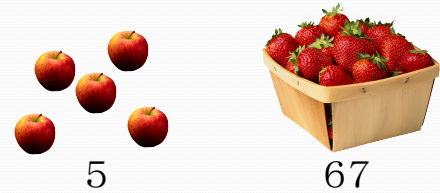


# コンピュータ基礎(5)

7章 情報の表現と基礎理論

## 数の表現 (書き方)

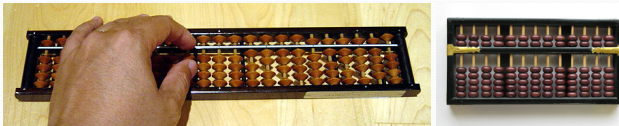
- 「数」と「数の書き方」をわけて考える
- 「数の書き方」と、「数そのものの性質」は別のも  
例：13 は素数・・・「13」という書き方とは無関係



- ここでは書き方 (表現方法) について考える

## 基数法とは？

- 位取り基数法 (基数法)
  - $67 = 6 \times 10^1 + 7 \times 10^0$  のような数の表現方法
  - 10 の部分は10でなくても良い。これを□という。
  - 基数を2にしたものを二進数と言う。
    - 二進数の1けたを□, 8けたを□と呼ぶ。
- 基数法の利点
  - 大きな数でも短く表現出来る (文字数が少ない)
  - ある数字の表現は、ただ1通りしかない



日本のそろばん: 数の表現が一通り

## 基数法の性質

- 表現出来る数の範囲 (基数を r とする)
  - n けたの r 進数で表せる数は 0 から  $r^n - 1$  である
- けたずらし (シフト)
  - r 進数の各けたを左に1けたずらすと r 倍になる  
例：10進数で 123 → 1230 にすると10倍  
同様に、2進数で 101 → 1010 にすると2倍
  - r 進数の各けたを右に1けたずらすと  $1/r$  倍になる  
例：10進数で 1230 → 123 にすると1/10倍  
同様に、2進数で 1010 → 101 にすると1/2倍
- この性質を使うと、一番下のけたの値を調べる事が出来る  
例：3456 / 10 = 345, 余り 6 となる  
同様に、1101 を 1/2 倍すると、110 余り 1

## 基数変換の方法

- 2進数を10進数に変換
  - 例：1101 =  $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$
- 10進数を2進数に変換
  - 2で割った余りを調べ、下の桁から順に並べる  
 $13 / 2 = 6$  余り 1  
 $6 / 2 = 3$  余り 0  
 $3 / 2 = 1$  余り 1  
 $1 / 2 = 0$  余り 1  
となるので、13 は二進数では 1101 になる
  - これは、前のスライドの「2進数を右に1けたシフトすると 1/2 になる」という性質を使っている  
1101(13) を 2 で割ると 110(6), 余り 1 になる

## 16進数と8進数

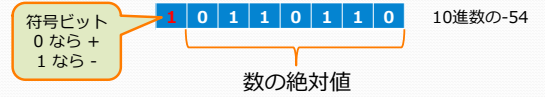
- 10進数と2進数の間の変換は、面倒くさい
  - でも、2進数での表記は、長すぎる  
→ 2進数を短く表現出来ないか？
- 2進数を4桁ずつ区切って表示する
  - 例：10110111 → 1011 と 0111 に分け、それぞれに記号を割り当てて表示しよう！
  - 4桁の2進数は 0~15 の 16通りなので、数字 (0~9) では足りない  
→ A~F を  $1010 (10)_{10} \sim 1111 (15)_{10}$  に割り当て。  
10110111 は B7 と表現出来る
- これは、□進数である。
  - 2進数を4桁ずつ区切っているので、 $2^4 = 16$
  - $B7 = B(11) \times 16^1 + 7 \times 16^0 = 183$

## 例題

- 10進数の 83 について
  - 8bit の 2進数で表せ.
  - 2桁の16進数で表わせ.
- 2進数の 01101011 について
  - 10進数で表せ.
  - 16進数で表わせ.
- 16進数の 7D について
  - 8bit の2進数で表せ.
  - 10進数で表せ.

## 負の数の表現

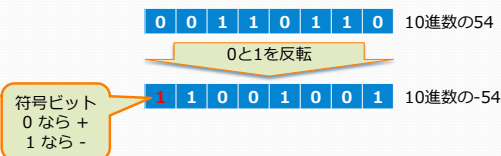
- 8bit(1byte)の2進数は、0~255 を表現出来る
  - 0と正の値しか表現出来ない
  - マイナス(-) の記号を表現出来ないか?
- アイデア1: 絶対値表現 (  )による方法)



- 問題点
  - 0の表現が二通りできてしまう  
00000000 . . . +0  
10000000 . . . -0

## 負の数の表現

- アイデア2:  表現

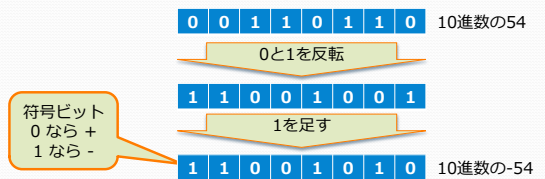


- 問題点
  - 0の表現が二通りできてしまう  
00000000 . . . +0  
11111111 . . . -0

## 負の数の表現

**重要!!**

- アイデア3:  表現



- 利点
  - 0の表現が一通りしかない
  - 負の数をそのまま加算すると正しい答えが得られる

もしも繰り上がりのために桁数が増えたら、その桁は単に捨てる!

## 例題(1)

- 21 を, 8bit の 2の補数で表現せよ.
  - 21 は, 2進数で 00010101
  - 反転して 11101010
  - 1を加えて 11101011 となる.
- 34 - 21 を, 2の補数を用いて計算せよ.
  - 34 は, 2進数で 00100010
  - これに, 上で求めた -21 の 2の補数を加える
  - 00100010
  - + 11101011
  - 10001101 一番上の桁をとって 00001101
  - これは, 10進数では 13 になる.

## 例題(2)

- 113 を, 8bit の 2の補数で表現せよ.
- 57 - 31 を, 2の補数を用いて計算せよ.

## 0の2の補数を計算してみる

- 0の2の補数を求める
  - 00000000
  - これを反転して 11111111
  - これに1を加えて 100000000
  - 最上位桁をすてると 00000000
- よって、0の表現は1通りしか存在しない
- n桁の2の補数による値の表現範囲
  - $-2^{(n-1)}$  から  $2^{(n-1)}-1$  まで
  - 例：8bit の場合、-128 から 127 まで(256通り)

## なぜ引き算ができるのか？(1)

- 10進数で  $845 - 268$  を計算することを考える
  - 繰り下がり計算が面倒！999から引くなら楽なのに！
- なんとかして 999 を使ってみる
  - $845 - 268$   
 $= 845 + (999 - 268) - 999$   
 $= 845 + (999 - 268) - 1000 + 1$   
 $= 845 + (999 - 268 + 1) - 1000$   
とすると計算できる。
- 999 から 268 を引算するのは、各桁の反転に相当
  - $0 \leftrightarrow 9, 1 \leftrightarrow 8, 2 \leftrightarrow 7, 3 \leftrightarrow 6, 4 \leftrightarrow 5$  で置換
  - なので、上の括弧内の計算( $999-268+1$ )は 各桁を反転して1を足すことに相当する

## なぜ引き算ができるのか？(2)

- 2進数の反転計算
  - 11111111 から引くことと同じ。
- 2の補数
  - 反転してから1を加えるので、100000000 から引くことに相当する
  - 桁あふれは無視するので、100000000 を足したり引いたりするのは、何もしないのと同じ。
- つまり、2の補数とは
  - 事前に引き算をしておいた値のようなもの。

## 小数の表現(1)

- 桁ずらし(シフト演算)について
  - 10進数の 123 を右に1桁ずらすと 12.3 となる。この値は 123 の  $1/10$  である。
  - 同様に、2進数の 1101 を右に1桁ずらすと 110.1 となる。これは  $1101 (13)_{10}$  の  $1/2$  である。つまり、6.5 である。
  - これを、と呼ぶ。
- 基数による解釈
  - 110.1 は  $1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} = 6.5$
  - 小数点以下の各桁の重みは、0.5, 0.25, 0.125, ..
- 割り切れない数について
  - 例えば、0.2 は2進数では循環小数になる  
 $0.00110011001100\dots$

## 例題

- 2進数の固定小数, 110.011 について
  - 10進数で表わせ。
- 10進数の小数, 7.825 について
  - 2進数で表せ。

- 
- 非常に大きい数の表現
  - 例：光の速度  $3 \times 10^8 = 300000000$  [m/s]
  - このように、桁をずらす桁数(指数)を使うことで非常に大きい数や小さな数を表現することが出来る
  - C言語では、float や double で使われている
    - float：単精度浮動小数点数
    - double：倍精度浮動小数点数
- 2進数では
  - 1桁ずらすと、2倍したことになるので、 $a \times 2^b$  のような表現になる
  - aを, bをと呼ぶ。

## 浮動小数点数の規格

- IEEE754 として規格化されている

s	指数部(e)	仮数部(m)
---	--------	--------

- float :全体で32bit
  - 符号ビット:1bit, 指数部:8bit, 仮数部:23bit の計32bit
- double :全体で64bit
  - 符号ビット:1bit, 指数部:11bit, 仮数部:52bit の計64bit
- 値は, 以下の式で計算できる
  - float :  $(-1)^s \times 2^{e-127} \times (1+m)$
  - double :  $(-1)^s \times 2^{e-1023} \times (1+m)$
  - s が 1 なら負の数である.
- 高度な話題 : げたばき表現, ケチ表現など.

## 10進数の表現

- コンピュータは, 2進数と10進数を変換している
  - 人に計算結果を見せるため(2→10)
  - プログラムをコンパイルするとき(10→2)
- 10進数も, なんらかの方法で表現する必要がある
- 10進数の一桁は, 2進数の4bit で表すことが出来る.
  - 10進数という. 23 なら "0010 0011"
  - 2進数でも, 16進数でもないことに注意!!
- 10進数の一桁を, 8bitで表すこともある.
  - 10進数という.

## 文字コード

- 文字を2進数で表すには?
  - a~z なら26通り
  - 大文字/小文字に, 10個の数字を加えて62通り
  - 記号!"#\$%&'()\*+,-\*/=^~;:}{<>?
  - ...を考えると, キリがいい所で8bitにしよう.
  - 256種類の文字が使える.
- 文字1つ1つに, 数値を割り当てる.
  - 文字'0'には48, 文字'A'は65, というふうに.
  - 0番から31番は特殊な用途に使われている.
  - 改行記号, 1文字消去, などなど.
- C言語の char 型変数は, 8bit の変数.

## ASCIIコード/JISコード

- コード
  - 7bit にアルファベット・数字・記号を入れたもの
  - 現在は, ほとんどのコンピュータで使われている
- コード(JIS X0201)
  - 残った半分にカタカナを入れたもの(半角カナ)
  - 濁点も1文字
  - ひらがな, 漢字は表現できない

	上位4bit																					
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F						
0	DE	SP	0	@	P		p										一	タ	ミ			
1	SH	D1	!	1	A	Q	a	q										。	ア	チ	ム	円
2	SX	D2	"	2	B	R	b	r										「	イ	ツ	メ	年
3	EX	D3	#	3	C	S	c	s										」	ウ	テ	モ	月
4	ET	D4	\$	4	D	T	d	t										、	エ	ト	ヤ	日
5	EQ	NK	%	5	E	U	e	u										・	オ	ナ	ユ	時
6	AK	SN	&	6	F	V	f	v										ヲ	カ	ニ	ヨ	分
7	BL	EB	'	7	G	W	g	w										ア	キ	ヌ	ラ	秒
8	BS	CN	(	8	H	X	h	x										イ	ク	ネ	リ	
9	HT	EM)	)	9	I	Y	i	y										ウ	ケ	ノ	ル	
A	LF	SB	*	:	J	Z	j	z										エ	コ	ハ	レ	
B	HM	EC	+	:	K	[	k	[										オ	サ	ヒ	ロ	
C	CL	→	,	<	L	¥												ヤ	シ	フ	ワ	
D	CR	←	-	=	M	]	m	]m										ユ	ス	ヘ	ン	
E	SO	↑	.	>	N	^	n	^										ヨ	セ	ホ	*	
F	SI	↓	/	?	O	_	o	_o										ツ	ソ	マ	*	

ASCIIコード

## 例題

- 次の4文字 "STAR" を, ASCIIコードで表わせ.
  - 16進数では □□ □□ □□ □□
- 次の16進数は, どのような文字か.
  - 54 6F 77 6E

## 漢字コード

- 漢字は数千種類も存在する
  - 8bit では表現出来ない・16bit (2バイト) 使う
- 数種類の漢字コードが使われている
  - JISコード・通信での標準的な規格
  - Shift-JISコード (Windows での標準)
  - EUCコード (UNIX でよく使われてきた)
  - コードが違っていると, 文字化けの原因に.
- 他の言語 (韓国, タイ, ...) の文字は?
  - Unicode が策定され, 普及し始めている
  - 多言語の文字を扱うことが出来る