

コンピュータで扱うデータ

http://cobayasi.com/koza/kihon/3_compdata.pdf

1. 2進数 ★★★
2. 負数の表現★★
3. シフト演算★★
4. 小数の表現★★
5. 誤差★
6. 論理演算と論理回路★★★★
7. 半加算器と全加算器★★
8. 文字データの表現★★
9. 音声や動画などのデジタルデータの表現★★★★



1. 2進数

- コンピュータ内部の扱うデータ
- 2進数→10進数の変換
- 10進数→2進数の変換
- 8進数,16進数→10進数の変換
- 10進数→16進数の変換
- ビットとバイト

● 10進数と2進数

■ 10進数

$$256 \Rightarrow 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 = 256$$

100の桁の値

100の位の重み

10の桁の値

10の位の重み

1の桁の値

1の位の重み

基数

■ 2進数

$$101 \Rightarrow 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

4の位の値

4の位の重み

2の位の値

2の位の重み

1の位の値

1の位の重み

基数

10進数	2進数
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

桁上がり

桁上がり

桁上がり

桁上がり

● 2進数 → 10進数の変換

基数

4の位の重み

2の位の重み

1の位の重み

$$111 \Rightarrow 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 7$$

4の位の値

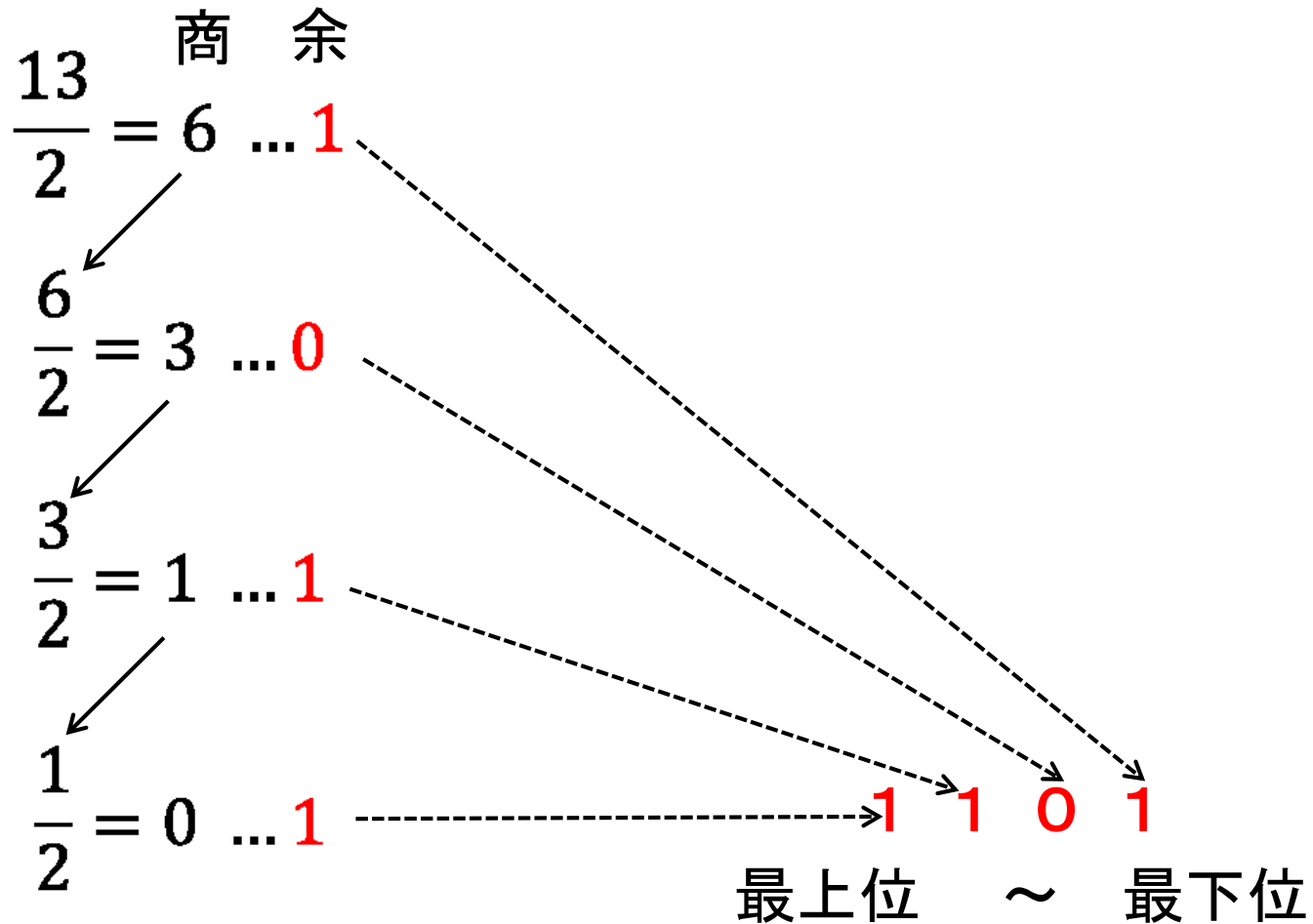
2の位の値

1の位の値

位の値 × 位の重み + …… + 位の値 × 位の重み

桁数だけ繰り返す

● 10進数 → 2進数の変換



● 8進数、16進数 → 10進数の変換

■ 8進数

基数

$$512 \Rightarrow 5 \times 8^2 + 1 \times 8^1 + 2 \times 8^0 = 330$$

64の位の重み (pointing to 8^2)

8の位の重み (pointing to 8^1)

1の位の重み (pointing to 8^0)

64の位の値 (pointing to 5)

8の位の値 (pointing to 1)

1の位の値 (pointing to 2)

■ 16進数

基数

$$1FB \Rightarrow 1 \times 16^2 + 15 \times 16^1 + 11 \times 16^0 = 507$$

256の位の重み (pointing to 16^2)

16の位の重み (pointing to 16^1)

1の位の重み (pointing to 16^0)

256の位の値 (pointing to 1)

16の位の値 (pointing to 15)

1の位の値 (pointing to 11)

10進数	8進数	16進数
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	10	8
9	11	9
10	12	A
11	13	B
12	14	C
13	15	D
14	16	E
15	17	F

桁上がり

桁上がり

● 10進数 → 8進数の変換

$$\frac{330}{8} = 41 \text{ ... } 2$$

商 余

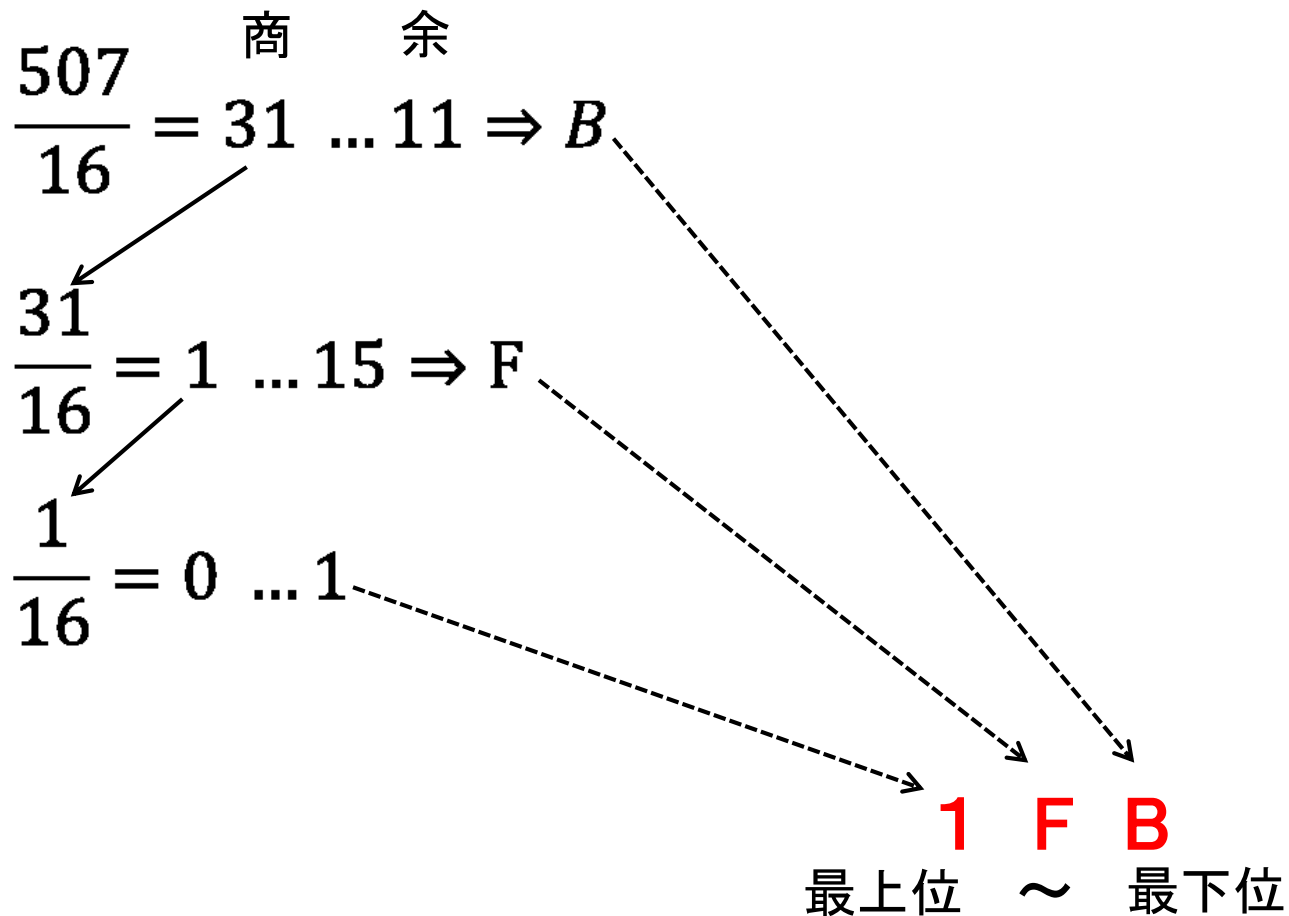
$$\frac{41}{8} = 5 \text{ ... } 1$$

$$\frac{5}{8} = 0 \text{ ... } 5$$

5 1 2

最上位 ~ 最下位

● 10進数 → 16進数の変換



< 小数の基数変換 >

■ 2進数 → 8進数

3桁ずつ区切って、8進数1桁にする

例えば、1010.01を変換するとき、

桁が足りなければ補う

基準

001010.010

1 2 . 2

■ 2進数 → 16進数

4桁ずつ区切って、16進数1桁にする

例えば、1010.01を変換するとき、

基準

1010.0100

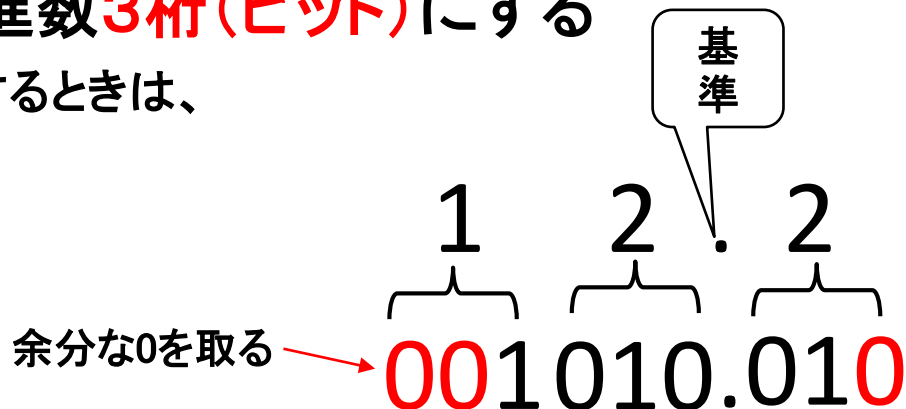
A . 4

桁が足りなければ補う

■ 8進数 → 2進数

8進数の**1桁**を、2進数**3桁(ビット)**にする

例えば、12.2を変換するとき、



■ 16進数 → 2進数

16進数の**1桁**を、2進数**4桁(ビット)**にする

例えば、A.4を変換するとき、



ビットとバイト

これも
知っとこ

■ 2進数1桁の単位 → **ビット(bit)**

2桁は2[bit]、4桁は4[bit]、8桁は8[bit]

■ 2進数8桁(bit)の単位 → **バイト(byte)**

16[bit]は2[byte]、32[bit]は4[byte]、64[bit]は8[byte]

■ 1[ビット]の組合せ数は、 $2^1 = 2$ 通り → 0,1

■ 2[ビット]の組合せ数は、 $2^2 = 4$ 通り → 00,01,10,11

■ 3[ビット]の組合せ数は、 $2^3 = 8$ 通り → 000,001,010
011,100,101
110,111

【過去問題】16進数(小数)3A.5Cを10進数の分数で表したものはどれか

ア $939/16$

イ $3735/64$

ウ $14939/256$

エ $14941/256$

基数変換は必ずできるように！
分数や小数についての問題が出る

16を基数として、問題の3A.5Cを表現すると、
 $(3 \times 16^1) + (10 \times 16^0) + (5 \times 16^{-1}) + (12 \times 16^{-2})$

整数部分は、

$$(3 \times 16) + (10 \times 1) = 58$$

小数部分は、

$$(5 \times 1/16) + (12 \times 1/256) = (80 + 12)/256 = 92/256 = 23/64$$

整数部分と小数部分を足して、

$$(3712/64) + (23/64) = 3735/64$$



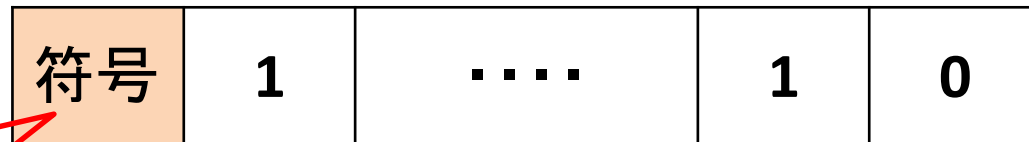
2. 負数の表現

- 負数の表現方法
- 絶対値表現
- 補数表現

● 負数の表現方法

正数または
負数

- 符号ありデータ(例: 0, 1, -1, 2, -2, ...)



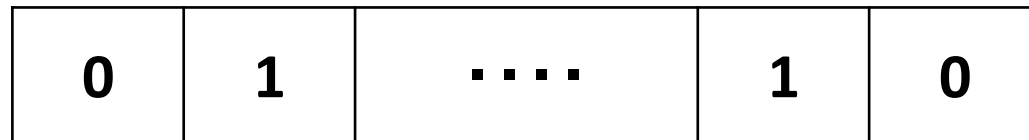
0: 正(+)
1: 負(-)

最上位
(MSB)

最下位
(LSB)

- 符号なしデータ(例: 0, 1, 2, 3, ...)

正数



最上位
(MSB)

最下位
(LSB)

● 符号ありデータの表現方法

■ 絶対値表現



正の「0」(ゼロ)と負の「0」(ゼロ)が存在する

■ 補数表現

10進数の「-5」を、8ビットの2の補数で表現する

<2の補数の作り方・その1>

① 10進数「5」を2進数に変換する

101

② 8ビットで表現する場合、**空のビットに0(ゼロ)**を入れる

00000101

③ **基準となる数(9ビットの1番小さい数)**から引き算する

100000000

- 00000101

11111011 ← -5

<2の補数の作り方・その2>

① 10進数「5」を2進数に変換する

101

② 8ビットで表現する場合、**空のビットに0(ゼロ)**を入れる

00000101

③ すべてのビットを、反転する

11111010

④ 1を足す

11111010

+ **00000001**

11111011 ← -5

2の補数は、10進数で-128~+127(2進数8ビットの場合)を表現できる

コンピュータで扱う負の数は、2の補数で表す

10進数	2の補数
+127	01111111
:	:
+3	00000011
+2	00000010
+1	00000001
0	00000000
-1	11111111
-2	11111110
-3	11111101
:	:
-128	10000000

「0」(ゼロ)
は1つ

【過去問題】負の整数を表現する代表的な方法として、次の3種類がある

- a 1の補数による表現
- b 2の補数による表現
- c 絶対値に符号をつけた表現(先頭ビットが0のとき正、1のとき負)

4ビット"1101"を上を示すa~cの方法で表現した場合、値が小さい順になるように3つの方法を並べたものはどれか

ア a,c,b イ b,a,c ウ b,c,a **エ c,b,a**

4ビット"1101"をa,b,c方法で表現すると、それぞれ以下のようになる

aでは $1101 \rightarrow 0010 \rightarrow 2$ (10進数) 従って $1101 \rightarrow -2$ (負の10進数)

bでは $1101 \rightarrow 0010 \rightarrow 0011 \rightarrow 3$ (10進数) 従って $1101 \rightarrow -3$ (負の10進数)

cでは $1101 \rightarrow -5$ (負の10進数)

$-5 < -3 < -2$ 従って **エ c,b,a**

2の補数は必ず計算できるように！

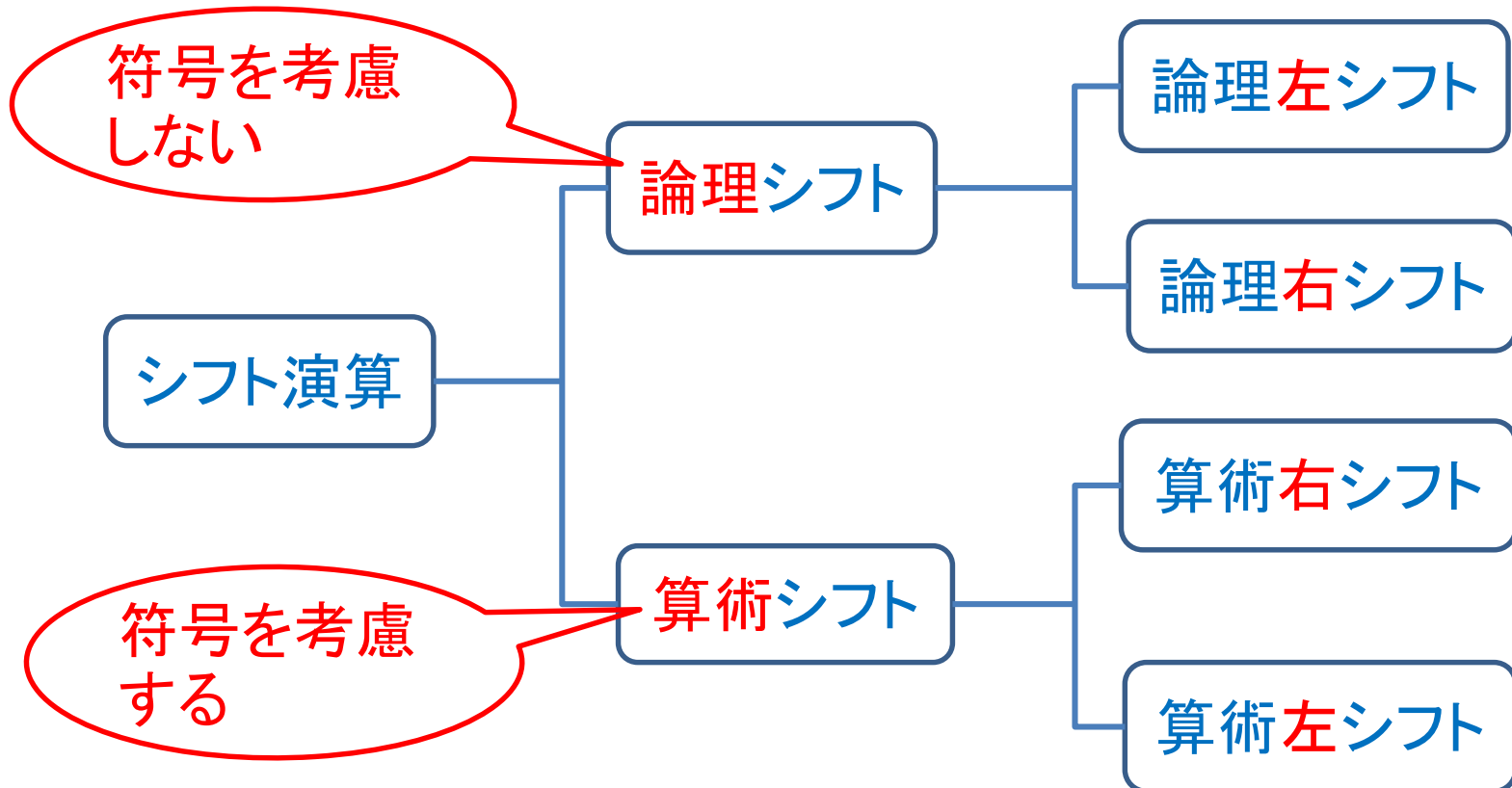


3. シフト演算

- シフト演算の体系
- 論理左シフト
- 論理右シフト
- 算術左シフト
- 算術右シフト
- シフト演算と加算の演算

● シフト演算の体系

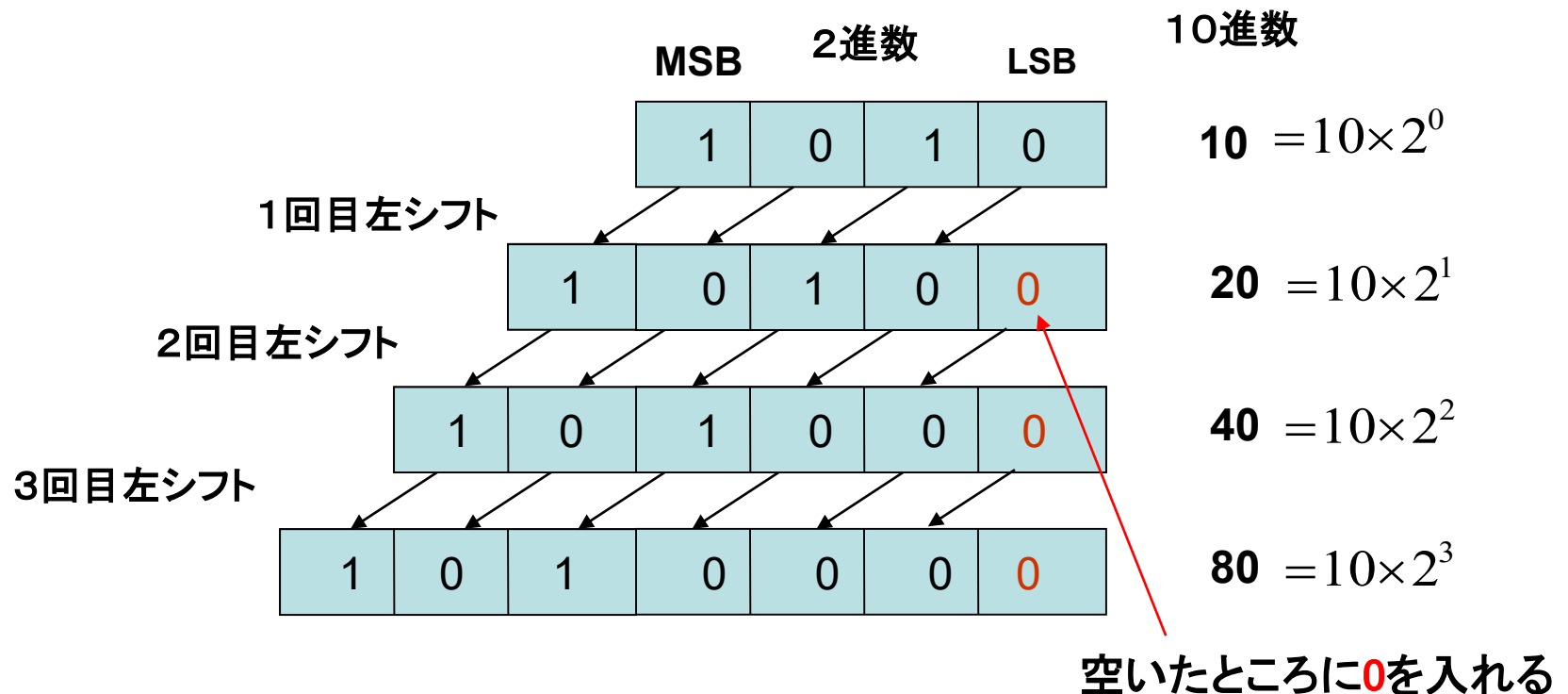
2進数で掛け算や割り算をするための演算



● 論理左シフト

■ 論理左シフト(符号を考慮しない)

左へ1回シフトすると、値が2倍になる



■ 論理右シフト(符号を考慮しない)

右へ1回シフトすると、値が1/2倍になる

10進数

$$8 = 8 \times 2^0$$

$$4 = 8 \times 2^{-1}$$

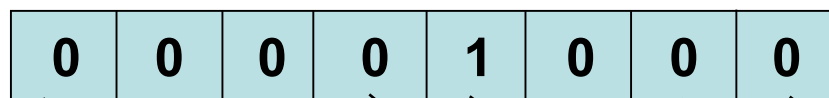
$$2 = 8 \times 2^{-2}$$

$$1 = 8 \times 2^{-3}$$

2進数

MSB

LSB

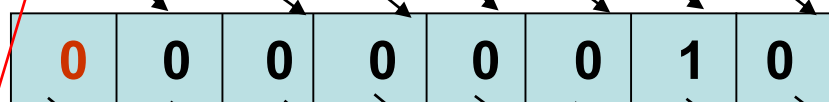


1回目右シフト



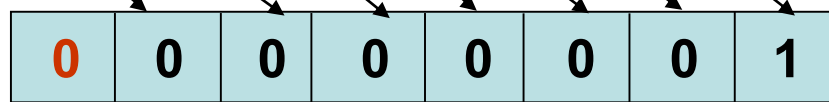
捨てる

2回目右シフト



捨てる

3回目右シフト



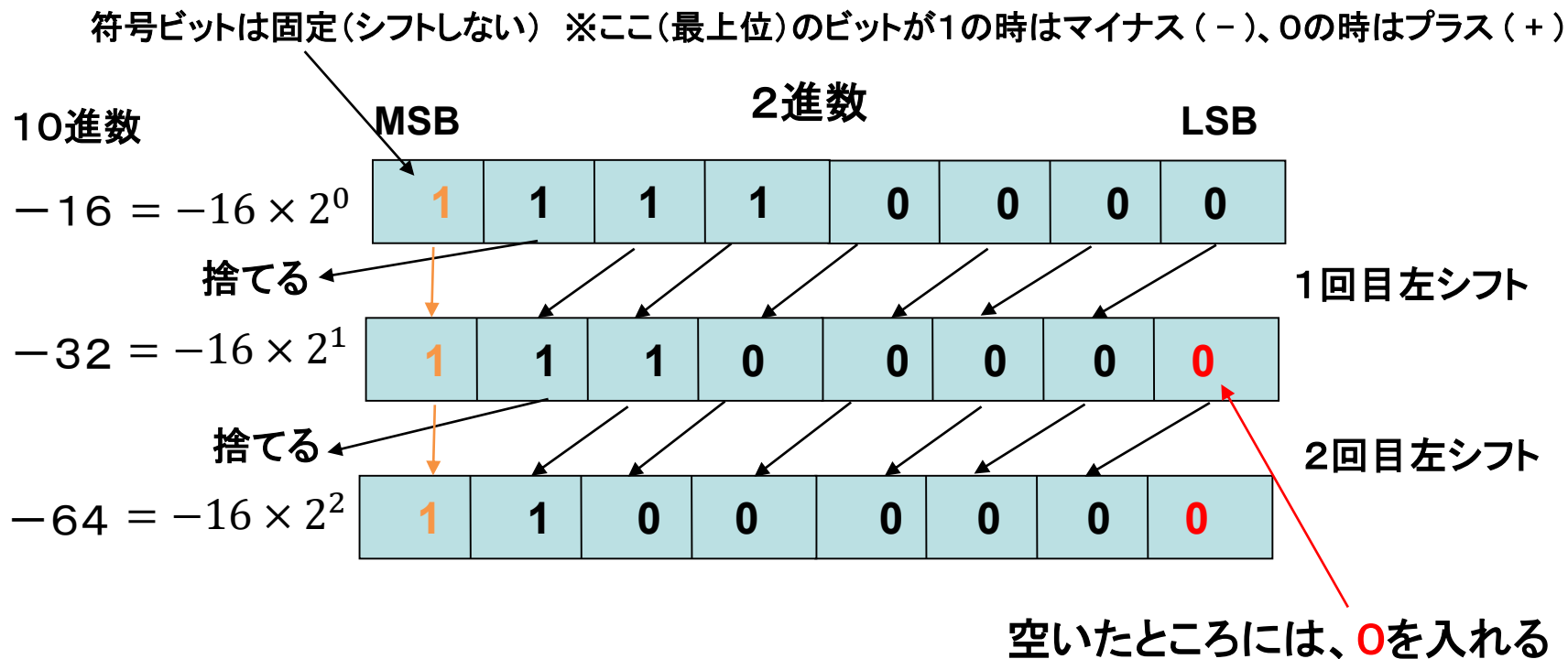
捨てる

空いたところに0を入れる

●算術シフト

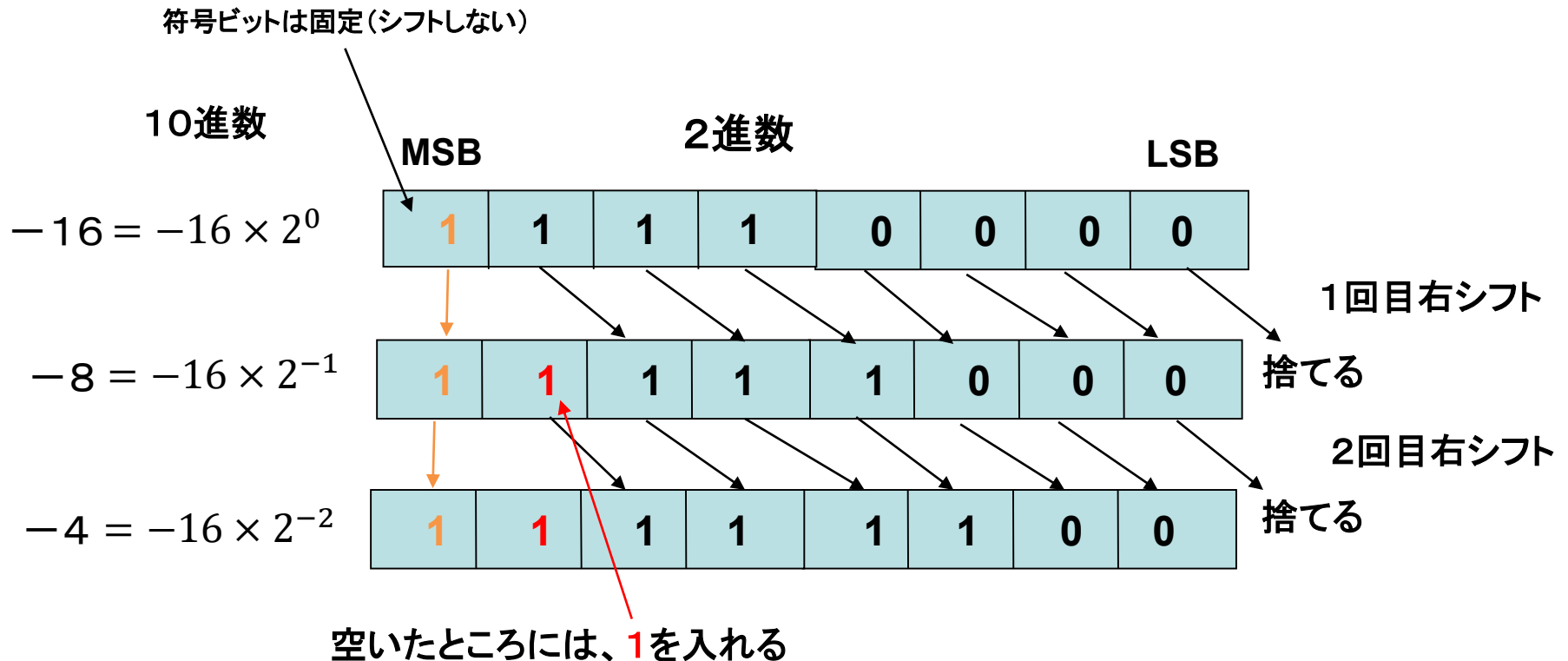
■ 算術左シフト(符号を考慮する)

左に1回シフトすると、値が2倍になる



■ 算術右シフト(符号を考慮する)

右に1回シフトすると、値が1/2倍になる



● シフト演算と加算の演算

2^n や 2^{-n} の掛け算や割り算以外の演算

<掛け算>

$$x \times 7 = x \times (2^2 + 2^1 + 2^0) = (x \times 2^2) + (x \times 2^1) + (x \times 2^0)$$

左シフト2回 左シフト1回 シフトしない

2^n ではない

【過去問題】

数値を2進数で表すレジスタがある。このレジスタに格納されている正の整数 x を10倍する操作はどれか。ここで、桁あふれは、起こらないものとする。

- ア x を2ビット左にシフトした値に x を加算し、
更に1ビット左にシフトする。 20倍
- イ x を2ビット左にシフトした値に x を加算し、
更に2ビット左にシフトする。 12倍
- ウ x を3ビット左にシフトした値と、 x を2ビット
左にシフトした値を加算する。 18倍
- エ x を3ビット左にシフトした値に x を加算し、
更に1ビット左にシフトする。 18倍

$$\text{ア } (2^2 + 1) \times 2^1 = (4 + 1) \times 2 = 5 \times 2 = 10\text{倍}$$

左2ビット
シフト

加算

左1ビット
シフト

様々なシフト演算を、
できるように！



4. 小数の表現

- 固定小数点表示と浮動小数点表示
- 浮動小数点数の書き表し方
- 浮動小数点表示の正規化

● 固定小数点表示と浮動小数点表示

固定小数点表示

0.00000015

浮動小数点表示

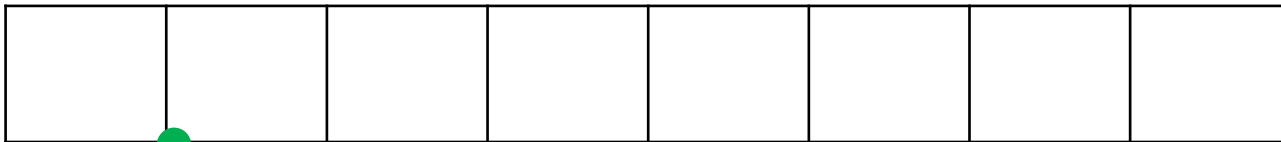
0.15 × 10⁻⁶

仮数

指数

基数(コンピュータでは、通常2)

<固定小数点表示の一例>

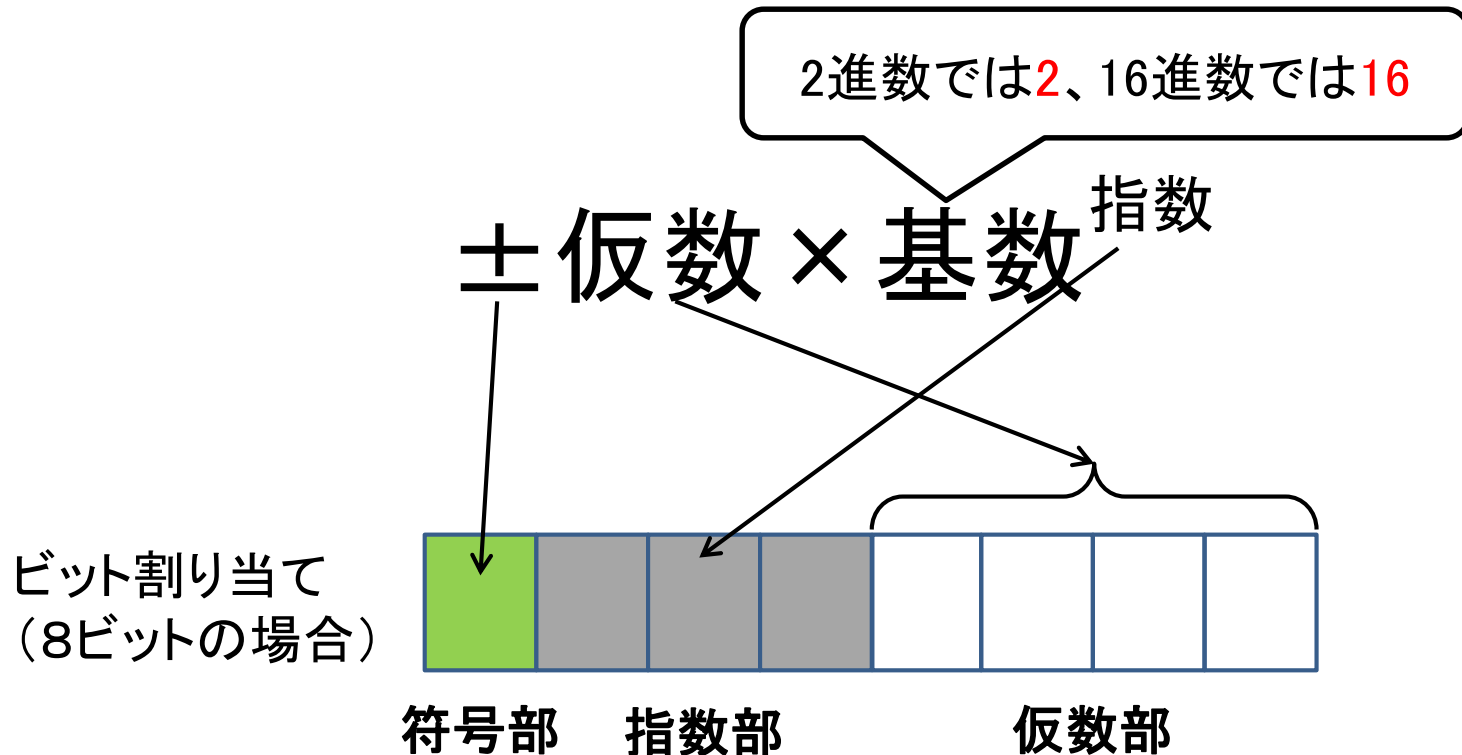


小数点の位置
を固定する

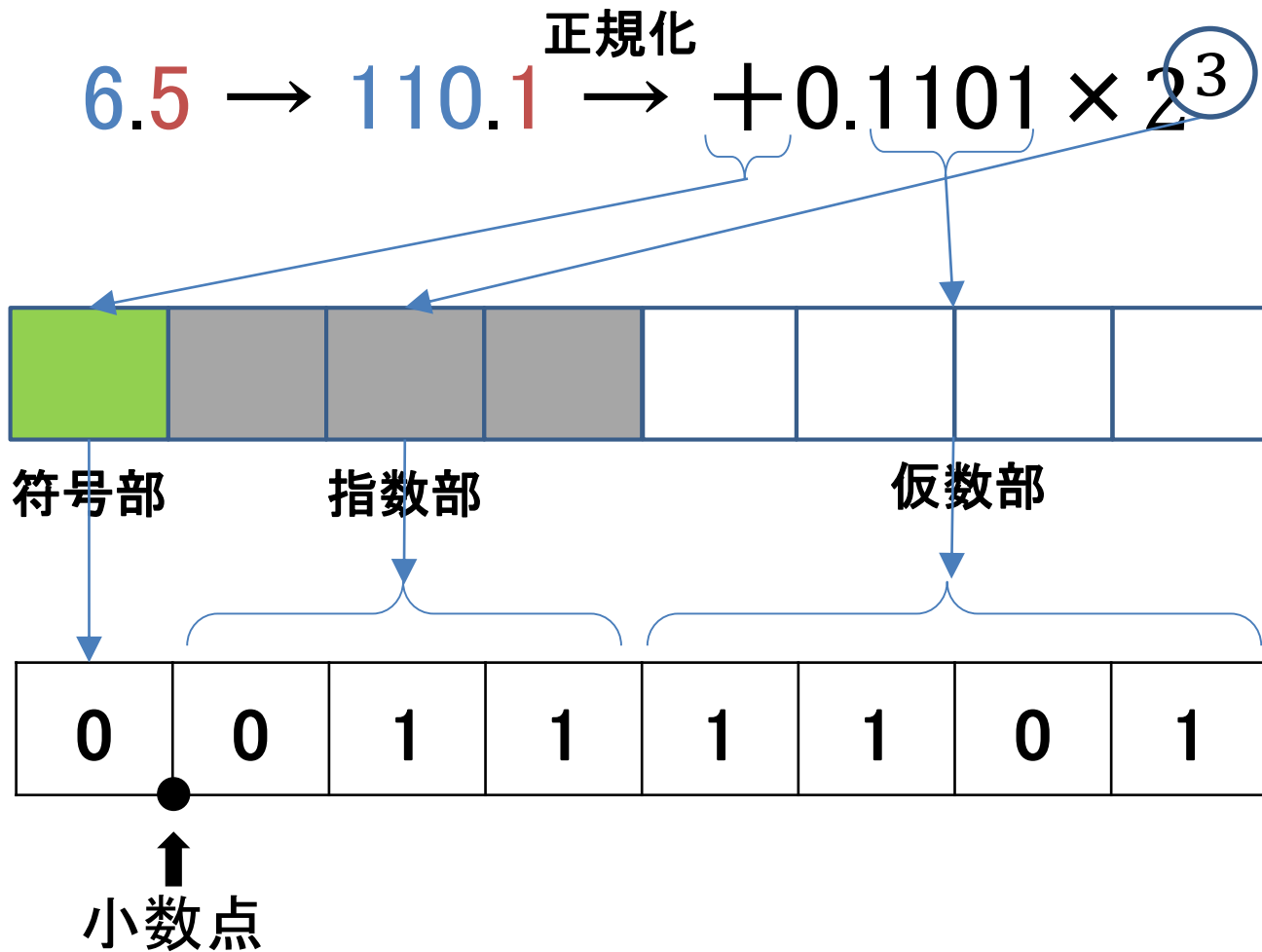
- ✓ 固定小数点は、0の数が増え(減)れば桁も増え(減)る
- ✓ 浮動小数点は、指数によって小数点の位置が動くので、数値を広範囲に少ない桁数で表現できる(コンピュータ向き)

● 浮動小数点数の書き表し方

小数点の位置を固定しないで、**符号部**、**指数部**、**仮数部**の各値で、小数点数を表す



● 浮動小数点数



● 浮動小数点の正規化

正規化とは

$$0.0015 \times 10^{-4}$$

小数点の右側の桁に0以外の数字がくるように指数を決める

$$0.15 \times 10^{-6}$$

■ 仮数部に、有効な桁数を多く入れられる

限られたビット数で有効数字の桁数を最大限に利用できる

$$0.12345 \times 10^{-4} = 0.012345 \times 10^{-3}$$

0.

1	2	3	4	5
---	---	---	---	---

0.

0	1	2	3	4
---	---	---	---	---

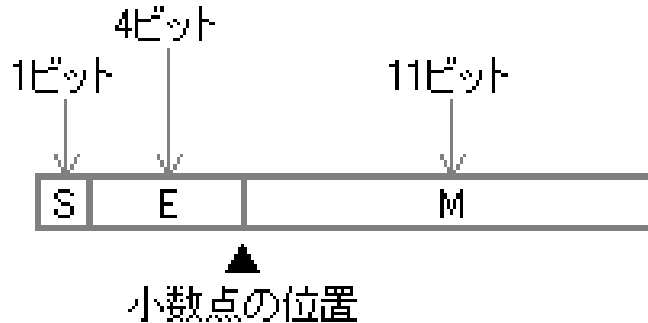
 5

入らない

5桁

【過去問題】

数値を16ビットの浮動小数点で、図に示す形式で表す。10進数0.375を正規化した表現はどれか。ここでの正規化は、仮数部の最上位けたが0にならないように指数部と仮数部を調節する操作である。



S : 仮数部の符号 (0 : 正、1 : 負)

E : 指数部 (2 を基数とし、負数は2の補数で表現)

M : 仮数部 (2進数 絶対値表示)

- | | | | | | |
|---|---|--|------|--|-------------|
| ア | 0 | | 0001 | | 11000000000 |
| イ | 0 | | 1001 | | 11000000000 |
| ウ | 0 | | 1111 | | 11000000000 |
| エ | 1 | | 0001 | | 11000000000 |

10進数の 0.375は、

$$\begin{aligned} 0.375 &= (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3}) \\ &= 0.011_{(2)} \\ &= 0.11_{(2)} \times 2^{-1} \end{aligned}$$

と表現できる。つまり、仮数部の情報に相当する $0.11_{(2)}$ という値を、指数部の情報に相当する 2^{-1} 倍したものとして考えることができる

符号部は、正の数なので、0 となる

仮数部は、11ビットで表現するので、11000000000 となる

指数部は、2 を基数とした2の補数なので、-1 の2の補数を4ビットで表現したものになる

-1 は負の数なので、4ビットの最大数からちょうど桁上がりした5桁の数 $10000_{(2)}$ から $0001_{(2)}$ を引く

$$10000_{(2)} - 0001_{(2)} = 1111_{(2)}$$

よって、ウが正解

正規化が正しくできるように！



5. 誤差

- 誤差とは
- 丸め誤差
- 打切り誤差
- 情報落ち
- 桁落ち
- 桁あふれ

● 誤差とは

実際の値を、2進数で表すことで発生する値の差

● 丸め誤差

限られた桁数の範囲で数値を表現するときに、四捨五入、切り上げ、切り捨てなどの操作で起こる誤差

10進数:0.1 → 2進数:0.000110011...
小数点以下8桁

入らない

● 打ち切り誤差

計算を途中で打ち切るために起こる誤差

円周率3.1415...などのような数値に対して、あらかじめ決めた数値(3.14)で打ち切る

● 情報落ち

絶対値が大きな値と絶対値が小さな値の足し算や引き算したとき、小さな値の情報が無視されて、結果に反映されないために発生する誤差

仮数部が4桁の計算では、

$$0.1234 \times 10^4 + 0.4321 \times 10^{-4} = 0.1234 \times 10^4 + 0.000000004321 \times 10^4$$

$$= 0.123400004321 \times 10^4$$

指数を同じにして計算する

4桁の仮数部に入らない

● 桁落ち

絶対数が等しい2つの数値で引き算をした時、有効桁数が減少するために起こる誤差

$$1.234 \times 10^{-1} - 1.233 \times 10^{-1} = 0.100 \times 10^{-3}$$

有効桁数が4桁 (1.234)
有効桁数が1桁 (0.100)
信頼性がない値 (0.100)

● 桁あふれ

計算結果が、コンピュータで扱うことができるビット数を超えたことにより起こる誤差

ビット数の最大値を超えた場合 → オーバーフロー

ビット数の最小値を下回った場合 → アンダーフロー

- 各誤差の性質を理解しよう！
- 丸め誤差、打ち切り誤差、情報落ち誤差は出題頻度が高いので集中して学習しよう！

【過去問題】

浮動小数点形式で表現された数値の演算結果における丸め誤差の説明はどれか

- ア 演算結果がコンピュータの扱える最大値を超えることによって生じる誤差である
- イ 数表現の桁数に限度があるので、最下位桁より小さい部分について四捨五入や切り上げ、切り捨てを行うことによって生じる誤差である
- ウ 乗除算において、指数部が小さい方の数値の仮数部の下位部分が失われることによって生じる誤差である
- エ 絶対値がほぼ等しい数値の加減算において、上位の有効数字が失われることによって生じる誤差である

ア 桁あふれ(オーバーフロー) エ 桁落ち



6. 論理演算と論理回路

- 論理演算とは
- 論理回路と表記法
- 3つの基本回路
- 論理回路の組合せ
- ドモルガンの法則

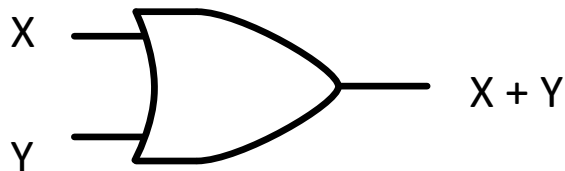
● 論理演算とは

情報の基本である0(偽)と1(真)の2値で行う演算

● 論理回路と表記法

MIL記号

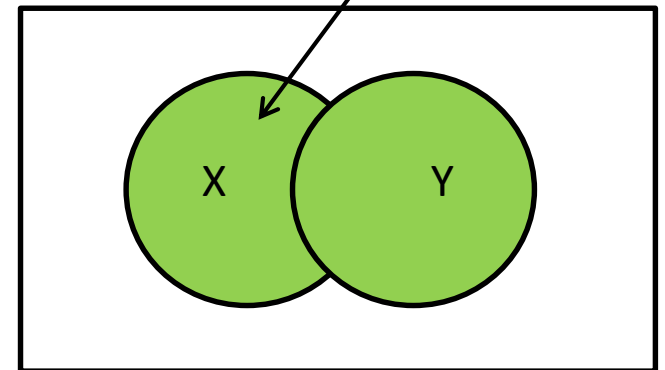
論理回路(論理記号)



真理値表

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

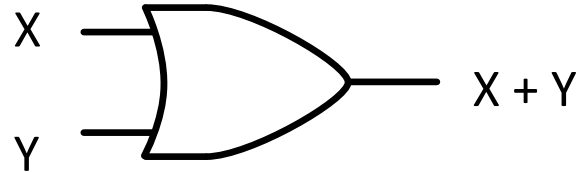
対象領域



ベン図

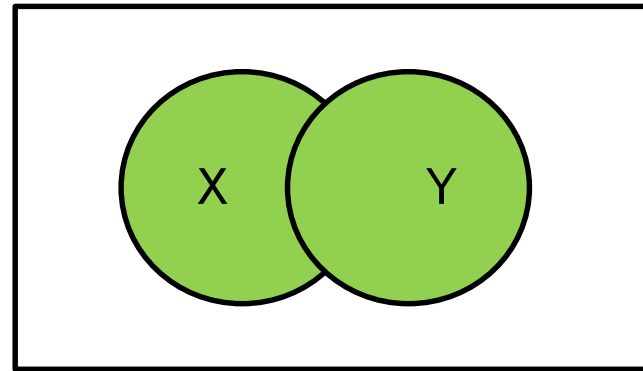
● 3つの基本回路

■ 論理和(OR)



真理値表

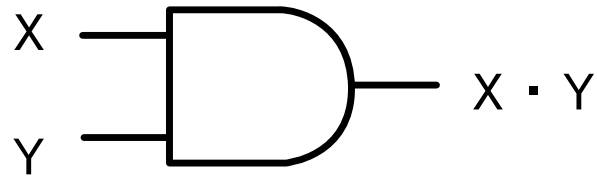
X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1



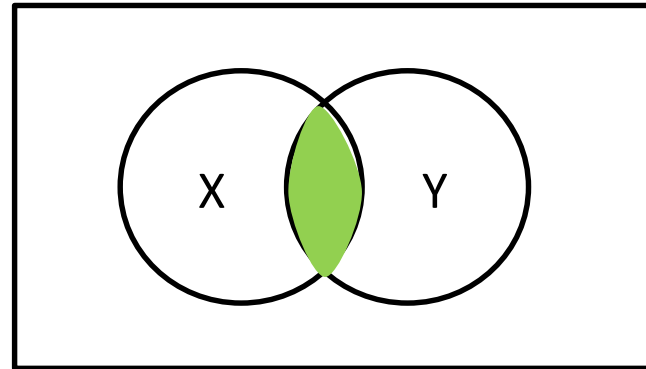
ベン図

XかYの**どちらか一方が1**ならば、1になる

■ 論理積(AND)

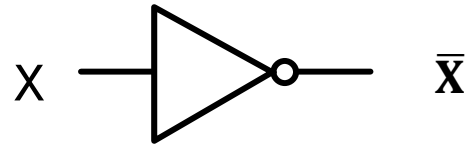


X	Y	X·Y
0	0	0
0	1	0
1	0	0
1	1	1

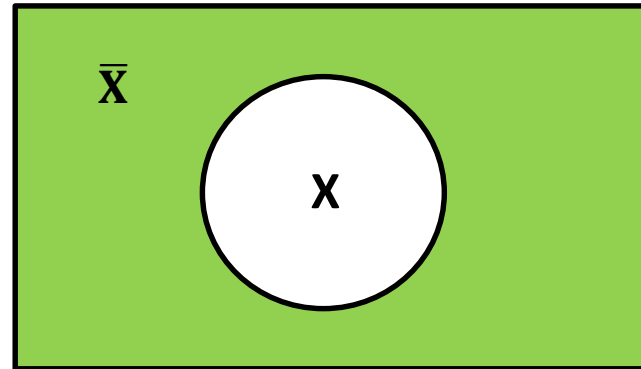


XとYが**どちらも1**ならば、1になる

■ 否定(NOT)



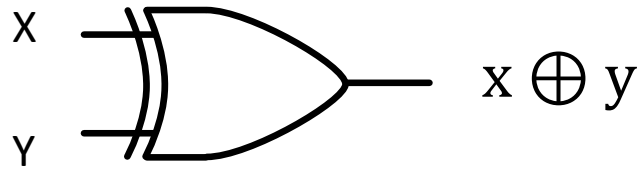
X	\bar{X}
0	1
1	0



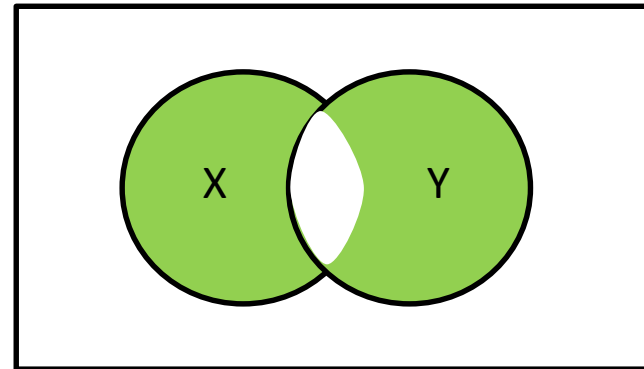
Xが0ならば、1になる

● 基本回路の組合せ

■ 排他的論理和(XOR)

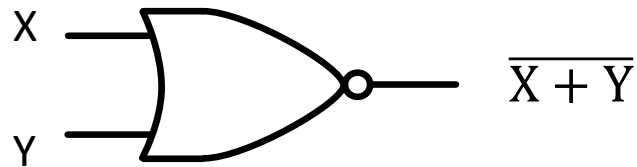


X	Y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

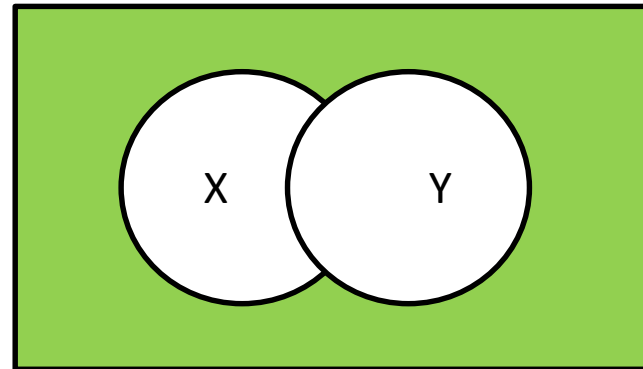


XとYが異なるときは1、同じときは0となる

■ 否定論理和(NOR:NOT+OR)

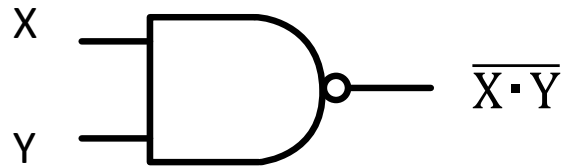


X	Y	$\overline{X + Y}$
0	0	1
0	1	0
1	0	0
1	1	0

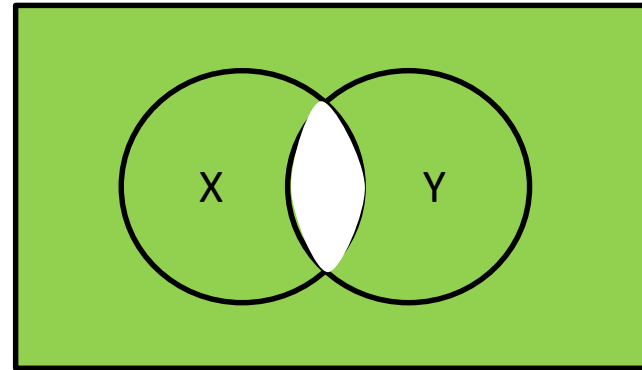


XとYの**どちらか一方でも1ならば、0になる**

■ 否定論理積(NAND:NOT+AND)



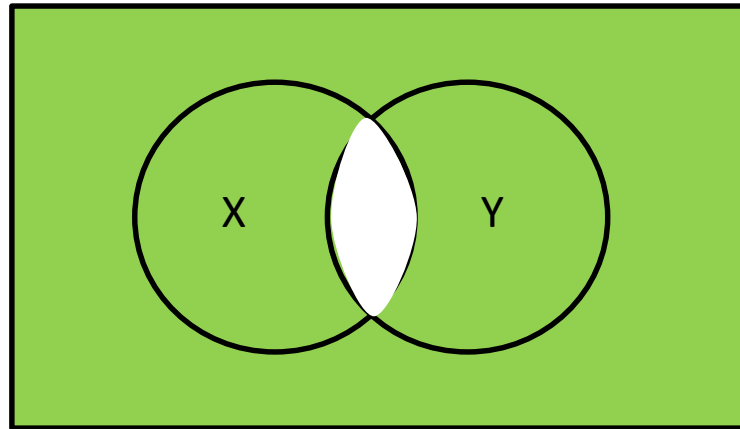
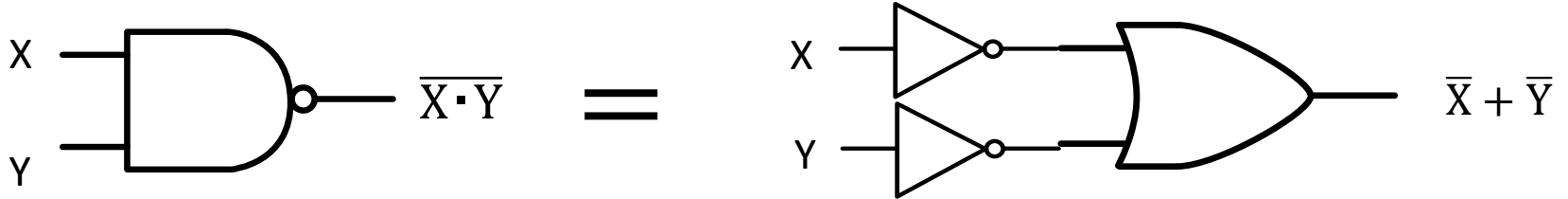
X	Y	$\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0



XとYが**どちらも1ならば、0**になる

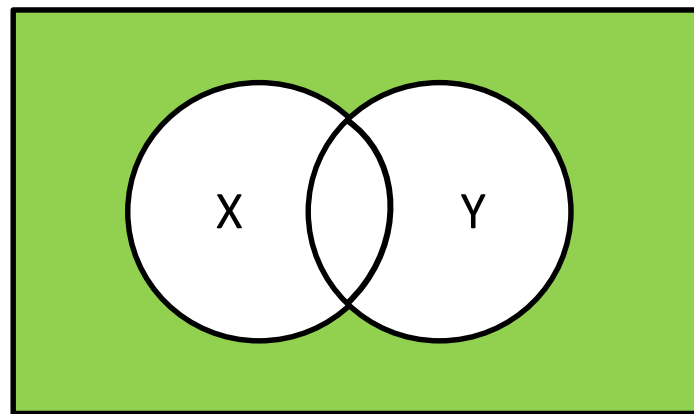
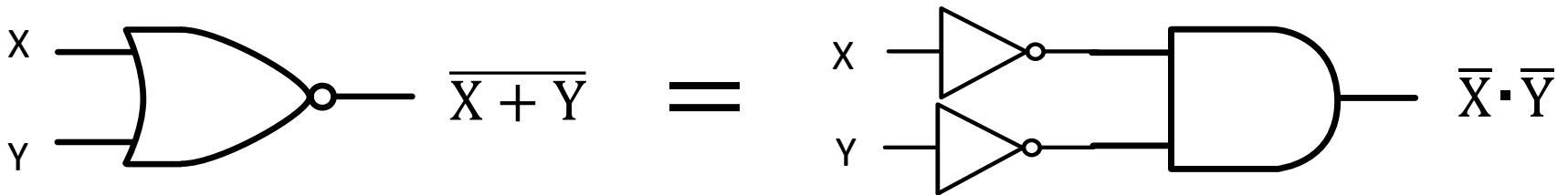
● ドモルガンの法則

- $\overline{X \cdot Y} = \bar{X} + \bar{Y}$ 論理積の全体の否定は、**それぞれの否定の論理和**に等しい



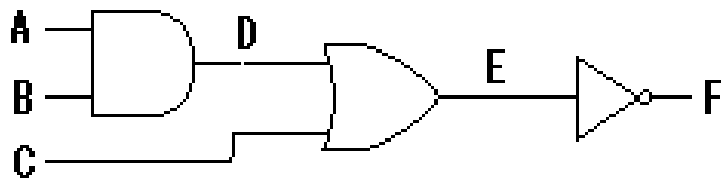
■ $\overline{X + Y} = \bar{X} \cdot \bar{Y}$

論理和の全体の否定は、
それぞれの否定の論理積
に等しい

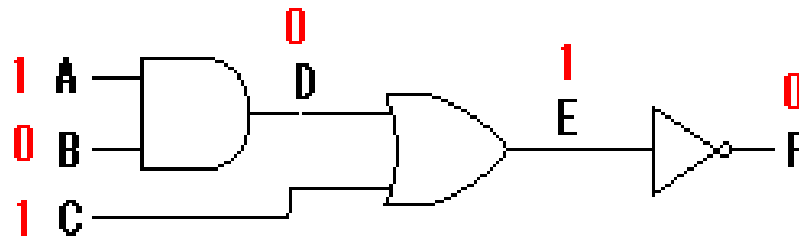


【過去問題】

下に示す論理回路において、 $A=1, B=0, C=1$ のとき、 D, E, F の値の適切な組み合わせはどれか



	D	E	F
ア	0	1	0
イ	0	1	1
ウ	1	0	1
エ	1	1	0



- 各論理回路の動作をしっかりと理解しよう！
- 論理回路の組み合わせたときの動作をしっかりと理解しよう！

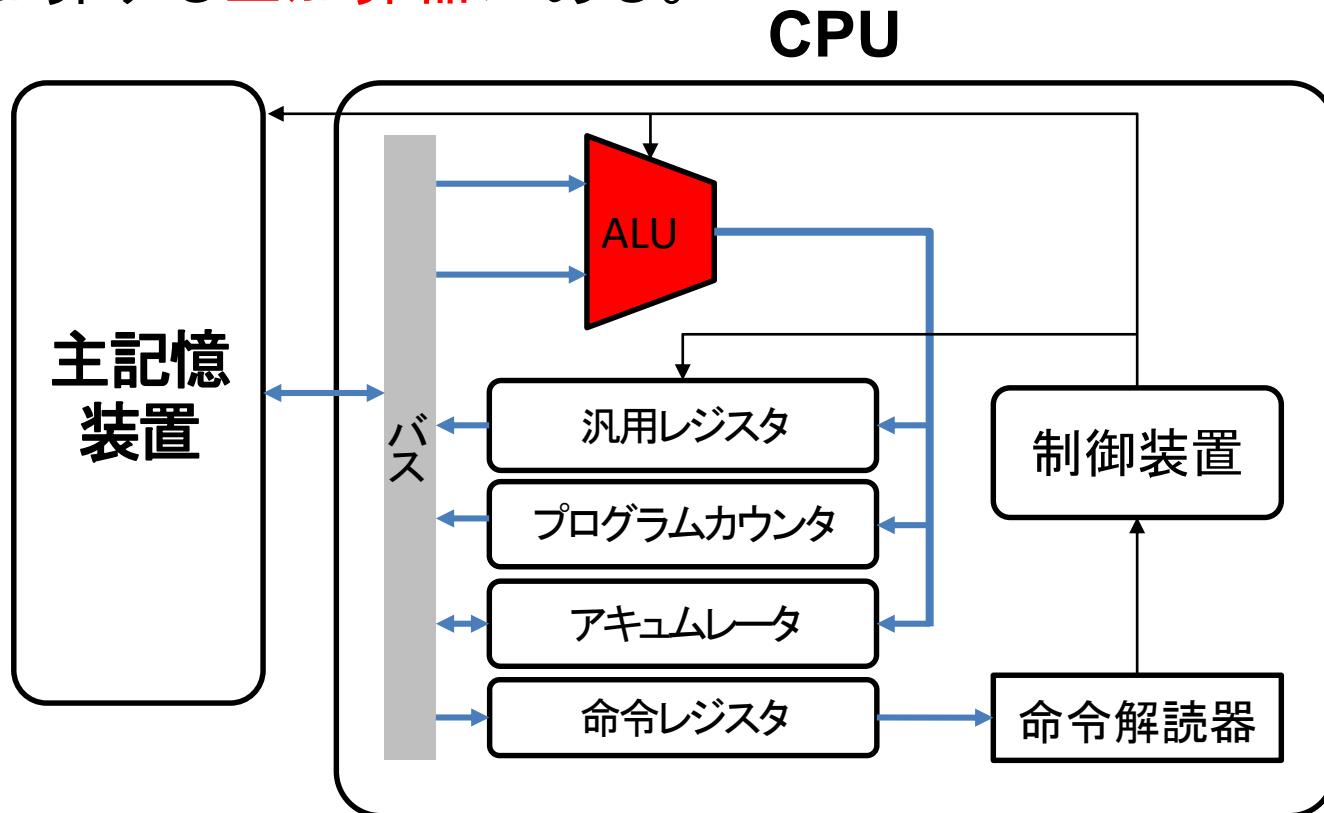


7. 半加算器と全加算器

- 加算器とは
- 半加算器
- 全加算器

● 加算器とは

コンピュータが、計算処理するための論理演算に使用する回路として、加算器がある。CPU内のALUによって、実現している。加算器には、1桁を加算する半加算器と2桁以上を加算する全加算器がある。



● 半加算器(Half Adder)

■ 2進数1桁の足し算(加算)

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

桁上がり

$$\begin{array}{r} X \\ + Y \\ \hline CS \end{array}$$

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

桁上がり

● 半加算器の動作を回路で表す

入力		出力	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

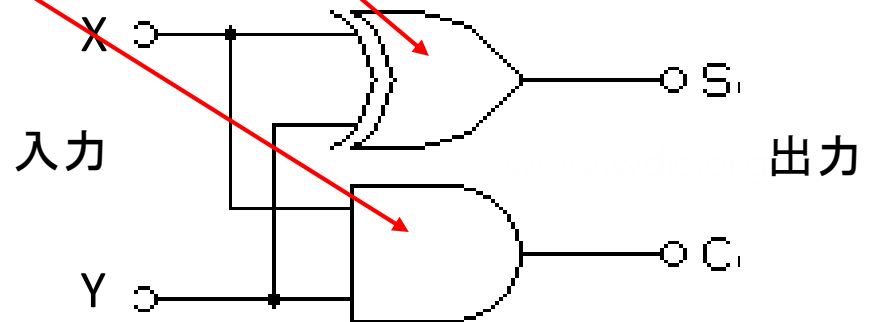
$$C = A \cdot B$$

$$S = \bar{A} \cdot B + A \cdot \bar{B}$$

論理積(AND)

排他的論理和(XOR)

半加算器(1桁専用加算器)

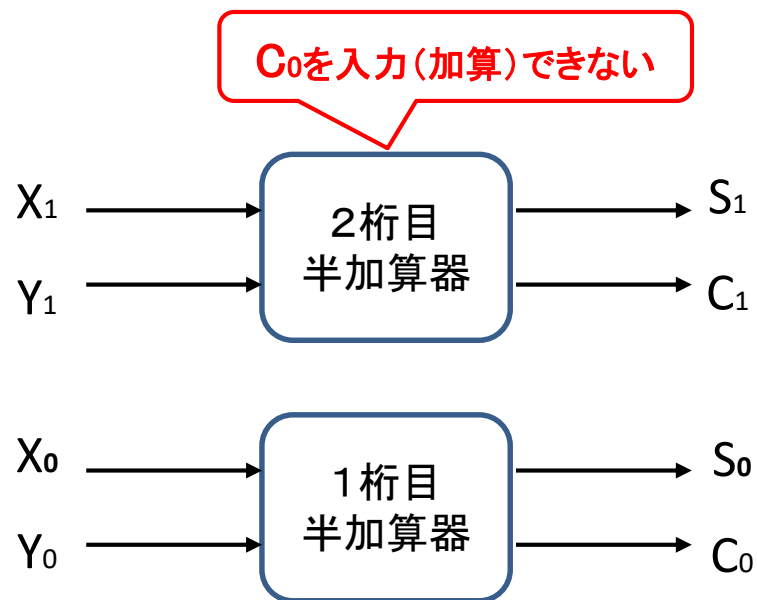


● 全加算器(Full Adder)

■ 2進数2桁の加算式

$$\begin{array}{r} C_0 \\ X_1 X_0 \\ + Y_1 Y_0 \\ \hline C_1 S_1 S_0 \end{array}$$

■ 2進数2桁の加算回路？



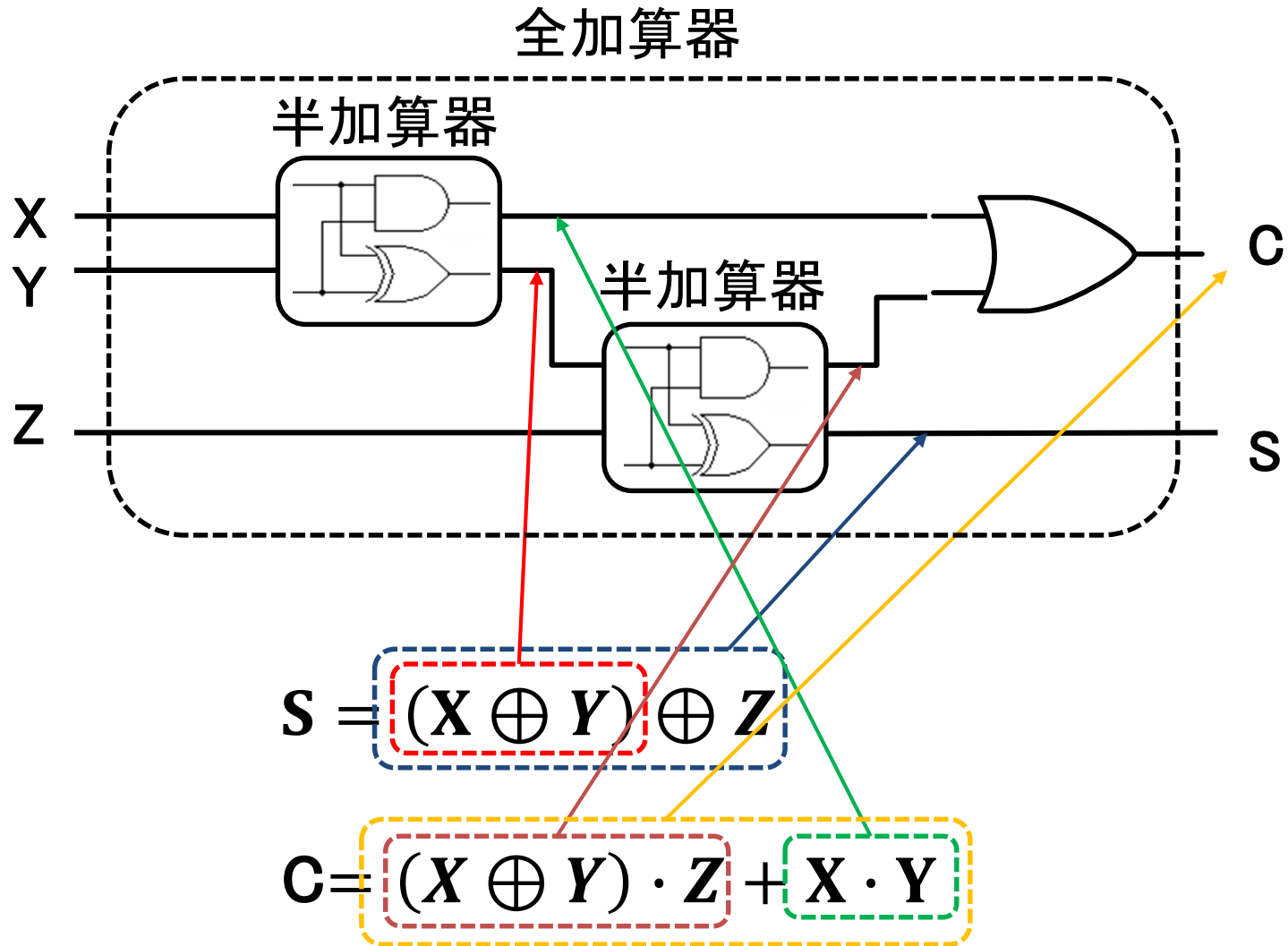
■ 全加算器 (2桁以上の加算に使える)



ここから、下位の桁Cを入力する

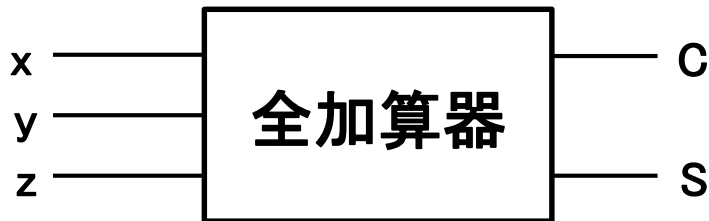
入力			出力	
X	Y	Z	C	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

■ 半加算器で全加算器を作る



【過去問題】

下に示す全加算器において、 $x=1, y=0, z=1$ を入力したとき、出力 c (桁上げ), s (和) の値はどれか



	C	S
ア	0	0
イ	0	1
ウ	1	0
エ	1	1

$$\begin{array}{r} 1 \quad z \\ 1 \quad y \\ + 0 \quad x \\ \hline \end{array}$$

1 0

C S

半加算器と全加算器の動作をしっかりと理解しよう！



8. 文字データの表現

- 文字コードの種類
- フォントの種類

● 文字コードの種類

コンピュータは、文字や記号などのデータを、コンピュータ内部(ハードウェアやOS)で扱うことができる形に変えて用いる

例えば、A→1000001(ASCIIコード)など

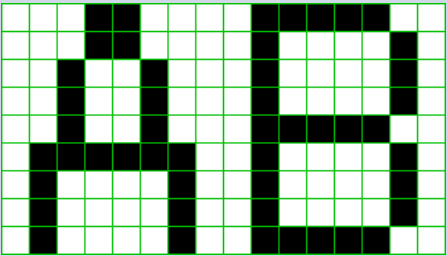
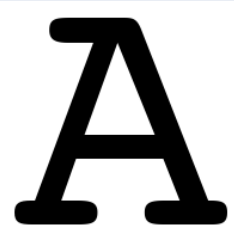
内容を覚えよう

種類	特徴	日本語
ASIIコード	米国規格協会が制定 7bit(文字コード)+1bit(誤り訂正コード) 半角英数字	非対応
シフトJIS コード	日本で多くのPCで使用 標準使用	日本語Windows標 対応
EUC	UNIXで使用	対応
JISコード	電子メール標準使用	対応
Unicode	ISOで規格 多国籍言語	対応

● フォントの種類

文字をモニターに表示するときやプリンタで印刷するときには、人間が分かるように文字の形に変換する必要がある。このとき、文字コードに対応した書体データを**フォント**と呼ぶ。

内容を覚えよう

種類	特徴	表示
ビットマップ・フォント	<ul style="list-style-type: none">• 点の集合体として表現する• 拡大や縮小すると形が崩れる• データ量が少ない	
アウトライン・フォント	<ul style="list-style-type: none">• 文字の輪郭を線で表現する• 拡大や縮小しても形が崩れない• データ量が多い• 表示するまでに時間がかかる	

【過去問題】

英字の大文字(A-Z)と数字(0-9)を同一のビット数で一意にコード化するには、少なくとも何ビット必要か

ア 5bit **イ** 6bit ウ 7bit エ 8bit

英字の大文字(A-Z)は26種類、数字(0-9)は10種類あるので、合計36種類の文字種になる。従って、 $36 = 2^6 \Rightarrow 6\text{bit}$ が必要。



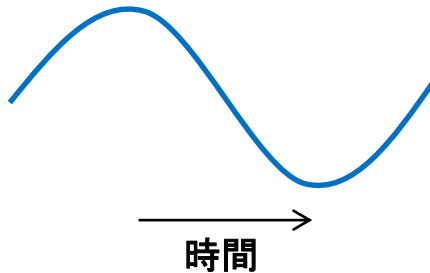
9. 音声や動画などの デジタルデータの表現

- アナログとデジタル
- 音声データのデジタル化
- コンピュータアニメーションの技法

● アナログとデジタル

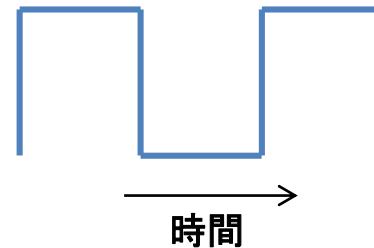
アナログ(データ)

切れ目なく連続した値が続く



デジタル(データ)

途切れ途切れの値が続く

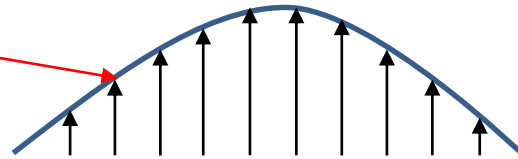


<https://youtu.be/uluJTWS2uvY>

● 音声データのデジタル化

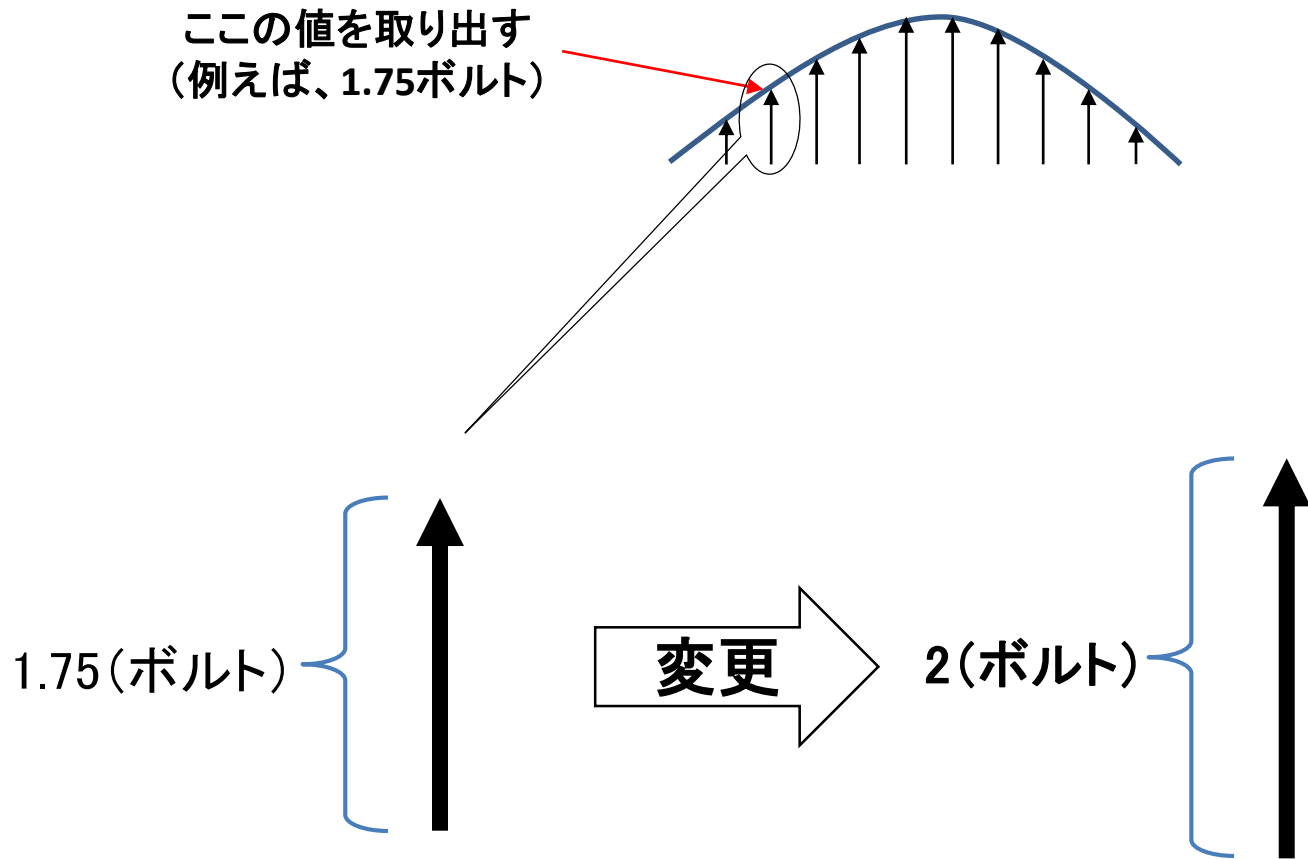
- アナログ信号をデジタル信号にするには、
3つの過程(①標本化、②量子化、③符号化)がある
- ①標本化(Sampling):一定間隔で値を取り出す
※取り出す間隔(等間隔)は、標本化(サンプリング)周波数という信号で決まる

ここの値を取り出す
(例えば、1.75ボルト)



②量子化 (Quantization) : 標本化で取り出した値を、
あらかじめ決めた値 (量子化値) にする

※量子化値が多いほど、元のアナログに近くなる



※量子値 (0, 1, ..., 15ボルト) の内で一番近い値に変更する

③符号化 (Coding) : 量子化した値を、0と1の符号にする

量子化値	符 号
0	0 0 0 0
1	0 0 0 1
:	:
15	1 1 1 1

量子値が0~15ならば、符号は4ビット必要

● コンピュータアニメーションの技法

- モーフィング (morphing)

ある画像が滑らかに次の画像へと変化するよう
に、複数の画像の中間の画像を生成して連続的に
変化させる技法

- モーションキャプチャ (motion capture)

人間などの動きをデジタルデータ化して取り込
むための技法。人体の関節などにセンサーを取
り付けることで動きを記録する。

【過去問題】

アニメーションの作成過程で、センサやビデオカメラなどを用いて人間や動物の自然な動きを取り込む技法はどれか。

- ア キーフレーム法
- イ ピクセルシェーダ
- ウ モーションキャプチャ
- エ モーフィング

モーションキャプチャは、現実の人間や動物の関節などに装着されたマーカー(センサー)ごとの動きをデジタルデータとして連続的に記録する技術。得られたデータはスポーツ分野において身体の動きのデータ収集・分析や、CGアニメーションに実物のような動きを与えるために利用される。

令和元年度 秋期

基本情報処理技術者試験問題・解答(コンピュータで扱うデータ)

【問23】

コードから商品の内容が容易に分かるようにしたいとき、どのコード体系を選択するのが適切か。

- | | |
|----------------|---------|
| ア 区分コード | イ 桁別コード |
| ウ 表意コード | エ 連番コード |

表意コードは、ニモニックコード(Mnemonic Code)とも言い、値から対象のデータが容易に連想できる英数字・記号の組合せをコードとして割り当てる方式。他のコードよりも桁数が多いが、利用者が記憶しやすい利点があり、商品番号や製品の型番としてよく使用されている。

例)

国名：日本→JP, アメリカ→US

色名：黒→BK, 白→WH

プロジェクトマネージャ→PM など

平成31年度 春期

基本情報処理技術者試験問題・解答(コンピュータで扱うデータ)

【問題22】

二つの入力と一つの出力をもつ論理回路で、二つの入力A,Bがともに1のときだけ、出力Xが0になるものはどれか。



- ア AND回路
- イ NAND回路
- ウ OR回路
- エ XOR回路

入力1	入力2	出力
0	0	1
0	1	1
1	0	1
1	1	0

図 NAND回路の真理値表

【問題25】

音声のサンプリングを1秒間に11,000回行い、サンプリングした値をそれぞれ8ビットのデータとして記録する。このとき、 512×10^6 バイトの容量のフラッシュメモリに記録できる音声の長さは、最大何分か。

- ア 77
- イ 96
- ウ 775
- エ 969

1秒間に11,000回データを取得し、1つのデータを8ビットで記録するので、1分間のデータ量は、

$11,000 \text{回} \times 8 \text{ビット} \times 60 \text{秒} = 5,280,000 \text{ビット}$
ビット単位からバイト単位に変換すると、
 $5,280,000 \div 8 = 660 \text{kバイト} = 0.66 \text{Mバイト}$
と計算することができる。

フラッシュメモリの容量は「 $512 \times 10^6 = 512 \text{Mバイト}$ 」なので、記録可能な音声の長さは、
 $512 \text{Mバイト} \div 0.66 \text{Mバイト} \doteq 775 \text{分}$