初版：2012年8月24日
改訂：2018年7月14日
監修：津田伸生

金沢工業大学

情報工学科

2018 年度版

コンピュータシステム基礎教科書

（補足編）

目次

- 毎週，受講前に必要な部分を印刷して持参することを推奨する
- 毎週の授業では各自のノート等に要点のメモをとること
- 毎週の授業の予習・復習を欠かさないこと．特に復習では「何が約束事なのか」を整理して例題と問題を見直すこと．

第０章　10 進数と 2 進数

コンピュータで扱う数表現の基礎は「工学大意（情報）」で学んだ．この補足編では，より正確に数値データを扱えるようになるために数表現について復習する．

0.1　数体系

- 10 進数で数値を表すことを **10 進法**，2 進数で数値を表すことを **2 進法**という．正式には**十進法**，**二進法**と書くがここでは数字を用いる．以下，この節の説明では**ゼロを含む正整数のみ**を扱う．
- 10 進数の 10 や 2 進数の 2 などを**基数**(radix)という．基数を替えて数の表現を変えることを**基数変換**という．ある基数のもとで整数や小数の記述方法を定めたものを**数体系**という．
- 10 進数の各桁は，0～9 の 10 種類の数字を用いて 0～9 の 10 通りの状態（何個ある）を表している．9 を超えた場合は次の桁へ 1 桁上がりする．3 桁の 10 進数 213 では，最下位桁の 3 は 10 進数の 1 が 3 個ある，2 桁目の 1 は 10 進数の 10 が 1 個ある，3 桁目の 2 は 10 進数の 100 が 2 個ある，を意味している．
- 2 進数の各桁は，0～1 の 2 種類の数字を用いて‘0’または‘1’の 2 通りの状態（「ない」または「ある」）を表している．‘1’を超えた場合は次の桁へ‘1’が桁上がりする．2 進数では，各桁の数値が示す情報量(状態)，すなわち ‘0’または‘1’を 1 ビットとして数える．これにより **2 進数の桁数もビットで数える**．3 ビット(桁)の 2 進数“101”では，最下位ビットの‘1’は 2^0=1 がある，2 ビット目の‘0’は 2^1=2 がない，3 ビット目の‘1’は 2^2=4 がある，を意味しており，全体で 10 進数の 5 に対応する数値を表している．

10進数と2進数

●10進数

・各桁で0～9の10通りの数字を使う

・ある桁の値が9を超えると次の桁に1が桁上がりする

	10^4	10^3	10^2	10^1	10^0
桁:	万	千	百	十	一
例）	・	・	1	1	7

●2進数

・各桁で0または1の2通りの数字を使う

・ある桁の値が1を超えると次の桁に1が桁上がりする

（対応する10進数）	128	64	32	16	8	4	2	1
桁:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
例）$(117)_{10}$ ＝	（・	1	1	1	0	1	0	1 $)_2$

＊2進数では桁をビットという(注：情報の単位もビットという)

・ 2 進数で，後で述べる**「データ形式」**を指定していない場合は，上位ビットに数値の意味がない場合は‘0’を省略する．これを**「通常表記」**という

　　例）　“101”を　“0000101”のように書く必要はない

・ 2 進数では最上位ビット(左端の桁)を **MSB**：most significant bit，最下位ビット(右端の桁)を **LSB**：least significant bit という

0.2　2 進数符号なし絶対値形式

・ n ビット 2 進数の全ビットでゼロを含む正整数を表す方法（データ形式）を**「n ビット 2 進数符号なし絶対値形式」**という．**この形式では負数は表現できない．**

　　注１）符号とは正(+)または負(−)を表す記号のこと．単に**「n ビット 2 進数符号なし形式」**ということもある．

　　注２）n の値は，一般には 4，8，16，24，32，64 など 4 の倍数を使う．

・ 8 ビット 2 進数符号なし絶対値形式では，通常表記の“101”は，“0000 0101”となる．

　　注３）**上位ビットの‘0’は省略できない．**必ず 8 ビットで数値を表現する．

・ 8 ビット 2 進数符号なし絶対値形式では“0000 0000”～“1111 1111”の範囲の数値を表現できる．この範囲に対応する 10 進数は 0～255 である．表現できる数値の種類は 2^8（=256）通り，最大値の 255 は $2^8 - 1$ で与えられる．

・ n ビット 2 進数符号なし絶対値形式では，表現できる数値の種類は式(1)，最大値は式(2)で与えられる．

$$2^n \tag{1}$$

$$2^n - 1 \tag{2}$$

0.3　10 進正整数の 2 進数変換

・最大 255 までの 10 進正整数を**「8 ビット 2 進数符号なし絶対値形式」**に変換する操作を考える．10 進数を変数 D で表すと，その値は式(3)に示す 2 のベキ乗の項からなる 10 進数多項式で表現できる．

$$D = (b_7 \times 2^7) + (b_6 \times 2^6) + (b_5 \times 2^5) + (b_4 \times 2^4) + (b_3 \times 2^3) + (b_2 \times 2^2) + (b_1 \times 2^1) + (b_0 \times 2^0) \tag{3}$$

ここで b_7～b_0 はそれぞれ‘0’または‘1’の値をもつ変数

・ 式(3)における b_7～b_0 を取り出して式(4)の右辺のように並べたものが 8 ビット 2 進数符号なし絶対値形式である．

$$(D)_{10} = (b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)_2 \tag{4}$$

　　注４）基数が異なる値を同じ式に書く場合は数値を括弧で囲んで添字に基数を書く．

　　例）　$(61)_{10} = (0011\ 1101)_2$ (5)

・ **除算法による 10 進 2 進変換**

$(D)_{10}$ から $(b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)_2$ を求めるには D を 2 で割る操作を繰り返す.

式(3)の両辺を 2 で割ると式(6)となる.

$$D/2 = \{(b_7 \times 2^6) + (b_6 \times 2^5) + (b_5 \times 2^4) + (b_4 \times 2^3) + (b_3 \times 2^2) + (b_2 \times 2^1) + (b_1 \times 2^0)\} \quad (6)$$

式(6)の左辺と右辺はそれぞれを 2 で割って得られる商であり，左辺と右辺で共通に得られる剰余（余り）は b_0 となる.

これ以降，商をさらに 2 で割って剰余を得る操作を繰り返すことで b_1 ～b_7 が求められる.

・ **除算法による 10 進 2 進変換の実際**

例）$(61)_{10}$ の 2 進変換

2) 61

2) 30　剰余　$1 = b_0$

2) 15　剰余　$0 = b_1$

2) 7　剰余　$1 = b_2$

2) 3　剰余　$1 = b_3$

2) 1　剰余　$1 = b_4$

0　剰余　$1 = b_5$

商が 0 になったら終了.　以下 $b_6 = 0$, $b_7 = 0$

答え　$(61)_{10} = (0011\ 1101)_2$

・ **なぜ 2 で割ると 10 進 2 進変換できるのか**

2 進数 $(b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)_2$ を 2 で割る操作は 1 ビット右シフト（下位シフト）で行える．このとき桁落ち（アンダーフロー）する b_0 は剰余に相当する.

$(b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)_2$　1 ビット右シフト→　$(0\ b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1)_2$

アンダーフロー b_0

このような 2 進数の右シフトの繰り返しでアンダーフローする 2 進数のビットを順次取り出す操作は，対応する 10 進数を 2 で割る操作を繰り返す操作と等価である.

・ 255 以上の 10 進正整数も同様な方法で 2 進数に変換できる.

・ **基数が 2 以外の p 進数へも p で割る除算法で変換できる**（詳細は省略）.

・ **減算法による 10 進 2 進変換**

2 のベキ乗の値を暗記しておくと除算法より簡単に 10 進 2 進変換を行える.

2^0=1，2^1=2，2^2=4，2^3=8，2^4=16，2^5=32，2^6=64，2^7=128，2^8=256，2^9=512，2^{10}=1024

例）$(61)_{10}$ の 2 進変換

61

－ 32　61 を超えない 2 のベキ乗の値 $32 = 2^5$を引く

29　差

－ 16　29 を超えない 2 のベキ乗の値 $16 = 2^4$を引く

13　差

－ 8　13 を超えない 2 のベキ乗の値 $8 = 2^3$を引く

5　差

－ 4　5 を超えない 2 のベキ乗の値 $4 = 2^2$を引く

1　差　$= 2^0$　(LSB)

差が 1 または 0 になったら終了.

最後の差 1 と引いた 2 のベキ乗の値が元の値に含まれるとして 2 進数を得る.

答え　$(61)_{10} = (0011\ 1101)_2$

・ **2 進 10 進変換**

8 ビット 2 進数符号なし絶対値形式 $(b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0)_2$を 10 進数に変換するには, 式(3)において b_7～b_0 に'0'または'1'を代入し和を求める.

例) $(0011\ 1101)_2$の場合

$$D = (0 \times 2^7) + (0 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

$1 = 2^0$

$4 = 2^2$

$8 = 2^3$

$16 = 2^4$

$+ \quad 32 = 2^5$

61　答え　$(61)_{10}$

注 1）以上, 8 ビット 2 進数符号なし絶対値形式を用いて基数変換を説明したが, 8 ビット以上の 2 進数符号なし絶対値形式の場合, 上位ビットの'0'を省略した通常表記の 2 進数でも同様に変換できる.

注 2）2 のベキ乗の 10 進数 $2^0 = 1$　～　$2^{10} = 1024$ は暗記すること.

問題 0.1　次の 10 進数を 2 進数に変換せよ（上位ビットの 0 は省略せよ).

(1)　$(24)_{10}$　　(2)　$(29)_{10}$

(3)　$(101)_{10}$　　(4)　$(936)_{10}$

＊変換結果が正しいかどうかは逆変換で確かめる（検算する）こと

問題 0.2　次の通常表記の 2 進数を 10 進数に変換せよ．

(1)　$(1\ 1010)_2$　　(2)　$(11\ 1101)_2$

(3)　$(101\ 1000)_2$　　(4)　$(1100\ 1011)_2$

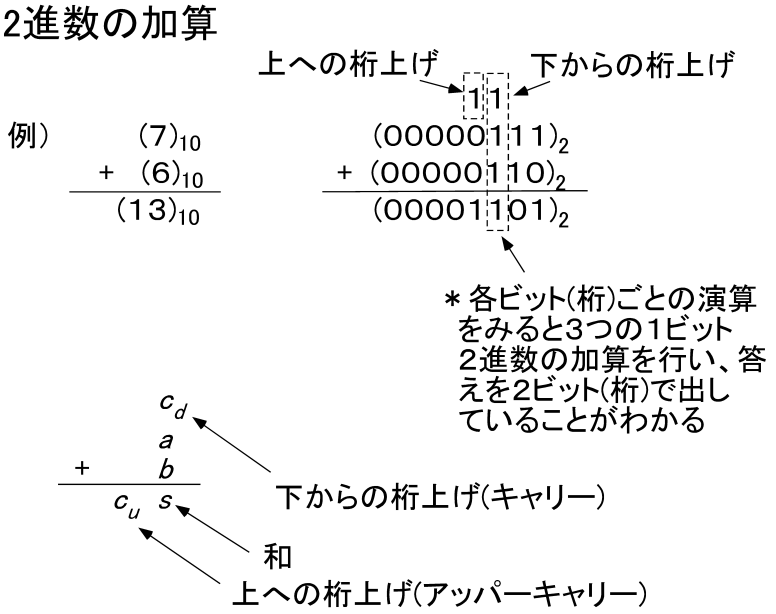
＊変換結果が正しいかどうかは逆変換で確かめる（検算する）こと

0.4　2 進正整数の四則演算

ここでは，2 進数はゼロの場合を含む正整数とする．なお図では，8 ビット 2 進数符号なし絶対値形式で示すが, 上位ビット'0'を省略した通常表記の 2 進数でも同様に演算できる．

・ **2 進正整数の加算**

各桁（ビット）で，下からの桁上げ，被加数，加数の 3 つの 1 ビット 2 進数の和を計算し，和が'1'を超えたら次の桁（ビット）へ'1'が桁上がりする．



・ **2 進正整数の減算**

各桁（ビット）で，'0'－'0'＝'0'，'0'－'1'＝'1'（上の桁から'1'を借りる），'1'－'0'＝'1'，'1'－'1'＝'0' の計算を行う．

ここでは，減算結果が負数にならない場合のみを扱う．

2進数の減算

例）

$$\begin{array}{r} (7)_{10} \\ -\ (4)_{10} \\ \hline (3)_{10} \end{array} \qquad \begin{array}{r} (00000111)_2 \\ -\ (00000100)_2 \\ \hline (00000011)_2 \end{array}$$

補数加算による減算

例）$(-4)_{10}$ を8-bit 2の補数形式へ変換

$$\begin{array}{r} (256)_{10} \\ -\ (4)_{10} \\ \hline (252)_{10} \end{array} \qquad \begin{array}{r} (100000000)_2 \\ -\ (00000100)_2 \\ \hline (11111100)_2 \end{array} = (-4)_{10}$$

符号ビット（＋／－）
補数形式では最上位ビットが
'0'は正数、'1'は負数

$$\begin{array}{r} (00000111)_2 \\ +\ (11111100)_2 \\ \hline 1\ (00000011)_2 \end{array}$$

切り捨てる

注）補数加算による減算は **0.7.3** で詳細説明する

問題 0.3 次の通常表記 2 進正整数の加減算を行え．また，10 進数で検算せよ．なお，ここでは減算は通常の減算で行え．

(1) $(1110)_2 + (1\ 1101)_2$　　(2) $(1101\ 0100)_2 + (111\ 0110)_2$

(3) $(10\ 0010)_2 - (111)_2$　　(4) $(1100\ 1010)_2 - (101\ 0101)_2$

- **2 進正整数の乗算**

  被乗数の左シフト（上位シフト）と加算の繰り返しで行う．

2進数の乗算

例）

$$\begin{array}{r} (6)_{10} \\ \times\ (5)_{10} \\ \hline (30)_{10} \end{array} \qquad \begin{array}{r} (00000110)_2 \\ \times\ (00000101)_2 \\ \hline (00011110)_2 \end{array}$$

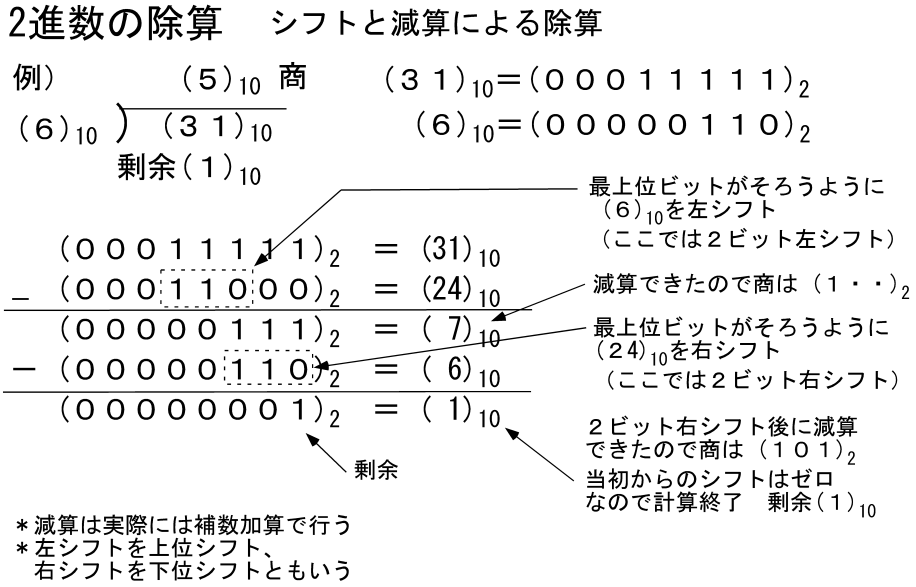
左シフトおよび加算による乗算

$$\begin{array}{l} (00000110)_2 \times (1)_2 \\ (00001100)_2 \times (0)_2 \\ +\ (00011000)_2 \times (1)_2 \\ \hline (00011110)_2 \end{array}$$

all '0' になり加算しなくてよい

- **2 進正整数の除算**

  除数の左シフト（上位シフト）による桁合わせと減算・右シフト（下位シフト）の繰り返しで行う．



問題 0.4 次の通常表記 2 進数の乗除算を行え．また，10 進数で検算せよ．

(1) $(1111)_2 \times (1111)_2$　　(2) $(10\ 1101)_2 \times (1\ 1011)_2$

(3) $(10\ 0010)_2 \div (111)_2$　　(4) $(1010\ 1100)_2 \div (1010)_2$

0.5　小数の 10 進 2 進変換

・ 式(7)に示すように，正の 10 進小数部 F を 4 ビットの 2 進小数部 $b_{-1}\ b_{-2}\ b_{-3}\ b_{-4}$で近似的に表現することを考える．

$$(.F)_{10} \fallingdotseq (.b_{-1}\ b_{-2}\ b_{-3}\ b_{-4})_2 \qquad (7)$$

式(7)では数値の左に小数点のみを記述し，整数部のゼロは省略してある．近似的の意味は，2 進小数部のビット数が有限の場合，10 進小数部 F が 1 桁であったとしても変換誤差が発生する場合があるためである．

式(7)の右辺の 2 進小数部は式(8)の右辺の 10 進数多項式で表現できる．

$$(.F)_{10} \fallingdotseq (.(b_{-1} \times 1/2^1) + (b_{-2} \times 1/2^2) + (b_{-3} \times 1/2^3) + (b_{-4} \times 1/2^4))_{10}$$
$$= (.(b_{-1} \times 0.5) + (b_{-2} \times 0.25) + (b_{-3} \times 0.125) + (b_{-4} \times 0.0625))_{10} \qquad (8)$$

・ 10 進小数部 F の 2 進変換は，式(9)に示すように，式(8)の両辺に 2 を掛ける操作で行える．これを**乗算法による小数変換**という．

$$(.F \times 2)_{10} \fallingdotseq ((.(b_{-1} \times 0.5) + (b_{-2} \times 0.25) + (b_{-3} \times 0.125) + (b_{-4} \times 0.0625)) \times 2)_{10}$$
$$= (b_{-1} . (b_{-2} \times 0.5) + (b_{-3} \times 0.25) + (b_{-4} \times 0.125))_{10} \qquad (9)$$

すなわち，2 を掛けることによって 2 進小数部 1 ビット目の b_{-1}が整数部へ桁上りするのでその値を取り出す．以降，桁上りした整数部をゼロにして 2 を掛ける操作を 3 回繰り返すことにより同様に b_{-2}～b_{-4}を得る．2 進小数部が 4 ビット以上の場合も同様にして変換できる．

・乗算法による 10 進小数の 2 進小数変換の実際

例) $(.7)_{10}$ の 4 ビット 2 進小数変換

$(.7 \times 2)_{10} = (1.4)_{10}$　$b_{-1} = 1$

$(.4 \times 2)_{10} = (0.8)_{10}$　$b_{-2} = 0$

$(.8 \times 2)_{10} = (1.6)_{10}$　$b_{-3} = 1$

$(.6 \times 2)_{10} = (1.2)_{10}$　$b_{-4} = 1$　　答え　$(.7)_{10} \fallingdotseq (.1\ 0\ 1\ 1)_2$

- 2 進小数の各ビットは 1 ビット上位の値の 1/2 になっている．この性質は 2 進整数と同じである．
- ビット数を指定していない通常表記の 2 進小数では下位の意味のない '0' は省略できる．

　　例)　".1010" は ".101" と書いてよい

- 2 の負のベキ乗の値，$1/2^1 = 2^{-1} = 0.5$，$1/2^2 = 2^{-2} = 0.25$，$1/2^3 = 2^{-3} = 0.125$，$1/2^4 = 2^{-4} = 0.0625$ を暗記しておくと，減算法により 4 ビットまでの 2 進小数に変換できる．(例は省略)

- **2 進小数の 10 進小数変換**

2 進小数$(.b_{-1}\ b_{-2}\ b_{-3}\ b_{-4})_2$が与えられたとき，10 進小数$(.F)_{10}$を得るには式(8)の右辺に b_{-1} ～b_{-4}を代入し和を求める．

例) $(.1\ 0\ 1\ 1)_2$の 10 進小数変換

1×0.5

0×0.25

1×0.125

$+\quad 1 \times 0.0625$

0.6875　　答え　$(.1\ 0\ 1\ 1)_2 = (.6875)_{10}$

この 10 進小数の値を$(.7)_{10}$と比較すると差は$(.0125)_{10}$となり誤差率は約 1.8%となる．

・2 進小数のビット数と変換誤差の評価

例) $(.7)_{10}$ の 8 ビット 2 進小数変換

$(.7)_{10} \fallingdotseq (.1011\ 0011)_2$

上記の逆変換

$(.1011\ 0011)_2 = (.69921875)_{10}$

$(.7)_{10}$と比較すると差は$(.00078125)_{10}$となり誤差率は約 0.11%となる．

問題 0.5 次の 10 進小数を通常表記の 2 進小数に変換せよ．

(1)　$(.5625)_{10}$　　(2)　$(.4375)_{10}$

問題 0.6 次の整数部と小数部をもつ通常表記の 2 進数を 10 進数に変換せよ．

(1)　$(10\ 1101.101)_2$　　(2)　$(1110\ 0101.1101)_2$

問題 0.7 次の設問について理由を説明せよ.

(1) なぜ 10 進整数を 2 で割ると 2 進整数に変換できるのか.

(2) なぜ 10 進小数に 2 を掛けると 2 進小数に変換できるのか.

0.6 固定小数点 2 進数符号なし絶対値形式

・ n ビットの 2 進数において，整数部を n－n_fビット，小数部を n_fビットとしてゼロを含む正の数値を表すデータ形式を**「n ビット固定小数点 2 進数符号なし絶対値形式」**という．n と n_fはあらかじめ決めておく．単に**「n ビット固定小数点 2 進数符号なし形式」**という場合もある.

例えば，16 ビット(2 バイト)固定小数点 2 進数符号なし絶対値形式で n_fを 4 ビットとした場合は，整数部は 12 ビットとなり，全体は

$$(b_{11}\ b_{10}\ \cdots\ b_1\ b_0\ b_{-1}\ b_{-2}\ b_{-3}\ b_{-4})_2$$

となる．**注）小数点は書かない．小数点は文字のため数値データでは使用できない.**

例) $(61.7)_{10} \fallingdotseq (0000\ 0011\ 1101\ 1011)_2$

・ 固定小数点形式を用いる場合の注意事項

小数部の桁数が決まった 10 進数を n ビット固定小数点 2 進数符号なし絶対値形式に変換する場合は，次のようにすることで 2 進小数を使わないようにできる.

例えば日本円で 61.7 円という金額がありこれを 2 進数で表現するには，最小単位を銭(0.01 円)として 6170 銭とすると, 整数部のみの 16 ビット固定小数点 2 進数符号なし絶対値形式により

$$(6170)_{10} \fallingdotseq (0001\ 1000\ 0001\ 1010)_2$$

と表現される.

注１）上記の 2 進数の'0'と'1'の配列は$(61.7)_{10}$のものとは異なる

注２）固定小数点形式で，小数部のビット数を小さく設定すると小数部の誤差が大きくなり，逆に小数部のビット数を大きくすると整数部のビット数が小さくなる．このため，整数部と小数部をもつ数値には，0.11 で述べる**「浮動小数点形式」**が一般に用いられる.

0.7 負数の表現

・通常表記の 2 進数では，負を示す－記号を用いて負数であることを表現できる.

例) $(-5)_{10} = (-101)_2$ または $-(5)_{10} = -(101)_2$

注）“－”の位置は括弧の内側でも外側でもよい

・“＋”や“－”の記号は文字であるため直接計算処理を行う数値データ（数値データワード）では使用できない.

・ このため数値データワードでは，2 値状態を表す数字'0'と'1'を用いて「ゼロまたは正」と「負」を表す以下の方法を用いる．

0.7.1 2 進数符号付き絶対値形式

・ n ビット 2 進数絶対値形式において，最上位ビット(MSB)を符号ビットとしてゼロおよび正(+)の場合は'0'，負(−)の場合は'1'を置くデータ形式を**「n ビット 2 進数符号付き絶対値形式」**という．

例） 8 ビット 2 進数符号付き絶対値形式では以下となる．

$(61)_{10} = (0011\ 1101)_2$

$(-61)_{10} = (1011\ 1101)_2$

・8 ビット 2 進数符号付き絶対値形式ではデータ部は 7 ビットとなるため，表現できる範囲は，　ゼロおよび正側　$(0000\ 0000)_2 = (0)_{10}$ ～ $(0111\ 1111)_2 = (127)_{10}$

負側　$(1000\ 0001)_2 = (-1)_{10}$ ～ $(1111\ 1111)_2 = (-127)_{10}$

となり，表現できる数値の種類は $2^8 - 1$（=255）通り，最大値は$\pm(2^7 - 1)$で与えられる．

n ビットの場合は，表現できる数値の種類は式(10)，最大値は式(11)で与えられる．

$$2^n - 1 \qquad (10)$$

$$\pm(2^{n-1} - 1) \qquad (11)$$

・ **「2 進数符号付き絶対値形式」は，数値データワードとして記憶はできるが，そのままでは直接演算できない**ため，これが可能な **0.7.3** で述べる**「2 の補数形式」**へ変換するための**「中間表現」**として使用される．

0.7.2 補数による負数表現

・ コンピュータでは加算と減算を加算回路のみを使って実行する．このため，正の数から正の数を減算する場合は引く値を補数に変換して負の数とし，引かれる値に加算する．加算においても負の数は補数で表現する．

・ **10 進数による補数の説明：**　補数とは，2 桁の 10 進数（0～99）で説明すると，0～49 はそのままゼロおよび正の値に対応させ，負の値には 2 桁を超える最初の数である 100（これを上界値という）からゼロを除いた正の絶対値である 1～49 を減算して得られる 99～51 を−1～−49 に対応づけるものである．

ゼロおよび正の値	負の値（対応する符号付き絶対値）
0	
1	99 (−1)
2	98 (−2)
：	：
49	51 (−49)

すなわち，補数とは，使用する 数の領域の桁数が決まっている場合 に，その数の領域を２分し，下位側をゼロおよび正の数，上位側を負の数として使用するものである．ただし，負の数が示す値の大小関係は対応する絶対値と逆になる．

注１）50（−50）は正数（絶対値）に変換できないため，ここでは表現の範囲外とする．

注２）上界値 100 から 2 桁の絶対値を減算することは，**2 桁の 00 の 3 桁目に 1 の借り（ボロー（borrow）ともいう）を仮定して減算を行うことと等価**である．

・ 上記の **10 の補数**は絶対値を上界値 100 から減算する代わりに次の規則で簡単に変換できる．

「変換したい絶対値の最下位桁からみてゼロはそのまま，最初のゼロ以外の桁は 10 から減算し，次の桁からは 9 から減算する」

例１） 絶対値が 31 → 69（−31）

例２） 絶対値が 30 → 70（−30）

・ 補数表現した負の数を再度補数変換すると元の正の数（絶対値）に戻る．

例１） 負数　69（−31）→ 絶対値　31

例２） 負数　70（−30）→ 絶対値　30

・ 補数加算による減算の例（10 進数 2 桁のデータ形式の場合）

例１）正の数からの減算

	30	（正の数）
＋	99	（負の数：−1）
	29	（正の数）

↑3 桁目への桁上りは切捨て

例２）負の数からの減算

	99	（負の数：−1）
＋	98	（負の数：−2）
	97	（負の数：−3）

↑3 桁目への桁上りは切捨て

注）正の数どうしを加算して結果の 2 桁目が 5～9（負の数）になるとオーバーフロー(overflow）となる．同様に，負の数どうしを加算して結果の 2 桁目が 0～4（ゼロまたは正の数）になるとオーバーフローとなる．**（オーバーフローとは数表現の限界を超えること）**

0.7.3　2 進数の補数形式

・ **2 の補数形式：　8 ビットの 2 の補数形式**について説明する．この形式の場合，符号付き絶対値形式と同様に，最上位ビット（MSB）が符号ビット（‘0’がゼロおよび正(+)，‘1’が負(−)），それ以下の 7 ビットがデータビットとなる．表現できる範囲は$(-127)_{10}$～$(127)_{10}$．ゼロおよび正の数値$(0)_{10}$～$(127)_{10}$には$(0000\ 0000)_2$～$(0111\ 1111)_2$を対応させる．負の数値$(-1)_{10}$～$(-127)_{10}$には 8 ビットを超える最初の数である$(1\ 0000\ 0000)_2$（上界値）からゼロを除いた絶対値$(0000\ 0001)_2$～$(0111\ 1111)_2$を減算して得られる

$(1111\ 1111)_2$～$(1000\ 0001)_2$を対応させる．

ゼロおよび正の値	負の値
$(0000\ 0000)_2 = (0)_{10}$	
$(0000\ 0001)_2 = (1)_{10}$	$(1111\ 1111)_2 = (-1)_{10}$
$(0000\ 0010)_2 = (2)_{10}$	$(1111\ 1110)_2 = (-2)_{10}$
：	：
$(0111\ 1111)_2 = (127)_{10}$	$(1000\ 0001)_2 = (-127)_{10}$

すなわち，2 の補数とは，使用する 8 ビットの数の領域を 2 分し，下位側をゼロおよび正の数，上位側を負の数として使用するものである．ただし，負の数が示す値の大小関係は対応する絶対値と逆になる．

注１） 負側の最大値 $(1000\ 0000)_2$ **$=(-128)_{10}$ は絶対値（正数）に変換できない**ため，ここでは表現の範囲外とする．ただし，符号変換等で使用することもある．

注２） 2 の補数形式では負の数のみならず正の数も MSB は符号ビットである点に注意．

注３） ビット幅が 8 の場合，9 ビットの上界値$(1\ 0000\ 0000)_2$から 8 ビット絶対値を減算する操作は，**8 ビットのオールゼロ$(0000\ 0000)_2$から 9 ビット目に借り（ボロー: borrow）1 を仮定して減算する操作と等価**となる．ビット幅が 16 の場合，32 の場合なども同様．

・ 上記の 2 の補数は対応する 2 進数絶対値を上界値$(1\ 0000\ 0000)_2$から減算する代わりに次の規則で簡単に変換できる．コンピュータの ALU（算術論理演算ユニット）ではこの操作を 2 の補数変換回路で行う．

「変換したい絶対値の最下位ビット(LSB)からみて最初の‘1’まではそのまま，次のビットからは最上位ビット(MSB)までは‘0’を‘1’に‘1’を‘0’に反転する」

	絶対値		負の数
例１）	$(0000\ 0001)_2 = (1)_{10}$	→	$(1111\ 1111)_2 = (-1)_{10}$
例２）	$(0000\ 0010)_2 = (2)_{10}$	→	$(1111\ 1110)_2 = (-2)_{10}$
例３）	$(0111\ 1111)_2 = (127)_{10}$	→	$(1000\ 0001)_2 = (-127)_{10}$

・ 2 の補数変換は,「変換したい絶対値のすべてのビットについて‘0’を‘1’に‘1’を‘0’に反転し（これを 1 の補数という），最下位ビット(LSB)に‘1’を加算する」方法でも行える．ただし，これには加算が必要なため，コンピュータの ALU では使われていない．

・補数表現した負の数を再度補数変換すると元の正の数（絶対値）に戻る．

例１）	$(1111\ 1111)_2 = (-1)_{10}$	→	$(0000\ 0001)_2 = (1)_{10}$
例２）	$(1111\ 1110)_2 = (-2)_{10}$	→	$(0000\ 0010)_2 = (2)_{10}$

例３）　$(1000\ 0001)_2 = (-127)_{10} \rightarrow (0111\ 1111)_2 = (127)_{10}$

注）補数とは負数を表す言葉ではなく，正数の補数，負数の補数がある点に注意．

・　補数加算による減算の例（8 ビット 2 の補数形式の場合）

例１）正の数からの減算

$(0001\ 1110)_2 = (30)_{10}$

$+\ (1111\ 1111)_2 = (-1)_{10}$

$(0001\ 1101)_2 = (29)_{10}$

↑9 ビット目への桁上りは切捨て

例２）負の数からの減算

$(1111\ 1111)_2 = (-1)_{10}$

$+\ (1111\ 1110)_2 = (-2)_{10}$

$(1111\ 1101)_2 = (-3)_{10}$

↑9 ビット目への桁上りは切捨て

注）2 の補数形式の加減算では MSB の符号ビットも演算対象となる．正の数どうしを加算して結果の MSB が‘1’（負の数を示す）になるとオーバーフロー(overflow)となる．同様に，負の数どうしを加算して結果の MSB が‘0’（ゼロまたは正の数を示す）になるとオーバーフローとなる．（オーバーフローとは数表現の限界を超えること）

・　**16 ビット 2 の補数形式**：　MSB が符号ビット，それ以下の 15 ビットがデータビットとなる．表現できる範囲は$(-32{,}767)_{10}$～$(32{,}767)_{10}$.

ゼロおよび正の値	負の値
$(0000\ 0000\ 0000\ 0000)_2 = (0)_{10}$	
$(0000\ 0000\ 0000\ 0001)_2 = (1)_{10}$	$(1111\ 1111\ 1111\ 1111)_2 = (-1)_{10}$
$(0000\ 0000\ 0000\ 0010)_2 = (2)_{10}$	$(1111\ 1111\ 1111\ 1110)_2 = (-2)_{10}$
:	:
$(0111\ 1111\ 1111\ 1111)_2 = (32{,}767)_{10}$	$(1000\ 0000\ 0000\ 0001)_2 = (-32{,}767)_{10}$

注）　$(1000\ 0000\ 0000\ 0000)_2 = (-32{,}768)_{10}$ は絶対値（正数）に変換できないため，ここでは表現の範囲外とする．ただし，符号変換等で使用することもある．

・n ビットの 2 の補数形式では，表現できる数値の種類は式(12)，最大値は式(13)で与えられる．注）式(12)と(13)は，n ビット符号付き絶対値形式の式(10)と(11)と同じ．

$$2^n - 1 \qquad (12)$$

$$\pm(2^{n-1} - 1) \qquad (13)$$

0.8　2 進数の 16 進表記

（この節の内容は教科書前編第 2 章・第 5 章にも記述してある）

・　2 進数のデータ形式は，一般に 8，16，32 など 4 の倍数のビット長（ビット幅ともいう）からなるため，4 ビット単位で 16 進数に変換して表記する．16 進数では$(0)_{10}$～$(9)_{10}$

は 10 進数と同じ $(0)_{16}$～$(9)_{16}$を用いるが，$(10)_{10}$～$(15)_{10}$はアルファベットを用いて $(A)_{16}$～$(F)_{16}$で表記する．

$(0000)_2$	$=(0)_{16}$ $=(0)_{10}$	$(1000)_2$	$=(8)_{16}$ $=(8)_{10}$
$(0001)_2$	$=(1)_{16}$ $=(1)_{10}$	$(1001)_2$	$=(9)_{16}$ $=(9)_{10}$
$(0010)_2$	$=(2)_{16}$ $=(2)_{10}$	$(1010)_2$	$=(A)_{16}$ $=(10)_{10}$
$(0011)_2$	$=(3)_{16}$ $=(3)_{10}$	$(1011)_2$	$=(B)_{16}$ $=(11)_{10}$
$(0100)_2$	$=(4)_{16}$ $=(4)_{10}$	$(1100)_2$	$=(C)_{16}$ $=(12)_{10}$
$(0101)_2$	$=(5)_{16}$ $=(5)_{10}$	$(1101)_2$	$=(D)_{16}$ $=(13)_{10}$
$(0110)_2$	$=(6)_{16}$ $=(6)_{10}$	$(1110)_2$	$=(E)_{16}$ $=(14)_{10}$
$(0111)_2$	$=(7)_{16}$ $=(7)_{10}$	$(1111)_2$	$=(F)_{16}$ $=(15)_{10}$

- **16 ビット 2 の補数形式の 16 進表記（16 の補数形式）**

  16 ビットの 2 進数を 4 桁の 16 進数で表す．16 進表記の場合，$(\ )_{16}$と書く代わりにデータの右に h（hexadecimal の頭文字）をつける記述法を用いるのが一般的である．2 の補数形式を 16 進表記すると **16 の補数形式**となる（16 ビットでは 4 桁）．

  ゼロおよび正の値

  $(0000\ 0000\ 0000\ 0000)_2 = (0)_{10} = 0000h$
  $(0000\ 0000\ 0000\ 0001)_2 = (1)_{10} = 0001h$
  $(0000\ 0000\ 0000\ 0010)_2 = (2)_{10} = 0002h$
  :
  $(0111\ 1111\ 1111\ 1111)_2 = (32{,}767)_{10} = 7FFFh$

  負の値

  $(1111\ 1111\ 1111\ 1111)_2 = (-1)_{10} = FFFFh$
  $(1111\ 1111\ 1111\ 1110)_2 = (-2)_{10} = FFFEh$
  :
  $(1000\ 0000\ 0000\ 0001)_2 = (-32{,}767)_{10} = 8001h$

  注１）16 進数 1 桁（4 ビット）を 1 ニブル(nibble)という．最上位桁を **MSN**（most significant nibble），最下位桁を **LSN** (least significant nibble)という．**16 の補数形式では，正の数の MSN は 0～7，負の数の MSN は F～8となる．**

  注２）32 ビット 2 の補数形式では 16 進 8 桁（8 桁の 16 の補数形式）となる．

  注３）2 進数符号なし絶対値形式，2 進数符号付き絶対値形式の場合も同様に 16 進表記する．

  注４） 特に形式を指定しない通常表記の 16 進数の場合，上位の 0 を省略して Ch,

3Dh，1E5h のように書くこともある．

・16 進表記での加減算

ここでは 16 ビット 2 の補数形式（4 桁の 16 の補数形式)とする．16 進表記の h は省略．

加算の例)　　1AD5　= (6,869)$_{10}$

＋　35 EC　= (13,804) $_{10}$

50C1　= (20,673) $_{10}$

各桁(ニブル)の加算結果が F (=(15) $_{10}$)を超えたら上位桁に 1 桁上がりする．

減算の例)　　4BDE　= (19,422)$_{10}$

－　35 EC　= (13,804) $_{10}$

15 F2　= (5,618) $_{10}$

各桁(ニブル)で被減数が減数より小さい場合は上位桁から 1（当該桁では(16) $_{10}$）を借りる．

・16 進表記での補数変換（16 の補数変換）

ここでは 16 ビット 2 の補数形式（4 桁の 16 の補数形式)とする．次の規則で変換する．

「LSN から見て最初のゼロ以外のニブルはそのニブルの値を(16) $_{10}$から引き，それ以上のニブルは(15) $_{10}$から引く」

例)　30A0h　= (12,448) $_{10}$

↓

CF60h　= (－12,448) $_{10}$

注) 再度補数変換すると元の数値に戻る．

・16 進表記でのデータ形式変換

16 ビット 2 進数符号付き絶対値形式 16 進表記から同 2 の補数形式 16 進表記（4 桁の 16 の補数形式）への変換を考える．正数は同一の数表現となるため変換する必要はない．負数は次のように行う．

例）負の符号付き絶対値形式　　CF60h　= (1100 1111 0110 0000) $_2$

16 の補数変換を行う　　↓

正の 16 の補数形式になる　　30A0h　= (0011 0000 1010 0000) $_2$

負にするため MSN に(8) $_{16}$を

を加える(MSB を'1'にする)　　↓

負の 16 の補数形式　　B0A0h　= (1011 0000 1010 0000) $_2$

(負の 16 ビット 2 の補数形式)

注１）符号付き絶対値形式の符号を正にしてから 16 の補数に変換してもよい．

注２）2 の補数形式から符号付き絶対値形式への変換も上記と同じ手続きで行える．

0.9　2 進数データ形式のまとめ

- **n ビット 2 進数符号なし絶対値形式**：n ビット（通常は 4 の倍数）で 2 進数の絶対値（ゼロおよび正整数のみ，負数は表現できない）を表す．n ビットに満たない小さい数値では上位ビットに 0 をおく．
- **n ビット 2 進数符号付き絶対値形式**：上記の n ビット 2 進数絶対値形式の最上位ビット（MSB）を符号ビットとし，0 は正数，1 は負数を表すようにしたもの．
  データ部は n　— 1 ビットになる．
- **n ビット 2 進数 2 の補数形式**：上記の n ビット 2 進数絶対値形式において，正数はそのまま，負数は MSB の符号を 0 にしてから 2 の補数に変換したもの．MSB は符号ビットとなり，0 は正数，1 は負数を表す．**データ部は n — 1 ビットになる．この形式では符号ビットを含めて加減算できる．**
- **n ビット固定小数点 2 進数符号なし絶対値形式**：整数部を $n-n_f$ ビット，小数部を n_f ビットとしてゼロを含む正の数値を表すデータ形式．n と n_fはあらかじめ決めておく．すなわちこの形式は，**n ビット 2 進数符号なし絶対値形式で小数部のビット数を定義したもの．**負数は表現できない．
- **n ビット固定小数点 2 進数符号付き絶対値形式**：上記の n ビット固定小数点 2 進数符号なし絶対値形式において，MSB を符号ビット（0 は正数，1 は負数を表す），整数部を $n-n_f-1$ ビット，小数部を n_f ビットとしてゼロを含む正および負の数値を表すデータ形式．データ部は n — 1 ビットになる．すなわちこの形式は，**n ビット 2 進数符号付き絶対値形式で小数部のビット数を定義したもの．**
- **n ビット固定小数点 2 進数 2 の補数形式**：上記の n ビット固定小数点 2 進数符号付き絶対値形式において，正数はそのまま，負数は MSB の符号を 0 にしてから小数部を含めて 2 の補数に変換したもの．MSB は符号ビットとなり，0 は正数，1 は負数を表す．データ部は n— 1 ビットになる．この形式では符号ビットを含めて加減算できる．すなわちこの形式は，**n ビット 2 進数 2 の補数形式で小数部のビット数を定義したもの．**

注 1）上記のいずれの形式も，符号ビットがある場合はそれを含めて，16 進表記できる．

注 2）**コンピュータで直接扱うデータの表現方法（内部表現ともいう）を「データ形式」という．**コンピュータで直接扱わないデータは，2 進数，10 進数，16 進数にかかわらず単に**「通常表記」**または**「外部表現」**という．この場合，負数は数値の左に －記号を書き，**上位桁の 0 は省略する**．（　）$_2$のように括弧で基数を指定する場合，**－記号は括弧の内側でも外側でもよい．**

注 3）固定小数点形式は，小数部が 0 ビットの場合でも，後述する浮動小数点形式に対応してこの名称を使う場合がある．

問題 0.8　4 ビット 2 の補数形式で表現できるすべての数値を 10 進数に対応させて示せ．また，各 4 ビット 2 の補数形式の値を 16 進数 1 桁で表現（16 進表記）せよ．

問題 0.9　次の 10 進負数を 8 ビット 2 の補数形式に変換せよ．また，それぞれを 16 進表記せよ．

(1)　$(-25)_{10}$　　(2)　$(-111)_{10}$

問題 0.10　次の 8 ビット 2 の補数形式を 10 進負数に変換せよ．

(1)　$(1011\ 0010)_2$　　(2)　$(1101\ 1011)_2$

問題 0.11　次の 10 進数演算を各数値を 8 ビット 2 の補数形式に変換して行え．なお，答えは 2 進数絶対値表現（－記号を用いる表現）と 10 進数に変換して検算せよ．

(1)　$(51)_{10} + (19)_{10}$　　(2)　$(-51)_{10} + (19)_{10}$

(3)　$(51)_{10} + (-19)_{10}$　　(4)　$(-51)_{10} + (-19)_{10}$

問題 0.12 次の 10 進正数を 2 進数 8 ビット符号なし絶対値形式に変換せよ．また，それぞれを 16 進表記せよ．

(1)　$(25)_{10}$　　(2)　$(101)_{10}$

問題 0.13 次の 10 進負数を 2 進数 8 ビット符号付き絶対値形式に変換せよ．また，それぞれを 16 進表記せよ．

(1)　$(-25)_{10}$　　(2)　$(-111)_{10}$

問題 0.14 次の 10 進負数を 2 進数 8 ビット 2 の補数形式に変換せよ．また，それぞれを 16 進表記せよ．

(1)　$(-25)_{10}$　　(2)　$(-111)_{10}$

問題 0.15 次の 16 進負数を 32 ビット符号付き絶対値形式 16 進表記と 32 ビット 2 の補数形式 16 進表記に変換せよ．

(1)　$(-1\ 11)_{16}$　　(2)　$(-78\ 9A\ BC\ DE)_{16}$

問題 0.16 次の 32 ビット 2 の補数形式 16 進表記を 32 ビット符号付き絶対値形式 16 進表記と 16 進数絶対値表現（－記号を用いる）に変換せよ．

(1)　$(FE\ DC\ BA\ 98)_{16}$　　(2)　$(89\ AB\ CD\ EF)_{16}$

問題 0.17 次の 32 ビット 2 の補数形式 16 進表記による加減算を行い，結果を同じ 2 の補数形式 16 進表記で示せ．また，その結果を 32 ビット符号付き絶対値形式 16 進表記と 16 進数絶対値表現（－記号を用いる）に変換せよ．

(1) $(\mathrm{FE\ DC\ BA\ 98})_{16} + (\mathrm{89\ AB\ CD\ EF})_{16}$

(2) $(\mathrm{FE\ DC\ BA\ 98})_{16} - (\mathrm{89\ AB\ CD\ EF})_{16}$

(3) $(\mathrm{89\ AB\ CD\ EF})_{16} - (\mathrm{FE\ DC\ BA\ 98})_{16}$

0.10　10 進数の扱い

・ コンピュータの内部ではすべての数値は 2 進数で表現される．16 進表記をしてもその元の値は 2 進数である．レジスタやメモリに 10 進数を置く場合は，その各桁の値(0～9)を 4 ビット 2 進数で表現する．これを **BCD 形式** (binary-coded decimal: 2 進化 10 進形式)という．この形式は 16 進数 0h～9h に対応する．

10 進数 1 桁の BCD 形式

$(0)_{10} = (0000)_2 \quad = 0\mathrm{h}$

$(1)_{10} = (0001)_2 \quad = 1\mathrm{h}$

:

$(9)_{10} = (1001)_2 \quad = 9\mathrm{h}$

・ **16 ビット 10 の補数形式**

16 ビットすなわち 4 ニブルに 4 桁（10 進数は桁のことを**ディジット(digit)**という）の 10 の補数形式の 10 進数値を記入する形式．コメント欄にこの形式の数値を書く場合は LSN の右に **d (decimal の頭文字)**を書く．

例)　ゼロおよび正の整数　　負の整数

$(0000)_{10} = 0000\mathrm{d}$

$(0001)_{10} = 0001\mathrm{d}$　　$(-0001)_{10} = 9999\mathrm{d}$

$(0002)_{10} = 0002\mathrm{d}$　　$(-0002)_{10} = 9998\mathrm{d}$

:　　:

$(4999)_{10} = 4999\mathrm{d}$　　$(-4999)_{10} = 5001\mathrm{d}$

注１） この形式の数値は 2 進数用の加減算回路では直接演算できないため，2 進数に変換して演算するか 10 進演算用のプログラムで演算する．

注２） C プログラムなどでは h や d を付けていない数値はデータ形式を指定しない 10 進数として扱われる．d を付けた数値でも，通常表記や絶対値形式の場合もあるため，10 の補数形式の場合はそのことを明示しておく必要がある．

注３） 数列 101 が 2 進数であることを示す場合に LSB の右に b (binary の頭文字)をつけて 101b と書くこともある．

0.11　浮動小数点形式（参考）

先に述べた固定小数点 2 進数 2 の補数形式では小数点位置が固定であるため，桁数が極めて大きい整数や極めて小さい小数を扱うことができない．このような数値を扱えるようにしたデータ形式が浮動小数点形式（floating point representation）である．C プログラムでは，小数点をもつ 10 進数を書いた場合（実数という）はこの形式で扱われる．

・ 浮動小数点形式の考え方を 10 進実数で説明する．

$$(100.5)_{10} = (0.1005 \times 10^3)_{10} \qquad (14)$$

式(14)の左辺は整数部 3 桁，小数部 1 桁からなる 10 進実数である．この値は右辺のように，小数部のみからなるように右へ桁ずらしを行い，その値にずらした桁数の 10 のベキ乗を掛けた値と等価となる．右辺の 0.1005 を**仮数部**，10^3を**指数部**という．このように，小数点位置を意味のある最上位桁（ここでは 1 の桁）の左まで移動させることを浮動小数点という．ただし，実用的な形式では次のように 2 進数の 16 進表記で表現する．

・ **32 ビット IBM 浮動小数点形式**

$(100.5)_{10}$の整数部と小数部をそれぞれ 2 進数に変換し，それを 16 進表記すると式(15)の右辺となる

$$(100.5)_{10} = (64.8)_{16} \qquad (15)$$

式(15)の右辺は 16 進表記であるため，桁ずらしを 16 のベキ乗で行うと式(16)の右辺となる．ここで仮数部の整数部 1 桁目の 0 は省略している．

$$(64.8)_{16} = (.648)_{16} \times (16^2)_{10} \qquad (16)$$

実際の 2 進数 32 ビットのデータ形式は以下となる．

(1 ビット符号)(7 ビット指数部)(24 ビット仮数部)

- 1 ビット符号は正(+)なら$(0)_2$，負(−)なら$(1)_2$とする．
- 7 ビット指数部にはずらした桁数$(2)_{10}$に$(64)_{10}$を加えた値を 2 進数で書く．$(64)_{10}$を加えるのは逆方向へ桁ずらしをする際に負の値にならないようにするためである．これを**バイアス値**という．上の例では，1 ビット符号と 7 ビット指数部は
  $(0\ 100\ 0010)_2$ となる．16 進表記では 42h となる．
- 24 ビット仮数部には小数点以下の仮数を左づめで書く．
- 全体を合成すると以下となる．

  $(100.5)_{10}$ = 42 64 80 00 h　(1 バイト単位でスペースを入れてある)

その他の例)

$(-100.5)_{10} = (-64.8)_{16}$ =C2 64 80 00 h

$(10.5)_{10} = (A.8)_{16}$ = 41 A8 00 00 h

$(1.5)_{10} = (1.8)_{16}$ = 41 18 00 00 h

$(0.5)_{10} = (.8)_{16}$ = 40 80 00 00 h

注１）負数の場合は仮数部には補数を用いず，正数と同様に絶対値で表記する．

注２）IBM 形式以外に IEEE 形式があるが，やや複雑なため説明を省略する．

・ **32 ビット IBM 浮動小数点形式での演算**

正数データどうしの加算： ２つのデータの指数部の桁数が同じ場合は仮数部をそのまま 16 進数で加算し，整数部へ１桁上りした場合は，加算結果の指数部を＋１して仮数部を右へ１桁桁ずらしを行う．16 進 6 桁(24 ビット)からアンダーフローした桁は切り捨てる． ２つのデータの指数部の桁数が異なる場合は，指数部の値が大きい方の仮数部に合わせて指数部の値が小さい方の仮数部を右へずらす．桁上りやアンダーフローの処理は上記と同様.

負数データの加算（減算）： 正数データの加算と同様に仮数部の桁ずらしを行った後，正数データ，負数データともに仮数部に整数部 1 桁（0h）を付加する，負数データは全体を 2 の補数に変換する．整数部 1 桁を付加したデータどうしを加算する．その際，整数部 2 桁目への桁上がりは切り捨てる．加算結果の整数部 1 桁目が Eh または Fh の場合は負数であるため，整数部 1 桁目を含めて絶対値に戻す．整数部 1 桁目へ桁上りがある場合は整数部が 0 になるように仮数部を右へずらす．整数部 1 桁目から仮数部上位桁が 0h の場合は仮数部の最上位桁が 0h 以外になるまで仮数部を右へずらす．ずらした桁数を指数部に加算または減算する．結果が負数の場合は符号ビットに‘1’を立てる.

乗算： 16 進 6 桁分の積が得られるまで仮数部のシフト加算を繰り返す．被乗数と乗数の指数部から積の指数部を決める．積の符号ビットは被乗数と乗数の符号ビットから決める.

除算： 16 進 6 桁分の商が得られるまで仮数部のシフト減算を繰り返す．被除数と除数の指数部から商の指数部を決める．商の符号ビットは被乗数と乗数の符号ビットから決める．浮動小数点形式では剰余は存在しない.

問題 0.18　次の 10 進数値を 32 ビット IBM 浮動小数点形式 16 進表記に変換せよ.

(1)　1000　　(2)　－1000　　(3)　30.75　　(4) 0.03125

問題 0.19　次の 32 ビット固定小数点 2 の補数形式（小数部なし）16 進表記を 32 ビット IBM 浮動小数点形式 16 進表記に変換せよ.

(1)　$(\text{AB CD EF 00})_{16}$　　(2)　$(\text{7F FF 00 00})_{16}$

問題 0.20　次の 10 進数加減算を 32 ビット IBM 浮動小数点形式 16 進表記で行え.

(1) 1000 ＋ 10　　(2) 1000 － 10　　(3) 10 － 1000

0.12　桁数が大きい 10 進数のデータ形式（参考）

・ **符号付きパック 10 進形式**

n バイトワードの最下位ニブル（LSN）に符号として正数(+)の場合は $(C)_{16}$，負数(－)の場合は $(D)_{16}$を書き，その上位ニブルに 10 進数値を BCD で下位から順に並べる．

例）4 バイトパック 10 進形式

$(12\ 3456)_{10}$ = 01 23 45 6C h（LSN の C は正の符号）

$-(12\ 3456)_{10}$ = 01 23 45 6D h（LSN の D は負の符号）

加算は，符号ニブルを 0 にし，負数は 10 の補数に変換してから 10 進加算する．減算は 10 の補数加算で行う．結果の最上位ニブル（MSN）が 0～4 の場合は正数と判定し LSN に C を立てる．MSN が 5～9 の場合は負数と判定し，絶対値に戻して LSN に D を立てる．オーバーフロー判定は，正数どうしの加算で結果が負数になる場合，負数どうしの加算で結果が正数になる場合を検出する．また，負数を補数変換しても結果が負数になる場合も検出する．

注）符号ニブルに正なら 3，負なら 7 を用いる形式もある．

・ **符号なしパック 10 進形式**

ゼロまたは正数のみを扱う形式．上記の符号付きパック 10 進形式において，LSN に符号を置かずに数値を LSN から順に並べる．

・ **ゾーン 10 進形式**

n バイトワードの各バイトの下位ニブルに 10 進数値を BCD で下位から順に並べる．最下位バイトの上位ニブルに符号として正数(+)の場合は$(C)_{16}$，負数(－)の場合は$(D)_{16}$を書く．最下位以外のバイトの上位ニブル（ゾーンニブル）には$(F)_{16}$を書く．

例）8 バイトゾーン 10 進形式

$(12\ 3456)_{10}$ = F0 F0 F1 F2 F3 F4 F5 C6 h　　（LSN の次の C は正の符号）

$-(12\ 3456)_{10}$ = F0 F0 F1 F2 F3 F4 F5 D6 h　　（LSN の次の D は負の符号）

注１）この形式は演算に使用しない中間表現として用いる．

注２）符号ニブルに正なら 3，負なら 7 を用いる形式もある．また，ゾーンニブルに 3 を用いる形式もある．

問題 0.21　次の 10 進数値を 6 バイト符号つきパック 10 進形式に変換せよ．

(1)　$(2013\ 0910)_{10}$　　(2)　$-(1020\ 2015)_{10}$

問題0.22　問題0.21の10進数値を8バイトゾーン10進形式に変換せよ．

以下，第１章～第５章は前編，第６章・第７章は後編に記載.

第０章著者：　津田 伸生 教授