情報科学

2019年　神戸大学　担当　片山　修

目次

[アナログとデジタル 1](#_Toc22648465)

[アナログデータ 1](#_Toc22648466)

[デジタルデータ 1](#_Toc22648467)

[標本化定理（サンプリング定理） 2](#_Toc22648468)

[周期と周波数 2](#_Toc22648469)

[エイリアシング 3](#_Toc22648470)

[コンピュータで扱われる数値 4](#_Toc22648471)

[2進数 4](#_Toc22648472)

[2進数から10進数へ 4](#_Toc22648473)

[10進数から2進数へ 4](#_Toc22648474)

[2進数の四則演算 5](#_Toc22648475)

[2進数と16進数 5](#_Toc22648476)

[「符号つき整数」と「符号なし整数」 6](#_Toc22648477)

[補数 7](#_Toc22648478)

[符号つき2進数を10進数に変換する 7](#_Toc22648479)

[キャリーオーバーとオーバーフロー 8](#_Toc22648480)

[2進数のシフト演算 9](#_Toc22648481)

[論理シフト 9](#_Toc22648482)

[算術シフト 10](#_Toc22648483)

[固定小数点数 11](#_Toc22648484)

[符号なし固定小数点数 11](#_Toc22648485)

[符号つき固定小数点数 11](#_Toc22648486)

[固定小数点数のデメリット 12](#_Toc22648487)

[浮動小数点数 12](#_Toc22648488)

[IEEE754形式 13](#_Toc22648489)

[単精度浮動小数点数 13](#_Toc22648490)

[倍精度浮動小数点数 14](#_Toc22648491)

[情報の単位 15](#_Toc22648492)

[情報量 15](#_Toc22648493)

[データの符号化 16](#_Toc22648494)

[文字コード 16](#_Toc22648495)

[画像データ 17](#_Toc22648496)

[解像度 18](#_Toc22648497)

[相対解像度 18](#_Toc22648498)

[絶対解像度 18](#_Toc22648499)

[デジタル画像のファイル形式 19](#_Toc22648500)

[データ圧縮 20](#_Toc22648501)

[可逆圧縮と非可逆圧縮 20](#_Toc22648502)

[ランレングス法 20](#_Toc22648503)

[Huffman法 20](#_Toc22648504)

[ハフマン符号化の手順 21](#_Toc22648505)

[論理演算 22](#_Toc22648506)

[ビット操作 23](#_Toc22648507)

[ビットマスク 23](#_Toc22648508)

[ビット反転 23](#_Toc22648509)

[ビット列のクリア 23](#_Toc22648510)

[加算器 24](#_Toc22648511)

[半加算器 24](#_Toc22648512)

[全加算器 24](#_Toc22648513)

[コンピュータの仕組み 26](#_Toc22648514)

[CPU 26](#_Toc22648515)

[CPUの基本動作 27](#_Toc22648516)

[CISCとRISC 28](#_Toc22648517)

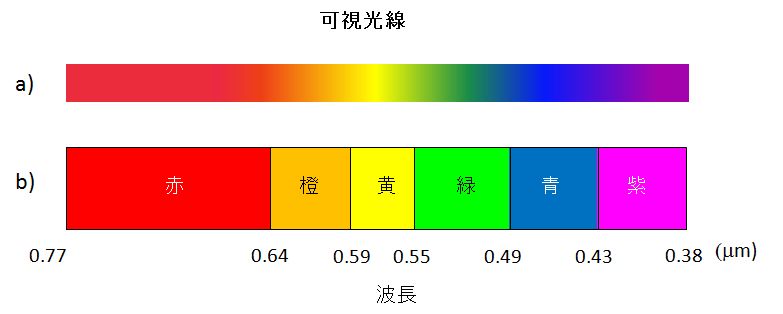
[CPUの性能 28](#_Toc22648518)

# アナログとデジタル

## アナログデータ

アナログとは時計の針の動きのように「連続した物理量」を表します。

音は空気の振動であり、映像などの色も光の波長によって表されるアナログであり、これらを電気の波である「アナログ信号」に変換して保存したものがアナログデータとなります。



a)連続的な色の変化　b)便宜的に波長情報を付加したもの

メリットは実際の音や映像に近いということがあげられますが、録音・録画機器、再生機器やモニタの性能に大きく依存します。また、雑音が載ったり時間やコピーによってデータが劣化したりするという問題があります。

## デジタルデータ

デジタルとは「連続した物理量を離散的に数値化」したものを表します。デジタル時計では1秒、2秒というように間が抜けたものになっています。

アナログデータをデジタルデータに変換することをアナログ/デジタル変換(A/D変換)言います。A/D変換は**標本化**(sampling)と**量子化**(quantization)という2段階の工程があります。音や映像は波で表わされるとしましたが、この波の形をある短い間隔（サンプリング周波数で決まる時間間隔）で区切って数値で表すことが**標本化**で、波の実際の小数点などが入った値(アナログ)を得ることができます。これを離散的な近似値に置き換えることを**量子化**といい、これがデジタルデータとなります。

標本化におけるサンプリング周波数を大きくし、量子化における近似値の取り方を細かくするほど元に忠実なデータが得られますが、データサイズは大きくなってしまいます。

赤い波形を一定間隔で区切った時

の値が標本化データ。

t

標本化データを近似した量子化データ。

t

デジタルデータのメリット：

* 雑音が載らない。
* パソコンで編集や加工が容易で扱いやすい。
* ネットワークを使った通信（やり取り）が容易。
* 劣化しない。

デジタルデータのデメリット：

* 誤差が生じる。
* 誤差を減じるためにはデータ量が増え、処理時間が長くなる。

# 標本化定理（サンプリング定理）

サンプリングとは・・・

データ を時間間隔でサンプリングするということは、(n=0,1,2,…)の時の値を取り出すということ。

時間間隔はどれくらい？

間隔が広いと、元の波形に戻らない

t

間隔が狭いと元の波形に忠実

t

## 周期と周波数

波が一度上下するのにかかる時間を周期[]と呼び、単位はsecとなります。1秒間に波が上下する回数を周波数[]と呼び、単位は「Hz(ヘルツ)」となります。例えば、半径1の円があり(1,0)の位置から反時計回りに周期で一周する場合のy座標の値は、 となります。ここでとなりこれを角速度といいます。

が大きくなれば、振動が細かくなることは容易に想像できます。

t

y

x

周期T

**サンプリング定理(標本化定理)**

が最大周波数をもつとき、間隔 (でサンプリングすれば、サンプリング後のデータから元のを復元できる。

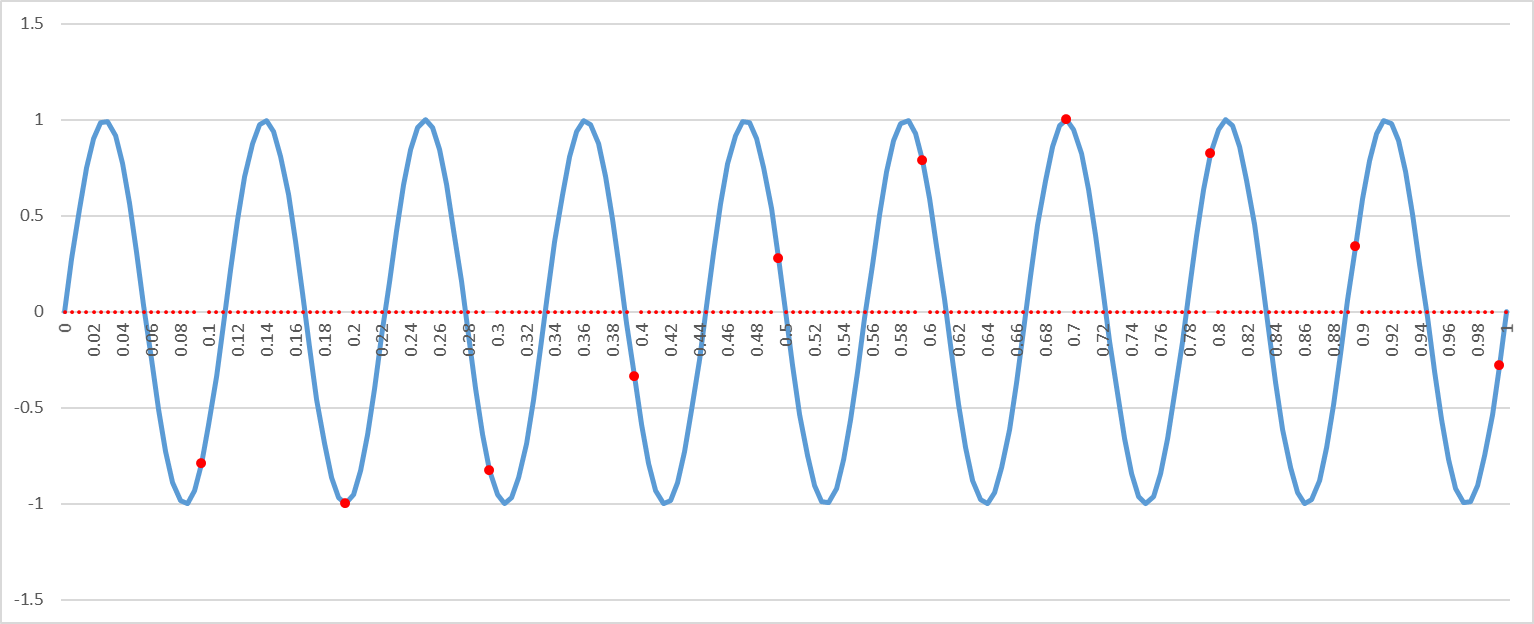
(証明は、行いません。感覚的には、細かい波はより細かく点を取り出せば、元の波が復元できるということ。)

## エイリアシング

大きい（高い）周波数の波を小さい（低い）周波数でサンプリングをおこなうと、元の周波数よりも低い波が現れることが有ります。下の図は、9Hzの波（青色）を10か所（10Hz）で観測したときの観測点を赤い丸とすると、1Hzの波形（緑色）が観測されます。このような現象をエイリアシングといいます。

これは、TV番組などで高速で回転するタイヤが、ゆっくり逆回転しているように見える現象で実感することができます。

参考URL: http://dsl4.eee.u-ryukyu.ac.jp/DOCS/DSP/aliasing.html



# コンピュータで扱われる数値

コンピュータで扱われる数値は次のように分類されます。

コンピュータの中では、数値はすべて２進数で表わされ、その桁数をビット(bit)と呼びます。

ビットの並びの一番左を最上位ビット：MSB(Most Significant Bit)と呼び、一番右を最下位ビット:LSB(Least Significant Bit)と呼びます。

# 2進数

ここでは、2進数のつくり方や計算方法について説明していきます。

まず、10進数は、0～9の10個の数字を使って、0,1,2…,9と数えていくと次は位が上がって10となり2桁では99まで表されます。2進数も同じ原理で0と1の2個の数字を使って、0,1,10,11,100,…というように数字が表されます。

## 2進数から10進数へ

10進数で123.4という数字は、と表すことができます。同様に2進数で1111011.101という数字は、

となります。(は10進数であることを表します。)

## 10進数から2進数へ

10進数の小数(A.B)で表される数値を整数部分(A)と小数部分(B)に分けて、それぞれ次の方法で2進数へ変換を行います。

整数部：

1. Aを2で割り、余りを2進数の整数部の最小桁とします。
2. 手順1のとき商が0であれば終了です。
3. 商が0でなければ、さらに商を2で割り、余りを2進数の整数部の上位桁とし、続けて手順２、３を繰り返します。  
   原理は、, , …, となり、

小数部分：

1. Bに2を掛け、結果の整数部分の値を2進数の小数部分の最初の桁とします。
2. 手順1の結果の小数部分が0であれば終了です。
3. 0でなければ、さらに結果の小数部分に２を掛け、結果の整数部分の値を2進数小数部分の次の下位の桁とし、手順２、３を繰り返します。

## 2進数の四則演算

2進数の四則演算は10進数と同様に行うことができます。

|  |  |
| --- | --- |
| 足し算  0+0=0 1+0=1 1+1=10(繰り上がり)  111  ＋101  1100 | 引き算  0-0=0 1-0=1 1-1=0 0-1=1  111  －101  10 |
| 掛け算  0x0=0 1x0=0 1x1=1  111  ×101  111  000  111  100011 | 割り算  101  101 11101  101  1001  101  100 |

## 2進数と16進数

16進数では、0,1,2,…,9,A,B,C,D,E,Fと16種類の文字を使って表現します。

8bitの2進数と16進数の関係は、次の通りです。

110 = 0000 00012 = 0116

910 = 0000 10012 = 0916

1010 = 0000 10112 = 0A16

1510 = 0000 11112 = 0F16

1610 = 0001 00002 = 1016

20010 = 1100 10002 = C816

23510 = 1110 10112 = EB16

25510 = 1111 11112 = FF16

変換ルールは次のようになっています。

2進数を4bitに分ける。それぞれを単独で10進数に変換する。さらに16進数に変換する。

2桁の16進数とする。

# 「符号つき整数」と「符号なし整数」

「符号なし(unsigned)」は０と正の整数だけを表現し、n桁の2進数では0～2n-1の数字が扱われます。「符号つき(signed)」では、正と負の符号を持つ数値を表現します。たとえば4桁の2進数では、最上位が正負を表していて、０の場合は「＋」、1の場合は「－」となります。残りの3桁で符号なし2進数を表現します。このように3桁の符号なし2進数の4桁目に符号を付けたものを「符号つき絶対値表現」といいます。たとえば、3は「0011」で－3は「1011」となりますが、０の場合は「0000」と「1000」の２つの表現が存在する欠点があります。

符号つき補数表現は、0000は正の数と扱い、正の数と負の数は8種類ずつとなります。一般にn桁2進数の符号つき補数表現では扱われる数字の範囲は、-2n-1～2n-1-1となります。

|  |  |
| --- | --- |
| 符号なし | |
| 2進数 | 10進数 |
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | 10 |
| 1011 | 11 |
| 1100 | 12 |
| 1101 | 13 |
| 1110 | 14 |
| 1111 | 15 |

|  |  |  |  |
| --- | --- | --- | --- |
| 符号なし表現 | | 符号つき補数表現 | |
| 2進数 | 10進数 | 2進数 | 10進数 |
| 0111 | 7 | 0111 | 7 |
| 0110 | 6 | 0110 | 6 |
| 0101 | 5 | 0101 | 5 |
| 0100 | 4 | 0100 | 4 |
| 0011 | 3 | 0011 | 3 |
| 0010 | 2 | 0010 | 2 |
| 0001 | 1 | 0001 | 1 |
| 0000 | 0 | 0000 | 0 |
| 1000 | 0 | 1111 | -1 |
| 1001 | -1 | 1110 | -2 |
| 1010 | -2 | 1101 | -3 |
| 1011 | -3 | 1100 | -4 |
| 1100 | -4 | 1011 | -5 |
| 1101 | -5 | 1010 | -6 |
| 1110 | -6 | 1001 | -7 |
| 1111 | -7 | 1000 | -8 |

## 補数

符号つき補数表現の2進数を作るには、「2の補数」という方法を用います。

10進数にも「10の補数」というものがあり、元の数字と補数を加算すると10のべき乗になるものが「10の補数」となります。

　例：6の補数は「４」(足すと10)、65の補数は「35」(足すと100)

10進数での引き算で、90-65=25となりますが、引き算ができない場合、65の補数を使って、

　90+35=125

と計算を行い、最上位(MSB)の「1」を取り除くことで、「25」という答えを求めることができます。このように、65の補数35は「-65」と負の表現になることが分かります。

「2の補数」の計算には、「10進数での-1は0から1を引いたもの」と同じ考え方を使います。4桁２進数での「-1」は「00002から12を引いて11112(5桁目から借りている)」となります。「-2」は、さらに1を引いて「11102」となります。このように、2進数の２の補数の計算は、-A=-1-(A-1)=(-1-A)+1で行うことができます。ここで（-1-A）は２進数Aの全ての桁を反転させることに相当しますので、２の補数は簡単に次のルールで作成することができます。

**「A2の全桁を反転させ(0は1へ、1は0へ)、これに1を加える。」**

０2から予め１を引くと、必ずそれは「全桁１」になります（例えば3桁の場合は111）。この全桁１からAを引くと、そのパターンは必ず「Aの全桁を反転させたもの」となります。

（例えばA10＝5(01012)ならば、1111－0101＝1010。これは0101の全桁を反転させたものと同じ）  
このパターンに「最初に０2から引いておいた１」を、元に戻すために足してやれば結果的に10112となり「(-1-A)+1」を計算したことと同じになります。(-510=10112)

また補数を使うことによって、B-Aという減算はB+(-A)として、加算で計算できることになります。

注意！！

ここで説明したのは符号を反転させる手順ですので-810(10002)を手順通りに変換すると  
反転：01112

１を加える：10002

となって-8に戻ってしまいます。４桁の符号つき補数表現では表現できる数値の範囲が-8～7までなのでそもそもが-8の符号の反転ができない事になっています。そこで、10002の場合は、10012へ強制的に変更して「反転：01102」し「１を加える：01112(710)」となるように考慮しておかなければなりません。

## 符号つき2進数を10進数に変換する

最上位が0の場合は、そのまま10進数に変換します。

最上位が1の場合は、例えば、10112(-510)では、

**-1**×23+0×22×+1×21+1×20 =-8+2+1=-5

となります。また別の方法では、

1. 最上位以外を反転させる（011→100）。
2. 手順1の結果に+1する（101）。
3. 10進数に変換し、「-」を付ける。1×22×+0×21+1×20 =4+1=5→「-5」

## キャリーオーバーとオーバーフロー

符号なし2進数n桁の数字では、0から2n-1までの数字しか表すことができません。従って2つの数字の加算を行う場合、n+1桁目に桁が上がってしまうキャリーオーバーが発生する場合があります。最初に桁数をnに固定して扱うためにこのn+1桁目は基本的には、**無視されて**しまいます。そこで、キャリーオーバーがあったかを判定しておき、計算結果が正確かどうかを見る必要があります。

一方で、符号つき2進数n桁の2つの数字の計算では、符号の桁以外のn-1桁での計算がn桁目に繰り上がる（0…+0…=1…,正の数同士の加算が負の数になる）というオーバーフローが発生する場合があります。この場合は、符号が逆転した誤った計算結果が表示されることになります。従って、どのような数値を扱っているかによって、キャリーオーバー(C)とオーバーフロー(O)をそれぞれ見なければなりません。

例：4桁の2進数の計算

|  |  |
| --- | --- |
| 符号なし | 符号つき |
| X=1110(1410)  Y=1101(1310)  X+Y=1011(1110：C=有,O=無)  本来2710が1110になっている | X=0110(610)  Y=0101(510)  X+Y=1011(-510：C=無,O=有)  本来1110が-510になっている |

# 2進数のシフト演算

2進数Aでm回のシフト演算を行うとA×2mを行ったことになります。m回のシフトとは、n桁の数値の位置を右または左にm回ずらしていくことをいいます。左にずらすとmは(+)となり、右ずらすとmは(-)となります。

これには、**論理シフト**と**算術シフト**の2つの方法があります。

## 論理シフト

符号なし2進数に対して操作を行います。

ずらした場合に空いた桁やはみ出た桁の処理は、次の通りになります。

|  |  |  |
| --- | --- | --- |
|  | 空いた桁 | はみ出た桁 |
| 右シフト | 0を入れる | 割り算での余り |
| 左シフト | 0を入れる | 無視 |

■論理シフト(左)：５桁の2進数01010(1010)を左に1回ずらすと10100(2010)となり、2倍したことになります。



無視される

0で埋められる

■論理シフト（右）：５桁の2進数01010(1010)を右に２回ずらすと00010(210)、余り10(210)となり、4で割り算(2-2)したことになります。



余りとなる

0で埋められる



## 算術シフト

符号つき2進数に対して操作を行います。

符号を表す最上位桁以外に対してシフト演算を行います。ずらした場合に空いた桁やはみ出た桁の処理は、次の通りになります。

|  |  |  |
| --- | --- | --- |
|  | 空いた桁 | はみ出た桁 |
| 右シフト | 符号桁と同じ数値を入れる | 割り算での余り |
| 左シフト | 0を入れる | 無視 |

m回のシフト演算では、論理シフトと同様に左シフトの場合は、元の2+m倍になり、右シフトの場合は、元の2－m倍となります。

■算術シフト(左)：５桁の2進数11010(-610)を左に1回ずらすと10100(-1210)となり、2倍したことになります。

無視される

0で埋められる



■算術シフト(右)：５桁の2進数11010(-610)を右に２回ずらすと11110(-210)となり、4で割り算(2-2)したことになります。ただし、結果は割り切れませんので、「切り捨て」が行われています。

余りとなる

符号桁と同じ1で埋められる



# 固定小数点数

固定小数点数は、ビット内の特定の位置に小数点を固定して数値を表します。これにも「符号なし」と「符号つき」の２つの表現方法があります。

## 符号なし固定小数点数

10進数の0と正の小数点数を2進数に変換し、小数点位置に合わせて配置することで表現されます。

例：

上位4ビットを整数部、下位4ビットを小数部とします。

表現できる数値の範囲は、0～15.937510(1111.11112)となります。

11.7510 の場合は、1110→10112、0.7510→0.112　なので、1011.112となります。

小数点位置

0を埋めた

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

### 符号つき固定小数点数

小数点のついた正負の数を扱います。MSBが符号ビットとなり、正の数は符号なし固定小数点数と同じ表現ですが、負の数は2の補数で表わします。

例：

上位4ビットを整数部、下位4ビットを小数部とする。

表現できる数値の範囲は、－8(1000.00002)～7.9375(0111.11112)となります。

-7.7510 の場合、絶対値7.7510は、710→1112、0.7510→0.112なので111.112となります。

これの2の補数を求めると、反転した000.002に0.012足して000.012となり、負の数であるためMSBに1を追加した1000.012となります。小数点の位置が決まっているため1000012と表記することもあります。

小数点位置

0を埋めた

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

### 固定小数点数のデメリット

例えば、11.1111112という数値を整数部が上位4ビット、小数部が下位4ビットで表記しようとすると0011.11112となり小数点以下の数値には切り捨てが行われます。これは正確な数値が表現できていないことになっています。また整数部分は0が埋められて無駄が生じています。このように固定小数点表記では、数値の精度を求められる計算には向いていないといえます。

## 浮動小数点数

浮動小数点数は、A=±m×bnとして表現することをいいます。ここでそれぞれの記号は次のことを表しています。

m：仮数部　b：基数部　n：指数部

これによって、大きな数字や小さな数字が少ないビット数で表現できることになります。

例：

10進数では、12400000010=12.4×107=1.24×108

2進数では、0.000101012=1.0101×2-4=0.10101×2-3

2進数表現では、浮動小数点数を何bitで表わすかを決め、その内符号（±）を決める１bitと指数部を表す数をx bitとし仮数部を表す数をy bitと決定すると、浮動小数点数は次のフォーマットで表わされます。bは２と決まっていますので省略されます。

|  |  |  |
| --- | --- | --- |
| 1bit | x bit | y bit |
| 符号  正の数：0  負の数：1 | 指数部 | 仮数部 |

この表現の欠点は、同じ数字の表現が複数あることですが、仮数部が1≦m<bとなるように指数部を調整する**正規化**によって統一することができます。

例：正規化表現では、１つの表現に決まります。

10進数では、12400000010=1.24×108

2進数では、0.000101012=1.0101×2-4　・・・(a)

2進数では、正規化を行うと必ず1.XXX…となるので、仮数部ではXXX…だけを格納し、扱える数値の範囲を広げることができます（「けち表現：economized form」といいます）。

また、指数部で負の数を扱うために、２の補数表現を用いるか、指数部を表す最大値の半分-1を加えて正の値に替えてからbit表現する（「ゲタ履き表現:bias」といいます）方法が用いられます。下駄ばき表現では、指数部の大小比較が簡単にできるという利点があります。

例：指数部を4bitで表わすとすると、0～15の数字が表現できる。これを、bais=7として  
-7から8に対応させる。よって、(a)の例では、-4 →3となり、00112が4bitの指数部となる。

## IEEE754形式

浮動小数点数の指数部と仮数部のビット数を国際標準規格として決めたものです。主なものに32ビットの単精度と64ビットの倍精度の規格があります。

### 単精度浮動小数点数

単精度浮動小数点数は、次の表のように定義されます。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 符号 | 指数部 | 仮数部 |
| 単精度 | 1bit | 8 bit | 23 bit |

指数部は2の補数ではなく127のゲタ履き表現で符号なしの整数として扱われます。8bitなので0～255までの表現ができますが、実際は、-127～128の数値に対して127（最大値256の半分－１=127）を加えて0～255の正の数として扱います。

|  |  |  |  |
| --- | --- | --- | --- |
| 指数 | 127を加える | 指数部（2進数） |  |
| -127 | 0 | 00000000 | 予約(0を表す) |
| -126 | 1 | 00000001 |  |
| ・  ・ | ・  ・ | ・  ・ |  |
| -1 | 126 | 01111110 |  |
| 0 | 127 | 01111111 |  |
| 1 | 128 | 10000000 |  |
| ・  ・ | ・  ・ | ・  ・ |  |
| 127 | 254 | 11111110 |  |
| 128 | 255 | 11111111 | 予約（無限を表す） |

例１：15.62510を単精度浮動小数点数で表わす。  
15.62510=+1111.1012=1.111101×23  
仮数部は最初の１は外して23bitにすると111101000000000000000002となります。  
指数部は3+127=13010=10000102となります。  
結果は、0 1000010 111101000000000000000002となります。

例２：-0.0011001012の場合  
符号部は1となります。  
0.001100101=1.100101×2-3  
仮数部は最初の１は外して23bitにすると100101000000000000000002となります。  
指数部は-3+127=12410=011111002となります。  
結果は、1 01111100 100101000000000000000002となります。

扱える範囲は、正の数で次の範囲になります。

最小値：0 00000001 000000000000000000000002

指数部：000000012=1なので、127を引いて-126

仮数部：1. 000000000000000000000002となり1となります。よって

1.0×2-1262 ≑ 1.0×(23.321928)-37.92978 ≑ 1.0×(10)-37.92978 ≑ 1.17549×10-3810となります。

最大値：0 11111110 111111111111111111111112

指数部：111111102=254なので、127を引いて127

仮数部：1. 111111111111111111111112となりおよそ２となります。よって

2.0 × 21272 ≑ 3.40282×103810となります。

### 倍精度浮動小数点数

省略します。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 符号 | 指数部 | 仮数部 |
| 倍精度 | 1bit | 11 bit | 52 bit |

# 情報の単位

1ビット(bit)で表わされる状態は、0と1の2通りありこれが情報の最小単位となります。

Nbitでは2N個の状態が表されます。また、8bitを1バイト(Byte)として扱います。

数字が大きくなると、1000(103)倍ずつ単位が上がっていく国際単位系と1024(210)倍ずつ上がっていく2進数の単位での表記を行います。

|  |  |  |  |
| --- | --- | --- | --- |
| 単位 | 標記 | 情報量(国際単位系SI) | 2進数に基いた単位 |
| ビット | bit(b) |  |  |
| バイト | Byte(B) | 1B=8b |  |
| キロバイト | Kilo Byte(KB) | 1KB=1000B=103B | 210B=1024B(1KiB) |
| メガバイト | Mega Byte(MB) | 1MB=106B | 220B=1,048,576B(1MiB) |
| ギガバイト | Giga Byte(GB) | 1GB=109B | 230B=1,073,741,824B(1GiB) |
| テラバイト | Tera Byte(TB) | 1TB=1012B | 240B=1,099,511,627,776B(1TiB) |

２つの単位は、1KB=1000B≒1024Bとしてほぼ同じものとして扱われます。しかし、IEC(International Electrotechnical Commission=国際電気標準会議)は、2進数に基いた単位を別に定めています。これは、各単位に2進数を表す「binary」を付けてKilobinaryByte=KiB(キビバイト)というように表されます。

確認：

167772160bitは何MBでしょうか？1MB=220Bとします。

## 情報量

1bitで2通り、2bitで4通りの表現ができます。逆に16通りの表現をするには4bitが必要となります。このようにモノゴトを表現するのに必要なbit数が**情報量**となります。一般にn通りの表現に必要なbit数はで計算することができます。

また、Claude Elwood Shannonによる情報量の計算は、

「ある事象が確率pで起こるとき、この事象が持つ情報量はとなる」

と定義されています。

例えば、明日のテストで「A(1),B(5),C(10),D(14),E(17)の問題のうちいずれかが等確率(1/5)で出題される。」となった場合、どの問題が出題されるかを知るにはbitの情報量が必要となります。また、「カッコ内の数字が10番台から出題します。」という情報を受け取った場合は、3問に絞られるので、必要な情報はbitとなります。このとき、受け取った情報量は、2.32-1.58=0.74bitということになります。

# データの符号化

コンピュータで扱われる文字や記号のデジタル化について説明していきます。

## 文字コード

コンピュータ内での文字・記号は、「文字コード」といわれる固有の番号が割り振られています。文字と番号のペアの集合を「符号化文字集合」とよび、非負整数値である番号をバイト表現に変換する方法を「文字符号化方式(エンコーディング)」とよびます。

* ASCIIコード：  
  アルファベットa～Z, 数字0～9や記号、改行などの制御文字を表現しています。1文字が1byte(8bit)で表現されます。
* 日本語用コード：  
  JIS（電子メールなど）, Shift-JIS（Microsoft社製）, EUC-JP(Unix), ISO-2022-JPなどがあります。コードは2byteで表現されます。
* 多言語に対応したもの  
  UTF-8 , UTF-16など

Shift-JISは、半角文字は1バイト、全角文字は2バイトで表されます。

次の表は、全角の「あ」「い」を表すそれぞれの文字コードで16進数での表現になっています。

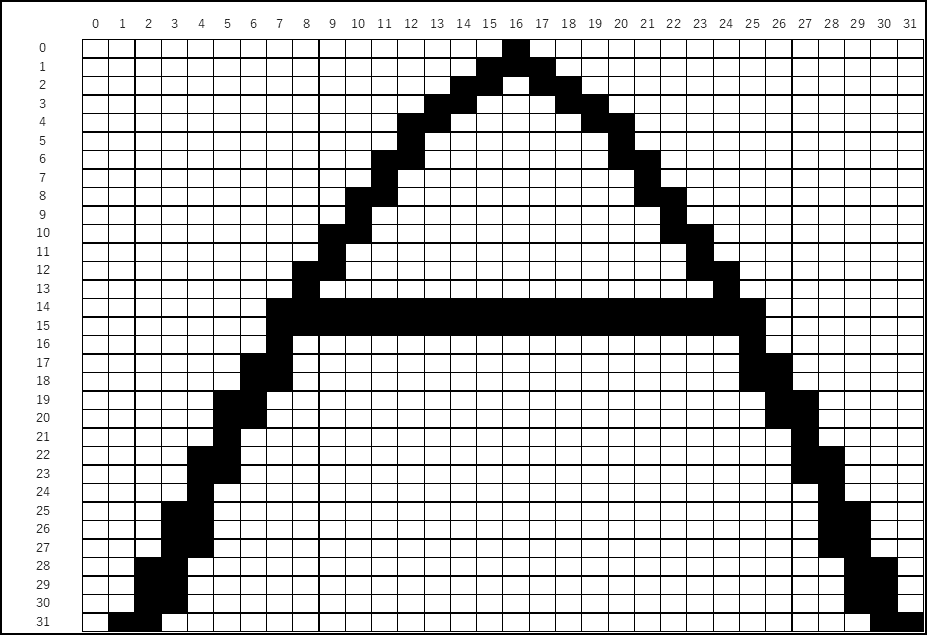
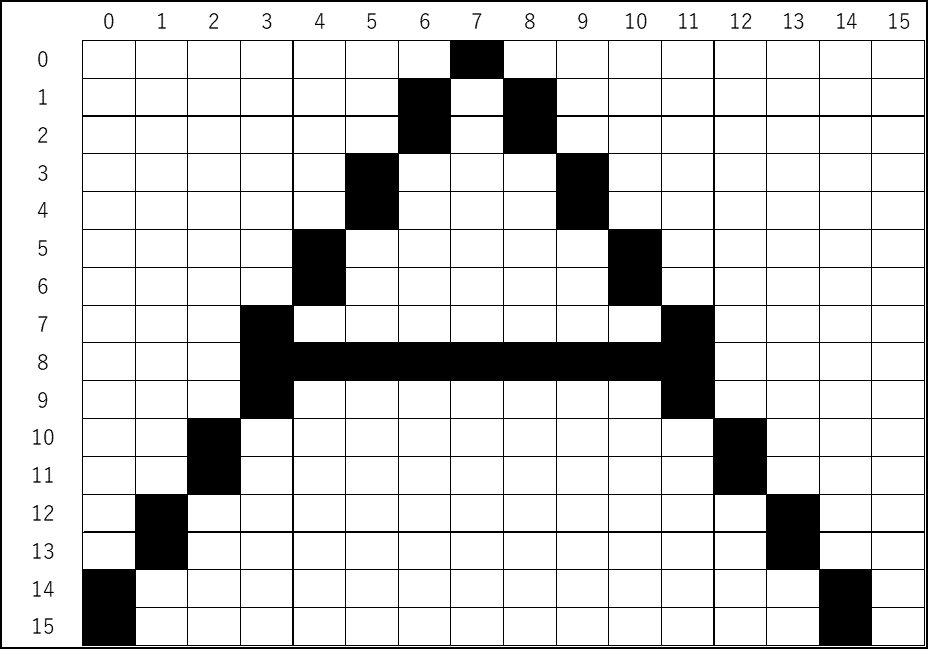
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文字 | Shift-JIS | UTF-8 | UTF-16 | EUC-JP |
| あ | 82A0 | E38182 | 3042 | A4A2 |
| い | 82A2 | E38184 | 3044 | A4A4 |

UTF-8には、BOM付とBOM無しの2種類が有ります。BOMはバイトオーダーマーク（byte order mark）の略で、Unicodeで符号化したテキストの先頭に付与される数バイトのデータのことです。BOM付きであれば、ファイルの先頭に３バイト(0xEF 0xBB 0xBF)のデータが付いていて、これでファイルの 種類を判断しています。

確認：Terapad xeditで文字コードの確認

# 画像データ

デジタル画像とは、画素を2次元に並べたものをいいます。画素とは色情報を持つ最小単位でピクセル（pixel）ともいいます。下の図は16×16の画素数で１つの画素を単純に0(白)と1(黒)で色分けしたモノクロ画像です。この図の1マスが1ピクセルを表しています。ピクセルの大きさは一定ではなく同じサイズで画素数が倍になると、ピクセルは小さくなりより滑らかな画像が表現できることになります。また、ドットとは、画像を表現する1つの点であることから、このモノクロ画像の場合は、1ピクセル＝1ドットと考えることができます。8ビットグレースケール画像は1ピクセルの色を8bitとし、薄い～濃いを256階調で表わしています。



カラー画像の場合は、1つの画素に光の3原色である赤（Red）、緑（Green）、青（Blue）（「RGB」といいます。）の重なり具合と濃い薄いの階調で色を表しています。3色をそれぞれ8bitで表わす24ビットカラー画像では、16,777,216色(フルカラー)を扱うことができます。

よって、カラー液晶画面などでは、１つの画素を3色のドットで表現するため、1ピクセル=3ドットとなります。

液晶

1ピクセル

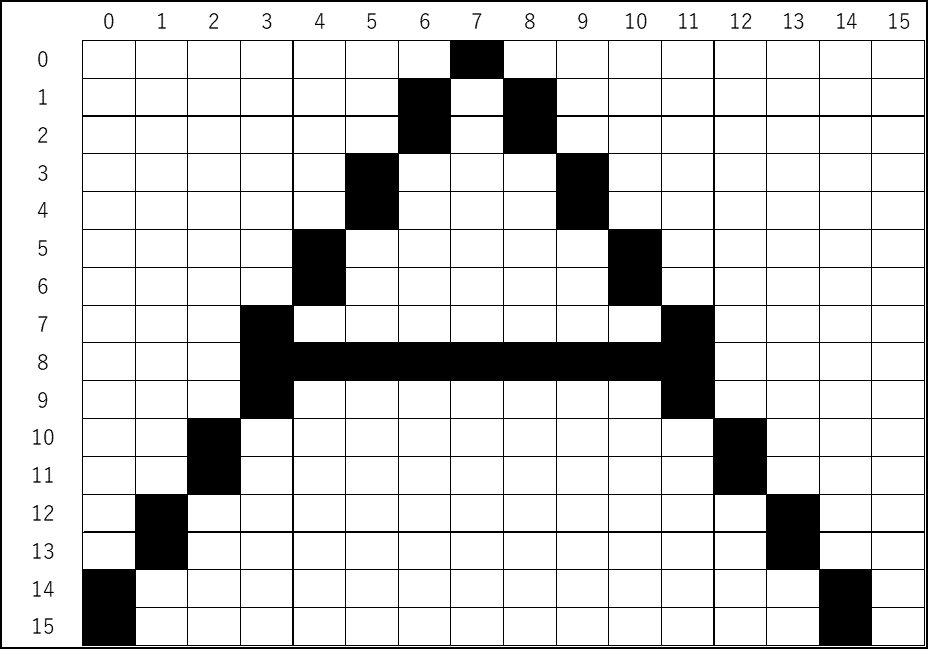
1ドット

デジタルカメラやスキャナの光を読み取る素子が3色分を読み取っていますので、1ピクセル=1ドットとなります。写真など印刷されたものは１ドットに色が付いてますので、1ドット=1ピクセルとして扱われます。

## 解像度

### 相対解像度

相対解像度は画像解像度ともいい、1inch(2.54cm)あたりのドット数あるいはピクセル数を表していいます。単位は、dpi(dot per inch)またはppi(pixel per inch)となります。



1 inch

上の画像が、横幅１inchであれば、16dpiということになります。画像が何dpiであるということと縦横のインチサイズが分からなければ全体の解像度が分からないので相対解像度といいます。

縦横4368×2912（1200万画素）の画像を350dpiで印刷する場合、約12inch(4368/350)×8inch(2912/350) =31cm×21cmとなりA3サイズで印刷できることになります。

### 絶対解像度

ディスプレイ上の画像の大きさをピクセル数(px)で表したもので、画面解像度とも言われています。

縦横のサイズが600(px)×800(px)でdpiが72dpiの画像と350dpiの画像は、同じモニタ上では、モニタ上のピクセルで表現されるので、いずれも600×800の同じサイズに表示されます。

## デジタル画像のファイル形式

まず、画像の形式としてビットマップ形式とベクトル形式があります。

* ビットマップ形式  
  画像を点（ピクセル）の集まりで表現した物
* ベクトル形式  
  画像を直線や曲線を座標と数式の組み合わせで表現した物

次に主な画像の形式を示します。

|  |  |
| --- | --- |
| 形式 | 特徴 |
| bmp (ビットマップ) | WindowsPCで標準に使われる画像形式。モノクロからフルカラー画像を扱うことができます。 |
| jpg (ジェイペグ) | ビットマップ形式の画像。色と色との境目を自然に、目立たなくする処理が可能でデジタルカメラ、Web用画像などで扱われます。また、ファイルサイズを小さくするために圧縮されます。ただし、「不可逆圧縮」であるため画質は劣化していまいます。また、輪郭がはっきりしている画像や文字が含まれている画像をこの形式で保存するとぼやけたりします。 |
| gif (ジフ) | ビットマップ形式の画像です。色数が256色と制限されていて圧縮によってファイルサイズを小さくすることができます。圧縮は「可逆圧縮」のため画像の劣化も起こりません。  色と色の境目がはっきりしているWebページのロゴマークなどによく使われています。また、「gifアニメーション」という短い動画を作ることができます。 |
| png (ピーエヌジー,ピング) | jpgやgifなどに代わる画像形式です。gifと同じ規格の「png-8」と「png-24」があります。png-24は、写真から図形までの幅広い画像の保存に向いており、可逆圧縮なので画質も維持されます。しかし、jpgと同じ画質であってもファイルサイズが大きくなってしまうデメリットがあります。 |

# データ圧縮

## 可逆圧縮と非可逆圧縮

可逆圧縮は、圧縮したデータを復元すると完全に元に戻る圧縮方法で、非可逆圧縮は、復元しても完全には元に復元できない圧縮のことを言います。

テキストなどのデータは完全に元に戻せなければデータとしての価値がなくなってしまいますので、可逆圧縮でしか圧縮できません。一方、画像や音、動画といったデータの場合は完全に復元できなくてもある程度似ていれば、それほどデータの価値に影響はありません。非可逆圧縮は、なくても良いデータを完全に除いてしまうので圧縮後のデータを小さくすることができます。しかし、画像であっても人が描いたイラストなどべた塗の画像などはJPEGの非可逆圧縮よりもGIFやPNGなどの可逆圧縮である方がデータ品質が良いことがあります。

## ランレングス法

ランレングス法は、同じデータが並んでいる場合に有効な圧縮法です。例えば、aaabbccccと9文字のデータは、a3b2c4とすることで6文字に圧縮できることになります。

## Huffman法

1952年にDavid Albert Huffmanによって考案された、可逆圧縮のための符号化法です。簡単には、データ内の各記号の出現率を求め、出現度の高い記号を短い符号で、出現度の低い文字を長い符号で表現することで、平均のデータ長を短くするという原理になっています。

1回につき「A,B,C,D」の4つ事象のうちいずれかが現れる情報を100回分作成すると仮定します。事象が4つであるので１つの事象に2bitの情報量が必要となり、例えば、A=00,B=01,C=10,D=11というように表すことができます。100回分では合計200bitとなります。このように記号をすべて一定の2bitで表わすことを「**固定長符号**」といいます。それぞれの出現回数が「1/2,1/4,1/8,1/8」であるとすると、一番確率の大きな現象Aを1bitで符号化し、現象Bを2bit、小さい確率の現象C、Dを3bitで符号化すると、必要な平均情報量は

となり情報量を減らすことができます。このように長さを変えて符号化することを「**可変符号化**」といいます。

ここで、Aを1bitで「1」と符号化したとし、他の事象を「1・・」と符号化するとAと区別する仕組みを入れなければなりません。これを回避するためにある符号のパターンが他の符号のパターンの先頭部分にならない様に符号化を行う必要があります。例えば、今回は、A→[1]、B→[01]、C→[001]、D→[000]とします。

### ハフマン符号化の手順

さきほどの例での手順を説明します。

A(1/2)

B(1/4)

C(1/8)

D(1/8)

1. ある文字列において、すべての記号の出現率を調べる。  
   A,B,C,Dの出現率が「1/2,1/4,1/8,1/8」
2. 各記号に出現率を書いた“葉”を用意し、出現率の高い順に並べる。
3. 出現率の最も低い2枚の“葉” を“節”から結び、左側の“枝”に「0」、右の枝に「1」を書き、節には、２つの出現率の合計を書く。この節を新たな葉とします。

A(1/2)

B(1/4)

C(1/8)

D(1/8)

0

1

1/4

1. 節より下は、考慮せずに、新たな葉で手順３を行います。

A(1/2)

B(1/4)

C(1/8)

D(1/8)

0

1

1/4

0

1

1/2

0

1

A(1/2)

B(1/4)

C(1/8)

D(1/8)

0

1

1/4

0

1

1/2

1. 符号の木（ハフマン木）ができたら、根元の節から各記号への枝に書かれた「0」「１」で符号化を行う。  
   　  
   結果：  
   Ａ →　0 、B →　10 、C →　110 、D →　111

# 論理演算

コンピュータ内での計算は、電子回路で行われていてその元のなるものが論理演算です。ここで扱う演算は、{0,1}の2値をとる集合に対してブール代数の基本法則に従います。また、論理関数は、２入力1出力（否定のみ1入力1出力）であるとします。論理演算を表す回路はMIL記号と呼ばれる図記号となっています。

演算の種類は次の表のものがあります。

|  |  |  |  |
| --- | --- | --- | --- |
| 演算 | 標記 |  | MIL記号 |
| 論理積(AND) | *A ・B* | *A* AND *B* | AND |
| 論理和(OR) | *A B* | *A* OR *B* | OR |
| 排他的論理和(XOR) | *A B* | A XOR(eXclusive OR) B | XOR |
| 否定論理積(NAND) | *A*｜*B* | *A* NAND *B* =NOT(A AND B) | NAND |
| 否定論理和(NOR) | *A*  *B* | *A* NOR *B* =NOT(A OR B) | NOR |
| 否定(NOT) |  | NOT *A* |  |

真理値表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | AND | OR | XOR | NAND | NOR |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

演算では、次の法則が成り立ちます。

交換則　A+B=B+A A・B=B・A

結合則　A+(B+C)=(A+B)+C A・(B・Ｃ)=(A・B)・C

分配則　A・(B+C)=A・B+A・C A+(B・C)=(A+B)・(A+C)

吸収則　A+(A・B)=A A・(A+B)=A

べき等則　A+A=A A・A=A

ド・モルガンの法則

の回路の例：

C



A

B

## ビット操作

### ■ビットマスク

あるビット列から一部を取り出すために、取り出したいビット位置に1、それ以外を0としたビットマスクを作成し、AND演算を行います。

例：8bitの01010101から下位4bitを取り出すためにビットマスクを00001111とします。

　　　　　　　　元　01010101

　　　ビットマスク　00001111

AND

　　　　　　　結果　00000101

### ■ビット反転

ビット列の1を0に、0を1に反転するには、元と同じbit数ですべて1のビット列とのXOR演算を行います。

　　　　　　　　元　01010101

　　　すべて1の列　11111111

XOR

　　　　　　　結果　10101010

### ■ビット列のクリア

ビット列をすべて0にするには、元と同じビット列でXOR演算を行います。

　　　　　　　　元　01010101

　　　　元と同じ列　01010101

XOR

　　　　　　　結果　00000000

## 加算器

論理回路を組み合わせて2進数の加算を行います。加算は1桁ずつ行われますが、下位からの桁上がりを考慮しない半加算器と考慮する全加算器があります。

### 半加算器

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | S(1桁目) | C(桁上がり) |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

下の桁からの桁上がりを考慮しません。計算結果の１桁目をSとし、桁上がりをCとします。



A

B

S

C

### 全加算器

下位の桁からの桁上がりを考慮して加算を行います。半加算器が２つとOR回路１で構成されます。Ckは下位からの桁上がりです。



A

B



Ck

S

C

Sa

Ca

Cb

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | Sa | Ca | Ck | S(SaとCk) | Cb | C |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |

練習1

1. XORが  
   であることを証明しなさい。解答での表記は、としてよい。  
   また、式は=でつないで左辺から右辺を導くこと。
2. XORの身近な例を考えなさい。

練習２

X,YおよびZを２の補数表現での４bitの符号つき2進数とし、それぞれX1～X4,Y1～Y4,Z1～Z4(添え字4が最上位)と4桁で表現します。

このとき、XとYの加算を行い、結果をZに求め、桁上がりをC1～C4とする加算器は、次の図で表わされます。

半加算器

X1

X2

X3

X4

Y1

Y2

Y3

Y4

加算器

加算器

加算器

C1

C2

C3

C4

S1

S2

S3

S4

X=-210、Y=-310として計算を行ったとき、C1～C4の組み合せで正しいものを選びなさい。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | C1 | C2 | C3 | C4 |
| ア) | 0 | 0 | 0 | 0 |
| イ) | 0 | 1 | 1 | 0 |
| ウ) | 0 | 0 | 1 | 1 |
| エ) | 0 | 1 | 0 | 1 |
| オ) | 1 | 1 | 0 | 0 |
| カ) | 1 | 0 | 1 | 0 |
| キ) | 1 | 0 | 0 | 1 |

練習3

文字列の中で同じ文字が繰り返される場合に、その部分を文字と繰り返し回数の組に置き換える圧縮方法は次のうちどれか。

* 1. ASCII符号化 B)巡回符号化　C)ランレングス符号化　D)ハフマン符号化

# コンピュータの仕組み

コンピュータは、入力装置、制御装置、演算装置、記憶装置、出力装置の５つの装置から成り立っています。これらの機械的装置をハードウェアと呼びます。

記憶装置

主記憶装置

(メモリ)

補助記憶装置

(HDD/SSD)

入力装置

(キーボード

マウス)

出力装置

(ディスプレイ

プリンタ)

中央処理装置(CPU)

制御装置

演算装置

画像処理装置 (GPU)

:データの動き

:制御の動き

* 中央処理装置(Central Processing Unit:CPU)
  + 制御装置：プログラムの指示に従って他の装置を制御する装置
  + 演算装置：主記憶装置のデータに対して演算を行う装置
* 記憶装置
  + 主記憶装置：プログラムとデータを一時的に格納する装置
  + 補助記憶装置：プログラムとデータを永続的に格納しておく装置
* 入力装置：データを主記憶装置に読み込むための装置
* 出力装置：主記憶装置のデータを外部に出力するための装置

## CPU

CPUは主記憶装置にあるプログラムから命令を順に取り出して解読し、各装置の動作を制御したり、データ同士の演算を行ったりします。

CPUは、次の図のように構成され、各機能の内容は表に示した通りです。

主記憶装置

(メモリ)

ALU

汎用レジスタ

プログラムカウンタ

メモリアドレスレジスタ

バス

CPU

データの流れ

命令

デコーダ

命令レジスタ

|  |  |
| --- | --- |
| 機能 | 内容 |
| デコーダ | CPUで実行する命令を解読する |
| レジスタ | データやアドレス、命令などを保持しておく高速なアクセスが可能な記憶装置 |
| 命令レジスタ | メモリから取り出された命令を格納しておく |
| プログラムカウンタ (プログラムレジスタ) | 実行する命令が保持されている主記憶装置のアドレスを保持する |
| 汎用レジスタ | 演算に使用するデータを格納する |
| メモリアドレスレジスタ | 操作するメモリ上のアドレスを保持する |
| ALU | 四則演算や論理演算を行う |
| バス | データをやり取りする経路。アドレスバス、データバス、コントロールバスの3種類がある。 |

### CPUの基本動作

CPUはプログラムに記述された命令を順番に実行していきます。命令は、「命令取り出し」と「命令実行」の2つの動作を繰り返す「命令サイクル」で実行されます。基本的な命令サイクルは次の4つの過程[ステージ]で構成されています。

1. メインメモリから命令を読み出す[フェッチ]
2. 命令の解読[デコード]
3. 命令の実行[エグゼキュート]
4. 結果をメインメモリへ書き込む[ライトバック]

命令は、基本的に4つのステージが終了してから次の命令の実行に移る「逐次制御方式」で行われます。現在のCPUは、複数の命令をステージごとにずらして並列に処理することで短時間にたくさんの命令を実行するという「先行制御方式」(パイプライン方式)が採用されています。

## CISCとRISC

CPUのもつ機能によって次の2種類があります。

* CISC[Complex Instruction Set Computer]  
  CPUに高機能な命令を持たせることによって、一つの命令で複雑な処理を実現できます。  
  主なもの：intel製(Pentium,Atom)、AMD製（Phenom,Athlon）
* RISC[Reduce Instruction Set Computer]  
  CPU内部に単純な命令しか持たないかわりに、それらをハードウェアのみで実装して、一つの命令を高速に処理できます。  
  主なもの：IBM製(PowerPC)、ARM

## CPUの性能

CPUの性能を示す指標の１つにMIPS(Million Instructions Per Second)があります。これは、1秒間に実行可能な命令の数を百万単位で表したものです。例えば、50,000,000回実行可能であれば、50MIPSとなります。求めるためには、「１つの命令の実行にかかる時間」が分かることが必要ですが、そのためには、クロック周波数とCPI(Cycles Per Instruction)が分かる必要があります。

■クロック周波数

コンピュータ内部にあり、電子回路の信号の送受信のタイミングを揃えるために周期的に電圧の高低が繰り返されている信号があります。この繰り返しが1秒間に行われている回数をクロック周波数（単位:Hz[ヘルツ]）と呼び、CPUはこれに同期して命令を実行します。

1クロックにかかる時間=1秒÷(クロック周波数)

となります。

■CPI

CPI(Cycles Per Instruction)は、1つの命令を実行するのに必要なクロック数です。

よって、

１つの命令の実行にかかる時間=1クロックにかかる時間×CPI  
=1秒÷(クロック周波数)×CPI

となり、MIPSは、

MIPS=1秒÷(１つの命令の実行にかかる時間)÷106

となります。すなわち、クロック周波数が大きくて、CPIが小さいほどCPUの処理速度が高速動作という点で性能が良いといえます。

練習４：

1GHzで動作するCPUがある。このCPUは，機械語の1命令を平均0.5クロックで実行できることが分かっている。このCPUは約何MIPSか。

練習５：

平均命令実行時間が10ナノ秒のコンピュータがある。このコンピュータの性能は何MIPSか。

## 主記憶装置

いわゆるメモリは、1byteごとにアドレスがついていて、CPUはこのアドレスを指定してデータを読み書きしています。

メモリにはつぎの２種類があります。

■ROM（Read Only Memory）

不揮発性メモリとも呼ばれ、電源が供給されていなくても記憶内容が保持されています。基本的に読み出し専用ですが、内容を書き換えることができるものもあります。書き換えが可能なROMをPROM（ProgrammableROM）と呼び、次の種類のものがあります。

* EPROM：紫外線を使って内容をすべて消去し、電気的に書き込む
* EEPROM：内容の一部の消去と再書き込みを電気的に行う。
* フラッシュメモリ：内容の一部またはすべての消去と再書き込みを高速に行う。

■RAM(Random Access Memory)

揮発性メモリとも呼ばれ、データは読み書きができますが電源がなくなると内容が消えます。

* DRAM(Dynamic RAM)：コンピュータのメインメモリに利用され容量が大きいものがあります。コンデンサに電荷を蓄えることによって内容を保持しています。この電荷が時間と共に失われていくため一定時間（1秒間に数回）ごとにリフレッシュと呼ばれるデータ更新が行われます。このように常に動いているためDynamicと呼ばれています。
* SRAM(Static RAM)：構造が複雑なので大容量のものには向きません。DRAMのようにリフレッシュが不要なので高速にデータの出し入れができます。また、低電力でデータが保持できるためCPUの内部やCPUとメインメモリとの間に配置して、使用頻度が高いデータを保持して高速な動作を維持するキャッシュメモリとして利用されます。

### キャッシュメモリ

　CPUとメインメモリの間に配置され、メインメモリの一部のデータをコピーしておきます。CPUが同じデータを読み込むときには、キャッシュメモリから読み込むことで高速な動作が維持されます。キャッシュメモリを使う手順は次のようになります。

1. CPUが、キャッシュメモリにアクセスする。
2. [キャッシュメモリにデータがある]CPUへ転送し終了  
   [キャッシュメモリにデータがない]メインメモリへアクセス
3. メインメモリからデータをCPUへ転送し、同時にキャッシュメモリにコピーする。コピーするための空き場所がない場合は、使用頻度の低いデータを解放することが必要になり、これを「キャッシュをフラッシュする」といいます。

CPUでデータを更新した場合、次の2つの方法のいずれかでメインメモリのデータの更新を行います。

* ライトスルー：キャッシュメモリとメインメモリの両方を更新する。この方式では、書き込みの時間はメインメモリのアクセス時間と同じなので、高速化はされない。しかしキャッシュの内容をフラッシュする必要が生じても、何もしなくてもよいので、回路が簡単になります。
* ライトバック：キャッシュメモリだけを更新し、読み出し時間だけではなく、書き込み時間をも短縮しています。しかし実際に書き込まれたデータはキャッシュメモリ上にしか存在しないため、キャッシュをフラッシュするときは、キャッシュメモリの内容をメインメモリに書き戻さなければならなりません。この方式は、ライトスルー方式よりも実装は困難ですが、全体的な性能はライトスルーよりもよくなります。

## 補助記憶装置

補助記憶装置は、大容量でデータを長期に保管できることが特徴になっています。種類としては、FD(フロッピーディスク)、CD(Compact Disc)、DVD(Digital Versatile Disc)、USB(フラッシュメモリ)、HDD(Hard Disk Drive)、SSD(Solid State Disk)等が挙げられます。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 記憶装置 | FD | CD | DVD | USB | HDD | SSD |
| 容量 | 1.4MB | 640MB | 4.7～9.4GB | 4GB～2TB | 500GB～16TB | 64GB～2TB |

### HDDの仕組み

HDDは、プラッタと呼ばれる磁気ディスクが円盤形になったものを記憶媒体として用いる記憶装置です。HDDには、複数枚のプラッタが搭載されていて、それぞれにデータを読み書きする磁気ヘッドが用意されています。



スピンドルモータ

アクチュエータ

電源コネクタ

インターフェイス  
コネクタ

アクセスアーム

磁気ヘッド

プラッタ

プラッタには、次の図に示すようにトラック・セクタ・シリンダと呼ばれる部位があります。

セクタ

1枚のプラッタ

トラック

|  |  |
| --- | --- |
| トラック | プラッタを同心円上に分けた領域。 |
| セクタ | トラックを一定角度で分けたときの1つの領域。データはセクタ単位で記録されます |
| シリンダ | 複数枚のプラッタの同じ位置のトラックの集合。同心円の数がシリンダ数となります。 |

### HDDの容量

HDDの容量は、シリンダ数C、1シリンダあたりのトラック数T、１トラックあたりのセクタ数S、セクタの長さをB[byte]としたとき次の式で求めることができます。

　HDDの容量[byte]=C×T×S×B

### HDDのアクセス時間

HDDからデータを読み出したり書き込んだりする時間（アクセス時間）は、CPUが動作の指示を出してから完了するまでの時間で次のように表されます。

アクセス時間=平均位置決め時間+平均回転待ち時間+データ転送時間

■平均位置決め時間(平均シーク時間)  
磁気ヘッドが目的のトラックに移動するのにかかる時間です。平均シーク時間のことを平均位置決め時間や平均ポジショニング時間ともいいます。

■平均回転待ち時間  
磁気ヘッドの真下に目的のセクタが来るまでの時間です。最小時間はデータが磁気ヘッドの下にある場合で0、最大で1回転待たなくてはなりませんので、平均は1/2回転にかかる時間となります。1回転にかかる時間は60秒÷(1分間当たりの回転量:RPM[Revolutions Per Minute])となります。  
すなわち  
　　平均回転待ち時間=60÷RPM÷2 [s]  
となります。

■データ転送時間  
磁気ディスクのデータを読み書きする間の時間です。  
　　データ転送時間＝転送するデータのバイト数÷データ転送速度  
　　データ転送速度=1トラックのバイト数(1周分のデータ量)÷1回転の時間

## SSD

SSDは、フラッシュメモリなどを用いた補助記憶装置です。HDDと比べてデータを高速に読み書きでき、磁気ヘッドやプラッタ等がないことから衝撃や振動に強いという特徴があります。しかし、使用頻度により記憶媒体が劣化するため、読み書きの回数に限界があります。

練習6：

12,000回転/分、平均シーク時間:10ms、転送速度10MB/秒のHDDにおいて1,000byteのデータの読み取りにかかる平均時間は何msか求めなさい。

平均シーク時間：

平均回転待ち時間：

データ転送時間：

よって、

アクセス時間=

# インターフェイス

コンピュータの構造を見てきましたが、それぞれの装置を接続させてデータのやり取りをする、または装置内でデータのやり取りをする決まりごとをインターフェイスといいます。また、人が操作する方法もインターフェイスといいます。ここでは、装置同士を接続するコネクタについて説明していきます。

## シリアルとパラレル

装置間でデータをやり取りするための通信方法には、シリアル伝送とパラレル伝送があります。シリアルはデータのビット列を1本の線で1bitずつ順番に送信する方法です。パラレルは、8本または16本の線を使って8bitまたは16bitを一度に送信する方法です。

使用例：

シリアル：USB(Universal Serial Bus)、IEEE1394(Firewire,iLink)、Serial ATA、RS-242C

パラレル：セントロニクス(パラレルポート)、IDE

パラレルの方が送ることができるデータ量が多いためパソコンの処理の高速・高性能化に役立つと思われます。しかし、使用例をみるとUSBやパソコンとHDDを接続するSATAにシリアルが使われています。例えば16bitを伝送できるパラレルの場合は、16個すべてのデータが同時に受け取られなければなりません。実際には、配線の長さや形状によってすべて同時ということはありませんので到着に時間差が生じます。この時間差は高速で処理しているCPUが全てのデータが揃うまでの待ち時間になりますので、効率の悪いことになります。この時間差のことを考慮する必要のないシリアルが現在主流になっています。

## 映像入力インターフェイス

### ディスプレイ解像度

コンピュータの高性能化に伴いディスプレイも高解像度化してきました。ディスプレイは、コンピュータの動きを視覚的に提示して人が操作を行うインターフェイスとなっています。解像度は、提示する領域の広さを表し、解像度が低いと表示できる領域が狭く、高いと領域が広くなります。



1024×768：15インチモニタ

1920×1080：21.5インチワイド液晶ディスプレイ

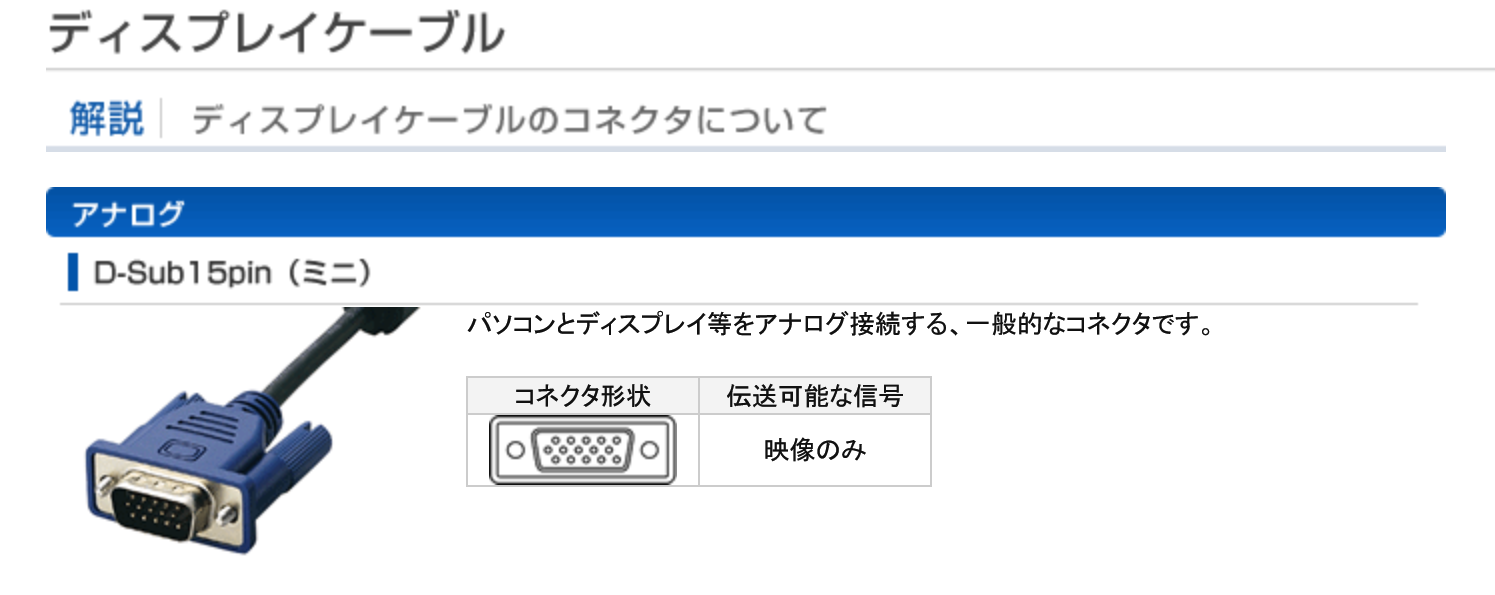
モニタには次の解像度があります。

* VGA・・640×480
* XGA・・1024×768
* SXGA・・1280×1024
* WXGA・・1280×768、1366×768、1280×800
* HD+・・1600×900
* UXGA・・1600×1200
* フルHD・・1920×1080
* 4K　UHDTV・・3840×2160　4K DCI 4096×2160(デジタルシネマ用)

### 接続ケーブル

コンピュータとモニタを接続するインターフェイスはアナログ方式とデジタル方式の2種類が有ります。モニタが液晶ディスプレイの場合のアナログ方式は、コンピュータからの信号をアナログに変換して液晶ディスプレイへ送信し、それをデジタルに変換して表示するという手間がかかり、変換の過程で映像の劣化が生じます。

接続ケーブルには主に次の種類のものがあります。

https://www2.elecom.co.jp/cable/display/guide.htmlより抜粋(2019.9)



## DisplayPortのバージョンによるデュアルモードの違い

DisplayPortには1.1～1.4までのバージョンがあります。

* 1080/60pや4K/30pに対応したDisplayPort1.1
* 4K/60pや5K/30pに対応したDisplayPort1.2
* 5k/60pや8K/30pに対応したDisplayPort1.3
* HDRや圧縮での8K/60pに対応したDisplayPort1.4

参考URL

https://www.eizo.co.jp/eizolibrary/other/itmedia02\_02/

# ソフトウェア

コンピュータを実際に制御するために作成されたプログラムです。基本ソフト（Operating System；オペレーティングシステム）とアプリケーションソフトなどがあります。ソフトウェアは操作をサポートするためにユーザインターフェースを持つものもあります。

## 基本ソフト

プリンタにデータを送ったり、ディスプレイに出力したりといったコンピュータがハードウェアなどを操作するための基本的なソフトウェア(Operating System)で、Windows、Mac OS、Linux（リナックス）などがあります。スマートフォンなどでは、Android、iOSなどがあります。

Windowsは、Microsoft社の製品で1993年のwindows3.1からはじまり現在widows7やwindows10が発売されています。これらは、電源スイッチを日常的にOn-Offすることを想定した個人利用向けのコンピュータ向けであり、常時稼働しているサーバ向けにはWindows Server 2013などのサーバOSといわれるものがあります。

Mac OSはApple社のMacintosh用に開発されたOSで、ユーザが直感的に操作できる機能を提供しています。マウスによって、プルダウンメニューを表示したり、ドラッグアンドドロップを行ったりといった操作ができるGUI（Graphical User Interface）が導入されています。また、デスクトップでのごみ箱といった概念も取り入れられています。

Linuxは、サーバなどで採用されていたUNIX（ユニックス）から派生したOSです。LinuxはOSの根本的な役割を担っている部分に様々な機能を持ったソフトウェアが組み合わさってOSとして機能します。その組み合わせによって、RedHat、CentOSなどディストリビューションと呼ばれるものがあります。ディストリビューションには、無償のものと有償のものがあります。

## アプリケーションソフト

コンピュータで特定の作業を行うためのソフトウェア。文書をつくるためのワープロソフト、表を作成するための表計算ソフトや電子メールをやり取りするためのメールソフトなどがあります。また、特定の機能や機器の設定を行うものをユーティリティーソフトと呼んでいます。さらに、OSがプリンタなどの周辺機器を制御するためのソフトをドライバソフトと呼んでいます。ドライバソフトは周辺機器に伴って提供されます。これによって、OSが常に未知の周辺機器に対応している必要がなくなっています。