

# 情報学基礎

## 2章

# コンピュータにおける情報の表現

担当：管理工学科

篠沢 佳久

# 本日の内容

- コンピュータにおける情報の表現(2章)
  - 2進数(2.1節)
  - 16進数(2.2節)
  - 整数の内部表現(2.3節)
  - 符号なし整数(2.4節)
  - 実数(2.5節)
  - 文字(2.6節)
  - 音声や音楽(2.7節)
  - 情報量について(コラム)

# 2進数(2.1節)

まず算数の世界で考えます

# 整数の10進数と2進数①

- 10進数整数

0, 1, 2, 3, 4, ..., 9, 10, 11, ..., 99, 100, ...

- 2進数整数

0, 1, 10, 11, 100, 101, 110, 111, 1000, ...

# 整数の10進数と2進数②

- 10進数整数

-4, -3, -2, -1, 0, 1, 2, 3, 4, ...

- 2進数整数

-100, -11, -10, -1, 0, 1, 10, 11, 100, ...

# 2進数から10進数への変換①

- 2進数

$b_{n-1} \quad b_{n-2} \quad b_{n-3} \quad \dots \quad b_2 \quad b_1 \quad b_0$

$b_i = \{0, 1\}$



10進数に変換

$$a = b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

# 2進数から10進数への変換②

- 10進数への変換例

1	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---



10進数に変換

$$\begin{aligned} &1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 128 + 32 + 16 + 4 + 2 + 1 \\ &= 183 \end{aligned}$$

(問) 10進数に変換しなさい

1	1	0	1	1	0	1	0
---	---	---	---	---	---	---	---

# 10進数から2進数への変換①

2で割る	商	余り
2 ) 44		0
2 ) 22		0
2 ) 11		1
2 ) 5		1
2 ) 2		0
2 ) 1		1
0		

この順に並べる

2進数という意味です

44<sub>(10)</sub> = 101100<sub>(2)</sub>

10進数という意味です

0になるまで2で割る

(問) 3213を2進数に変換しなさい



# 2進数小数(10進数小数への変換)

- 2進数小数

$$0.k_1 k_2 k_3 \dots k_n \quad k_i = \{0, 1\}$$



10進数に変換

$$a = k_1 \cdot 2^{-1} + k_2 \cdot 2^{-2} + k_3 \cdot 2^{-3} \dots + k_n \cdot 2^{-n}$$

(例)

$$\begin{aligned} 0.1011 (2) &= 1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125 + 1 \times 0.0625 \\ &= 0.6875 (10) \end{aligned}$$

# 10進数小数から2進数小数への変換

1.25の小数部

1.25の整数部

$$0.625 \times 2 = 1.25 \quad 1$$

$$0.25 \times 2 = 0.5 \quad 0$$

$$0.5 \times 2 = 1 \quad 1$$

この順に並べる

小数部がなくなるまで続ける

$$0.625_{(10)} = 0.101_{(2)}$$

(問)  $0.1_{(10)}$  を2進数へ変換しなさい

$$0.1 \quad \times 2 = 0.2 \quad 0$$

$$0.2 \quad \times 2 = 0.4 \quad 0$$

$$0.4 \quad \times 2 = 0.8 \quad 0$$

$$0.8 \quad \times 2 = 1.6 \quad 1$$

$$0.6 \quad \times 2 = 1.2 \quad 1$$

$$0.2 \quad \times 2 = 0.4 \quad 0$$

▪

▪

この部分を  
繰り返す

すなわち  $0.1_{(10)}$  は  $0.0001\dot{1}_{(2)}$  と  
いう無限小数になる

(問)  $0.3_{(10)}$  を2進数へ変換しなさい

# 実数の10進・2進変換

- 整数部と小数部に分けて変換する

(例)  $5.75_{(10)} = 101.11_{(2)}$

$$44.6875_{(10)} = 101100.1011_{(2)}$$

# 16進数(2.2節)

- 2進数を8進数や16進数で表現すると省スペースになる

10進数	2進数	8進数	16進数
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	a
11	1011	13	b
12	1100	14	c
13	1101	15	d
14	1110	16	e
15	1111	17	f

# 16進数の例

- $31_{(10)} = 11111_{(2)} = 1f_{(16)}$
- $100_{(10)} = 1100100_{(2)} = 64_{(16)}$
- $0.6875_{(10)} = 0.1011_{(2)} = 0.b_{(16)}$
- $10.5_{(10)} = 1010.1_{(2)} = a.8_{(16)}$



$0.1_{(2)} \rightarrow 0.1000_{(2)}$

# 整数の内部表現(2.3節)

ここまでは算数の世界でした

では、コンピュータ内部では  
どう表現しているのでしょうか？



# 整数のコンピュータ内部での表現

- 一定のビット幅を使って表現
- 例として「整数を4ビットで表す」ことを考える  
(10進数) (2進数) (コンピュータ内部)

正の整数  
の場合

0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111



- (前ページでわかるように)  
正数は素直に変換できる
- 負数は**2の補数表示**で表す  
(10進数)    (2進数)    (コンピュータ内部)

－1	－1	1111
－2	－10	1110
－3	－11	1101
－4	－100	1100
－5	－101	1011
－6	－110	1010
－7	－111	1001
－8	－1000	1000

**2の補数表示**

# 2の補数表示①

- 正の整数と0(ゼロ)は最左ビットが0で, 算数の世界と同様
- 負の整数は最左ビット(most significant bit; **MSB**)を1
- ある負数( $-n$ )の2の補数表示を求めるには,  $n$ のビット表現の0と1を反転させて1を加える  
(例)  $-6$ の**2の補数表示**は

0110		1001		1010
6	(ビット反転)		(+1)	-6

## 2の補数表示②

- 2の補数表示を使うと負数であっても足し算で計算できる
  - $6 + 6 = 1010 + 0110 = 0000$ （繰上りは無視）
  - $2 + (-4) = 1110 + 1100 = 1010$
- 一定のビット幅しか使っていないので、**表せる数の上限と下限がある**ことに注意！
  - （例）4ビットだと  $-8 \sim +7$  ( $-2^3 \sim 2^3 - 1$ ) の間の16種類のみ
  - （問1）4ビットで  $-6 + (-6)$  はどうなるか？
  - （問2）32ビットで表せる上限と下限は？

# 符号なし整数①

- 負の整数を使う必要のない場合は最左ビット (MSB) を符号として使いたくない

- 4ビットで考えると,

(10進数)	(2進数)	(コンピュータ内部)
--------	-------	------------

8

1000

1000

9

1001

1001

10

1010

1010

11

1011

1011

12

1100

1100

13

1101

1101

14

1110

1110

15

1111

1111

## 符号なし整数②

- $n$  ビットで符号なし整数を表す場合は  
     $0 \sim 2^n - 1$   
の範囲の整数を表すことができる

(問) 8ビットで符号なし整数を表す場合の上限と下限は？

# 整数の表現方法のまとめ

ビットパターン (2進数)	16進数	符号なし整数 (10進数)	2の補数表示
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	8	− 8
1001	9	9	− 7
1010	a	10	− 6
1011	b	11	− 5
1100	c	12	− 4
1101	d	13	− 3
1110	e	14	− 2
1111	f	15	− 1

# 実数(2.5節)

- 10進数の場合
  - $M \times 10^e$
  - $6.0221415 \times 10^{23}$
  - $23.4 = 234 \times 10^{-1}$   
 $= 2.34 \times 10^1$
  - Mを仮数部, eを指数部
- 2進数の場合
  - $10.01 = 1001 \times 2^{-2}$   
 $= 1.001 \times 2^1$

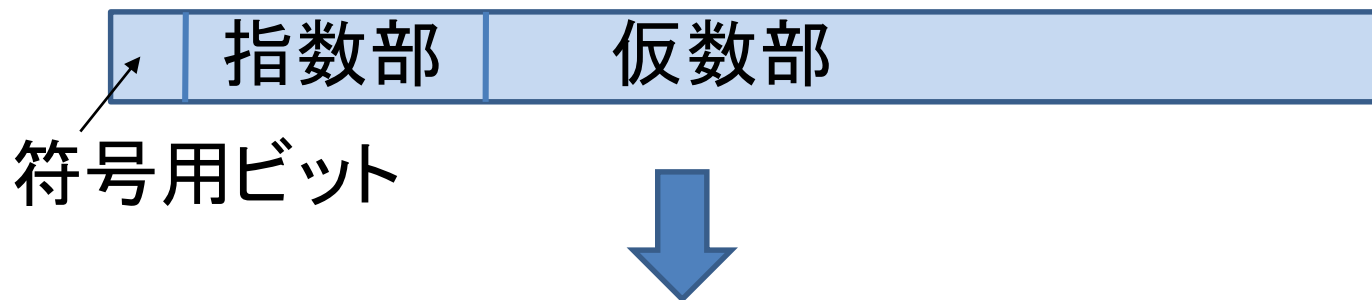


# 実数のコンピュータ内部での表現方法①

- 実数を **仮数**(2進数)  $\times$  **2<sup>指数</sup>** の形式で表す  
その際、仮数を 1.・・・ の形にする
  - この作業を正規化という
  - 正規化するときには小数点を動かすことから「**浮動小数点**」(floating point)と呼ばれる
- どの実数も1. の部分は共通となるので、これ以降の・・・の部分だけをコンピュータ内に収める
- **仮数の符号は最左ビットで表す**(1が負数)

## 実数のコンピュータ内部での表現方法②

- 指数も正負があるが, 最小を $00\cdots 0$ , 最大を $11\cdots 1$ とする(げたばき表示)  
通常はこの真ん中あたりを0乗とする
  - この0乗の指数を**げた**と呼ぶ
- コンピュータでの内部表現



$(-1)^{\text{最左ビット}} \times (1. + \text{仮数部}) \times 2^{(\text{指数部} - \text{げた})}$   
の実数を表していることになる

- $0.0_{(2)}$ は**例外**で内部表現はオールゼロ

# 実数のコンピュータ内部での表現方法③

## 浮動小数点数の規格 (IEEE754)

- 単精度 (32ビット):  
指数部8ビット (127のけた) と仮数部23ビット  
⇒ 結果として有効桁が7桁ほど
- 倍精度 (64ビット):  
指数部11ビット (1023のけた) と仮数部52ビット  
⇒ 結果として有効桁は15桁ほど

# 問題

- 10進数実数  $-3.625$  は単精度表現としてコンピュータ内でどのように表わされるか？

# 単精度浮動小数点数

- $-3.625_{(10)} = -11.101_{(2)} = -1.1101_{(2)} \times 2^1$   
 $= -1.\textcolor{blue}{1101}_{(2)} \times 2^{(\textcolor{blue}{128}-\textcolor{red}{127})}$

げた

- 1 10000000 1101000...0

符号

指数(8ビット)

仮数(23ビット)

$$10000000_{(2)} = 128_{(10)}$$

# 問題

- 10進数実数 10.75 は倍精度表現としてコンピュータ内でどのように表わされるか？

# 倍精度浮動小数点数

$$10.75_{(10)} = 1010.11_{(2)} = 1.01011 \times 2^3$$
$$= 1.01011 \times 2^{(1026-1023)}$$

げた

0 100000000010 0101100000.....0

指数部  
(11ビット)

仮数部  
(52ビット)

符号ビット

$$1026_{(10)} = 100000000010_{(2)}$$

# 丸め誤差

10進数の 0.1 は2進数で無限小数になる



仮数部は有限ビットしかない



収納できない部分は捨てる(丸める)



誤差が発生する (これを丸め誤差という)



# 丸め誤差の例①

(問) 10進小数 0.1 を単精度浮動小数点表示  
しなさい

## 丸め誤差の例②

$$\begin{aligned} 0.1_{(10)} &= 0.0001100110011\cdots_{(2)} \\ &= 1.10011001\cdots \times 2^{-4} \quad \leftarrow (123-127) \end{aligned}$$

0 01111011 100110011001100110011001100

指数部  
(8ビット)

仮数部  
(23ビット)

符号ビット

$$(123)_{(10)} = (01111011)_{(2)}$$

(注意 最後の数字を1に  
丸めることもある)

# 文字のコンピュータ内部での表現(2.6節)

- アルファベット文字や半角記号はASCIIコード表に基づく
- 各文字はASCIIコードでは7ビットのビットパターンで表される
- 日本語は複数バイトで表す  
コード体系にはEUC, JISコード, S-JISコード, UTFなどがあるが, 近年はUTFへ統一されつつある  
⇒ 今でもときとして文字化けが起こる

# ASCIIコード表①

コード	文字	コード	文字	コード	文字	コード	文字
0～8	(説明略)	53	5	78	N	103	g
9	水平タブ	54	6	79	O	104	h
10	改行	55	7	80	P	105	i
11～31	(説明略)	56	8	81	Q	106	j
32	空白	57	9	82	R	107	k
33	!	58	:	83	S	108	l
34	"	59	;	84	T	109	m
35	#	60	<	85	U	110	n
36	\$	61	=	86	V	111	o
37	%	62	>	87	W	112	p
38	&	63	?	88	X	113	q
39	'	64	@	89	Y	114	r
40	(	65	A	90	Z	115	s
41	)	66	B	91	[	116	t
42	*	67	C	92	\	117	u
43	+	68	D	93	]	118	v
44	,	69	E	94	^	119	w
45	-	70	F	95	_	120	x
46	.	71	G	96	`	121	y
47	/	72	H	97	a	122	z
48	0	73	I	98	b	123	{
49	1	74	J	99	c	124	
50	2	75	K	100	d	125	}
51	3	76	L	101	e	126	~
52	4	77	M	102	f	127	Del

- 憶える必要はない
- 普段は文字として意識しないような「ベル音(コードは7)」や「改行」にもコードが割り当てられている

## ASCIIコード表②

- Aが65ということは、文字Aがコンピュータ内部で1000001というビットパターンで表現されることを示す
- 0(ゼロ)が48ということは、文字0は110000というパターンであることを示す
- 文字0が10進数の48のパターン、文字1が49のパターン、というように0～9, A～Z, a～zは連続したパターンが割り当てられている

# ASCIIコード表③

- (例) Keio (文字)
- (10進数) 75 101 105 111
- (2進数) 1001011 1100101 1101001 1101111
  
- (例) 123 (文字)
- (10進数) 49 50 51
- (2進数) 0110001 0110010 0110011



文字と整数は表現方法が異なる

- (例) 123 (10進数の整数)
- (2進数) 1111011

# パリティ(参考)

- ASCIIコードは7ビット
- データの転送時などにどこかのビットの0と1が反転するような誤りを検出するために、ASCIIコードに1ビット加えて、その文字データの1の数が偶数になるようにする(これを偶数パリティという)  
たとえば、Aは10000001だが、この先頭にパリティビットを加えて010000001とする

# 音声や音楽(2.7節)

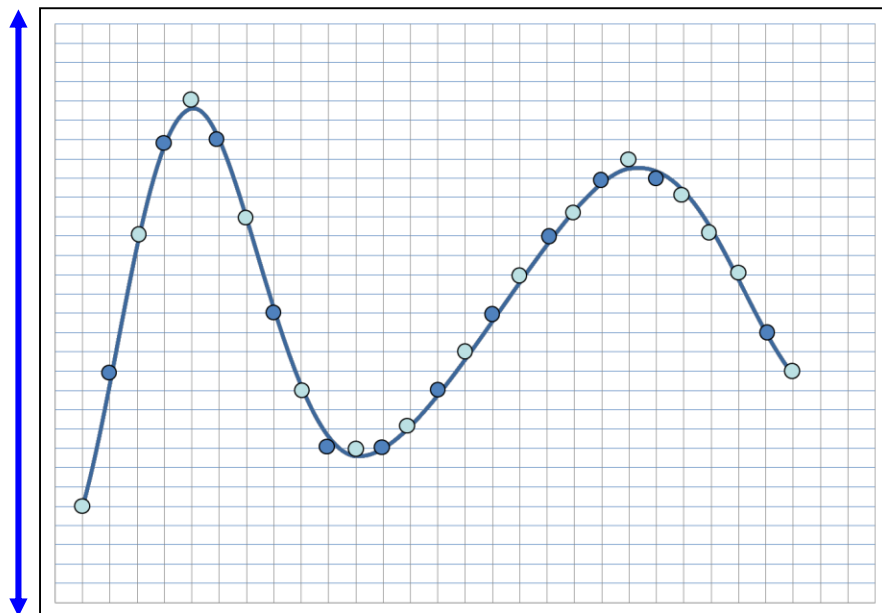
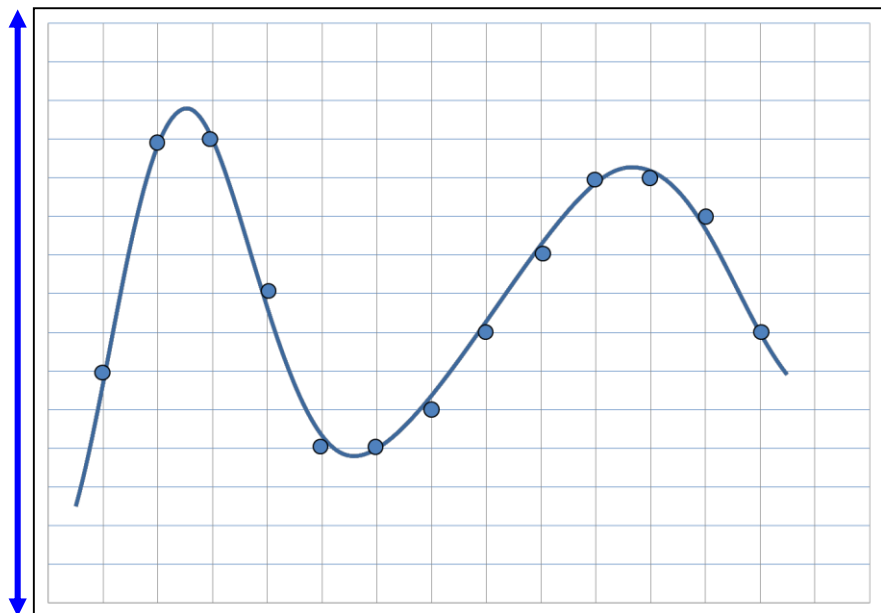
- 音声や音楽
  - アナログ信号
- デジタル化
  - ある時刻の信号レベルを一番近い離散値(とびとびの値)に置き換える
  - 時間間隔が短いほど, 離散値の幅が細かいほど, 元の信号を正確に置き換えられる



# 標本化と量子化①

量子化

量子化



標本化

標本化

## 標本化と量子化②

- 標本化・・・一定時間ごとの離散的な時点の値の系列で表現すること  
1秒間に何回標本化するかを**標本化周波数**として表す

**シャノン・染谷の標本化定理**より, 元の波の周波数の2倍の標本化周波数で標本化すれば再現できることがわかっている

- 量子化・・・採取した信号レベルを有限個の値のなかの一つで対応すること  
**量子化ビット数**で何ビットの値で表現するかを表す

# CDでの音楽データ

- 標本化周波数: 44.1 kHz
- 量子化ビット数: 16ビット
- 左右のチャンネルそれぞれのデータがある

(問) コンパクト・ディスクに入っている60分の音楽データの大きさを求めなさい

→ 次週, バイトの説明後に解説します

# まとめ：数の内部表現での注意

- たとえば0100100 というデータが与えられたとき、それをどう解釈すべきか（2の補数表示，符号なし整数，文字コード等）はデータからはわからない

# 情報量(コラム)

- シャノンによる定義:  
確率 $p$ で起こる事象をメッセージで伝えた場合  
の情報量を  $\log_2 (1/p) = -\log_2 p$  とする
- ある事象の確率が  $p = 1/2$  のとき,  
このメッセージの情報量は 1 となる.  
これを情報の単位として1ビットと呼んだ.
- 独立な2つの事象の同時確率はそれぞれの事  
象の確率の積である  $\Rightarrow$  情報量としては和  
になる

# 問題

- ①「さいころを1回ふったら偶数の目がでた」というメッセージの情報量を求めよ.
- ②「さいころを1回ふったら2の目がでた」というメッセージの情報量は①の $1/3$ か, 検証しなさい.

# 情報エントロピー

- 同時に起こらないM個の事象が、確率 $p_1, p_2, \dots, p_n$  (和は1) で出現するとき、この情報源の価値を情報量の期待値の和  
$$-\sum (p_i \times \log_2 p_i)$$
で表すことにする. これを情報エントロピーという
- 情報エントロピーは平均情報量とみなすことができ、単位はやはりビットである.

# 問題

- 英語のアルファベット26文字とスペースがすべて等確率で生じるとするときの情報エントロピーを求めなさい.



# 本日のまとめ

- 2章: コンピュータにおける情報の表現
- 次週は3章を読んで来て下さい