第二回 情報の表現

2002年9月12日

二進法と内部表現

- 現代の電子計算機の内部ではすべての情報が０か1かの2進法で表現される(2値表現、デジタル)
- なぜ2進法が使われるのか
  - 電子回路の設計の容易さ
  - 電子回路の動作上の信頼性
  - 論理演算との親和性
- 10進法と2進法

  10進法は10のn乗の集まりを使って、

  2進法は2のn乗の集まりを使って、

  それぞれ位表記する
- Excelにはなぜ表示形式があるのか
  - 内部表現をどのように解釈して人間に見せるかが異なる
    数値、文字列、日付、時刻など

十進法と二進法

十進法	位取記数法	二進法
0	0×2^0	0
1	1×2^0	1
2	$1\times2^1+0\times2^0$	10
3	$1\times2^1+1\times2^0$	11
4	$1\times2^2+0\times2^1+0\times2^0$	100
5	$1\times2^2+0\times2^1+1\times2^0$	101
6	$1\times2^2+1\times2^1+0\times2^0$	110
7	$1\times2^2+1\times2^1+1\times2^0$	111
8	$1\times2^3+0\times2^2+0\times2^1+0\times2^0$	1000
9	$1\times2^3+0\times2^2+0\times2^1+1\times2^0$	1001
…	…	…
16	1×2^4	10000
32	1×2^5	100000
64	1×2^6	1000000
…	…	…
100	$1\times2^6+1\times2^5+1\times2^2$	1100100
…	…	…

二進法と八進法・十六進法

- 計算機の内部で使うときは便利な2進法だが
  人間が読むには桁数が多すぎる。
- 2進法の表記を3桁ごとにまとめると8進法に
  4桁ごとにまとめると16進法になる。
  → 読みやすく2進法に戻しやすい
- 8進法の表記には 0 ~ 7 までの数字が、
  16進法の表記には 0 ~ 9 の数字と A~F までの文字が
  用いられる。(A=10, B=11, C=12, D=13, E=14, F=15)

  10進法: 19667 → 19667_{10}と書く (5桁)

  2進法: 100 110 011 010 011 (15桁)

  8進法: 4 6 3 2 3 → 46323_{8}と書く(5桁)

  2進法: 0100 1100 1101 0011

  16進法: 4 C D 3 → $4CD3_{16}$と書く(4桁)

  2進法: Binary(バイナリ)

  8進法: Octal(オクタル)

  16進法: Hexadecimal (ヘキサデシマル)

整数の表現

- 正の整数の表現は二進法の位取り記法
  $1_{10} \rightarrow 1_2$, $4_{10} \rightarrow 100_2$, $9_{10} \rightarrow 1001_2$, …
- 計算機内部の「数」は**有限桁**
  「0」は8桁の計算では「00000000」（上位桁の0を省略しない）
- 負の整数の表現
  1. 符号+絶対値表現
     最上位のビットが0の数を「正」1の数を「負」とする
     残りのビットは整数の絶対値
     → 「正の0」と「負の0」の二つができてしまう
  2. 1の補数
     各桁の 0 と 1 を入れ替えたもの (ビット反転)
     → これも「正の0」と「負の0」の二つができる
  3. 2の補数
     1の補数に1を足したもの(上位の桁に溢れたら無視する)
     → 正の数と負の数を「加算」すると引き算になる

  いずれの方法も最上位ビットは「1」になる

  2の補数が「負の整数」として用いられる

整数の引算

- (正の数)“引く”(正の数) → (正の数)“足す”(負の数)
- (負の数)として「2の補数」を採用すると、有限桁数の計算の場合、加算回路に正の数と同じ規則が使える。

例: 4桁の2進法の計算

$4 - 3 \rightarrow (4_{10}) + (-3_{10})$

$4_{10} = 0100_2$

$3_{10} = 0011_2$

→ 1の補数 1100_2

→ 2の補数 $1101_2 = (-3_{10})$

$$
\begin{array}{rl}
0100 & \leftarrow (4_{10}) \\
+)\ 1101 & \leftarrow (-3_{10}) \\
\hline
10001 & \text{溢れた5桁目を無視} \\
0001 & \leftarrow (1_{10})
\end{array}
$$

約束:
- 計算は有限桁、かつ、桁溢れは無視される
- 2の補数を(負の数)と定義する

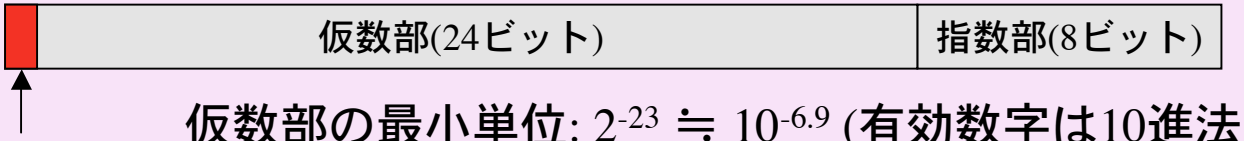
結果:
加算回路で”引算”ができるようになる

浮動少数

- 固定少数と浮動少数
  - 固定少数は約束だけの問題
    例えば「1を0.01とみなす」
    → 「100円を1ドルとみなす」
    例：10円の飴を3つ、20円のキャラメルを2個買った。
    合計で何ドル?
    10円 x 3 + 20円 x 2 = 70円 → 0.7ドル
    ドルと考えた場合「固定少数」
  - 浮動少数は仮数部と指数部で表現

$$(\text{仮数部}) \times 2^{(\text{指数部})}$$

32ビット浮動少数の例



仮数部の最小単位: $2^{-23} \fallingdotseq 10^{-6.9}$ (有効数字は10進法で約7桁)
$-128 \leqq$ 指数部 $\leqq 128$
→ $2^{-128} \fallingdotseq 10^{-38.5}$ から $2^{128} \fallingdotseq 10^{38.2}$
の数を表現することができる

桁落ち

- 計算機内部で行われる計算は常に有限桁
- 大きさの似通った数字の引算があると有効数字の桁数が下がる（桁落ち）

→ 精度を求められる計算をする場合に注意が必要
桁落ちが起きる引算ができるだけ少なくなるように
入力する値の取りかたや計算順序を工夫する

十進法での例:
ある対戦競技で身長差を5倍した値を身長の低い選手にハンディとして与える。選手Aの身長は 175.3 cm、選手Bの身長は 173.9 cm。選手Bのハンディはいくつになるか。

1) 4桁での計算
(175.3 - 173.9) x 5 = 1.4 x 5 = 7.0

2) 3桁での計算(有限桁未満切捨て)
(175.3 - 173.9) x 5 → (175 - 173) x 5 = 2 x 5 = 10

3) 3桁での計算(有限桁未満切捨て)
あらかじめ平均身長(例えば170cm)との差を入力しておく
(5.3 - 3.9) x 5 = 1.4 x 5 = 7.0

文字

- 文字データの表現形式の基本はASCII (ISO646)
- 1バイト(2^8 = 8ビット) のうち下位7ビットにアルファベットや幾つかの記号を割り当てる(いわゆる半角英数字)
- 8ビット目はパリティビット = かつて通信経路のエラー検出に
- 計算機とネットワークの世界の「**共通語**」
- ASCIIを使った文字列は世界中のほとんどすべての情報機器で表示したり入力することができる
  → 電子メールのアドレス、webのURLなど
- 1バイトで表現できるので「1バイト文字」と呼ばれることもある
- 日本語、中国語など各国の言語は「2バイト文字」が現在の主流で、国ごとの互換性は無い
- Unicodeと呼ばれる「万国共通」の文字コード体系の標準化作業が進みつつある (MacOS Xなどではその一部が実装)

ASCIIコード表

文字	16進	文字	16進	文字	16進	文字	16進	文字	16進	文字	16進	文字	16進	文字	16進
NUL	0x00	DLE	0x10	SP	0x20	0	0x30	@	0x40	P	0x50	`	0x60	p	0x70
SOH	0x01	DC1	0x11	!	0x21	1	0x31	A	0x41	Q	0x51	a	0x61	q	0x71
STX	0x02	DC2	0x12	"	0x22	2	0x32	B	0x42	R	0x52	b	0x62	r	0x72
ETX	0x03	DC3	0x13	#	0x23	3	0x33	C	0x43	S	0x53	c	0x63	s	0x73
EOT	0x04	DC4	0x14	$	0x24	4	0x34	D	0x44	T	0x54	d	0x64	t	0x74
ENQ	0x05	NAK	0x15	%	0x25	5	0x35	E	0x45	U	0x55	e	0x65	u	0x75
ACK	0x06	SYN	0x16	&	0x26	6	0x36	F	0x46	V	0x56	f	0x66	v	0x76
BEL	0x07	ETB	0x17	'	0x27	7	0x37	G	0x47	W	0x57	g	0x67	w	0x77
BS	0x08	CAN	0x18	(	0x28	8	0x38	H	0x48	X	0x58	h	0x68	x	0x78
HT	0x09	EM	0x19	)	0x29	9	0x39	I	0x49	Y	0x59	i	0x69	y	0x79
NL	0x0a	SUB	0x1a	*	0x2a	:	0x3a	J	0x4a	Z	0x5a	j	0x6a	z	0x7a
VT	0x0b	ESC	0x1b	+	0x2b	;	0x3b	K	0x4b	[	0x5b	k	0x6b	{	0x7b
NP	0x0c	FS	0x1c	,	0x2c	<	0x3c	L	0x4c	¥	0x5c	l	0x6c	|	0x7c
CR	0x0d	GS	0x1d	-	0x2d	=	0x3d	M	0x4d	]	0x5d	m	0x6d	}	0x7d
SO	0x0e	RS	0x1e	.	0x2e	>	0x3e	N	0x4e	^	0x5e	n	0x6e	~	0x7e
SI	0x0f	US	0x1f	/	0x2f	?	0x3f	O	0x4f	_	0x5f	o	0x6f	DEL	0x7f

16進法の数字の頭に 0x をつけて **0x43** のように表記するのは
プログラミング言語の一つであるC言語における「お約束」
0x43 とは 43_{16} のこと
0x00 ~ 0x1f はいろいろな「制御記号」
0x41 ~ 0x5a, 0x61 ~ 0x7a がアルファベット文字

日本語文字コード

- 日本語（ひらがな、カタカナ、漢字）は2バイト(16ビット＝65536通り)のデータとして表現される（いわゆる全角文字）
  - JISコード (7ビットJIS, ISO-2022-JP)
    - 通信経路で8ビット目をエラー検出に使っていた時代に2バイト文字で通信をするために考案されたコード
    - 電子メールやNetNewsの送受信には今でも7ビットJISが使われる
  - EUCコード
    - 日本語UNIX諮問委員会の提言に基づき米AT&T社で規定されたコード
    - UNIXシステムでよく使われる
  - シフトJISコード
    - Windows, MacOSなどのパソコンで使用される
- 現代のwebブラウザには自働判別機能が備わっているので、あまり意識しなくてもよくなったが、ファイルを直接転送したりする場合には変換作業が必要。
- 「万国共通」を目指すUnicodeがやがて普及?
- 参考文献 「文字の海、ビットの舟」参照

マルチメディア情報(音声・画像)

- 音声や画像も 0 と 1 のデジタル情報に数値化すればいろいろな処理をすることができる
- 音声・音楽は約20Hz ~ 20kHzの空気の振動
  CD → 44.1kHzで音の強弱をサンプリング
- 画像 ~ 2次元情報を「画素(ピクセル)」に分解してサンプリング
- 動画 ~ 2次元情報に時間の情報を加えてサンプリング
- CD: 10分で約百メガバイト
- 静止画: デジタルカメラ300万画素 → 数百メガバイト
- CPUの性能向上につれて90年代半ばから発展
- 圧縮アルゴリズムの発展で扱いやすい情報量に

シャノンの情報理論

- 情報理論の基礎(1948年)
- 情報量を「エントロピー」と定義
- 二つの事象がどちらかが起きる確立をそれぞれ p と $(1-p)$ とするとき、エントロピー $H(p)$の定義は

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p)$$

単位:ビット

- デジタルデータは、**その情報が持っているエントロピーまで**圧縮することができる
- 例: 白黒の100万画素のデジカメ写真に黒い点が１万画素含まれている場合

  $p = 1 \div 100 = 0.01$

  $H(0.01) \fallingdotseq 0.0808$ ビット

  理論上、データ量を8.08%まで圧縮することが可能

  JPEG圧縮などは写真画像に適した圧縮アルゴリズムの例

データ圧縮

- それぞれのデータの特性に応じた圧縮アルゴリズムが開発されている
- 圧縮されたデータから元のデータを完璧に復元することのできるLossless圧縮と、元のデータとは異なるが人間の目や耳にはあまりその差が感じられないLossy圧縮がある
  - Lossless圧縮の例
    - **Run-length法** - 0や1が長く続く場合、その数を記録する
    - **Huffman法** - 出現頻度の高い単語や文字列に符号を割り当てていく
    - **LZW** - Lemple、Ziv、Welchの3人が開発した画像圧縮方式
      - GIF画像などで使用 - 米UNISYS社が特許
  - Lossy圧縮の例
    - **JPEG** - 画像圧縮アルゴリズムを制定する団体Joint Photographic Coding Experts Groupが制定した静止画像圧縮アルゴリズム。Lossless圧縮とLossy圧縮がある。
    - **MPEG** - Moving Picture Coding Experts Group／Moving Picture Experts Groupが制定した動画像圧縮アルゴリズム。

今週のレポート

- 問1: 「-108」を1の補数と2の補数でそれぞれ2進法表記せよ。有効桁は8桁とする。
- 問2: アルファベットの「A」「a」および半角記号の「!」をASCIIコードで2進法で表現すると、それぞれどうなるか。
- 問3: 圧縮アルゴリズムで用いられるHuffman符号とはなにか。インターネット検索を使って調べ、説明せよ。参考にしたwebページのURLも記せ。
- 問4: MP3について検索して分かったことを記せ。参考にしたwebページのURLも記せ。

参考文献

- 稲垣耕作 著「コンピュータ科学の基礎」コロナ社
  ISBN4-339-02338-8

- 小舘香椎子, 上川井良太郎, 中村克彦 共著
  「教養のコンピュータサイエンス情報科学入門 第2版」 丸善
  ISBN4-621-04871-6

- 菊沢正裕, 山川修, 田中武之共著「情報リテラシー：メディアを手中におさめる基礎能力」森北出版
  ISBN4-621-04871-6

- 小形克宏 著
  「文字の海、ビットの舟」―― 文字コードが私たちに問いかけるもの
  http://www.watch.impress.co.jp/internet/www/column/ogata/index.htm