n進数

情報数学入門
2025 年度後期
デジタル社会共創学環 只木進一

数字の表現

- ローマ数字: I, II, III, IV, V, VI,. . . , IX, X, XI,. . . , L,. . . , C, . . .
- 漢数字: 一, 二, 三, 四, 五, 六, 七, 八, 九, 十, 百, 千, 万,
  百二十五、二千二百五,. . .
- アラビア数字: 0, 1, 2, . . . , 9,10,11,12,. . . ,99,100,. . .
  125, 253, 0.23, 3.14, . . .
- アラビア数字は、"0"を使うことで桁を上手に表現できる
- "0"は、インドで「発見」され、アラビアを経由してヨーロッパに伝わった

10 進数の構造

- 0 から 9 までの 10 種類の記号を使う
- $9 + 1 = 10$ で桁が上がる
- "0"を使って、桁を表現する

$$125 = 1 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$$
$$203 = 2 \times 10^2 + 0 \times 10^1 + 3 \times 10^0$$
$$0.23 = 2 \times 10^{-1} + 3 \times 10^{-2}$$
$$3.04 = 3 \times 10^0 + 0 \times 10^{-1} + 4 \times 10^{-2}$$

n 進数

- 10 進数の構造を一般化したものが n 進数
- n 進数では、0 から $n-1$ までの n 種類の記号を使う
- $(n-1)+1$ で桁が上がる

コンピュータ内でのデータの取り扱い

- コンピュータ内では、全てが 2 進数
- 2 進数一桁をビット (bit) と呼ぶ
- 8 ビットを 1 バイト (Byte) と呼ぶ
- 1 バイトで表現できる数は、0 から 255 までの 256 通り
- 文字コード
  - ASCII コード: 7 ビットで英数字と記号を表現
  - 日本語: 2 バイト (16 ビット) で表現
    JIS, Shift-JIS, EUC-JP など
  - Unicode: 16 ビット以上で世界中の文字を表現

10 進数から 2 進数へ

- 2 のべき乗で表現する
- 情報技術では、先頭に 0b を付けることが多い

$$
\begin{aligned}
53 &= 32 + 16 + 4 + 1 \\
&= 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^0 \\
&= (00110101)_2 \\
130 &= 128 + 2 \\
&= 1 \times 2^7 + 1 \times 2^1 \\
&= (10000010)_2
\end{aligned}
$$

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

2進数への変換

- 2で除した商と余りを求める
- 商が0になるまで繰り返す
- 余りを下から順に並べると2進数になる

2)	53	
2)	26	1
2)	13	0
2)	6	1
2)	3	0
2)	1	1
	0	1

$$
\begin{aligned}
53 &= 2 \times 26 + 1 = 2 \times (2 \times 13) + 1 \\
&= 2^2 \times (2 \times 6 + 1) + 1 = 2^3 \times (2 \times 3) + 2^2 + 1 \\
&= 2^4 \times (2 + 1) + 2^2 + 1 = 2^5 + 2^4 + 2^2 + 1
\end{aligned}
$$

例 2.1: 30 を 2 進数に変換

なぜコンピュータは2進数を使うのか

- 素子の構造が単純
  - 状態はオンとオフの２つ
  - リレー、真空管、トランジスタ
- ノイズに強い
- 演算が簡単

a	b	$a+b$	$a \times b$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	10	1

例 3.1: 2進数加算乗算

$$\begin{array}{rr} & 101 \\ +) & 11 \\ \hline & 1000 \end{array} \qquad \begin{array}{rr} & 1011 \\ +) & 110 \\ \hline & 10001 \end{array}$$

$$\begin{array}{rr} & 101 \\ \times) & 11 \\ \hline & 101 \\ +) & 1010 \\ \hline & 1111 \end{array} \qquad \begin{array}{rr} & 101 \\ \times) & 101 \\ \hline & 101 \\ +) & 10100 \\ \hline & 11001 \end{array}$$

2進数の減算

引き算は、上の桁からの「借り」があり、足し算に比べて難しい。
コンピュータは、2進数でヒトと同じように引き算をしているか。

- コンピュータが扱うのは有限桁
- 8bit と考える: 扱えるのは 0 から 255 まで

8bit CPU

1970 年代
PC-8000 シリーズ、FM-7 シリーズなど

2 の補数: two's complement

- 正の整数 n に対する 2 の補数
  - n の二進表現で 0 と 1 を反転し、1 を加える
- $n = 5$ の場合

$$5 = (00000101)_2$$
$$\Rightarrow (11111010)_2 + (00000001)_2 = (11111011)_2$$

2 の補数を使った減算

- $9-5$ をそのまま実行

$$\begin{aligned} 9-5 &= (00001001)_2 - (00000101)_2 \\ &= (00000100)_2 = 4 \end{aligned}$$

- 9 に 5 に対する 2 の補数を足す

$$(00001001)_2 + (11111011)_2 = (100000100)_2$$

  - 二進表現が 9 桁になった。一番上の桁を無視し、8 桁部分として 4 を得る。

$$(00000100)_2 = 4$$

2 の補数を使った減算の仕組

- n に対する 2 の補数とは
  - 0 と 1 を反転させる

$$(11111111)_2 - n$$

  - 1 を足す

$$(11111111)_2 - n + 1 = (100000000)_2 - n$$

- m から n を引く代わりに、m に n に対する 2 の補数を足す

$$m + ((11111111)_2 - n + 1) = m - n + (100000000)_2$$

- 後で、$(100000000)_2$、つまり桁が溢れた部分を取り除けば良い

10 進での減算を見直し

$9-5$ の代わりに、9 に 5 に対する 2 の補数を足すことを 10 進で見てみる:8bit の場合

- 5 に対する 2 の補数

$$(256-1)-5+1$$

- 9+(5 に対する 2 の補数)

$$9+(256-1)-5+1=9-5+256$$

減算: $5 - 9$

- $9 = (00001001)_2$ に対する 2 の補数

$$(11110110)_2 + (00000001)_2 = (11110111)_2$$

- 5 に 9 に対する 2 の補数を足す

$$(00000101)_2 + (11110111)_2 = (11111100)_2$$

- これは $4 = (00000100)_2$ に対する 2 の補数

$$(11111011)_2 + (00000001)_2 = (11111100)_2$$

- 2 の補数は、対応する負の数を表している

例: $23 - 17$

- $23 = (00010111)_2$
- $17 = (00010001)_2$
- 17 に対する 2 の補数: $(11101111)_2$
- 23+(17 に対する 2 の補数)

$$(00010111)_2 + (11101111)_2 = (100000110)_2 = 6 + 256$$

負の数

- 8bit のうち、最上位を符号として扱う
- 例: $0 - 1 = (11111111)_2$
  - 1 の「2 の補数」に相当

$$
\begin{aligned}
-1 &= (11111111)_2 \\
-2 &= (11111110)_2 \\
-3 &= (11111101)_2 \\
&\cdots \\
-128 &= (10000000)_2
\end{aligned}
$$

- 最上位桁が 0 の数は、0 から 127 まで

例 4.1: $-13 = (11110011)_2$