第二章 情報の２進表現

- r 進法
- ２進数と１０進数の相互変換
- ８進数と１６進数
- ＢＣＤ符号
- 負数の表現と加減算
  - 符号ー絶対値表現
  - 補数表現（２の補数）

[r **進法**]

10 進法 - 人間が通常使用している方法
10 個ごとに桁が一つ上る
60 進法 - 時間 (1 分 =60 秒, 1 時間 =60 分)
2 進法 - 計算機, ディジタル回路で使用

[**基数**]

- r 進法における r のことを基数と呼ぶ
- r 進法での数 $a_{n-1}\cdots a_1a_0\,.\,a_{-1}a_{-2}\ldots a_{-m}$ は以下の意味を持つ。

$$a_{n-1}\cdots a_1a_0\,.\,a_{-1}a_{-2}\ldots a_{-m\,(r)} \;=\; \sum_{i=-m}^{n-1} a_i r^i$$
$$(0 \le a_i < r)$$

[**ビットとバイト**]
ビット: ２進数の１桁
バイト: ２進数の８桁 (１バイト = ８ビット)
[**例**]

$101_{(2)} = 5_{(10)}$ $0.1_{(2)} = 0.5_{(10)}$
$11.01_{(2)} = 3.25_{(10)}$
$12_{(8)} = 10_{(10)} = 1010_{(2)}$
$101_{(2)}$ は３ビットの２進数
$1011010100110101_{(2)}$ は２バイトの２進数

２進数と１０進数の相互変換

[**１０進整数 $\Longrightarrow$ ２進数**]

- ２で割った余りを次々に計算

[**１０進小数 $\Longrightarrow$ ２進数**]

- ２倍して小数点を越えた部分を次々に計算
- 必ずしも有限項で終わらない場合がある
  $0.35_{(10)} = 0.010110011001100110\cdots$

[**２進数 $\Longrightarrow$ １０進数**]

- 定義式に従って計算
  $10110_{(2)} = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$
- 効率的な計算法 (計算機向き)
  $10110_{(2)} = (((1\times2 + 0)\times2 + 1)\times2 + 1)\times2 + 0$

８進数と１６進数

[**必要性**]
２進数 - 人間にわかりずらい
$\Longrightarrow$ ８進数、１６進数ならばわかり易い

[**８進数**] ３ビットを１桁 (０～７までの数) で表す

$111\,011\,100\,110_{(2)} = 7\,3\,4\,6_{\,(8)}$
$100111_{(2)} = 47_{(8)}$ $10101_{(2)} = 25_{(8)}$

[**１６進数**] ４ビットを１桁 (0 ～ 9,A ～ F) で表す

$1010_{(2)} = \mathrm{A}_{(16)}$ $1011_{(2)} = \mathrm{B}_{(16)}$
$1100_{(2)} = \mathrm{C}_{(16)}$
$1101_{(2)} = \mathrm{D}_{(16)}$ $1110_{(2)} = \mathrm{E}_{(16)}$
$1111_{(2)} = \mathrm{F}_{(16)}$

$0011\,1010\,1111\,0101_{(2)} = 3\,\mathrm{A}\,\mathrm{F}\,5_{\,(16)}$
$10111001001_{(2)} = 5\mathrm{C}9_{(16)}$

2進化10進数 (BCD 符号)

[BCD 符号]

- 計算機内部で10進数を表現するのに必要
- 4ビットを用いて10進数の1桁を表現

[**例**] (8-4-2-1 符号)

$3071_{(10)} \ = \ 0011\,0000\,0111\,0001_{(BCD)}$

[**注意**] BCD 数と2進数は異なる

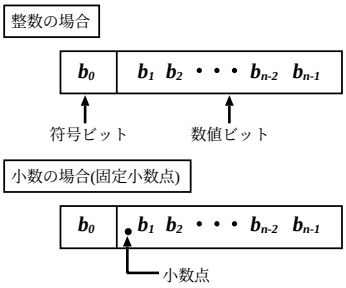
$39_{(10)} \ = \ 0011\,1001_{(BCD)} \ (\ = 10111_{(2)})$

負数の表し方と加減算

[**負数の表し方**]

- 符号と絶対値
- 2 の補数
- 1 の補数

[**符号と絶対値**]



[**例**]

$+5$	0	1	0	1	$+0.75$	0	1	1	0
$+1$	0	0	0	1	$+0.5$	0	1	0	0
$+0$	0	0	0	0	$+0.0$	0	0	0	0
-0	1	0	0	0	-0.0	1	0	0	0
-1	1	0	0	1	-0.5	1	1	0	0
-5	1	1	0	1	-0.75	1	1	1	0

[**2の補数**] (A^* を A の 2 補数とする)

- $A^* \ = \ 2^n - A$ (n ビットの場合)
- A のビットを反転 ($0 \leftrightarrow 1$) し、1 を足すことにより A^* を作成可
- MSB(最上位ビット) が、0 ならば正 (か 0) の数で、1 ならば負の数
  (LSB: 最下位ビット)

[**例**]

$+5$	0	1	0	1	-5	1	0	1	1
$+3$	0	0	1	1	-3	1	1	0	1
$+2$	0	0	1	0	-2	1	1	1	0
$+1$	0	0	0	1	-1	1	1	1	1
$+0$	0	0	0	0	-0	0	0	0	0

[**2の補数による加減算**]

$M - S$ は、2 の補数を用いて、$M + S^*$ で計算

$$\begin{aligned} M - S &= M + (2^n - S) - 2^n \\ &= M + S^* - 2^n \end{aligned}$$

[**桁あふれのおこる条件**]

符号ビット a_0	符号ビット b_0	あふれの条件
0	0	MSB への桁上げのあった時
1	1	MSB への桁上げのない時
0	1	あふれは起きない
1	0	あふれは起きない

同符号の加算(あふれ無し)

+8	0 0 1 0 0 0	-8	1 1 1 0 0 0
+(+3)	0 0 0 0 1 1	+(-3)	1 1 1 1 0 1
+11	0 0 1 0 1 1	-11	1 1 1 0 1 0 1

無視される

同符号の加算(あふれ有り)

+22	0 1 0 1 1 0	-22	1 0 1 0 1 0
+(+27)	0 1 1 0 1 1	+(-27)	1 0 0 1 0 1
+49	1 1 0 0 0 1	-49	1 0 0 1 1 1 1

桁あふれ　桁あふれ

異符号の加算

+8	0 0 1 0 0 0	+3	0 0 0 0 1 1
+(-3)	1 1 1 1 0 1	+(-8)	1 1 1 0 0 0
+5	1 0 0 0 1 0 1	-5	1 1 1 0 1 1

無視される

減算

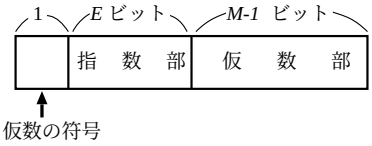
+8	0 0 1 0 0 0		0 0 1 0 0 0
-(-3)	- 1 1 1 1 0 1	補数化 →	0 0 0 0 1 1
+11			0 0 1 0 1 1

浮動小数点数

[**実数の計算機上での表現**]

- すべての実数を正確に表現することはできない
- 固定小数点方式
  処理が簡単、誤差の評価が簡単、表現範囲がせまい
- 浮動小数点方式
  表現範囲が広い、処理・誤差評価が面倒

浮動小数点表現　$r = m \times b^e$



仮数の符号

[**データ形式の例**](IBM370 系)

$b = 16,\ 1/16 \leq |m| < 1,\ E = 7,\ M = 24, 56, 120$

[**正規化**]

演算の結果、データ形式に会わない場合が出てくるので、
仮数部と指数部を調整する。

$e = 10, m = 0.036825_{(16)} \Longrightarrow e = 9, m = 0.36825_{(16)}$

[**その他**]

オーバーフロー、アンダーフローに注意

情報交換用符号

[**英数字符号**]

- 英字や数字を表現するための符号 (7 ビットか 8 ビット)
  (改行などの制御文字も含まれる)
- ISO, ASCII, EBCDIC など
- JIS 符号 - 8 ビット、カナ文字も含む

[**漢字符号**]

- 漢字を扱うには 1 バイト (8 ビット) では不足
- JIS 漢字符号
  2 バイト (16 ビット) を用いて漢字を表現
  英数字の時は 1 バイト (8 ビット) のみを用いる