# PC一般

## キーボード

良くあるのは以下の２つ

101 英数字のみ

106 キーにかな文字の刻印がある　例　「A ち」

109 106に２つのWinキーとアプリケーションキーが追加。

## ディレクトリ構成の記述方法

方法１　– または +で表現する

- root \

- directory1\

- file1.txt

- file2.txt

方法２　「けいせん」を使う

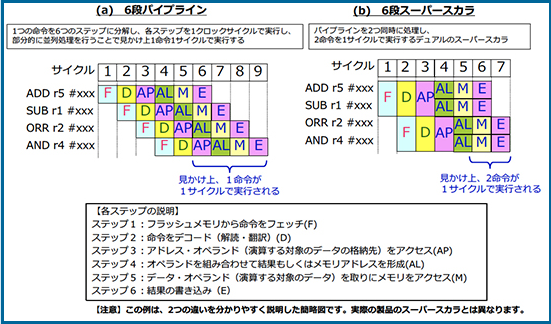
# Win10

## プログラミング関連

### PATH

PCを右クリック→システムの詳細設定→環境変数

# CPU

スーパースカラ

別名：スーパースケーラ、superscalar

見かけ上、２命令を１サイクルで実行する。

# メモリ

UV-EPROM

紫外線消去型EPROM

紫外線で全データ一括消去できる。

# ストレージ関連

## マスターブートレコード(MBR)

PCのブートセクタの一種。コンピュータの起動直後に実行される、ブートレコーダの情報を持つ。

全体のサイズは512バイト。

|  |  |  |  |
| --- | --- | --- | --- |
| アドレス | サイズ | 内容 |  |
| 0x0(0) | 446byte | ブートストラップローダ  （別名：ブートローダ） |  |
| 0x01BE | 64byte | パーティションテーブル | 第1～第4パーティション。各16byte |
| 0x01FF | 2byte | ブートシグニチャ |  |

## パーティションテーブル

### レイアウト

|  |  |  |  |
| --- | --- | --- | --- |
| オフセット | サイズ | 内容 |  |
| 0x00 | 1byte | ブートフラグ |  |
| 0x01 | 3byte | 開始セクタ（CHS） |  |
| 0x04 | 1byte | パーティーションタイプ |  |
| 0x05 | 3byte | 終了セクタ（CHS） |  |
| 0x08 | 4byte | 開始セクタ（LBA） |  |
| 0x0c | 4byte | 総セクタ数 |  |

### ブートフラグ

### ディスクラベル

MBR；Master Boot Record

1983年に導入。マスターブートコード、ディスクのパーティションテーブル、およびディスクシグネチャを含む3つの部分で構成。

GPT；GUID Partition Table

1990年に開発。比較的近年に普及。

４つのプライマリパーティションテーブルを保持できる。

## ファイルシステム

### 概要

ストレージに保存されたデータにアクセスする為のOSの機能。この機能が無いとディスクのセクタなどを直接指定しないとファイルを取得できない。

一般的には「フォーマットする時のタイプ」と言った感じで認識されている。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Win | Mac | Linux | 備考 |
| exFAT | 〇 |  | △ | 制限なし。外付けHDD にしか使えない。OS用の内臓HDD には使えない。旧機器△。 |
| FAT16 | 〇 |  | 〇 | Win95まで。１ファイル最大2GB、1パーティション4GBまで。 |
| FAT32 | 〇 | 〇 | 〇 | １ファイル最大4GB、1パーティション2TBまで。 |
| NTFS | 〇 | △ |  | WinXP あたりから登場。1パーティション16EBまで。  Mac非対応（読み込みは可能らしい） |
| HFS+ |  | 〇 |  | 1パーティション2TBまで。（OSによっては8EB） |
| APFS |  | 〇 |  | Apple File System。Mac系では比較的新しいファイルシステム。 |
| ext2 |  |  | 〇 | 以前のLinuxで標準的に使用されていた規格。inodeの数が制限される。 |
| ext3 |  |  | 〇 | ex2の後継（root可能）inodeの数が制限される。  ジャーナリングファイルシステム |
| ext4 |  |  | 〇 | ex3の後継（root可能）inodeの数が制限される。  ジャーナリングファイルシステム |
| XFS |  |  | 〇 | ジャーナリングファイルシステム（root可能）  動的inode　　　　IRIX 5.3において初めて搭載。ほとんどのLinuxで使用可能。 |
| JFS |  |  | 〇 | IBMが開発。ジャーナリングファイルシステム。  動的inode |

参考）<https://www.buffalo.jp/support/faq/detail/1079.html>

※1000G = 1P(ペタ) ; 1000p= 1E(エクサ)

### NTFS-3G

NTFSがベースだが、Linux、macOSなどで動作する。

かなり低速らしい。

## S.M.A.R.T.

Self-Monitoring, Analysis, and Reporting Technology

## HDD

### 回転数

大きく分けて5400と7200がある

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 回転数 | 読み書き | 騒音 | 消費電力 | 値段 | 備考 |
| 5400 | やや遅い | 静か | 少なめ | 安め |  |
| 7200 | 早い | ややうるさい | 多め | 高め |  |

### 大きさ

デスクトップ用の3.5inchとノートPC用の2.5inch

### ディスクキャッシュ

頻繁に読み込むファイルがキャッシュ容量以下であれば高速になる。が，それほどの効果はないという声も。

むしろバッファと呼ぶべきという声も。

## SDカード

SanDisk 米　　メモリーカード最大手

Extreme PRO, Extreme PLUS, Extreme, Ultra PLUS, Ultra の順に高速。

Transcend 台湾　USBメモリなどで高い信頼性

Ultimate 633x, Ultimate 600x, Premium400の順に高速。

High Endurance は書き換え信頼度高

Toshiba 日本　国内メーカーの中では安い

EXCERIA PRO, EXCERIA(黒), EXCERIA（赤）の順に高速

TEAM 台湾

U3 などと書いてあるのはUHS スピードクラス。ビデオ記録に必要な指標

V90などと書いてあるのはビデオスピードクラス。

|  |  |  |  |
| --- | --- | --- | --- |
| 最低保証速度 | スピードクラス | UHSスピードクラス | ビデオスピードクラス |
| 90MB/s |  |  | V90 |
| 60MB/s |  |  | V60 |
| 30MB/s |  | U3 | V30 |
| 10MB/s | C10 | U1 | V10 |
| 6MB/s | C6 |  | V6 |
| 4MB/s | C4 |  |  |

参考）http://sony-xperia-zl2-sol25.blogspot.com/2015/02/sandisk-samsung-team-microsdxc.html

## RAID

RAIDには「レベル」もある。採用の多いレベルをまとめる。

2020年6月頃まで勘違いしていたが，バックアップよりもアクセス速度改善に焦点が当たっている技術

まとめ（編集中）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ストライプ | ミラー | パリティ | 備考 |
| RAID0 | 2 | 0 | なし |  |
| RAID1 | 0 | 2 | なし |  |
| RAID10 | 2 | 2 | なし |  |
| RAID2 | 2 | 0 | 1:バイト単位 | 誤り訂正はハミングコード |
| RAID3 | 2 | 0 | 1:バイト単位；専用ディスク |  |
| RAID4 | 2 | 0 | 1:専用ディスク；ブロック単位 |  |
| RAID5 | 2 | 0 | ブロック単位；各ディスク |  |
| RAID6 | 4 |  |  |  |

RAID0： ストライピング ディスク２台以上

１つのファイルを複数のディスクに分散して記録。アクセス速度向上。バックアップ能力はない。

RAID1： ミラーリング ディスク２台

２台のディスクに全く同じファイルを書き込む。バックアップ機能のみ。

アクセス速度には変化なし。

RAID1+0 ディスク４台

文字通り1と0の組み合わせ。

※RAID10という表記も見受けられる

RAID5： 独立R/Wアクセス（パリティディスク分散） ディスク３台以上

ストライピングだがデータ（ブロック単位）とパリティをセットで別々のディスクに書き込む。

RAID4で発生していた「ライトペナルティ」による負荷が抑えられている。広く採用されている。

RAID6： 独立R/Wアクセス（パリティディスク分散(2台) ディスク４台以上

ストライピング＋２つのディスクにパリティデータを書き込み。安全面は高いが，効率と費用はやや高め。

以下はあまり使われないもの

RAID2： ストライピング（ハミングコード）

ストライピング＋パリティ専用ディスク。処理単位はブロックやバイト。ECC（Error Correction Code ；ハミングコード）と呼ばれる誤り訂正コードを用いる。データの信頼性確保。費用面の問題から，ほとんど採用されていない。

RAID3： ストライピング（パリティコード）

ストライピング＋パリティ専用ディスク。処理単位はバイト単位。こちらはパリティを使用。高いデータ転送速度。

（パリティには元データそのものも含まれ，障害時にはこれを利用してデータを復旧する）あまり採用されていない。

RAID4： 独立R/Wアクセス（パリティディスク固定）

ストライピング＋パリティ専用ディスク。（≒RAID3）処理単位はブロック単位。

パリティディスクにアクセスが集中する為，アクセス速度は遅くなる。

# アルゴリズム

## 逆ポーランド記法

別名：後置記法、Reverse Polish Notation

計算機に計算を指示する場合に都合が良い

|  |  |  |
| --- | --- | --- |
| 中置記法 | 逆ポーランド記法 |  |
| a+b | ab+ |  |
| a+b-c | ab+c- |  |
| a\*(b+c) | abc+\* |  |
| (a+b)/(c-d) | ab+cd-/ |  |

# 一般的なショートカットキー

Ctrl + 5 ： 取り消し線 e.g. あいうえお　※Wordでは1.5行 になっている。自作する必要がある。

# 論理演算

（英）Binary Operation

論理積 AND 両方真の時に真になる

|  |  |  |
| --- | --- | --- |
| 入力A | 入力B | 出力 |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

論理和 OR いずれかが真の時に真になる

|  |  |  |
| --- | --- | --- |
| 入力A | 入力B | 出力 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

否定論理積 NAND 論理積の結果を否定。両方偽である。

否定論理和 NOR ORの結果を否定。いずれかが偽である。

※C言語的に考えるなら否定は ! となる。

排他的論理和 XOR　(EOR) 入力値が同じ時に偽、異なる時に真

|  |  |  |
| --- | --- | --- |
| 入力A | 入力B | 出力 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# セキュリティ系

## 暗号化

### MD5

Message Digest Algorithm 5

・任意の長さの原文を元に128bitの値を生成するハッシュ関数の1つ

・与える値の長さがなんであれ、生成されるのは128bitの値になる

# 保守系

## システム構成

### ホットスタンバイ

現用（本番）系と同様のシステムを最初から起動しておく方式。障害発生時には自動的に処理を引き継ぎ、業務を継続する。3つの中では切替えに要する時間が最も短い。

### ウォームスタンバイ

コンピュータの電源を入れ OS を立ち上げておくが、業務システムは起動させない状態で待機させておく方式。待機時には主系のデータベースとの同期だけを行い、障害が発生してから待機系のデータベースシステムに切り替えて処理を引き継ぐ。

### コールドスタンバイ

待機時には予備系で別の処理を行わせておく方式、または予備系の電源を切った状態で待機させておく方式。障害発生時はシステムを再起動し、業務システムを立ち上げて処理の引継ぎを行う。3つの中では切替えに要する時間が最も長い。

# サービス系

## タスクスケジューラ

Windows版cronといった所

スタート → Windows管理ツール → タスクスケジューラ

# グラフィック系

## 画像ファイル

jpeg

最初のバイト：FF D3

圧縮率の指定ができ、圧縮率が非常に高いが、非可逆圧縮なので一度画質が低下すると元には戻せない。

さらに、保存をするたびにデータを損失する事になる。視覚的違和感の無い間引き圧縮。写真に向く。

png

非可逆圧縮で画質の劣化がない。透過処理が優秀、半透明も可能。高性能。

最初のバイト89 50 4E 47 0D 0A 1A 0A 更にこの後にチャンクというデータが続く

|  |  |  |
| --- | --- | --- |
| 種類 | バイト数 | 説明 |
| length | 4 | チャンクのdata部分のバイト数を表す |
| type | 4 | 4文字でチャンクの種類を表す |
| data | 可変長 | lengthで指定されたbyte長のチャンクのデータ |
| CRC | 4 | typeとdataのCRCで、データの破損をチェックできる |

giff

（ジフ）Web用にLZW圧縮した画像。写真以外の画像、例えば図形やロゴなどに向く。

透過処理、動画作成などができる。

BMP Windows標準。基本無圧縮

ICO アイコン専用の画像

SVG ベクター画像（直線や曲線などの集まりで描かれる画像）の筆頭。Scalable Vector Graphics

## 画像系用語

### シェーディング

立体感を生じさせるため，物体の表面に陰付けを行う処理

### アンチエイリアシング

スクリーンの画素数が有限なので図形の境界近くに生じる，階段状のギザギザを目立たなくする処理

### クリッピング

画像表示領域にウィンドウを定義し，ウィンドウの外側を除去し，内側の見える部分だけを取り出す処理。

### レンダリング

CG映像作成における最終段階として，物体のデータをディスプレイに描画できるように映像化する処理。

## HDMI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| バージョン | 1.2以前 | 1.4 | 2 | 2.1 |
| 帯域幅 | 4.95Gbps | 10.2Gbps | 18 Gbps | 48 Gbps |
| 解像度  リフレッシュレート |  | 4K 静的HDR/30Hz  フルHD 静的HDR/144Hz | 4K 静的HDR/60Hz  フルHD 静的HDR/240Hz | 8K/60Hz（圧縮）  8K 動的HDR/30Hz  4K/240Hz（圧縮）  4K 動的HDR/144Hz  フルHD 動的HDR/240Hz以上 |
| VRR |  | × | × | 〇（任意機能） |
| ARC |  | ARC（任意機能） | ARC（任意機能） | eARC  （任意機能） |
| DSC |  | × | × | 〇  （任意機能） |
| 対応するHDMIケーブル | Standard | High Speed | プレミアムハイスピード | ウルトラハイスピード |

## モニタ

|  |  |  |
| --- | --- | --- |
| 種類 | 長所 | 短所 |
| TN型 | 応答速度が速い  低コスト  透過度が高い→消費電力が少ない | 視野角が狭い  色の再現性が低い |
| VA型 | 応答速度が速い  コントラストが高い | 視野角が狭い  色の再現性が低い |
| IPS型 | 視野角が広い  色の再現性が高い | コントラストが低い  応答速度が遅い  高コスト |

# プログラミング標準

## メモ

・「考えることを減らす」のはシステム管理上とても重要。

## 正規表現

正規表現には、最も左から、最も長く一致するものにマッチするという法則がある

### 基本的な正規表現

|  |  |  |  |
| --- | --- | --- | --- |
| 文字 | 意味 | 例 | 備考 |
| . | 任意の１文字 | a.. ⇒ aから始まる3文字  ..a ⇒ aで終わる3文字 |  |
| \* | 直前の文字の0回以上の繰り返し | t.\*l ⇒ tol, tool, tail など  ^t\* ⇒ 注意。無意味。 | **0回も含む点**に注意  拡張の+に似ている。  ⇒数量詞　とよぶ |
| [] | 文字クラス（文字集合） | [abc] a,b,cのどれか１文字  [a-z] 小文字の英字１文字  [a-zA-Z] 英字１文字 |  |
| ^ | （先頭にある場合）  行頭 | ^a ⇒ aから始まる行 | （参考）　$　行末 |
| ^ | （ [] とともに使う場合）  後続の文字以外 | [^abc] a,b,c,以外の１文字 |  |
| - | 範囲指定 | [0-9] 数字 |  |
| (...) | 文字をグループにまとめる | goog(le|ol)　> google |  |
| { n } | 直前の文字の桁数指定 | a{3}?　　> aaa |  |
| { n, } | 直前の文字の最小桁数指定 | a{3,}?　> aaa, aaaa… |  |
| { n, m } | 直前の文字の最小桁数と最大桁数を指定 | a{3,4}?  > aaa, aaaa |  |
| { n, m }? | 直前の文字の最小桁数と最大桁数を指定。最短一致。 | a{3,4}?  > aaa, aaaa |  |
|  |  |  |  |

### 拡張正規表現

linuxだとgrep -E かegrepを使用する必要がある（grepでは使用できない正規表現）

|  |  |  |  |
| --- | --- | --- | --- |
| 文字 | 意味 | 例 | 備考 |
| | | 左右いずれかの文字列 | abc|xyz |  |
| + | 直前の文字が １回以上 繰り返す場合  \* に少し似ている。\*は0回も対象 | a+  >(a,aa,aaa …) | 直前の文字の繰り返し  ⇒数量詞　とよぶ |
| ? | 直前の文字が **０個か１個** の場合 | windows?  >(window, windows) |  |
| \*? | 直前の文字が ０個か１個 の場合 |  | ？ |
| ?? | 直前の文字が ０個か１個 の場合。 |  | ？ |

### 定義済の正規表現（メタキャラクタ）

|  |  |  |  |
| --- | --- | --- | --- |
| 文字 | 意味 | 例 | 備考 |
| \t | 任意の１文字 |  |  |
| \r | 改行。CR。 |  |  |
| \n | 改行。LF。 |  |  |
| \d | 全ての数字 |  | also [0-9] |
| \D | 全ての数字以外の文字 |  | also [^0-9] |
| \s | 垂直タブ以外のすべての空白文字 |  | also [ \t\f\r\n] |
| \S | 全ての非空白文字 |  | also [^ \t\f\r\n] |
| \w | アルファベット、アンダーバー、数字 |  | also [a-zA-Z\_0-9] |
| \W | \w 以外 |  | also [^a-zA-Z\_0-9] |

### 特定の位置関係の正規表現

|  |  |  |  |
| --- | --- | --- | --- |
| 文字 | 意味 | 例 | 備考 |
| $ | 一致した文字列全体に置換。 |  | 単発で$を使うと行末文字を示す |
| ＼< | 単語の先頭 | \<i 　iで始まる語がある |  |
| ＼> | 単語の末尾 | \is> isで終わる語 |  |
| ＼b | 単語の先頭か末尾 |  |  |
| ＼B | 単語の先頭か末尾以外 |  |  |
| ＼A | ファイルの先頭 |  |  |
| ＼z | ファイルの末尾 |  |  |
| ＼G | 直前の一致文字列の末尾 |  |  |

置換文字列

|  |  |  |  |
| --- | --- | --- | --- |
| 文字 | 意味 | 例 | 備考 |
| \0 | 一致した文字列全体に置換。 |  |  |
| \1～\9 | 一致した文字列の1～9番目に対応する文字列に置換 |  |  |

### 具体例

#### 数字のみ使用する

'^[0-9]+$'

比較

|  |  |
| --- | --- |
| 正規表現 | 意味 |
| '[0-9]+' | 少なくとも１文字は数字が利用されている |

### 資料（正規表現）

参考資料：<https://murashun.jp/blog/20190215-01.html>

言葉の定義

特殊文字 | や \（エスケープ文字）のように特殊な意味を持つ文字

文字クラス [] 内の文字集合の事

数量詞 \* や + などのように直前の文字の繰り返し回数を示す文字

アンカー ^ や $ などのように文字列内での位置を示す文字

## float(浮動小数点)

### 概要

float の最大値 = 3.402823e+38

float の最小値 = -3.402823e+38

float の浮動小数点で表現できる最小値 = 1.175494e-38

※このe+38と言う表現は10のべき乗の事。

### 計算の仕方

floatは符号部1bit、指数部8bit、仮数部23bitからなる。

・値を2進数化する。簡単なのは整数部分と小数部分に分けて考える事。（例）10.625 ⇒ 10 + 0.625

整数部は普通に2進数に変換する 10 = 23＋21 ⇒ 1010

小数部も似たような考え 0.625 = 2-1＋2-3 ⇒ 0.101

・整数部が１になるよう小数点の位置をずらす。 1010.101　⇒ 1.010101

⇒ 言い換えると 1010.101 = 1.010101 × 23 と言う事。（指数部の値）

・指数部は127 (0111 1111) を0とし、負の値も表現する。その為、127 + 指数部の値を2進数に変換する。⇒ 127 + 3 = 130 = 27 + 21 ⇒　1000 0010 となる。

仮数部の最初の1は必ず１になる為、最終的に取り払われ、符号部 0 (=正の値)，指数部 1000 0010，仮数部 010101 と言った形になる。⇒ 0100 0001 0010 1010 0000…(トータル32bit分)

Mitsubishi MXComponent向きに作ったC#ライブラリに、実際に計算を行った拡張メソッドがある

public static int SetDevice(this ActProgTypeLib.ActProgTypeClass act, string szDevice, float lplData)

## データ系ファイル

### json

### yaml

インデントする事でネストされた項目を示す。

加えて、以下が主な特徴

配列 行頭の「-」 は

ハッシュ キー: 値

スカラー 文字列からデータ型の自動判別 例：yes→bool 2005-01-01→date

コメント #　でコメントを入れられる。（jsonはコメントが入れられない）

### XAML

XAML（Extensible Application Markup Language）

・アプリの画面を定義

・XML の一種

・WPF : Windows Presentation Foundation

・VS 用のテンプレートがあるらしい

<http://bloggemdano.blogspot.jp/2014/11/evolution-of-f-empty-wpf-template.html>

・VS の新規プロジェクト作成時に、 オンライン → F# WPF で検索

・VS のデザイナは使えるが、イベント登録は自力でやる必要がある

・一部の Windows.Control が使えない e.g. DataGridView,

### Markdown

厳密には「データ系」というよりもWebブラウザで表示されるReadmeなどに良く使われるファイル

#### 表示アプリ

Chrome - Markdown Preview Plus

インストール後 > chrome://extensions > 詳細 > ファイルのURLへのアクセスを許可する　をON

## 開発手法

### ペアプログラミング

ソフトウェアの品質を高めるために，2人のプログラマが協力して，一つのプログラムをコーディングする。

### プロトタイピング

利用者からのフィードバックを得るために，提供予定のソフトウェアの試作品を早期に作成する。

### テスト駆動開発

動作するソフトウェアを迅速に開発するために，テストケースを先に設定してから，プログラムをコーディングする。

### リファクタリング

ソフトウェアの保守性を高めるために，外部仕様を変更することなく，プログラムの内部構造を変更する。

正しい。

XP(eXtreme Programming)で

## uuid

### 概要

バージョン 説明

1 時刻とMACアドレスを元にするバージョン

2 DCEセキュリティバージョン

3 バイト列を元にするバージョン。MD5ハッシュを使用

4 ランダム生成するバージョン

5 バイト列を元にするバージョン。SHA-1ハッシュを使用

### ulid

ソート可能なuuid

## 環境変数設定ファイル

概要

特定のアプリケーションで使う環境変数を管理するファイル

目的

・ソースコードから機密情報を分離する

・設定の一元管理。バージョン管理。

注意点

・公開レポジトリに含まれないようにする

具体例

|  |  |
| --- | --- |
| ファイル名 | 概要 |
| .env | 一般的な環境変数設定ファイル名 |
| dotenv | 1 |
| config/secrets.yaml | Ruby on Railsの場合 |

# CPU関連

## CPU処理区分の違い

プロセス

CPU1コアに対して1つ。他のプロセスとメモリを共有しない。

１つ以上のスレッド、ファイル、ヒープメモリなどで構成される。

最も独立性の高い実行単位だが，起動や切り替えに時間がかかる。

スレッド

プロセス内にはいくつものスレッドが存在。（サブセット）

一連のスレッド間ではメモリを共有する。

ファイバ

スレッドのサブセット。スレッドを更に軽量化したもの。

タスク

プロセス，スレッドと同意味で使われる事もあり（誤用）ややこしい。

スレッドで処理される事が多いが，タスク自体は単に１つの処理。

ジョブ

LinuxOSにおいて一連の複数のプロセスをまとめたもの。

ジョブと呼ばれるのはUpstartというシステムを使っているものだけで，Systemdを使っている場合にはユニットと呼ばれる。

## 命令

一般にCPUの演算は以下の流れで行われる

命令フェッチ

プログラムカウンタを参照して、次に実行すべき命令が格納されているアドレスを知る。

主記憶から命令レジスタに命令を読み込む

その次に実行する命令の主記憶上のアドレスをプログラムカウンタに格納。

命令デコード

命令レジスタの命令を命令デコーダで解読

実行アドレス計算

デコードされた命令語のオペランド部の値を用いて、演算対象のデータを保持する主記憶のアドレスを計算する。

オペランドフェッチ

計算されたアドレス値を用いて、演算で必要になる値（オペランド）を主記憶やレジスタから読み込む。

実行

命令演算を実行。実行結果を主記憶やレジスタに書き込む

## キャッシュメモリ

ライトスルー方式(write through)

キャッシュメモリと主記憶の両方を書き換える。

データの整合性は得られるが処理速度は低下する。

ライトバック方式

キャッシュメモリだけに書き込む方式。主記憶への書き込みはキャッシュメモリからデータが追い出されるときに行われる。

一貫性を保つための制御が複雑になるが、処理速度は向上。

## その他用語

ディスパッチ

実行可能なタスクに対してプロセッサの使用権を割り当てること

アキュームレータ

演算途中の結果を一時的に保持しておくレジスタ

# フレームワーク

## 概要

## Flutter

## React Native

## .NET

### 概要

.NET Coreの後継にあたり，.NET Frameworks(4.8が最終)と.NET Coreの両方を並立して使用する事ができる。

従来の.NET系とは異なり，OSのコンポーネントの一部としては動作しない。

Visual Studioでは「.NET Portability Analyzer」を用いる事で.NET5.0への互換性を調べる事ができる。

### .NET (5以降)に統合される技術

|  |  |
| --- | --- |
| 技術 | 備考 |
| WPF | リッチなUIを作れるフレームワーク。XAMLやMVVMなどは他の技術にも使われているので押さえておくべき。 |
| UWP | Windows用アプリを作るためのプラットフォーム。UIだけでなく、いろいろ便利なAPIセットが追加されている。(昔WinRTって言われてたやつ) |
| Xamarin | .NETでマルチプラットフォーム開発を行うためのプラットフォーム。.NET6以降に載る予定の次世代UIフレームワーク.NET MAUIはXamarin.Formsベースの模様。 |
| .NET Core | OSSでクロスプラットフォームな.NET実装。今後はこいつが.NET5に統合されて生き残ることになる。 |

（参考）<https://qiita.com/nendocode/items/f132745ea2ccb3c68fb9>

# プログラム言語

## 概要

|  |  |  |  |
| --- | --- | --- | --- |
| 言語 | 運営 | 概要／用途 | 備考 |
| C# | Microsoft | Windowsアプリ、ゲーム開発 |  |
| Dart | Google | モバイルアプリ開発（Flutter）、Web開発 |  |
| Go | Google | Webサーバー開発、クラウドコンピューティング |  |
| Java | Oracle | アプリケーション開発、Androidアプリ開発 |  |
| JavaScript | Ecma International | Web開発、フロントエンド開発 |  |
| Swift | Apple | iOSのネイティブアプリ開発向きの言語 |  |
| Rust | Mozilla | システムプログラミング、高性能な並行処理、安全なソフトウェア開発 |  |
| Swift | Apple | iOSアプリ開発 |  |
| TypeScript | Microsoft | Web開発、JavaScriptの拡張 |  |
| Python | Python Software Foundation | データ分析、人工知能、Web開発 |  |
| Ruby |  | Web開発、スクリプト言語 |  |
|  |  |  |  |

## Rust

### 概要

Mozillaが支援するオープンソースのプログラム言語。書き方はpythonに似ている。拡張子.rs

コンパイル型言語だが，スクリプト型での実行も可能。

マルチパラダイムプログラミング言語で，手続き型プログラミング、オブジェクト指向プログラミング、関数型プログラミングなどの実装手法をサポート。

ネイティブに近い環境で実行される（実行環境はC言語などに近い）為，デバイスドライバなどの低レベル層のソフトウェア開発が向いている。

### cargo

rustのビルドシステムであり、パッケージマネージャ。

コードのビルドやコードが依存しているライブラリのダウンロードや、それらのライブラリのビルドを行ってくれます。

参考：<https://qiita.com/yoshii0110/items/6d70323f01fefcf09682>

# ミニPC

## ODYSSEY - X86J4105

Arduinoとラズパイ互換GPIOを搭載した、Celeron J4105搭載ボード

メインストレージ：64GBeMMC SATA x 1

サポートOS

Windows 10 Enterprise

Ubuntu

OpenWrt

# 用語

## 画像系

## その他

UEFI；Unified Extensible Firmware Interface

BPM；Business Process Management

企業内の業務の流れを可視化し，業務改善サイクルを適用することによって，継続的な業務改善を図る手法

EAI；Enterprise Application Integration  
企業内の異なるシステムを互いに連結し，データやプロセスの統合を図ることによって，システムを全体として効率よく活用する手法

BPR；Business Process Reengineering

企業内の慣行などにとらわれず，業務プロセスを抜本的に再構築することによって，コスト・品質・サービスなどを改善する手法

ODD；Optical Disc Drive 光ディスクドライブ

オーバレイ

必要な部分を補助記憶装置から読み込みながら動作する。主記憶領域の大きさに制限があるときに，有効な手法。

アルファ版／ベータ版

開発段階：プロトタイプ版→アルファ版(α版) →ベータ版(β版)→マスターアップ版

α版 α版はまだ機能が足りない試作品

β版 動作は不安定だが、使ってもらえるレベルではある