2009 年度春期

電子計算機概論 I

(数学科)

(担当: 角皆)

本講義の概要

- 「計算」の定式化
- 「計算」の実現
- 「計算」の量と質

$\longrightarrow$ 「計算する」とは?

本講義の概要

- 「計算」の定式化
- 「計算」の実現
- 「計算」の量と質

$\longrightarrow$ 「**計算する**」とは?

「計算」

||

計算機で行なえること

計算機

- アナログ計算機
  - ⋆ 図式計算 (計算図表・ノモグラム)
  - ⋆ 計算尺
- ディジタル計算機
  - ⋆ プログラム機構方式
  - ⋆ プログラム入力方式
  - ⋆ プログラム内蔵方式 (von Neumann 方式)
- 量子計算機

計算機

- アナログ計算機
  - ⋆ 図式計算 (計算図表・ノモグラム)
  - ⋆ 計算尺
- ディジタル計算機
  - ⋆ プログラム機構方式
  - ⋆ プログラム入力方式
  - ⋆ プログラム内蔵方式
    (von Neumann 方式)
- 量子計算機

計算機

- アナログ計算機
  - ⋆ 図式計算 (計算図表・ノモグラム)
  - ⋆ 計算尺
- ディジタル計算機
  - ⋆ プログラム機構方式
  - ⋆ プログラム入力方式
  - ⋆ プログラム内蔵方式
    (von Neumann 方式)
- 量子計算機

本講義の概要

- 「計算」の実現: 計算機
  - ⋆ 数の表し方・二進法・Boole 代数
  - ⋆ 計算機の実現
    論理回路・演算回路・順序回路
  - ⋆ 計算 (プログラム) の実際
    簡易アセンブラ

本講義の概要

- 「計算」の定式化
  - ⋆ 計算機のモデル化
    有限オートマトン・Turing 機械など
  - ⋆ 計算可能性の理論
- 「計算」の量と質
  - ⋆ 計算量の理論

計算機での情報の表し方

計算機内では
　　全てのデータは 0,1 の組 (列) で表される。

情報の最小単位:
　　0 か 1 か : bit (binary digit)

実際には幾つかの bit を組にして一度に扱う
　　(通常は 8 bit $= 1$ byte (octet))

1 byte で $2^8 = 256$ 通りの値を表せる

計算機での数の表し方

計算機内の 0,1 の列を、
二進表記 (二進法) で数値として扱う

正の整数の場合、1 byte で

$$0 \leq x \leq 2^8 - 1 = 255$$

の範囲の値が表せる

十進表記 $\longleftrightarrow$ 二進表記の変換

十進	二進
0	**00000000**
1	**00000001**
2	**00000010**
3	**00000011**
4	**00000100**
5	**00000101**
⋮	⋮
253	**11111101**
254	**11111110**
255	**11111111**

二進の九九 (一一?) の表

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

演習問題 1:

次の各式の被演算子 (operand) を

二進表記で表した上で、

それを用いて筆算で計算せよ。

(1) $87 + 26$

(2) $87 + 50$

(3) $87 - 26$

(4) 13×11

二進では桁数が多くて煩わしい

$\longrightarrow$ 十六進 (hexadecimal) 表記

十進	二進	十六進
0	**0000**	0
1	**0001**	1
2	**0010**	2
3	**0011**	3
4	**0100**	4
5	**0101**	5
6	**0110**	6
7	**0111**	7

十進	二進	十六進
8	**1000**	8
9	**1001**	9
10	**1010**	a
11	**1011**	b
12	**1100**	c
13	**1101**	d
14	**1110**	e
15	**1111**	f

十六進表記を示すのに、屡々0x を頭置する

十進	二進	十六進
0	**00000000**	0x00
1	**00000001**	0x01
2	**00000010**	0x02
3	**00000011**	0x03
4	**00000100**	0x04
5	**00000101**	0x05
⋮	⋮	⋮
253	**11111101**	0xfd
254	**11111110**	0xfe
255	**11111111**	0xff

練習問題 (レポート問題の一つ):
十六進の九九 (FF?) の表を作ってみよう

	0	1	2	⋯	e	f
0						
1						
2						
⋮						
e						
f						

(表を出力するプログラムを書いて作成せよ)

文字の表し方:

符号化 (coding) して数値 (文字番号) で表す

(文字コード)

アスキーコード (ASCII code) 対応表

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
2		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

負の数 (符号付き整数) を表すには?
(− も 0,1 で表す)

単純に考えると、

- 符号に 1 bit (正なら 0・負なら 1)
- 絶対値に残り 7 bit ($0 \leq |x| \leq 2^7 - 1$)

$\longrightarrow$ 欠点あり

- 0 が 2 通りに表される ($+0, -0$)
- 演算で正負の場合分けが面倒

$\longrightarrow$ 他にもっと良い方法はないか?

負の数 (符号付き整数) を表すには?
($-$ も 0,1 で表す)

単純に考えると、

- **符号に 1 bit (正なら 0・負なら 1)**
- **絶対値に残り 7 bit ($0 \leq |x| \leq 2^7 - 1$)**

$\longrightarrow$ **欠点あり**

- **0 が 2 通りに表される ($+0, -0$)**
- **演算で正負の場合分けが面倒**

$\longrightarrow$ **他にもっと良い方法はないか?**

負の数 (符号付き整数) を表すには?

($-$ も 0,1 で表す)

単純に考えると、

- **符号に 1 bit (正なら 0・負なら 1)**
- **絶対値に残り 7 bit ($0 \leq |x| \leq 2^7 - 1$)**

$\longrightarrow$ 欠点あり

- **0 が 2 通りに表される ($+0, -0$)**
- **演算で正負の場合分けが面倒**

$\longrightarrow$ 他にもっと良い方法はないか?

負の数 (符号付き整数) を表すには?

($-$ も 0,1 で表す)

単純に考えると、

- **符号に 1 bit (正なら 0・負なら 1)**
- **絶対値に残り 7 bit ($0 \leq |x| \leq 2^7 - 1$)**

$\longrightarrow$ **欠点あり**

- **0 が 2 通りに表される ($+0, -0$)**
- **演算で正負の場合分けが面倒**

$\longrightarrow$ **他にもっと良い方法はないか?**