算術演算の方法が異なる

- 実数に対する演算では，整数部と小数部とを分ける小数点の扱いについて考慮する必要
  - ➢ コンピュータ（特に，演算器）のハードウェア機構では，小数点（の位置）も"0"と"1"で表す必要
- 整数演算の方が実数演算よりもやさしい
- 有限のハードウェアを使って実数表現するために発生する丸めや誤差への対処が必要
  - ➢ 丸め機能や誤差対策は，無限循環小数（有理数）の外に，平方根 $\sqrt[2]{\ }$，円周率 π，自然対数の底 e などの無理数に対しても必要