２進数とその簡単な計算

情報ネットワーク工学入門
2024 年度後期
佐賀大学理工学部　只木進一

1. 二進数と十進数: binary and decimal numbers
2. 二進数演算: binary operations
3. 減算: binary subtraction
4. 除算と小数: binary divisions and floating numbers
5. 接頭辞: Prefixes
6. 10 進数、2 進数、8 進数、16 進数

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、すべて 2 進数 (binary numbers) 表現
- 2 進数 1 桁 $[0, 1]$ を bit と呼ぶ
- 2 進数 8 桁 $[0, 255]$ を Byte と呼ぶ
- 文字コード
  - ASCII コード: 7bit で数字やアルファベットを表現
  - 日本語コード: JIS、SJIS、EUC は 2 バイト
  - 多言語混在: UTF-8 など

二進数と十進数

10 進数	2 進数
0	0b0000
1	0b0001
2	0b0010
3	0b0011
4	0b0100
5	0b0101
6	0b0110
7	0b0111
8	0b1000

十進数と桁の意味

- 十進数 (decimal numbers) では $\{0,1,2,3,4,5,6,7,8,9\}$ の 10 個の記号を使用
- k 桁目は 10^{k-1} が何個あるかを表す

$$
\begin{aligned}
1634 &= 1 \times 10^3 + 6 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 \\
3021 &= 3 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 1 \times 10^0
\end{aligned}
$$

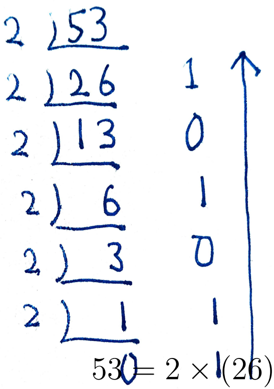
- $9 + 1 = 10$ という桁上がりの規則

十進数と二進数の相互変換

- 十進数から二進数へ
- 2 のべき乗の和で表す
- 二進数は、先頭に 0b を付けて表記

$$
\begin{aligned}
53 &= 32 + 16 + 4 + 1 = 2^5 + 2^4 + 2^2 + 2^0 \\
&= 0\mathrm{b}00110101 \\
130 &= 128 + 2 = 2^7 + 2^1 \\
&= 0\mathrm{b}10000010 \\
163 &= 128 + 32 + 2 + 1 = 2^7 + 2^5 + 2^2 + 2^0 \\
&= 0\mathrm{b}10100011
\end{aligned}
$$

10 進数から 2 進数へ



- 2 で割った商と余りを求める
- これを 0 になるまで繰り返す
- 余りを上から下に読む

$$53 = 0b00110101$$

$$\begin{aligned}53 &= 2 \times (26) + 1 = 2 \times (2 \times 13) + 1 \\ &= 2 \times 2 \times (2 \times 6 + 1) + 1 = 2^3 (2 \times 3) + 2^2 + 1 \\ &= 2^4 (2 + 1) + 2^2 + 1 = 2^5 + 2^4 + 2^2 + 2^0\end{aligned}$$

例 1.1: $73 = 0b01001001$

2^n はある程度覚えよう

$$
\begin{aligned}
2^0 &= 1, & 2^1 &= 2, \\
2^2 &= 4, & 2^3 &= 8, \\
2^4 &= 16, & 2^5 &= 32, \\
2^6 &= 64, & 2^7 &= 128, \\
2^8 &= 256, & 2^9 &= 512, \\
2^{10} &= 1024, & 2^{11} &= 2048.
\end{aligned}
$$

なぜ、コンピュータは2進数を使うのか

- 演算素子・規則素子の構造が単純
  - 状態はオン (on) とオフ (off) の二つ
  - リレー (relays)、真空管 (vacuum tubes)、トランジスタ (transistors)
  - 磁心記憶 (core memory)、トランジスタ
- 演算規則が簡素

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

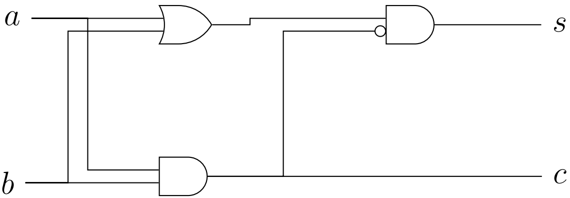
加算と乗算

$$\begin{array}{r} 101 \\ +)\quad 11 \\ \hline 1000 \end{array} \qquad \begin{array}{r} 101 \\ \times)\quad 11 \\ \hline 101 \\ +)\ 101 \\ \hline 1111 \end{array} \qquad \begin{array}{r} 101 \\ \times)\ 101 \\ \hline 101 \\ +)\ 101 \\ \hline 11001 \end{array}$$

$$\begin{array}{r} 1011 \\ +)\quad 110 \\ \hline 10001 \end{array}$$

半加算器: half adder

1 桁の加算をする論理回路



a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

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

$$
\begin{aligned}
5 &= 0\mathrm{b}00000101 \\
&\Rightarrow 0\mathrm{b}11111010 + 0\mathrm{b}00000001 \\
&= 0\mathrm{b}11111011
\end{aligned}
$$

2の補数を使った減算

- $9 - 5$ をそのまま実行

$$\begin{aligned} 9 - 5 &= 0\mathrm{b}00001001 - 0\mathrm{b}00000101 \\ &= 0\mathrm{b}00000100 = 4 \end{aligned}$$

- 9に5に対する2の補数を足す

$$0\mathrm{b}00001001 + 0\mathrm{b}11111011 = 0\mathrm{b}100000100$$

- 二進表現が9桁になった。一番上の桁を無視して4を得る。

$$0\mathrm{b}00000100 = 4$$

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

$$0\mathrm{b}11111111 - n$$

  - 1を足す

$$0\mathrm{b}11111111 - n + 1 = 0\mathrm{b}100000000 - n$$

- 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

$$m + (0\mathrm{b}11111111 - n + 1) = m - n + 0\mathrm{b}100000000$$

- 後で、$0\mathrm{b}100000000$、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: $5-9$

- $9 = 0\mathrm{b}00001001$ に対する 2 の補数

$$0\mathrm{b}11110110 + 0\mathrm{b}00000001 = 0\mathrm{b}11110111$$

- 5 に 9 に対する 2 の補数を足す

$$0\mathrm{b}00000101 + 0\mathrm{b}11110111 = 0\mathrm{b}11111100$$

- これは $4 = 0\mathrm{b}00000100$ に対する 2 の補数

$$0\mathrm{b}11111011 + 0\mathrm{b}00000001 = 0\mathrm{b}11111100$$

- 2 の補数は、対応する負の数を表している

例: $23 - 17$

- $23 = 0\mathrm{b}00010111$
- $17 = 0\mathrm{b}00010001$
- 17 に対する 2 の補数: $0\mathrm{b}11101111$
- 23+(17 に対する 2 の補数)

$$0\mathrm{b}00010111 + 0\mathrm{b}11101111 = 0\mathrm{b}100000110 = 6 + 256$$

負の数

- 8bit のうち、最上位を符号として扱う
- 例: $0 - 1 = 0\mathrm{b}11111111$
  - 1 の「2 の補数」に相当

$$
\begin{aligned}
-1 &= 0\mathrm{b}11111111 \\
-2 &= 0\mathrm{b}11111110 \\
-3 &= 0\mathrm{b}11111101 \\
&\cdots \\
-128 &= 0\mathrm{b}10000000
\end{aligned}
$$

```
x = 0b11011
y = (~ x) + 1  #~x は x のビット反転
```

例 3.1: $-13 = 0\mathrm{b}11110011$

除算

$$
\begin{array}{r}
101 \\
1011\,\overline{)\,01000001} \\
1011 \\
\hline
10101 \\
1011 \\
\hline
1010
\end{array}
$$

小数

- 小数の表現: ここでは $()_2$ と表現

$$
\begin{aligned}
(0.101)_2 &= 2^{-1} + 2^{-3} \\
&= \frac{1}{2} + \frac{1}{8} = \frac{5}{8} = 0.625
\end{aligned}
$$

- 浮動小数型

$$(0.101)_2 = 2^{-1} \times (1 + (0.01)_2)$$

小数の和

```
>>> x = 1/2
>>> y = 1/8
>>> z = x + y
>>> print(z)
0.625
>>> x = 0.1
>>> y = 0.2
>>> z = x + y
>>> print(z)
0.30000000000000004
```

小数を正しく扱うには、特別な処理が必要。例えば、二進化十進表現 (BCD, Binary-coded decimal)。

接頭辞: Prefixes

- 3 桁毎に名前を付ける
  - $1\mathrm{k} = 10^3$、$1\mathrm{M} = 10^3\mathrm{k}$、$1\mathrm{G} = 10^3\mathrm{M}$、$1\mathrm{T} = 10^3\mathrm{G}$、$1\mathrm{P} = 10^3\mathrm{T}$
  - $1\mathrm{m} = 10^{-3}$、$1\mu = 10^{-3}\mathrm{m}$、$1\mathrm{n} = 10^{-3}\mu$
- 2 進の場合には、1000 の代わりに $2^{10} = 1024$ を使う
- 2022 年 11 月開催の国際度量衡総会で新たな接頭語が追加された
  `https://unit.aist.go.jp/nmij/info/SI_prefixes/index.html`
- SI (Système International d'unités) 単位系
  `https://www.aist.go.jp/Portals/0/resource_images/aist_j/press_release/pr2004/pr20040120/si_all.pdf`

10 進数、2 進数、8 進数、16 進数

- n 進数: 使える記号が n 個
- 10 進数 (decimals): $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  $9 + 1 = 10$
- 2 進数 (binaries): $\{0, 1\}$
  $1 + 1 = 10$
- 8 進数 (octals): $\{0, 1, 2, 3, 4, 5, 6, 7\}$
  $7 + 1 = 10$
- 16 進数 (hexadecimals): $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\}$
  $\mathrm{F} + 1 = 10$

16 進数の利用: 文字コード

- 16 進 2 桁は 8bit: $0x00 \sim 0xFF$
- 16 進数は先頭に $0x$ を付けて表示
- ASCII コード: 英数文字を表現: 7bit $0x00 \sim 0x7F$
- 通常の日本語は 16 進 4 桁
- UNICODE `http://www.unicode.org/charts/`

16 進数の利用: インターネット

- インターネットのアドレス標記
- 8bit 毎 (octet) に区切って記述する
- ネットマスク
- MAC (Media Access Control) アドレス

Python

Python では、10 進数、2 進数、8 進数、16 進数を相互に変換できる。

```
x = 126
print(bin(x)) # 2進数として印刷
print(oct(x)) # 8進数として印刷
print(hex(x)) # 16進数として印刷
xb = 0b1001
xh = 0xf13e
xo = 0o32
```

Java

Java (jshell) でも試してみよう

```
jshell> x = 137;
x ==> 137
jshell> System.out.println(Integer.toBinaryString(x));
10001001
jshell> System.out.println(Integer.toOctalString(x));
211
jshell> System.out.println(Integer.toHexString(x));
89

jshell> x = -137;
x ==> -137
jshell> System.out.println(Integer.toBinaryString(x));
11111111111111111111111101110111
jshell> System.out.println(Integer.toOctalString(x));
37777777567
jshell> System.out.println(Integer.toHexString(x));
ffffff77
```

課題

UNICODE の最初の 7 ビット分は、ASCII 文字、つまり数字やローマ字等を表している。8 ビットまで拡張すると、全てではないが、ヨーロッパの英語以外の言語が使用しているアクセント付き文字を表現している。以下の URL を見て、確かめなさい。それぞれの PDF の 3 ページ以降には、それぞれの記号の説明がある。

- 0000 - 007F
  `www.unicode.org/charts/PDF/U0000.pdf`
- 0080 - 00FF
  `www.unicode.org/charts/PDF/U0080.pdf`