01-01. ソフトウェアの具体例

◆ OSS：Open Source Softwareとは

以下の条件を満たすソフトウェアをOSSと呼ぶ。

1. 利用者は、無償あるいは有償で自由に再配布できる。
2. 利用者は、ソースコードを入手できる。
3. 利用者は、コードを自由に変更できる。また、変更後に提供する場合、異なるライセンスを追加できる。
4. 差分情報の配布を認める場合には、同一性の保持を要求してもかまわない。 ⇒ よくわからない
5. 提供者は、特定の個人やグループを差別できない。
6. 提供者は、特定の分野を差別できない。
7. 提供者は、全く同じOSSの再配布において、ライセンスを追加できない。
8. 提供者は、特定の製品でのみ有効なライセンスを追加できない。
9. 提供者は、他のソフトウェアを制限するライセンスを追加できない。
10. 提供者は、技術的に偏りのあるライセンスを追加できない。

◆ OSSの具体例

ALL	OS	データベース	言語	Web サーバ	プロキシ サーバ	AP サーバ	フレームワーク	OR マッピング	ログ管理	SOAP
ビジネスプロセス	SOA	DNS	ファイルサーバ	認証サーバ	メールサーバ	POP3 IMAP	バージョン管理	クラスタリング	シングルサインオン	ID管理
運用監視	BIレポート作成	ポータル CMS 文章管理	グループウエア	オフィススイート	業務システム	ネットワーク	クラウド構築	全文検索エンジン	インフラ構築	機械学習 AI
ブロックチェーン	Java ユーティリティ	その他								

引用：https://openstandia.jp/oss_info/

- **OS**

  CentOS、Linux、Unix、Ubuntu

- **データベース**

  MySQL、MariaDB

- **プログラミング言語**

  言うまでもない。

- **フレームワーク**

  言うまでもない。

- **OR Mapper**

  言うまでもない。

- **バージョン管理**

Git、Subversion

- **Webサーバ**

  Apache

- **業務システム**

  Redmine

- **インフラ構築**

  Chef、Puppet

- **クラウド構築**

  Docker

◆ Linux Distribution（Linuxの種類）

現在、Linuxは3つを源流として、いくつもの派生系へ分岐している。

- **RedHat系**

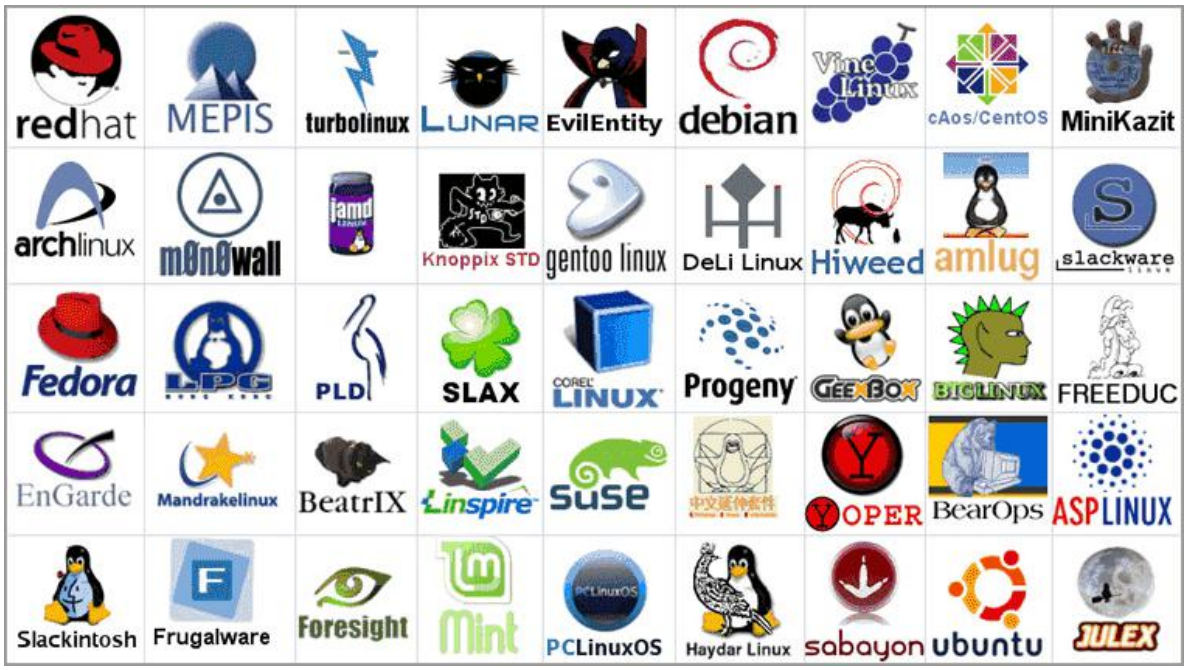
RedHat、CentOS、Fedora

- **Debian系**

Debian、Ubuntu、

- **Slackware系**

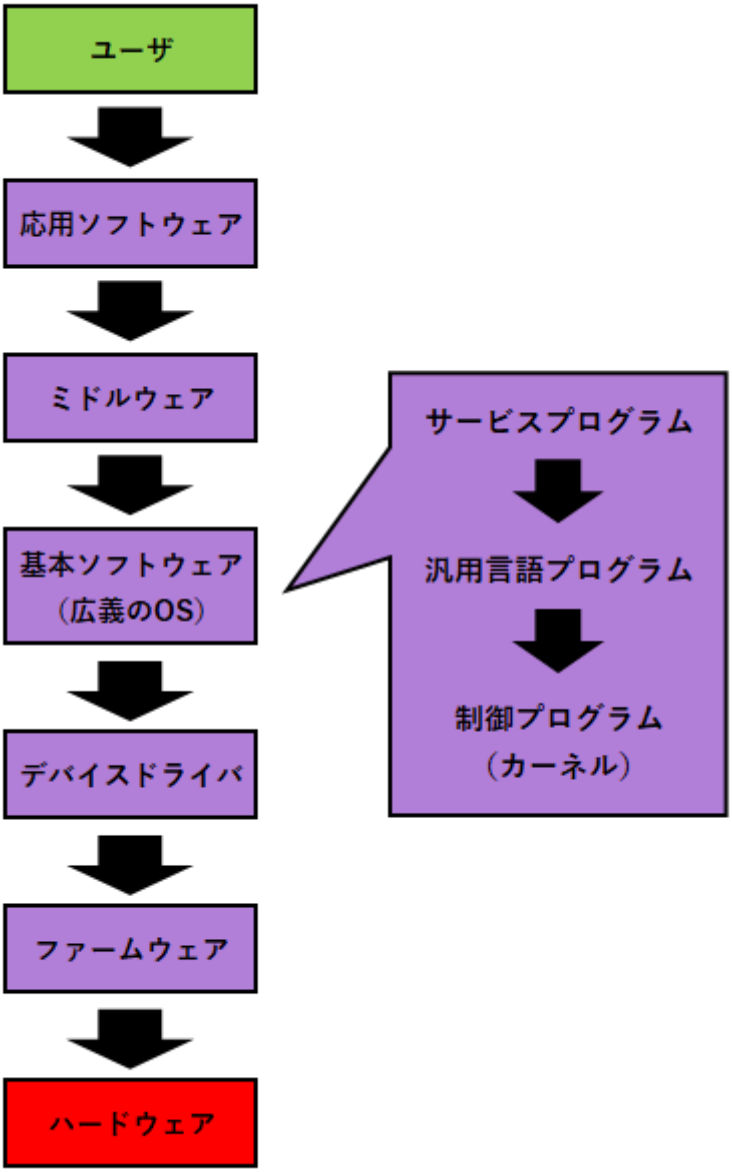
Slackware



01-02. ソフトウェアについて

◆ ユーザの操作が、命令としてハードウェアに伝わるまで

ソフトウェアには、応用ソフトウェアとシステムソフトウェアがある。



◆ 応用ソフトウェア

【具体例】

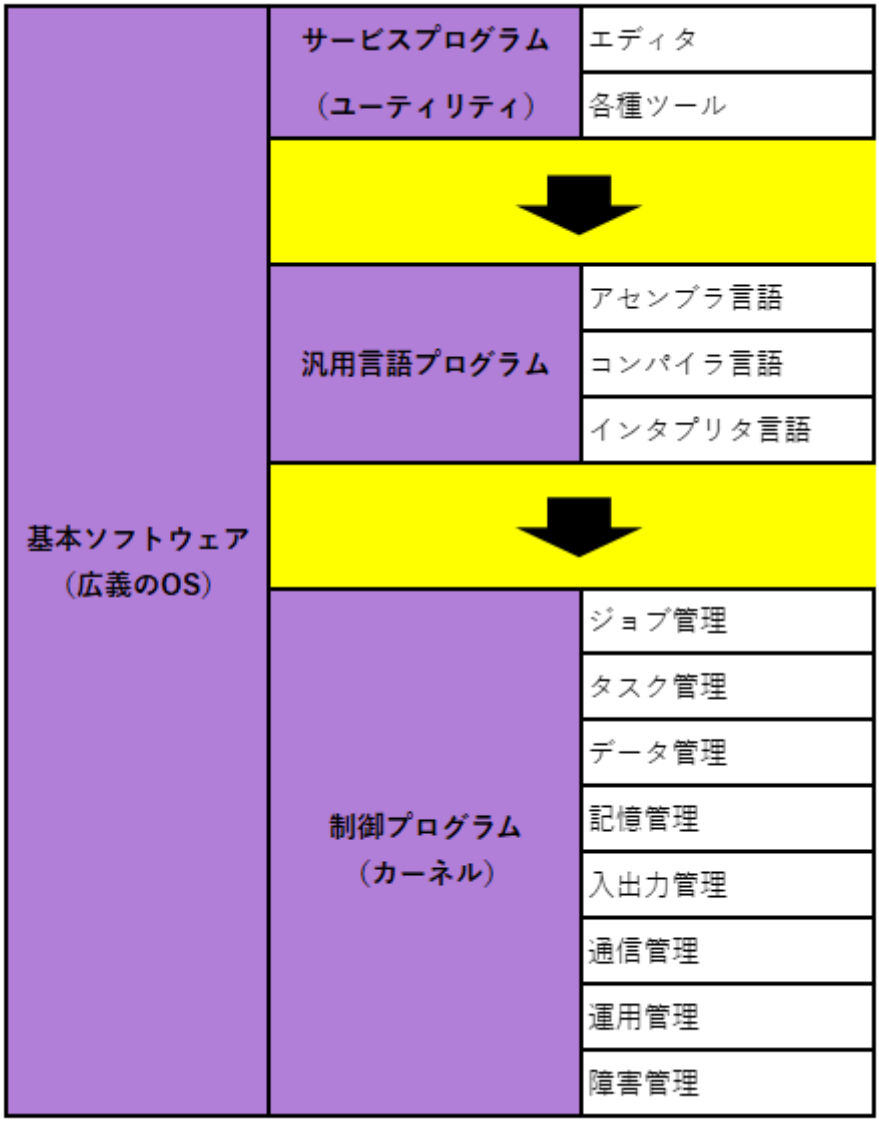
その辺のアプリ

◆ Middleware

【具体例】

データベース管理システム

◆ 基本ソフトウェア（広義のOS）



- **制御プログラム（：別名カーネル）**

  【具体例】

  カーネル、マイクロカーネル、モノリシックカーネル
- **汎用言語プログラム**

  【具体例】

  C、Java、PHP、Javascript、などで実装されたプログラム
- **サービスプログラム**

  【具体例】

  ファイル圧縮プログラム

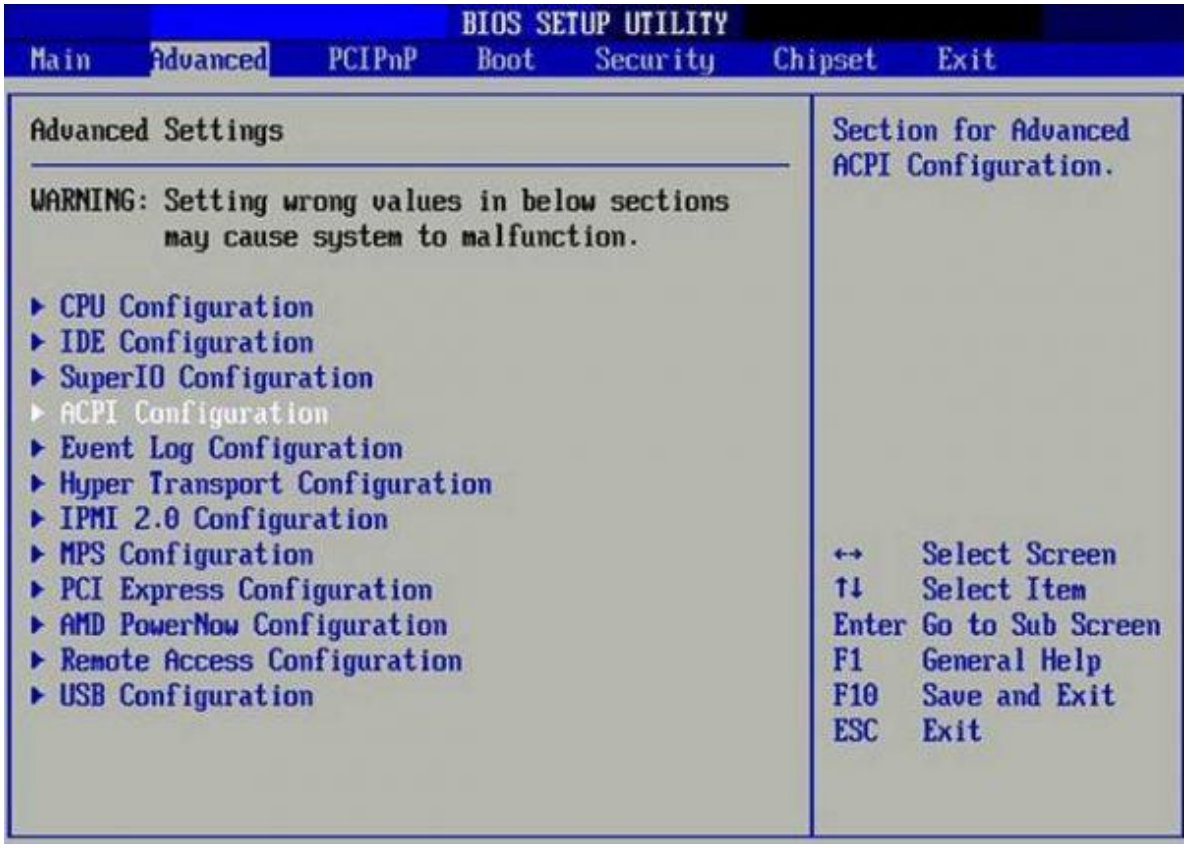
◆ デバイスドライバ

◆ Firmware

システムソフトウェア（ミドルウェア + 基本ソフトウェア）とハードウェアの間の段階にある。

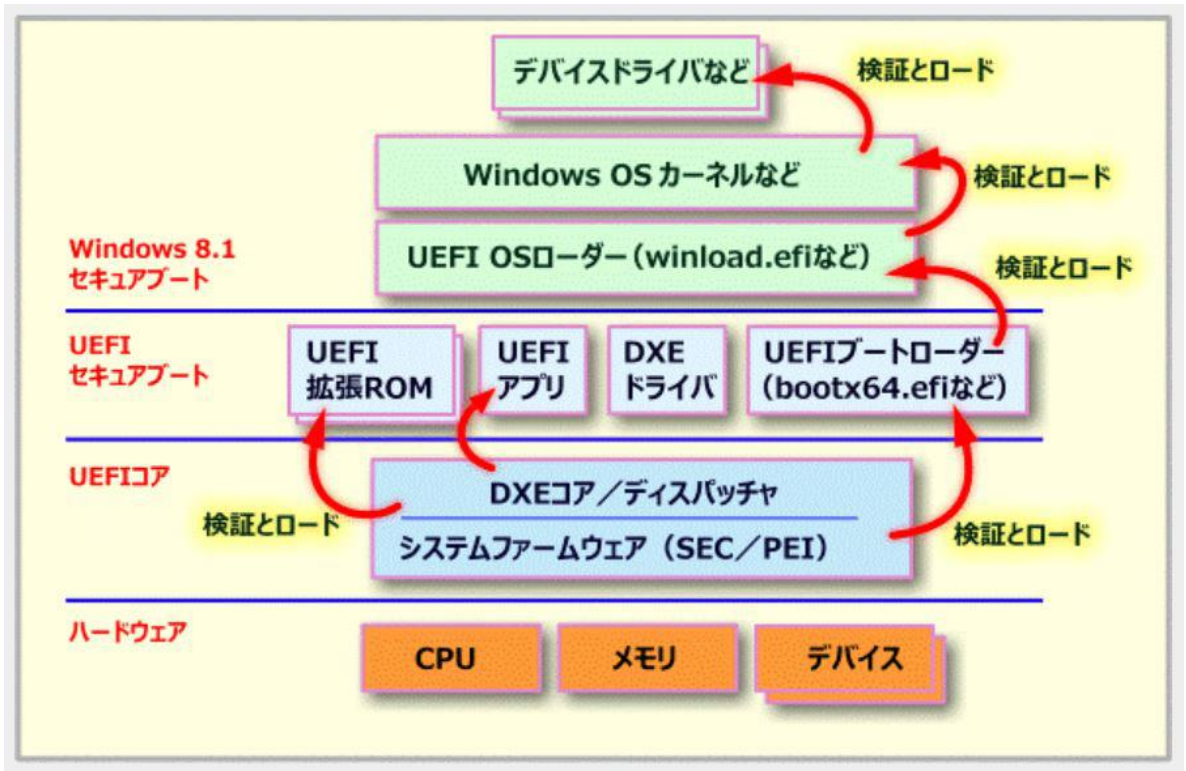
【具体例】

BIOS：Basic Input/Output System



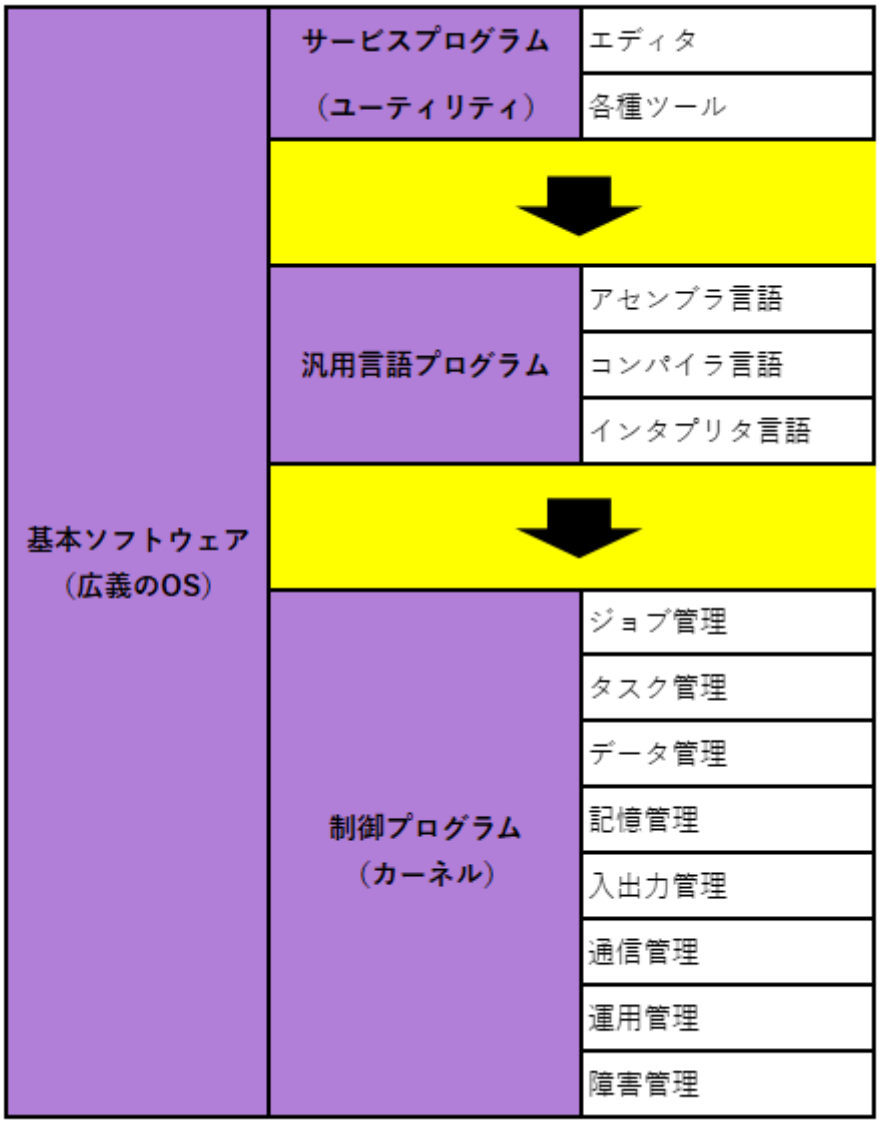
【具体例】

UEFI：United Extensible Firmware Interface



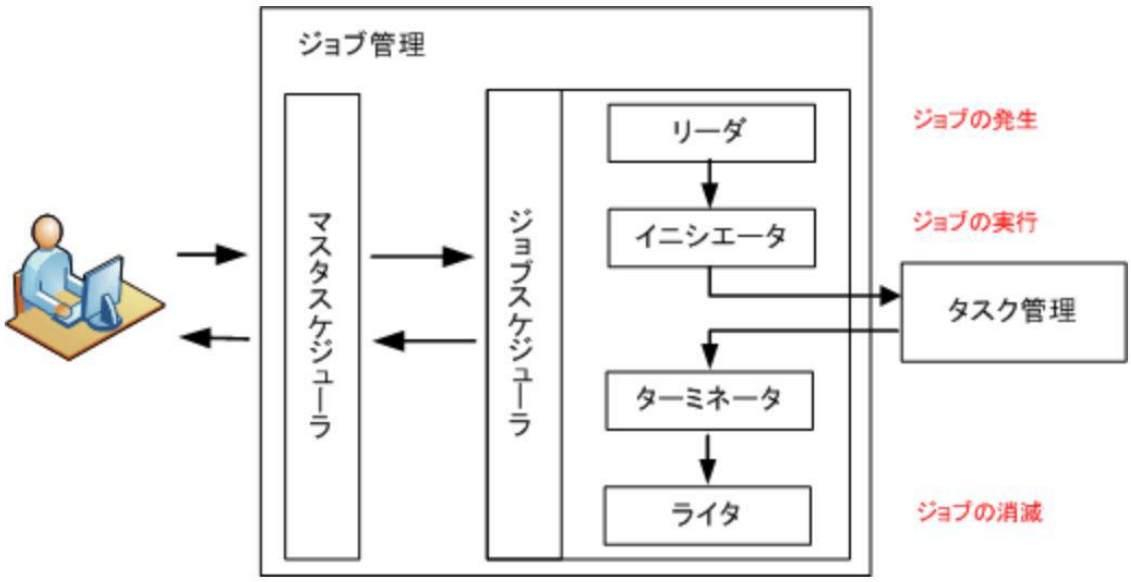
01-03. 基本ソフトウェアの制御プログラム（カーネル）

◆ 基本ソフトウェア（再掲）



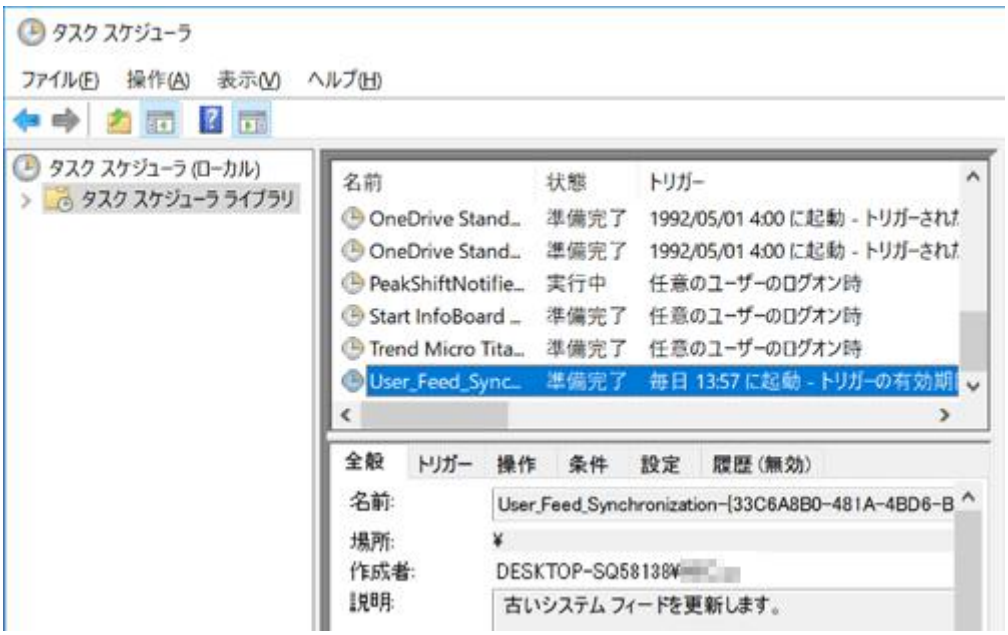
◆ ジョブ管理機能

クライアントは、マスタスケジュールに対して、ジョブを実行するための命令を与える。



【具体例】

Windowsのタスクスケジューラでは、決められた時間または一定間隔でプログラムやスクリプトを実行することができる。



◆ マスタスケジュラと、ジョブスケジュラ（タスクスケジューラ）

ジョブとは、プロセスのセットのこと。マスタスケジュラは、ジョブスケジュラにジョブの実行を命令する。データをコンピュータに入力し、複数の処理が実行され、結果が出力されるまでの一連の処理のこと。『Task』と『Job』の定義は曖昧なので、『process』と『set of processes』を使うべきとのこと。

引用：https://stackoverflow.com/questions/3073948/job-task-and-process-whats-the-difference/31212568

複数のジョブ（プログラムやバッチ）の起動と終了を制御したり、ジョブの実行と終了を監視報告するソフトウェア。ややこしいことに、タスクスケジューラとも呼ぶ。

- **Reader**

ジョブ待ち行列に登録

- **Initiator**

ジョブステップに分解

- **Terminator**

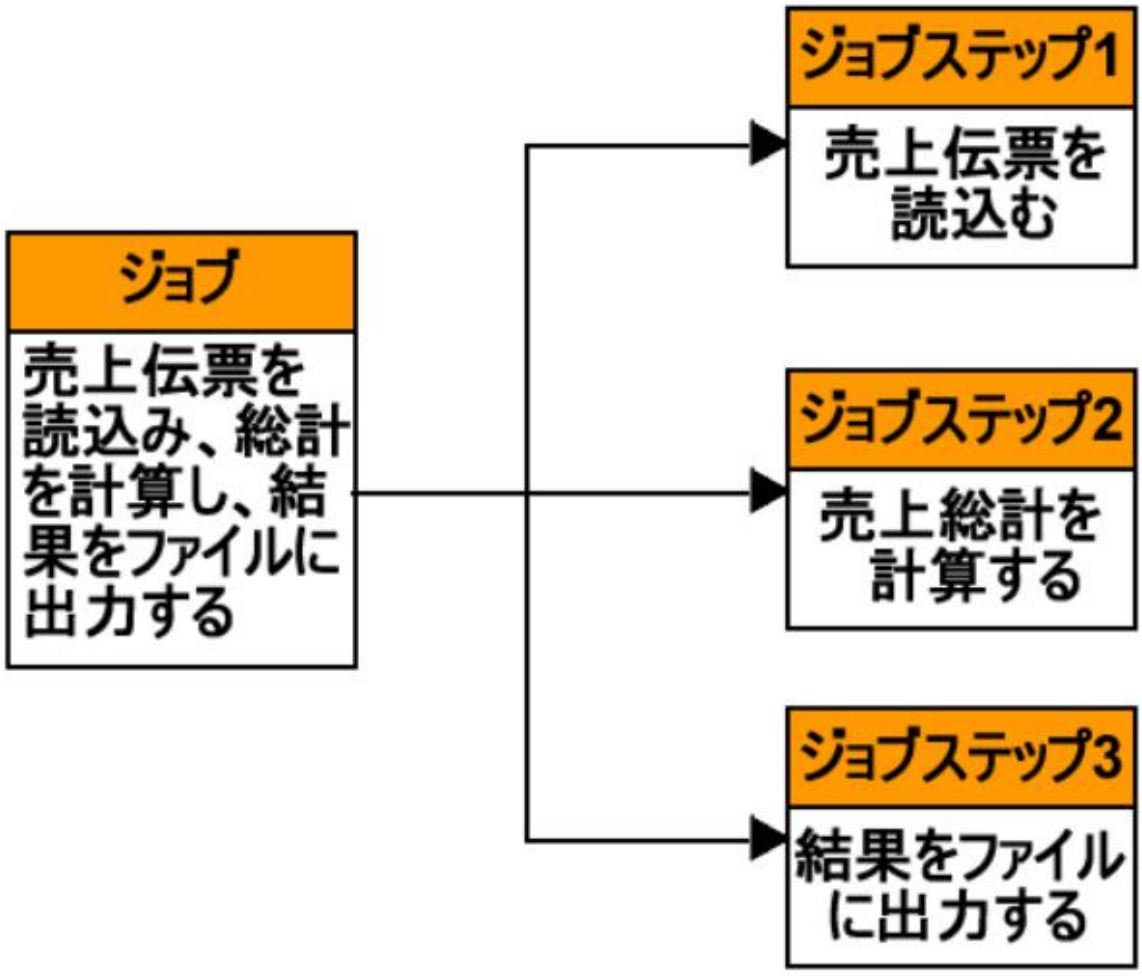
出力待ち行列に登録

- **Writer**

優先度順に出力の処理フローを実行

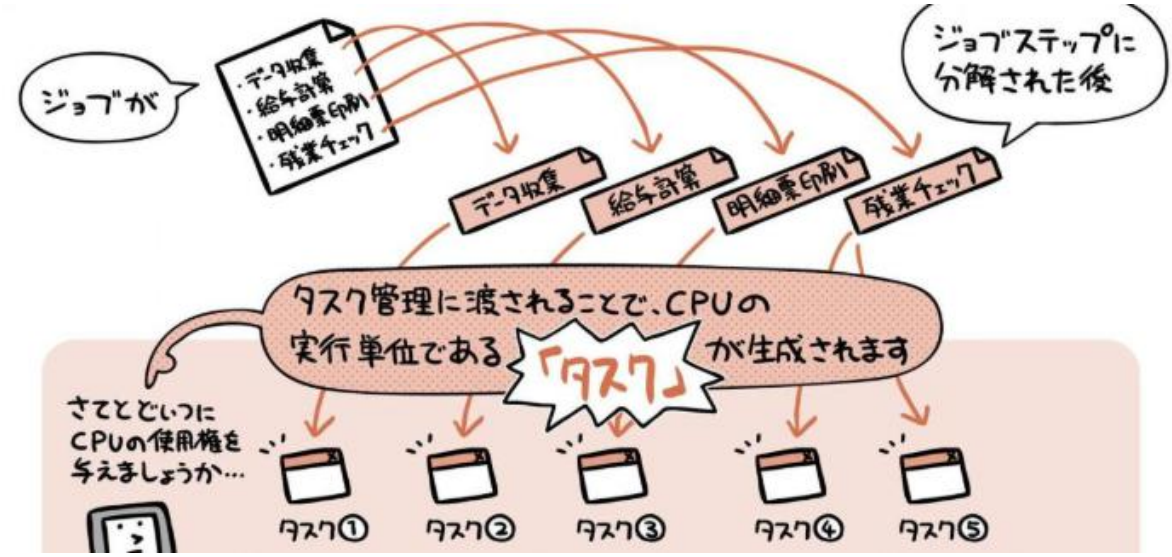
◆ Initiatorによるジョブのジョブステップへの分解

Initiatorによって、ジョブはジョブステップに分解される。



◆ タスク管理機能

タスクとは、スレッドに似たような、単一のプロセスのこと。Initiatorによるジョブステップから、タスク管理機能によって、タスクが生成される。



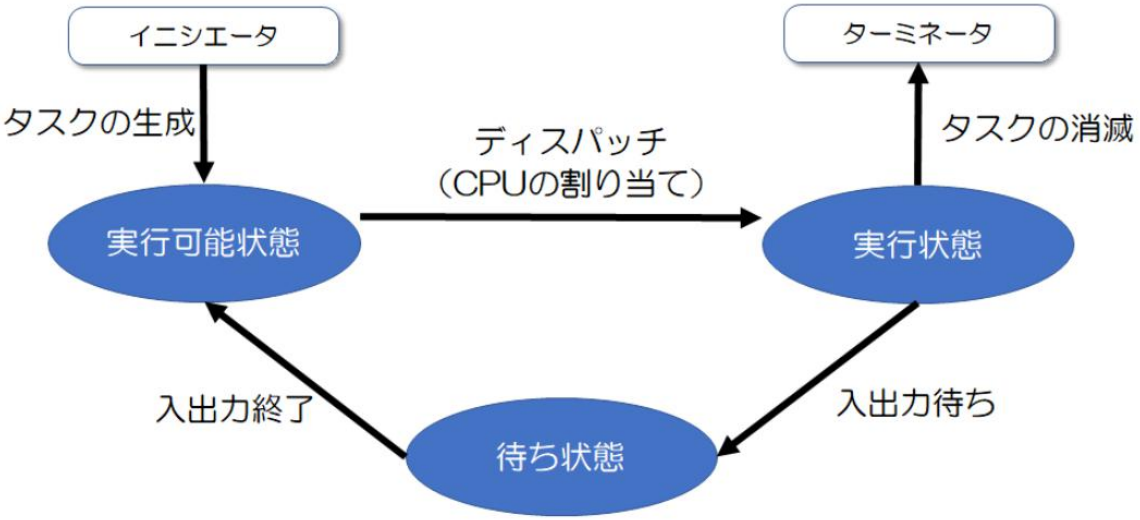
タスクが生成されると実行可能状態になる。ディスパッチャによって実行可能状態から実行状態になる。

- **優先順方式**

各タスクに優先度を設定し、優先度の高いタスクから順に、ディスパッチしていく方式。

- **到着順方式**

  実行可能な状態になったタスクから順に、ディスパッチしていく方式。



	処理内容	遷移前の状態	遷移後の状態
①	前処理（CPUを使用）	実行可能	実行
②	データの入力	実行	待ち
↓	データ入力の完了	待ち	実行可能
③	計算処理（CPUを使用）	実行可能	実行
④	データの出力	実行	待ち
↓	データ出力の完了	待ち	実行可能
⑤	後処理（CPUを使用）	実行可能	実行

- **Round robin 方式**

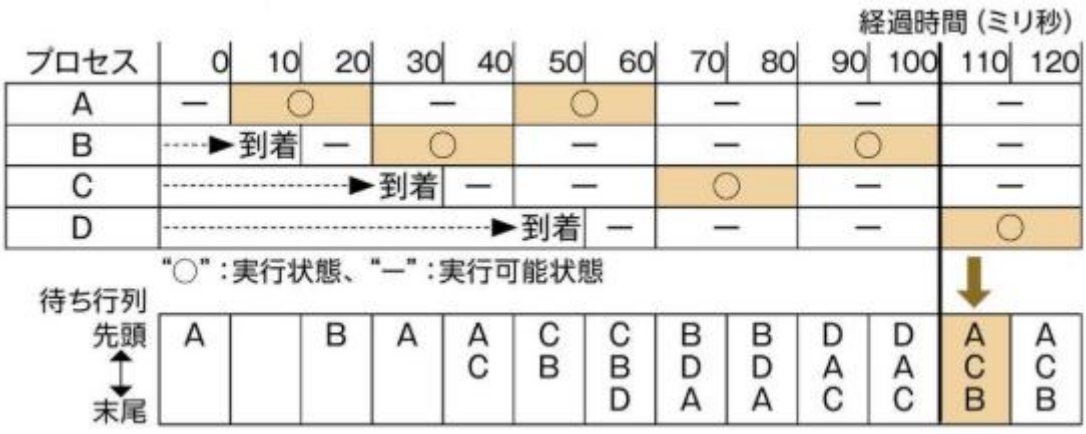
Round robinは、『総当たり』の意味。一定時間（タイムクウォンタム）ごとに、実行状態にあるタスクが強制的に実行可能状態の待ち行列に登録される。交代するように、他のタスクがディスパッチされる。

【具体例】

生成されたタスクの到着時刻と処理時間は以下のとおりである。強制的なディスパッチは、『20秒』ごとに起こるとする。

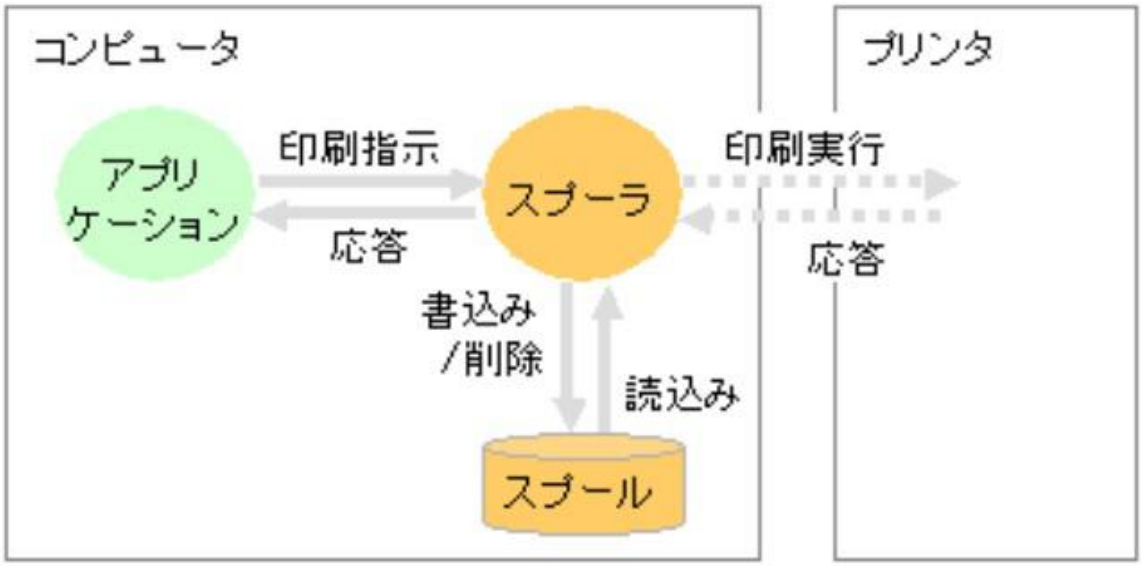
プロセス名	到着時刻（ミリ秒）	処理時間（ミリ秒）
A	0	120
B	10	90
C	30	60
D	50	30

20秒時点で、Aは、実行状態から実行可能状態の待ち行列に登録される。続けて、Bは、逆に実行可能状態から実行状態にディスパッチされる。



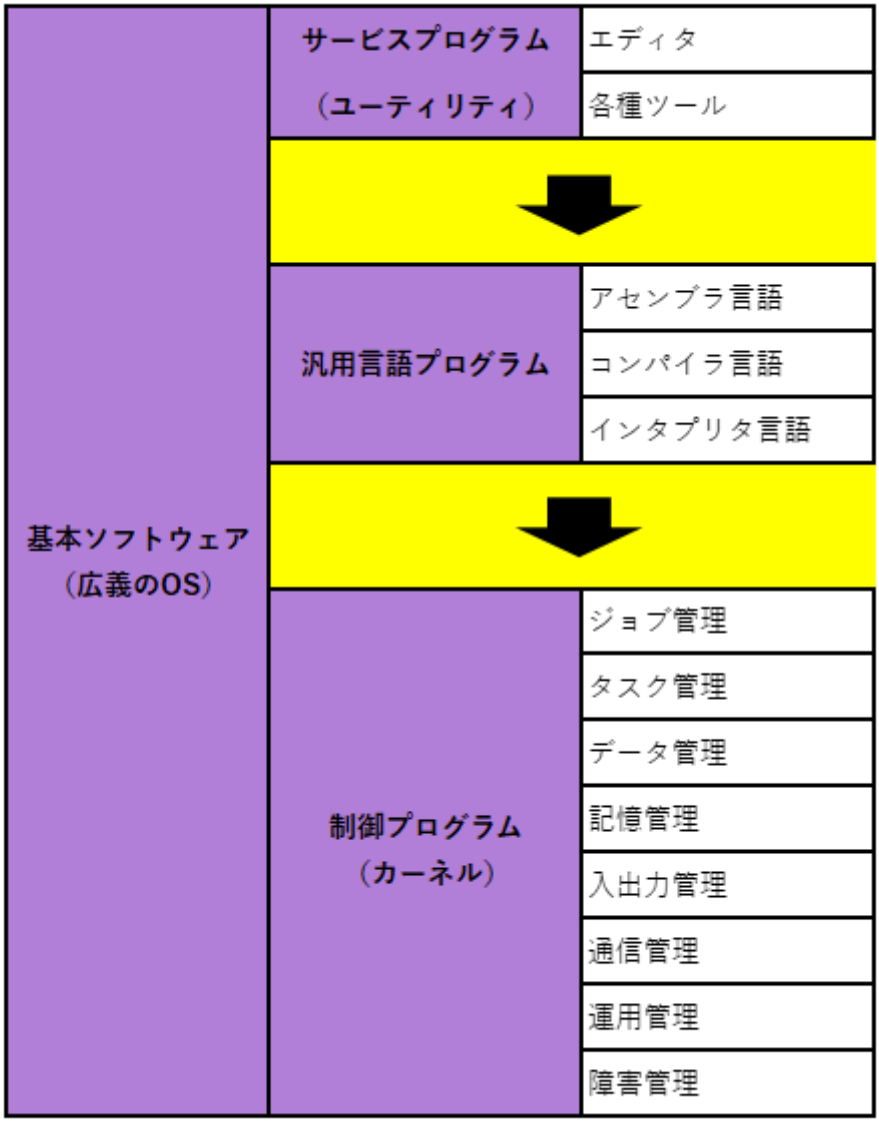
◆ Spooling

アプリケーションから低速な周辺機器へデータを出力する時、まず、CPUはスプーラにデータを出力する。Spoolerは、全てのデータをまとめて出力するのではなく、一時的に補助記憶装置（Spool）にためておきながら、少しずつ出力する（Spooling）。



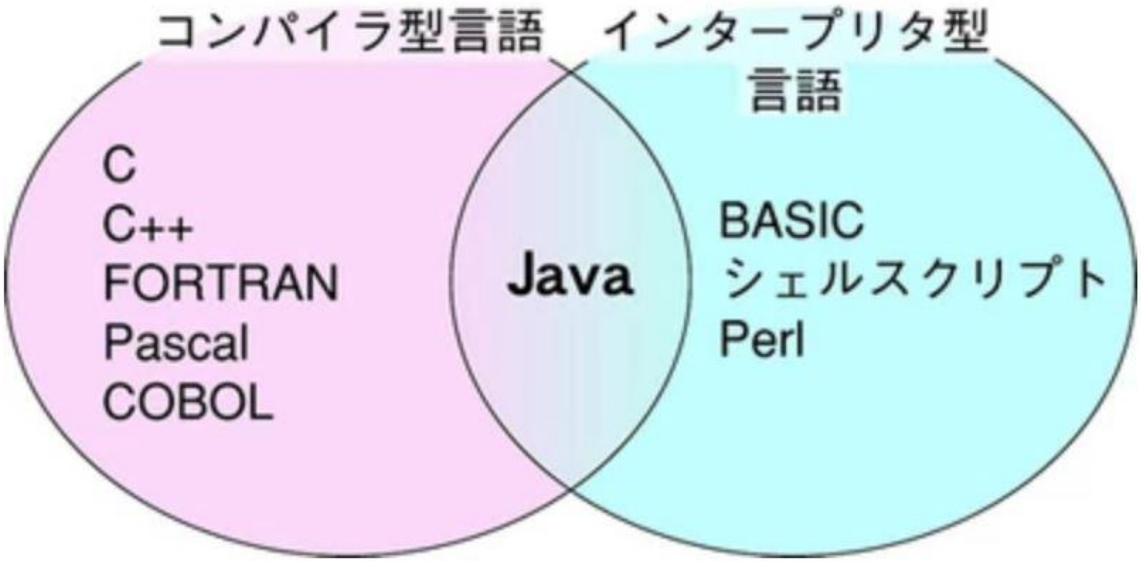
01-04. 基本ソフトウェアの汎用言語プログラム

◆ **基本ソフトウェア（再掲）**



◆ Compier言語とInterpreter言語の種類

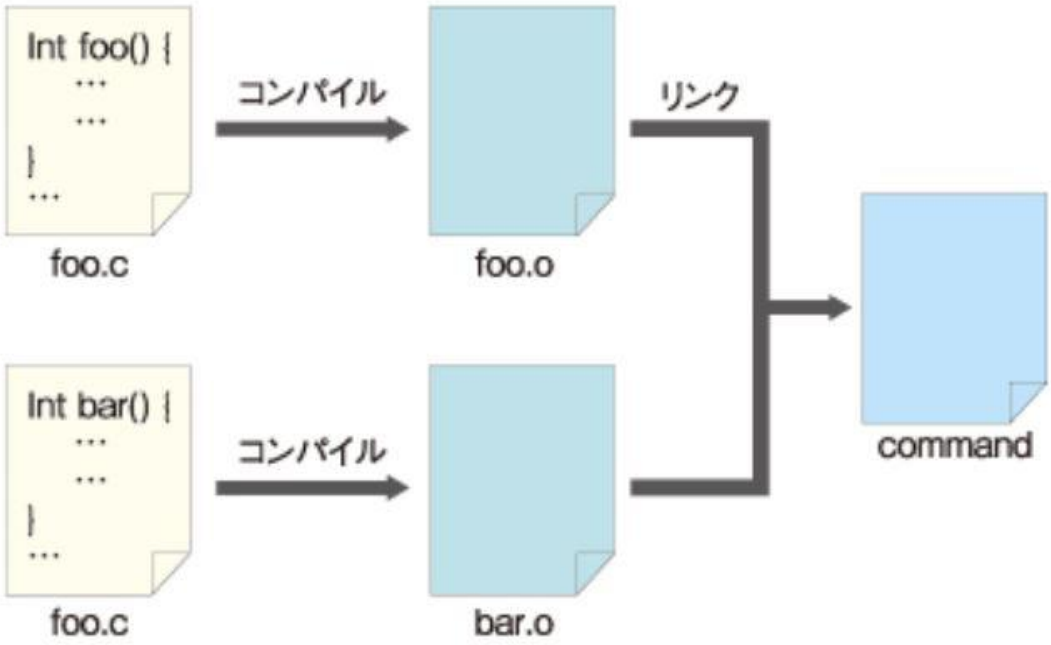
プログラム言語（ウェブサイトならJava、php、Javascriptなど）は、機械語に変換された後、CPUによって読み込まれる。そして、ハードウェア（ウェブサイトならパソコン）のCPUによって、ソースコードに書かれた様々な処理が実行される。



◆ ビルド

Compier言語やInterpreter言語では、ソースコードは機械語からなるオブジェクトコードに変換される（インタプリタ言語であっても、このプロセスはコンパイルと言うらしい...）。コンパイル後に、各オブジェクトコードは、ライブラリにリンクされる。この一連のプロセスを『ビルド』という。

※Vue.jsを使用するためには『asset:build』や『asset:watch』が必要であるが、まさにこのため。

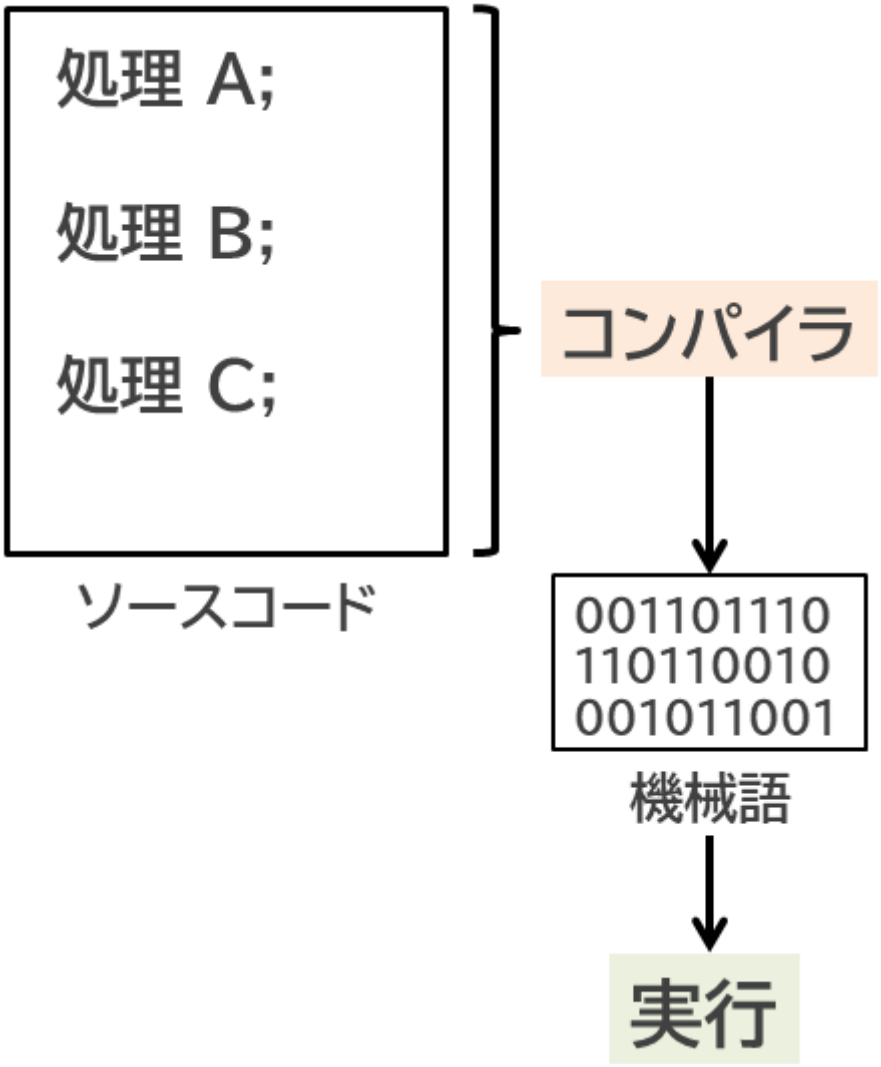


◆ Compiler言語による命令までの流れ

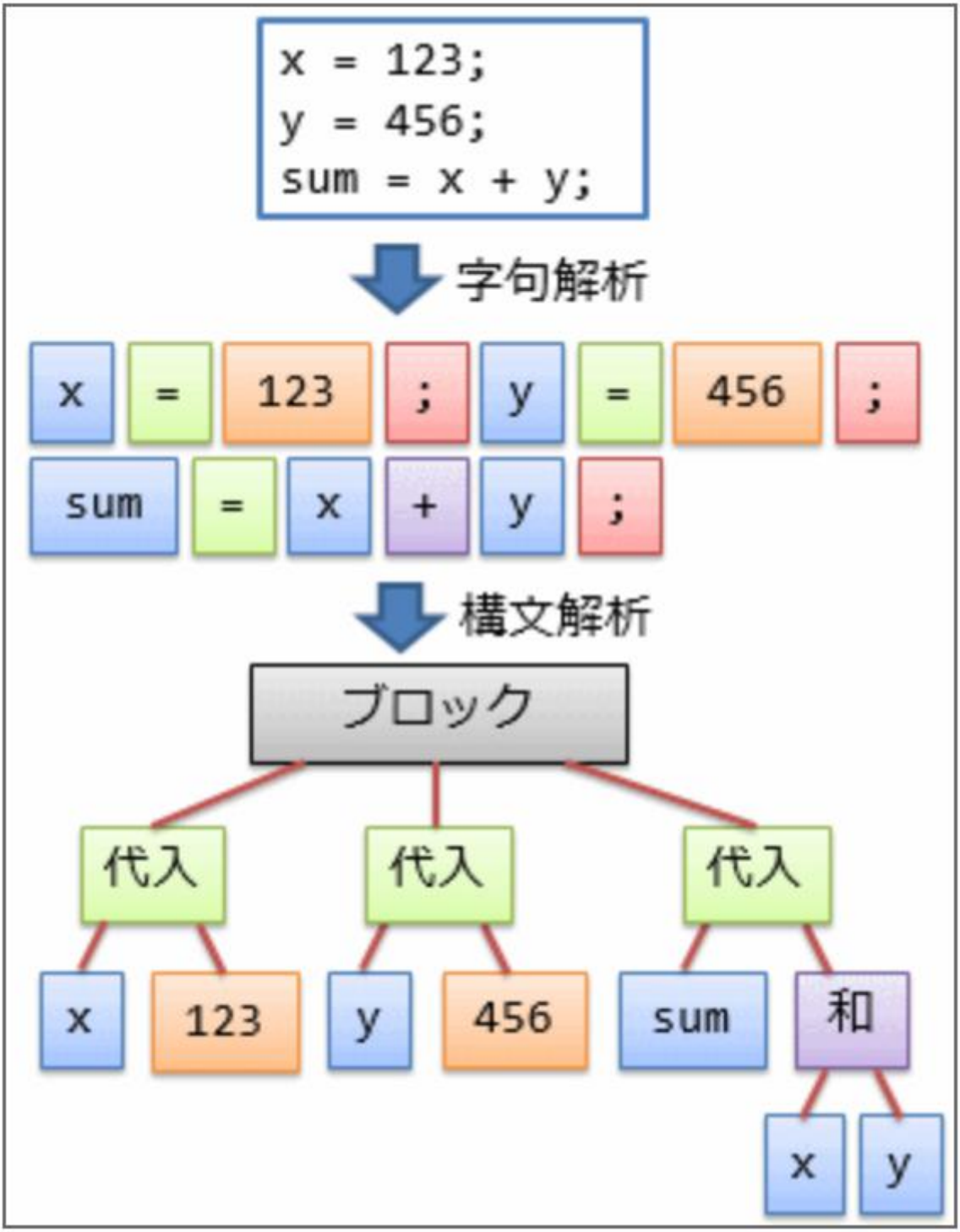
コードを、バイナリ形式のオブジェクトコードとして、まとめて機械語に翻訳した後、CPUに対して命令が実行される。

【具体例】

C#



- コンパイルの詳しい流れ



1. **Lexical analysis（字句解析）**

   ソースコードの文字列を言語の最小単位（トークン）の列に分解。

	構文規則	説明
(1)	符号なし浮動小数点定数 →小数点定数［指数部］｜数字列 指数部	符号なし浮動小数点定数は、「小数点定数 指数部」、「小数点定数」、または「数字列 指数部」のいずれかである
(2)	小数点定数→［数字列］.数字列｜数字列.	小数点定数は、「数字列.数字列」、「.数字列」、 または「数字列.」のいずれかである
(3)	指数部 → e［符号］数字列	指数部は、「e数字列」か「e 符号 数字列」である
(4)	数字列 → 数字｜数字列 数字	数字列は、「数字」か「数字列 数字」である
(5)	符号 → ＋｜－	符号は、「＋」か「－」である
(6)	数字 → 0｜1｜2｜3｜4｜5｜6｜7｜8｜9	数字は、0、1、…、9のいずれかである

2. **Syntax analysis（構文解析）**

トークンの列をツリー構造に変換。

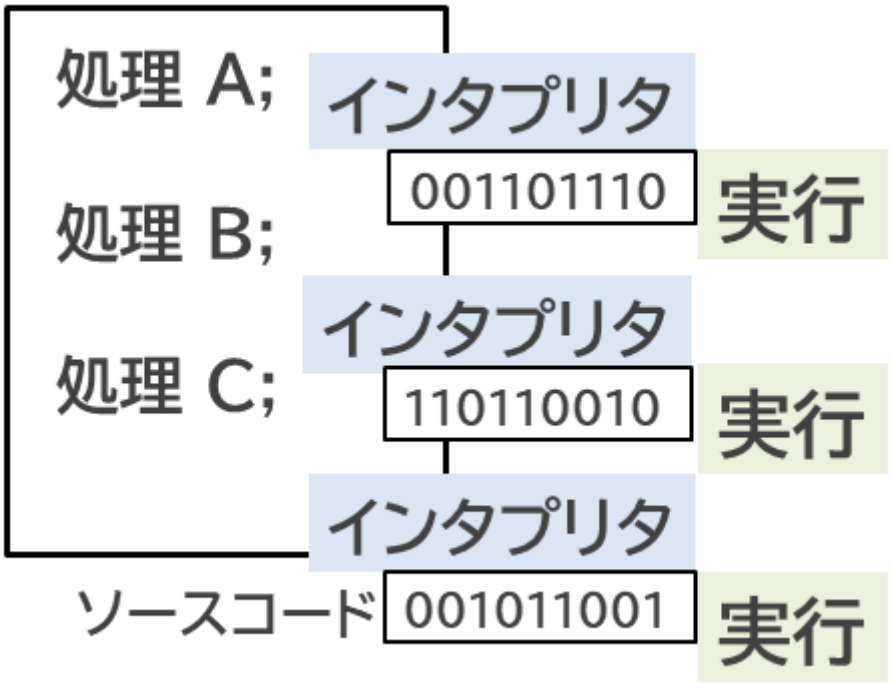
3. **Semantics analysis（意味解析）**
   ツリー構造を基に、ソースコードに論理的な誤りがないか解析。
4. **Code optimization（コード最適化）**
   ソースコードの冗長な部分を削除または編集。機械語をより短くするこができる。
5. **Code generation（コード生成）**
   最適化されたコードをバイナリ形式のオブジェクトコードに変換。
6. **命令の実行**
   オブジェクトコードを基に、命令が実行される。

◆ Interpreter言語による命令までの流れ

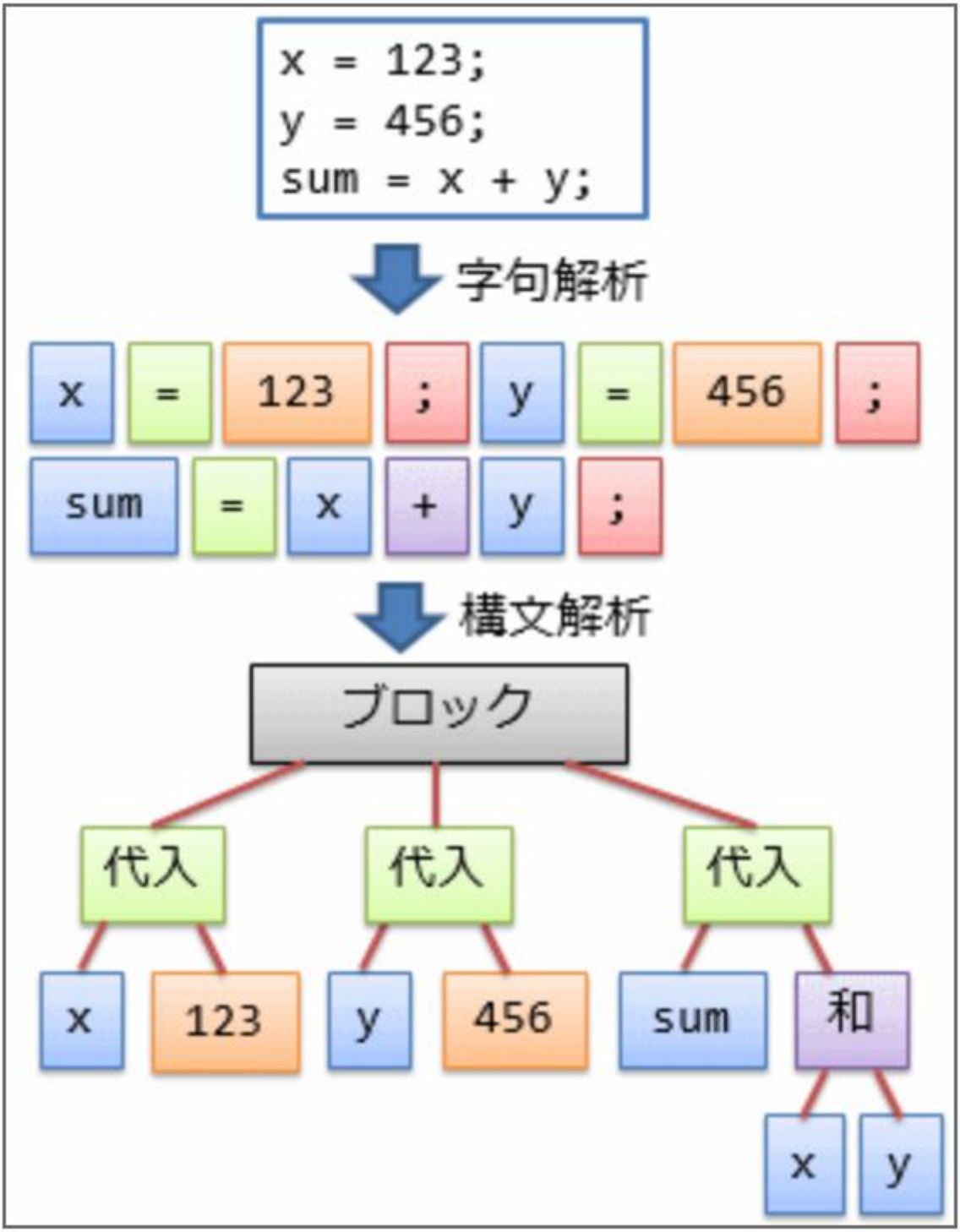
コードを、一行ずつ機械語に変換し、順次、命令を実行する言語。

【具体例】

PHP、Ruby、JavaScript、Python



- **コンパイルの詳しい流れ**



1. **Lexical analysis（字句解析）**

   ソースコードの文字列を言語の最小単位（トークン）の列に分解。

	構文規則	説明
(1)	符号なし浮動小数点定数 →小数点定数［指数部］｜数字列 指数部	符号なし浮動小数点定数は、「小数点定数 指数部」、「小数点定数」、または「数字列 指数部」のいずれかである
(2)	小数点定数→［数字列］.数字列｜数字列.	小数点定数は、「数字列.数字列」、「.数字列」、 または「数字列.」のいずれかである
(3)	指数部 → e［符号］数字列	指数部は、「e数字列」か「e 符号 数字列」である
(4)	数字列 → 数字｜数字列 数字	数字列は、「数字」か「数字列 数字」である
(5)	符号 → ＋｜－	符号は、「＋」か「－」である
(6)	数字 → 0｜1｜2｜3｜4｜5｜6｜7｜8｜9	数字は、0、1、…、9のいずれかである

2. **Syntax analysis（構文解析）**

トークンの列をツリー構造に変換。ソースコードから構造体を構築することを構文解析といい、Htmlを構文解析してDOMツリーを構築する処理とは別物なので注意。

3. **Semantics analysis（意味解析）**

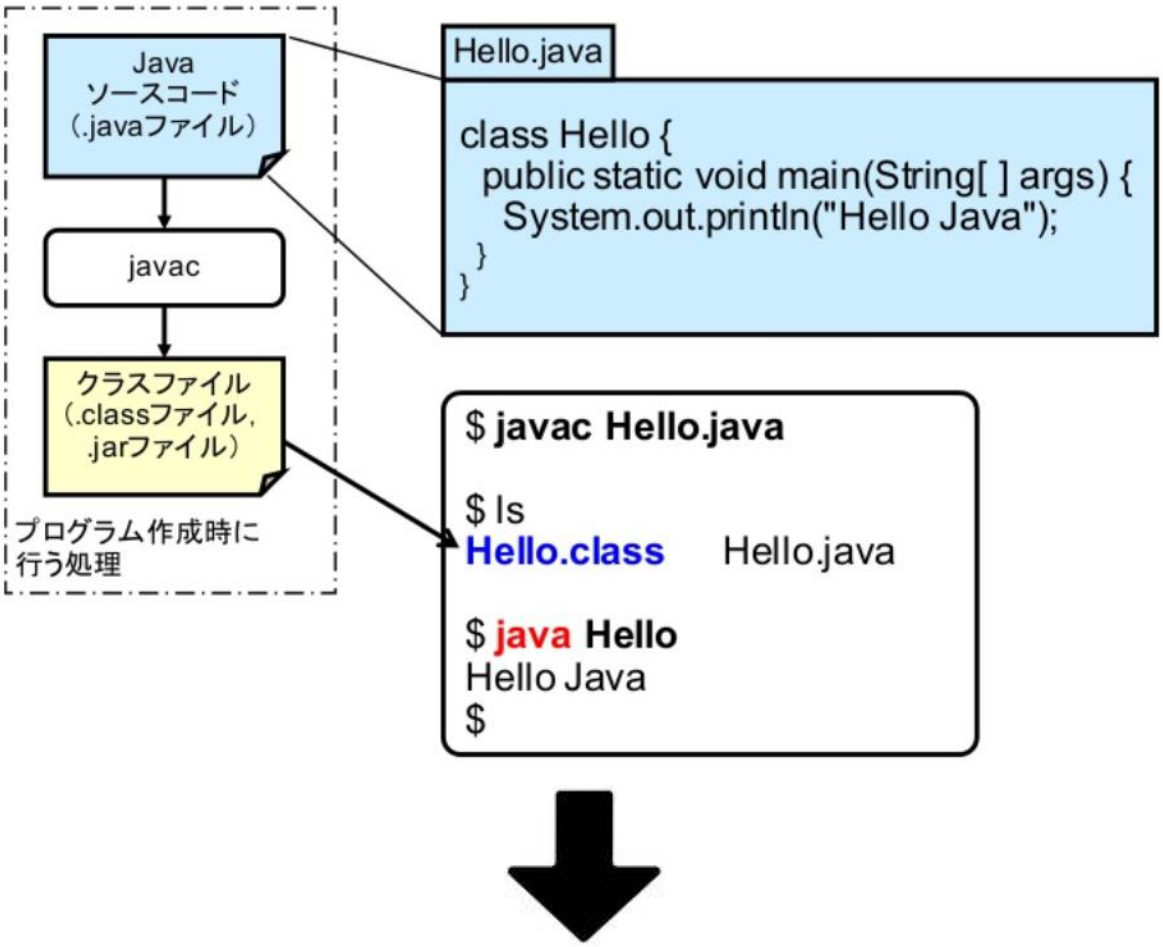
   ツリー構造を基に、ソースコードに論理的な誤りがないか解析。

4. **命令の実行**

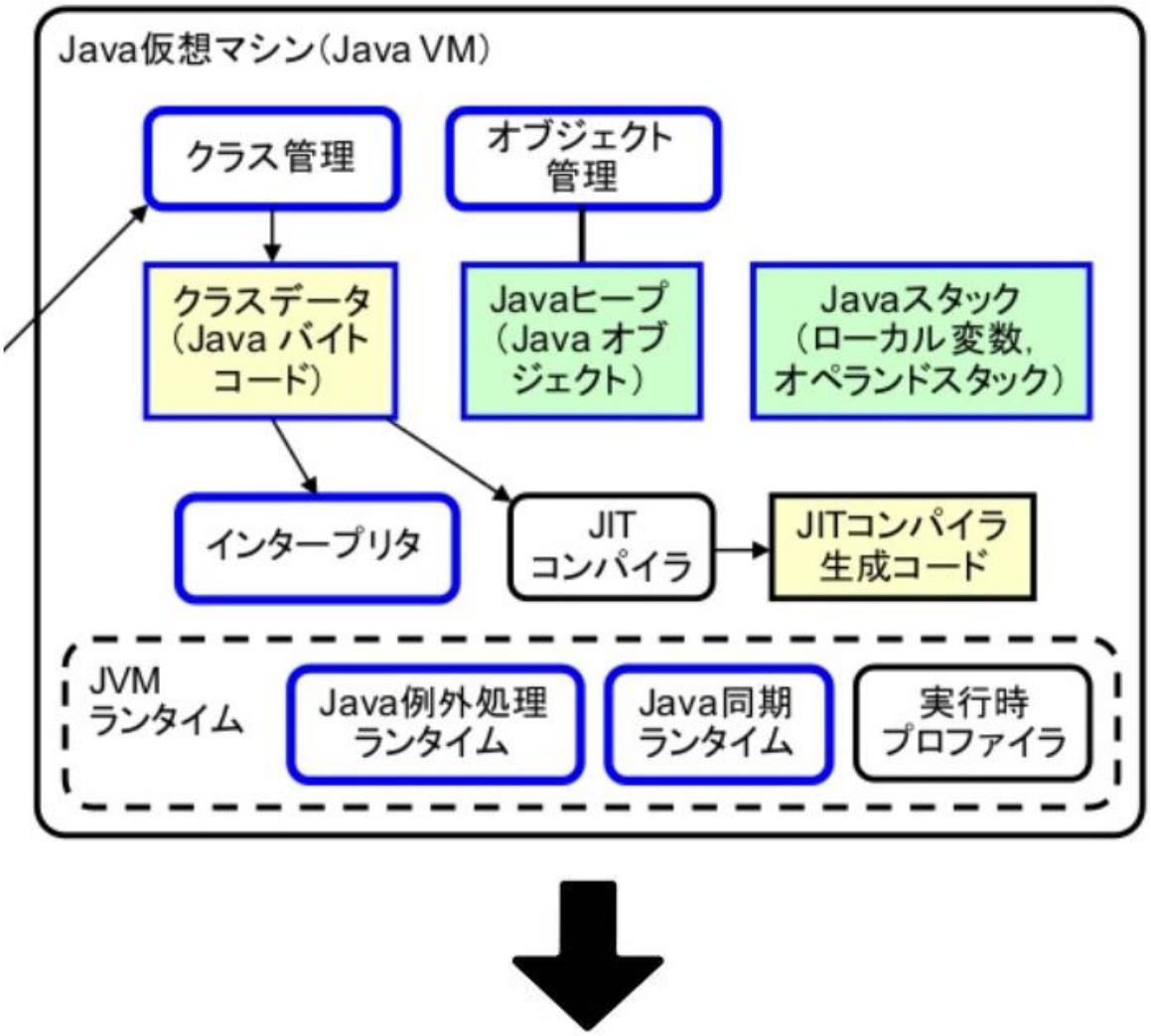
   意味解析の結果を基に、命令が実行される。

◆ 両言語の中間型であるJavaによる命令までの流れ

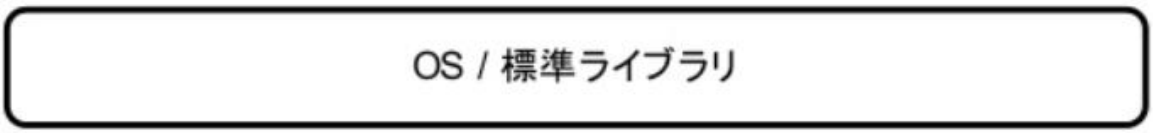
コードを、中間的なオブジェクトコードとして、変換する。



オブジェクトコードを、JVM：Java Virtual Machine内の仕組みによって、機械語に翻訳する。



結果的に、OS（制御プログラム？）に依存せずに、命令を実行できる。

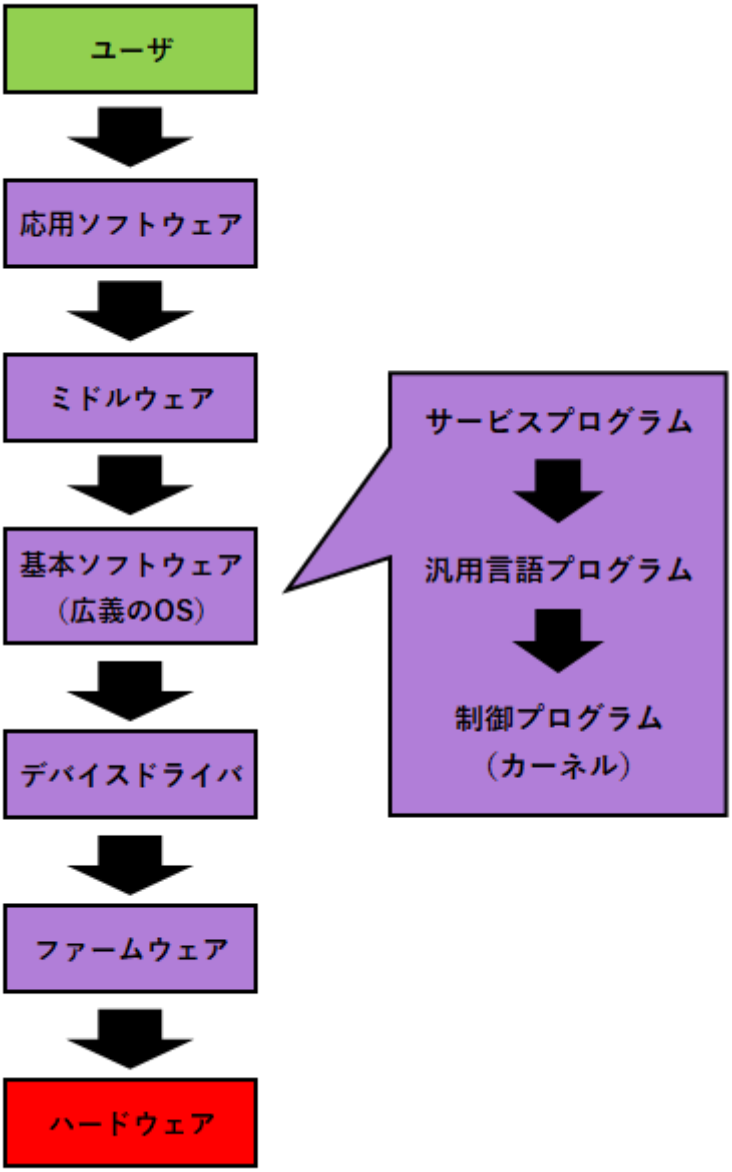


ヒープ領域とスタック領域について

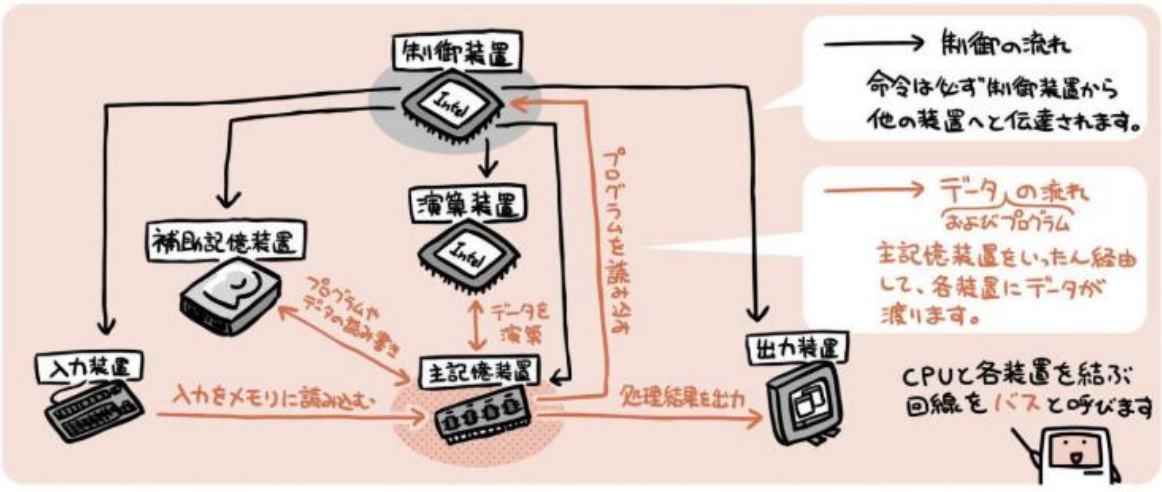
https://www.omotenashi-mind.com/index.php?title=Java%EF%BC%9A%E6%84%8F%E5%A4%96%E3%81%A8%E6%95%99%E3%82%8F%E3%82%8B%E6%A9%9F%E4%BC%9A%E3%81%AE%E5%B0%91%E3%81%AA%E3%81%84%E3%83%A1%E3%83%A2%E3%83%AA%E7%AE%A1%E7%90%86%E3%81%AE%E3%81%8A%E8%A9%B1(4)

02-01. ハードウェアについて

◆ ユーザの操作が、命令としてハードウェアに伝わるまで（再掲）



◆ 装置間の関係



◆ CPU（プロセッサ）

CPUは制御と演算を行う。CPUの制御部分は、プログラムの命令を解釈して、コンピュータ全体を制御。CPUの演算部分は、計算や演算処理を行う。特、『**算術論理演算装置（ALU：Arithmetic and Logic Unit）**』とも呼ぶ。

◆ RAM（メインメモリ+キャッシュメモリ）

プログラムやデータを一時的に記憶し、コンピュータの電源を切るとこれらは消える。

◆ ROM

プログラムやデータを長期的に記憶し、コンピュータの電源を切ってもこれらは消えない。

◆ 入力装置

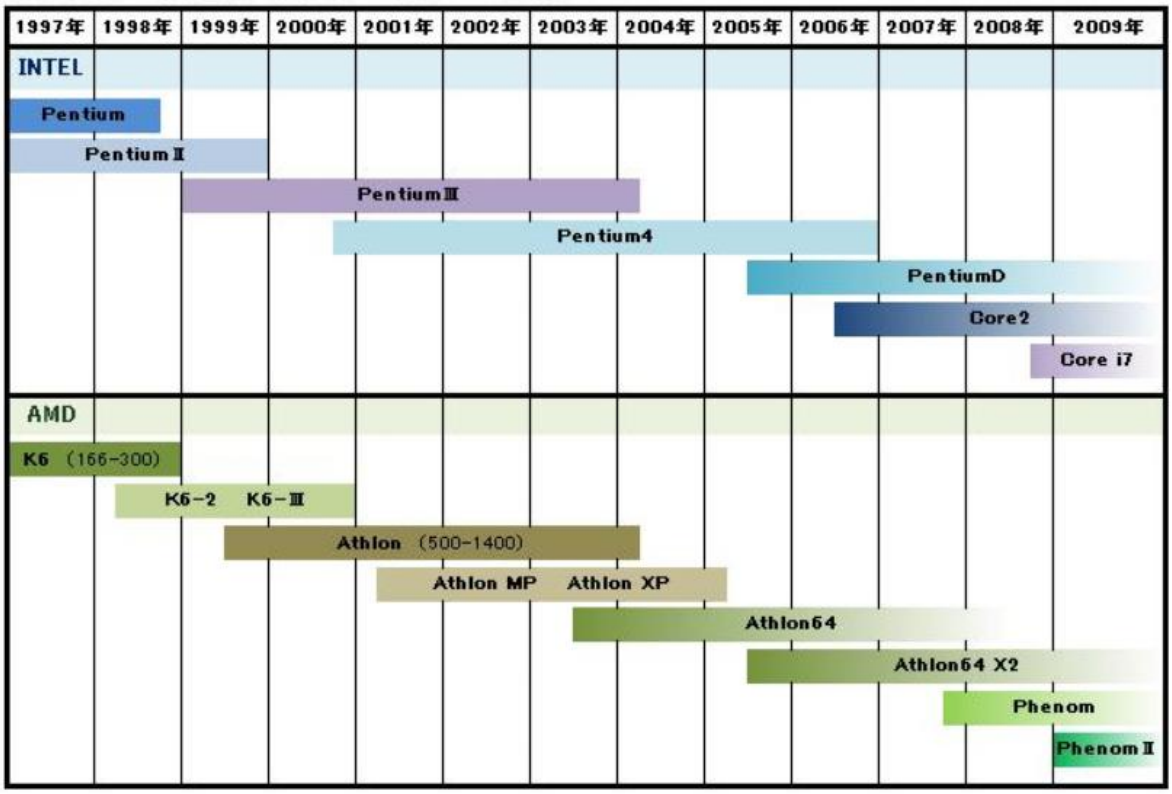
コンピュータにデータを入力。キーボード、マウス、スキャナなど。

◆ 出力装置

コンピュータからデータを出力。ディスプレイ、プリンタなど。

02-02. CPU（プロセッサ）

◆ IntelとAMDにおけるCPUの歴史（※2009年まで）

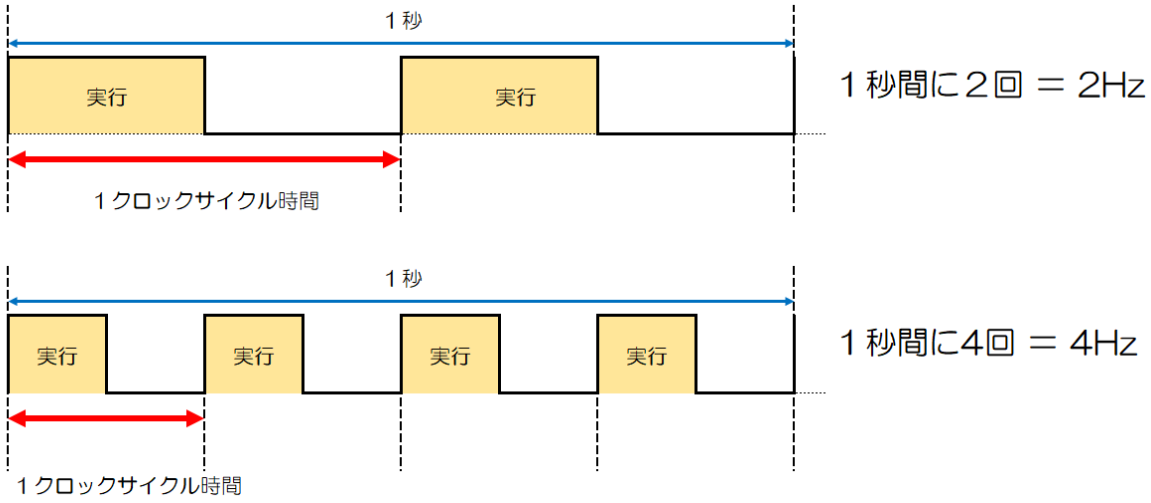


◆ クロック周波数

CPUの回路が処理と歩調を合わせるために用いる信号を、『クロック』と言う。一定時間ごとにクロックが起こる時、１秒間にクロックが何回起こるかを『クロック周波数』という。これは、Hzで表される。ちなみに、ワイのパソコンのクロック周波数は2.60GHzでした。

（例1）3Hz = 3 (クロック数／秒)

（例2）2.6GHz = 2.6×10^9 (クロック数／秒)



◆ MIPS：Million Instructions Per Second（×10^6 命令数／秒）

CPUが1秒間に何回命令を実行するかを表す。

（例題）

命令当たりの平均クロック数は、『(4×0.3)＋(8×0.6)＋(10×0.1) 』から、『7』と求められる。

700Hz (×10^6 クロック数／秒) ÷ 7 (クロック数／命令) = 100 (×10^6 命令数／秒)

動作クロック周波数が700MHzのCPUで，命令の実行に必要なクロック数とその命令の出現率が表に示す値である場合，このCPUの性能は約何MIPSか。

命令の種別	命令実行に必要なクロック数	出現率（%）
レジスタ間演算	4	30
メモリ・レジスタ間演算	8	60
無条件分岐	10	10

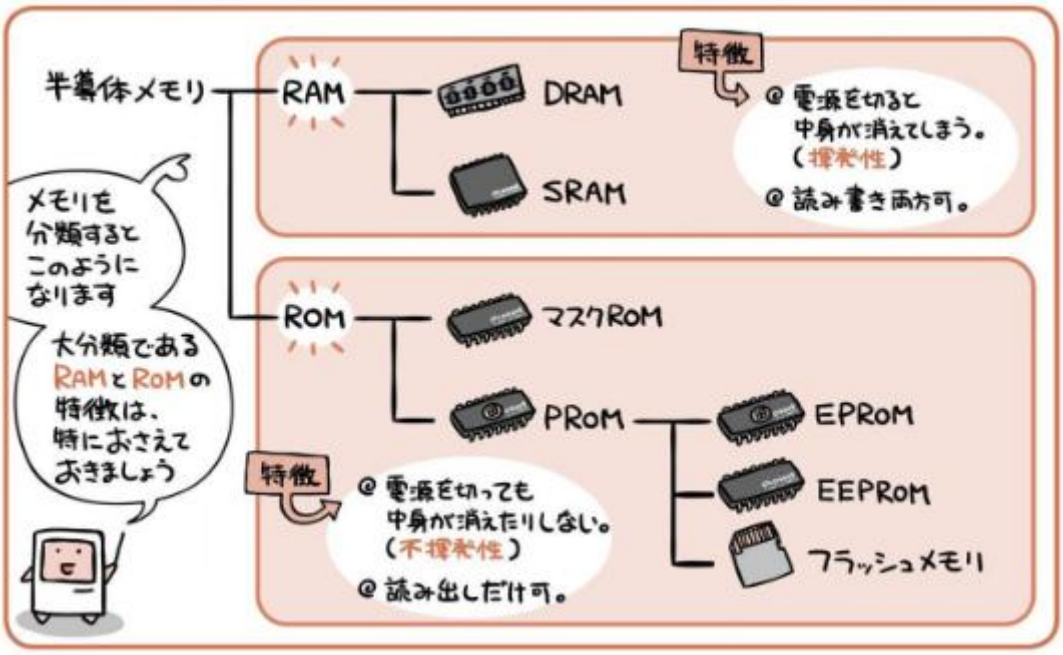
- **1命令当たりの実行時間 (秒／命令) の求め方**

  1 ÷ 100 (×10^6 命令／秒) = 10n (秒／命令)

02-03. 物理メモリ（RAM + ROM）

◆ 物理メモリの種類

『物理メモリ』は、RAMとROMに大きく分けられる。

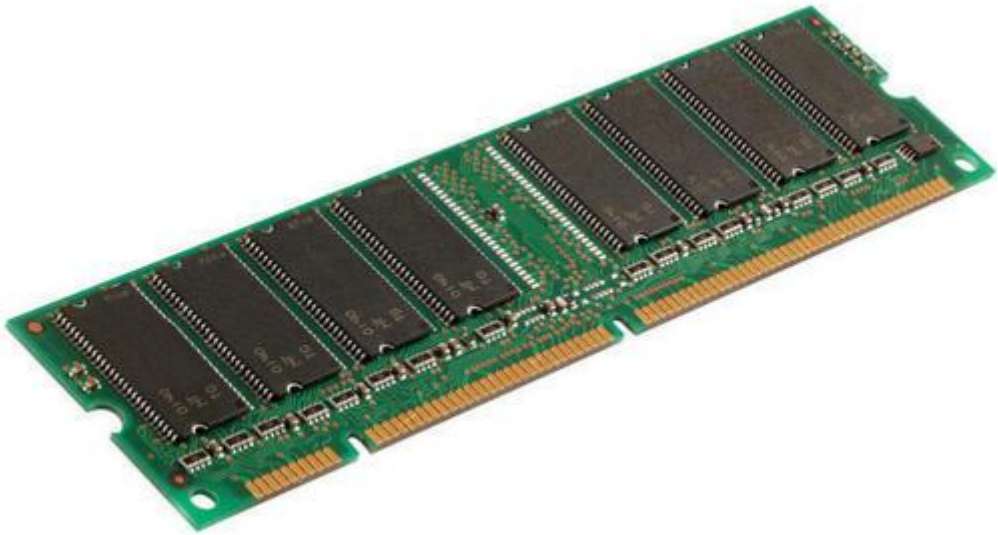


◆ RAM：Read Access Memory

RAMは、メインメモリとして使われる『Dynamic RAM』と、キャッシュメモリとして使われる『Static RAM』に分解される。

- **Dynamic RAM**

  メインメモリとして用いられる。よく見るやつ。



- **Static RAM**

  キャッシュメモリとして用いられる。

◆ ROM：Read Only Memory

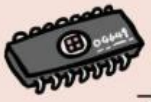
- **Mask ROM**

マスクROM

普通「ROM」と言ったらコレを指します

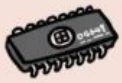
読み出し専用のメモリです。製造時にデータを書き込み、以降は内容を書き換えることができません。

- **Programmable ROM**



PROM (Programmable ROM)

プログラマブルなROM。つまり、ユーザの手で書き換えることができるROMです。下記のような種類があります。



EPROM (Erasable PROM)

紫外線でデータを消去して書き換えることができます。

EEPROM (Electrically EPROM)

電気的にデータを消去して書き換えることができます。

フラッシュメモリ

EEPROMの1種。全消去ではなく、ブロック単位でデータを消去して書き換えることができます。

◆ Garbage collection

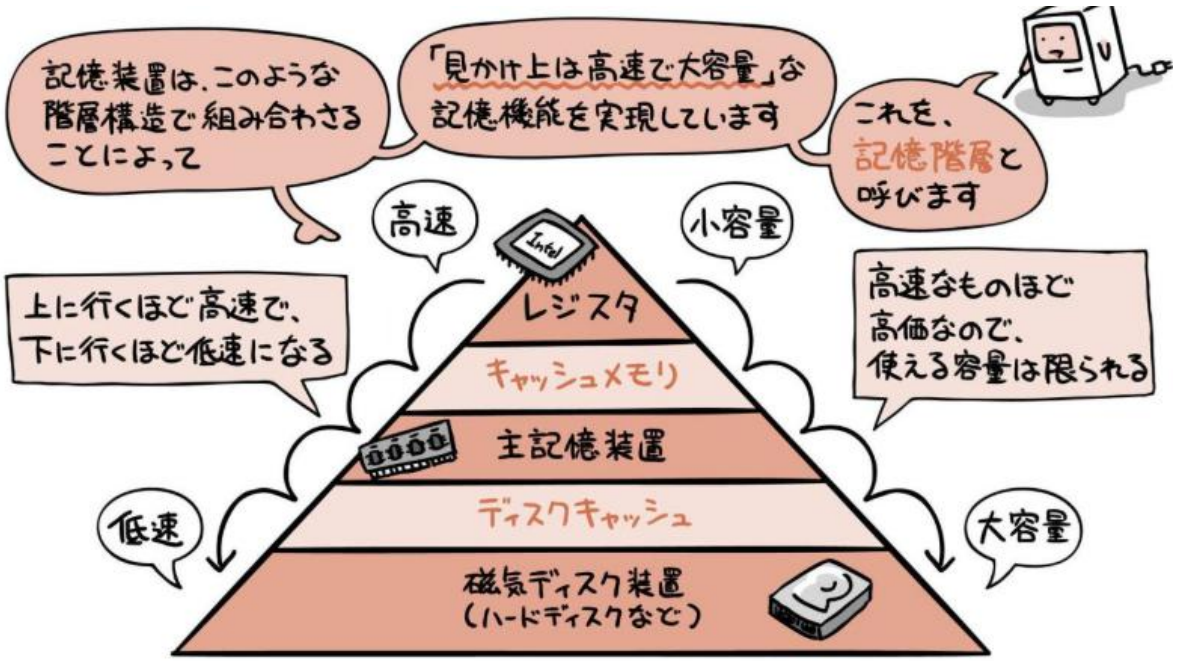
プログラムが確保したメモリ領域のうち、不要になった領域を自動的に解放する機能。

- **JavaにおけるGarbage collection**

Javaでは、JVM：Java Virtual Machine（Java仮想マシン）が、メモリ領域をオブジェクトに自動的に割り当て、また一方で、不要になったメモリ領域の解放を行う。一方で自動的に行う。

02-04. SRAM

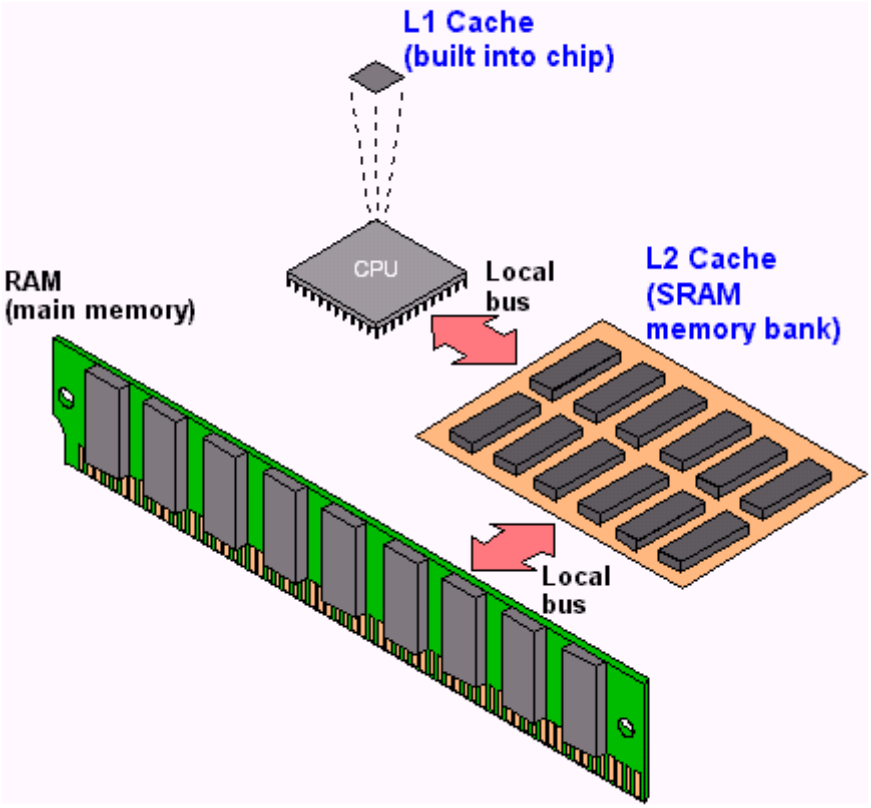
CPUから命令が起こるとき、CPU、DRAM、ハードディスク間には、読み込みと書き出しの処理速度に差がある。



◆ キャッシュメモリとは

- **一次キャッシュメモリと二次キャッシュメモリ**

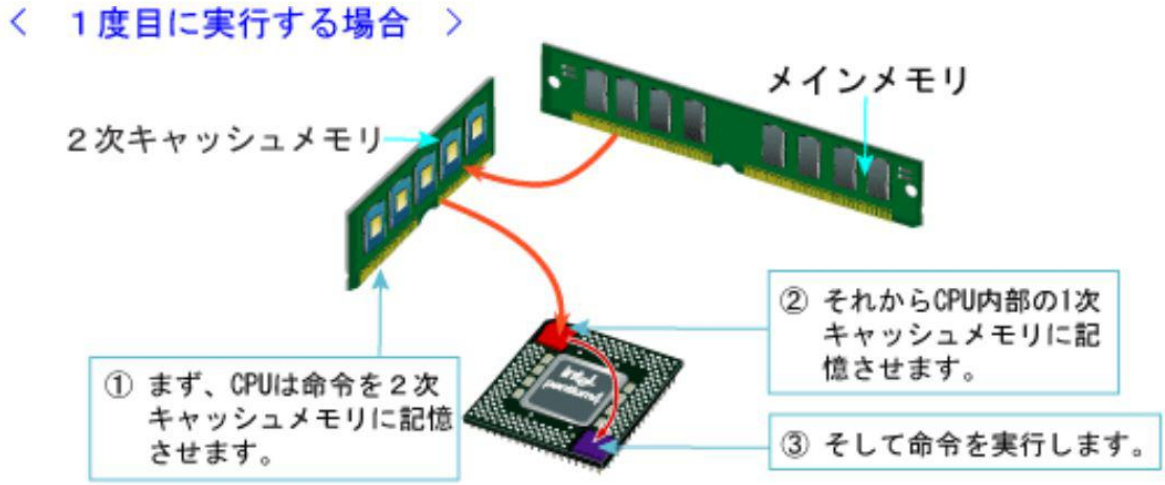
  CPUとメインメモリの間に、キャッシュメモリを何段階か設置し、CPUとメインメモリの間の読み込みと書き出しの処理速度の差を緩和させる。



◆ キャッシュメモリの読み込み方法

- **ユーザー → メインメモリ → 二次キャッシュメモリ → 一次キャッシュメモリ**

1. ユーザーが、パソコンに対して命令を与える。
2. CPUは、命令をメインメモリに書き込む。
3. CPUは、メインメモリから命令を読み出す。
4. CPUは、二次キャッシュメモリに書き込む。
5. CPUは、一次キャッシュメモリに書き込む。
6. CPUは、命令を実行する。



- **実例**

  タスクマネージャのパフォーマンスタブで、n次キャッシュメモリがどのくらい使われているのかを確認できる。

使用率	速度		基本速度:	2.60 GHz
4%	1.20 GHz		ソケット:	1
			コア:	2
プロセス数	スレッド数	ハンドル数	論理プロセッサ数:	4
185	1743	69364	仮想化:	有効
			L1 キャッシュ:	128 KB
稼働時間			L2 キャッシュ:	512 KB
0:01:07:42			L3 キャッシュ:	3.0 MB

◆ キャッシュメモリへの書き込み方式の種類

- **Write-throught 方式**

  CPUは、命令をメインメモリとキャッシュメモリの両方に書き込む。常にメインメモリとキャッシュメモリの内容が一致している状態を確保できるが、メモリへの書き込みが頻繁に行われるので遅い。

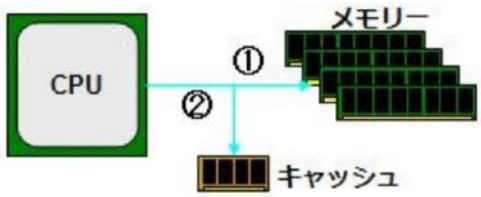
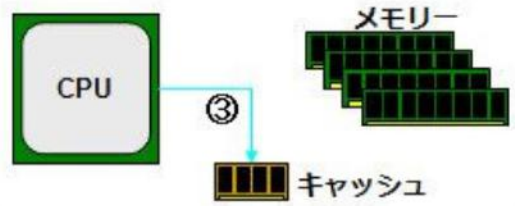


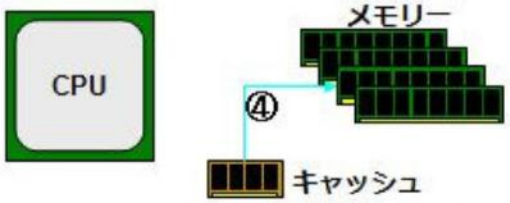
図3 ライトスルーキャッシュの基本的な仕組み。①メモリーに書き込む。②同時にキャッシュにも書き込む

- **Write-back 方式**

  CPUは、キャッシュメモリのみに書き込む。次に、キャッシュメモリがメインメモリに書き込む。メインメモリとキャッシュメモリの内容が一致している状態を必ずしも確保できないが、メインメモリへの書き込み回数が少ないため速い

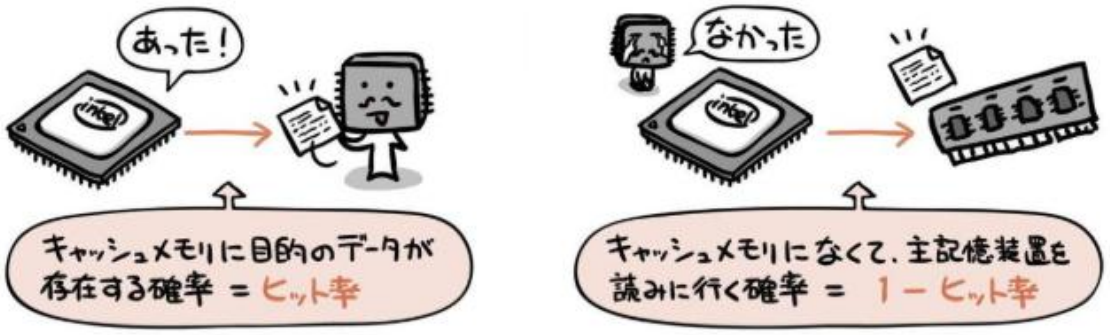


③ CPUからキャッシュにデータを書き込む



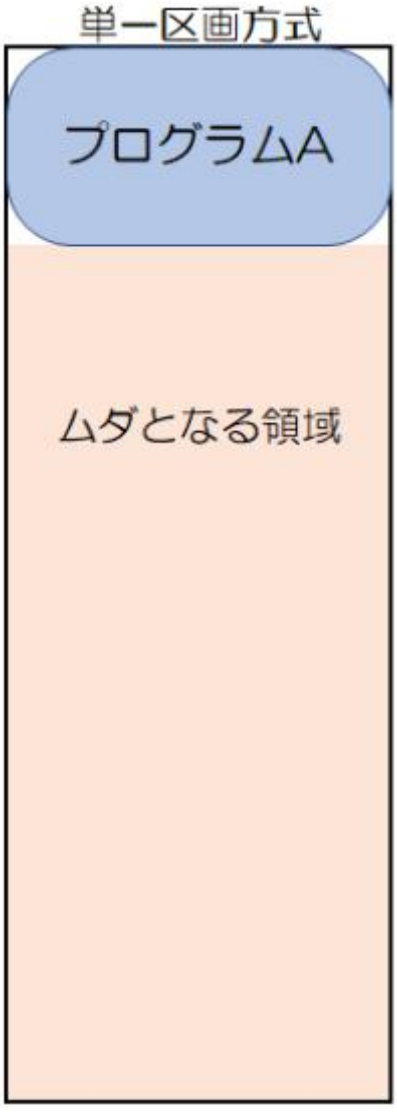
④ 次にキャッシュからメモリーにデータを書き込む

◆ 実効アクセス時間



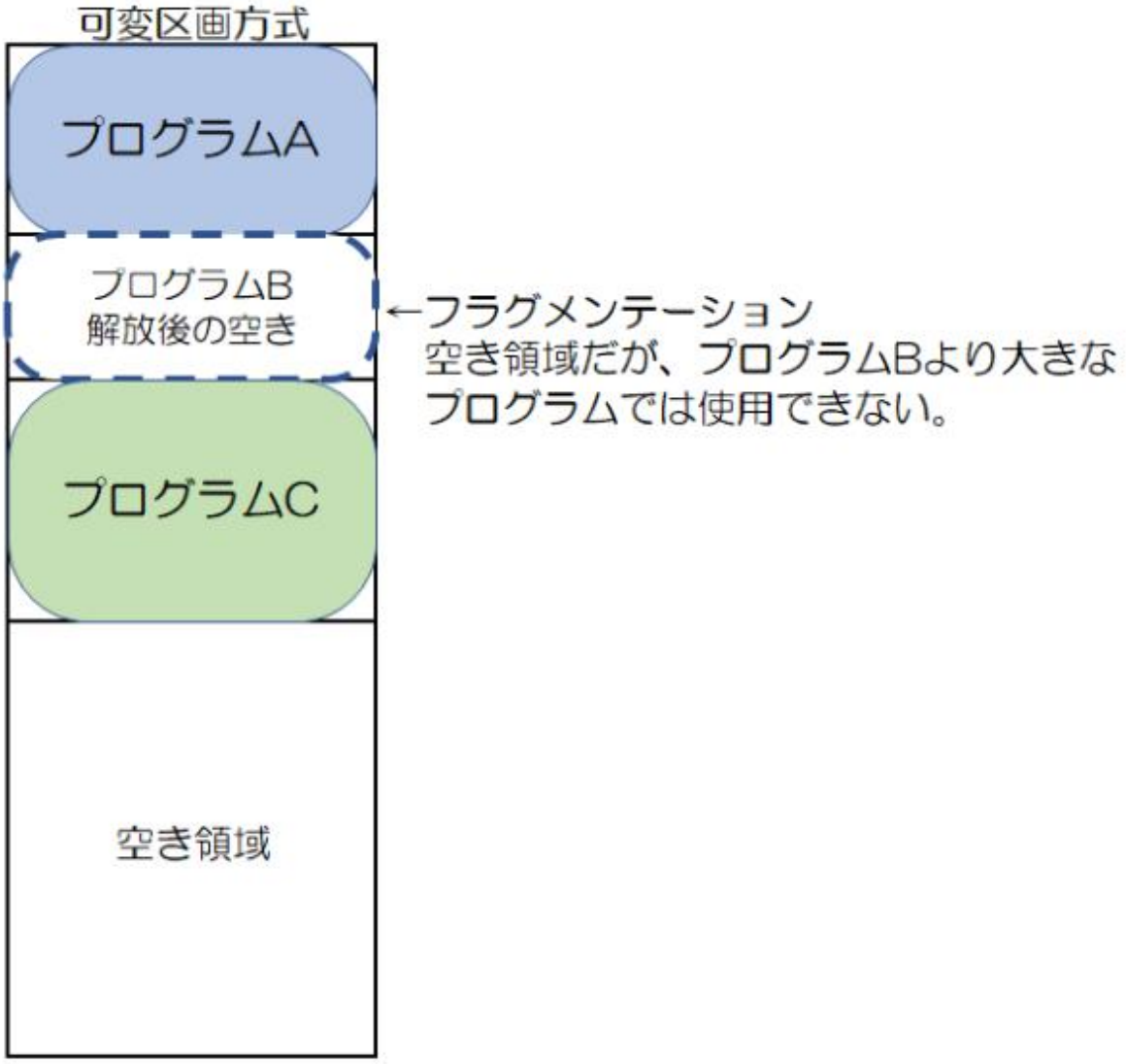
02-05. 物理メモリのアドレス空間管理

◆ 固定区画方式



◆ 多重区画方式

◆ **可変区画方式**



◆ オーバーレイ方式

◆ フラグメンテーション／メモリコンパクション

◆ スワッピング方式

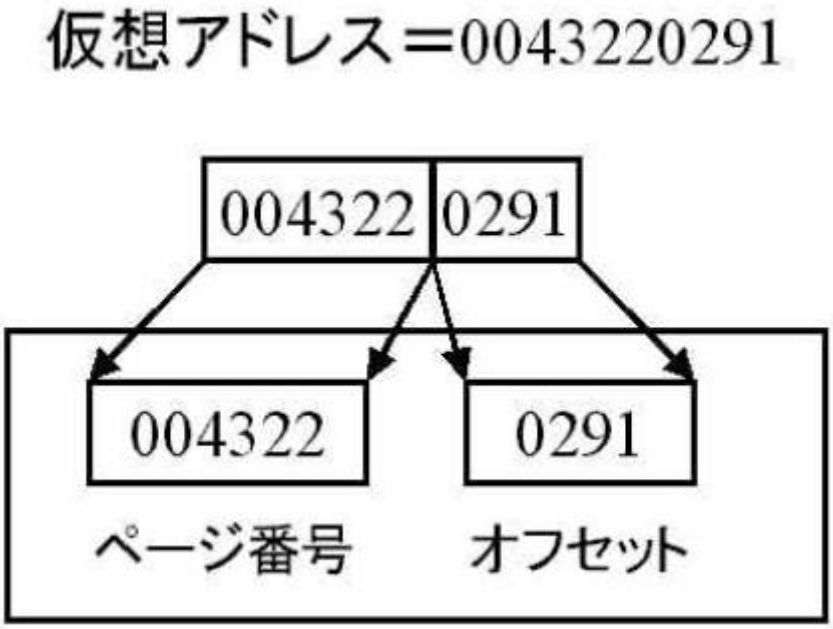
02-06. 仮想メモリのアドレス空間管理

以下の用語に統一する。

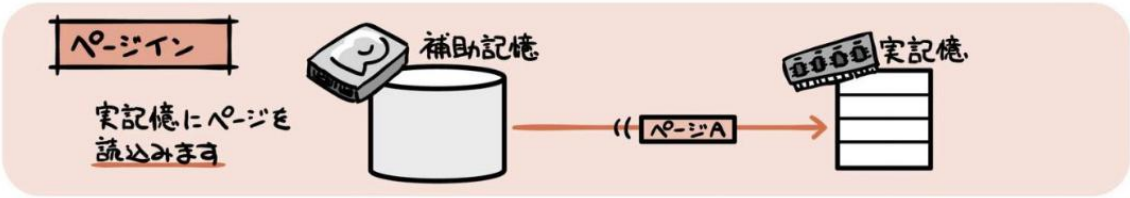
- **主記憶 ⇒ 物理メモリ（メインメモリ＋キャッシュメモリ）**
- **補助記憶 ⇒ ハードディスク**
- **仮想記憶 ⇒ 仮想メモリ**

◆ Paging方式とページ構造

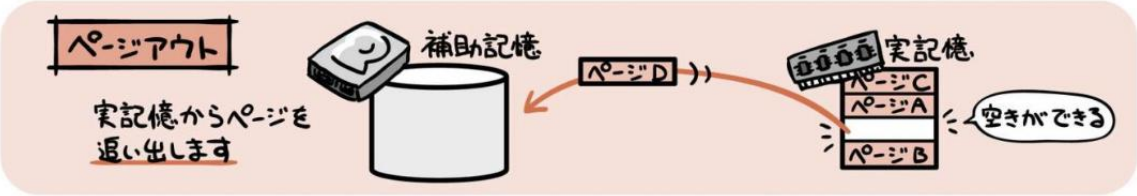
仮想メモリの実装方法の一つ。仮想メモリのアドレス空間を『固定長』の領域（ページ）、また物理メモリのアドレス空間を『固定長』の領域（ページフレーム）に分割し、管理する方式。



ハードディスクから物理メモリのページフレームにページを読み込むことを『Page-in』という。



物理メモリのページフレームからハードディスクにページを追い出すことを『Page-out』という。

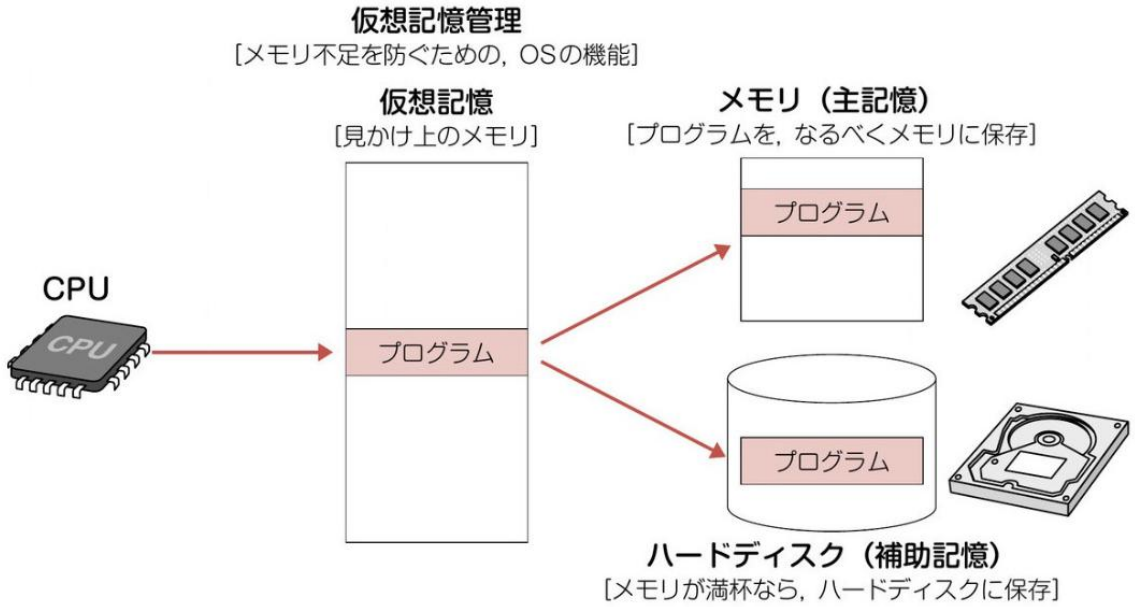


- **Demand paging方式**

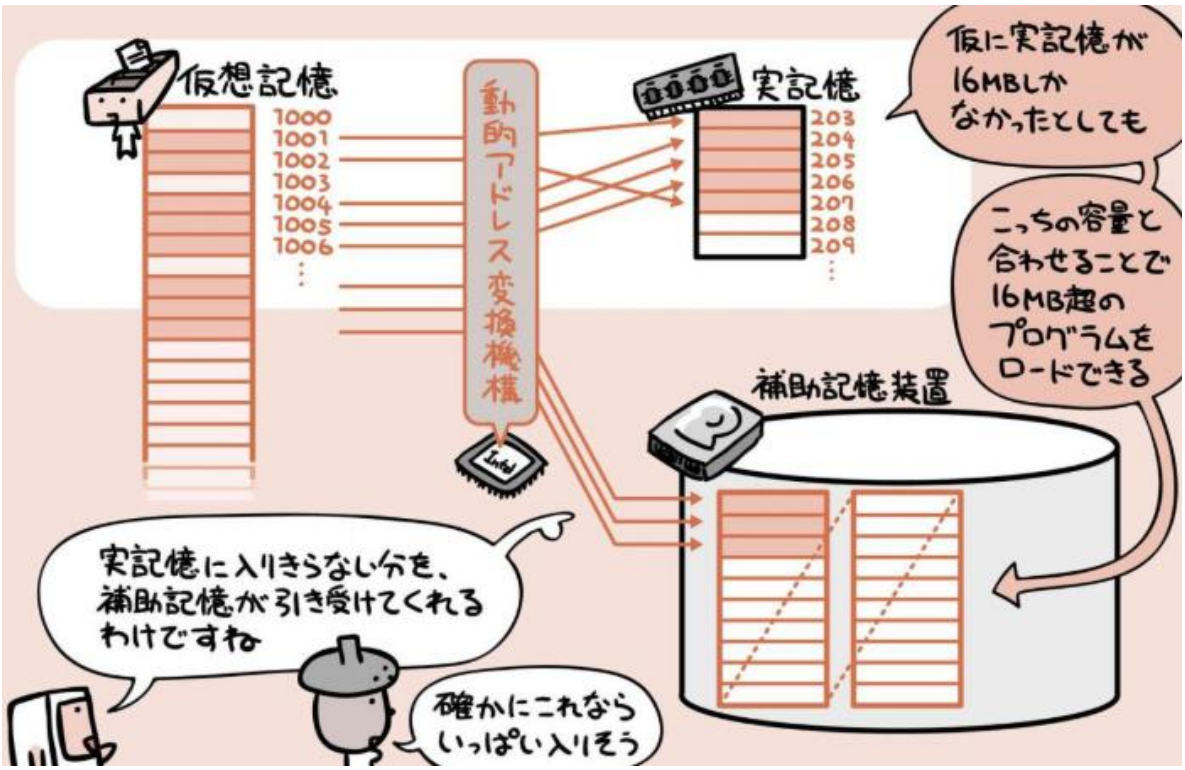
◆ Segment方式

仮想メモリの実装方法の一つ。仮想メモリのアドレス空間を『可変長』の領域（セグメント）、また物理メモリのアドレス空間を『可変長』の領域（セグメント）に分割し、管理する方式。

◆ 仮想メモリとのマッピングによる大容量アドレス空間の実現



仮想メモリのアドレス空間を、物理メモリのアドレス空間とハードディスクにマッピングすることによって、物理メモリのアドレス空間を疑似的に大きく見せかけることができる。

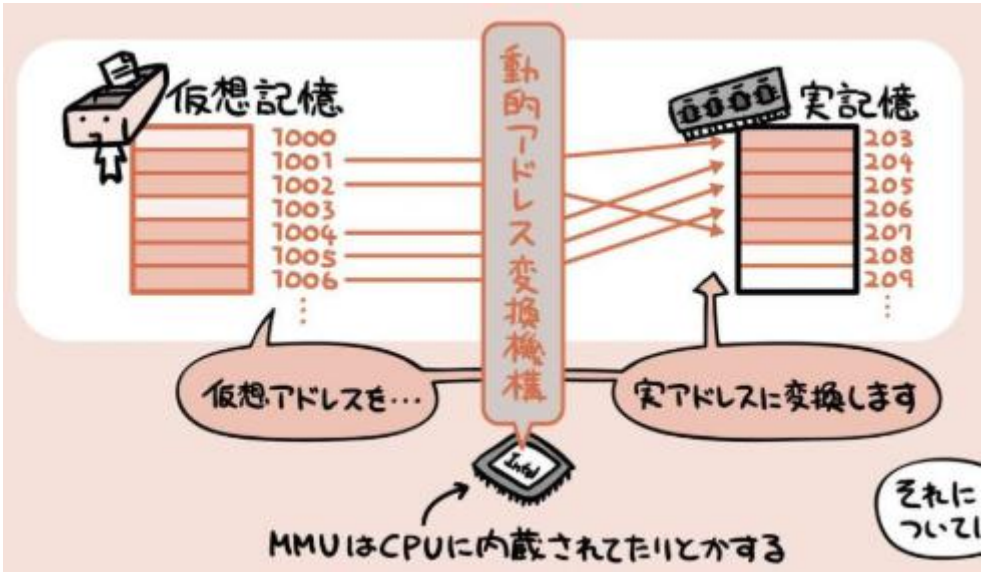


ちなみに、富士通の仮想メモリの容量は、以下の通り。

◆ MMU：Memory Management Unit（メモリ管理ユニット）

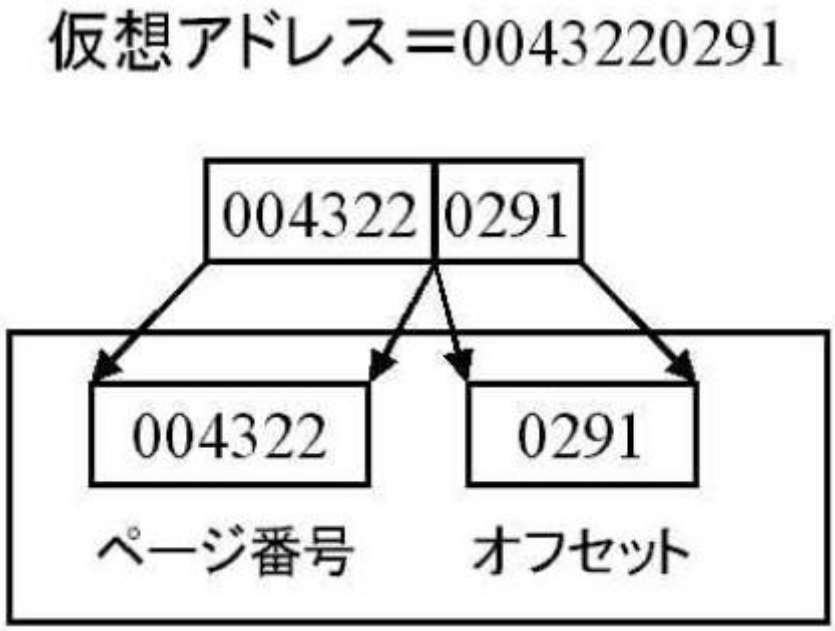
- **MMUにおける動的アドレス変換機構**

MMUによって、仮想メモリのアドレスは、物理メモリのアドレスに変換される。この仕組みを、『動的アドレス変換機構』と呼ぶ。

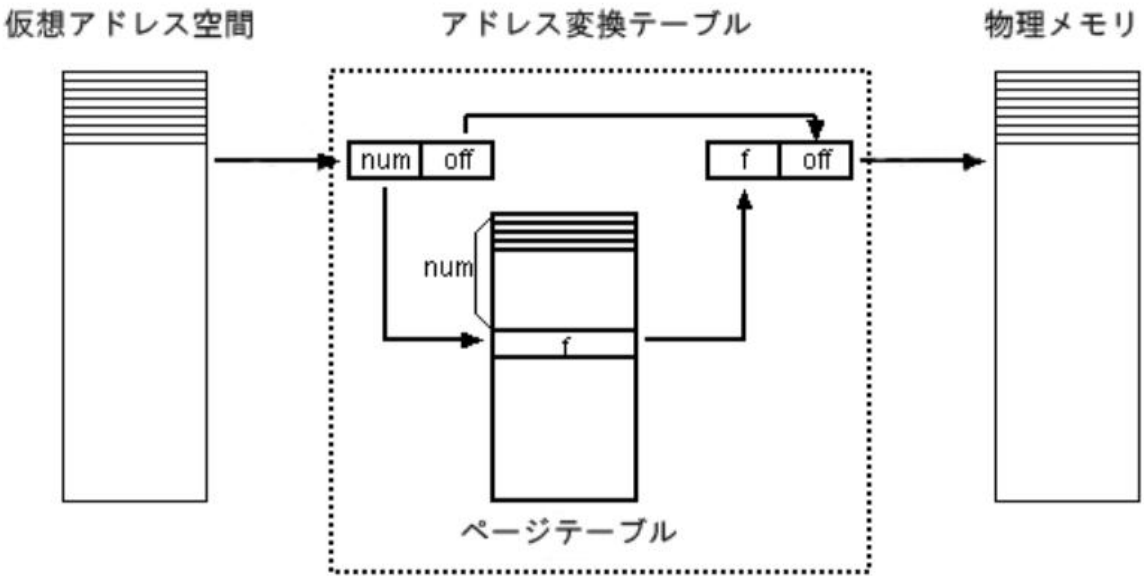


- **アドレス変換の仕組み**

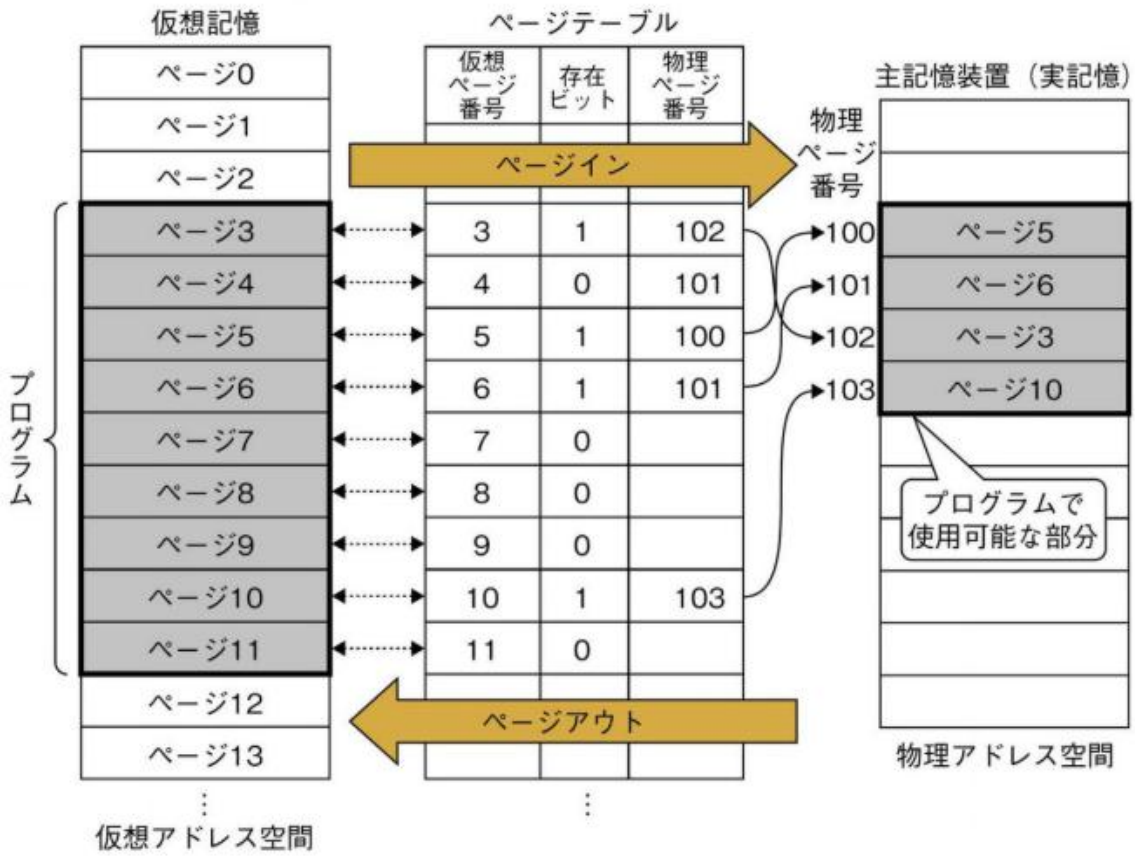
1. ページの仮想アドレスを、ページ番号とページオフセットに分割する。



2. ページ番号とページフレームのアドレスを対応づけるページテーブルを利用して、 仮想アドレスの指すページフレームを探す。
3. ページフレームのアドレスとページオフセットから物理アドレスを計算する。



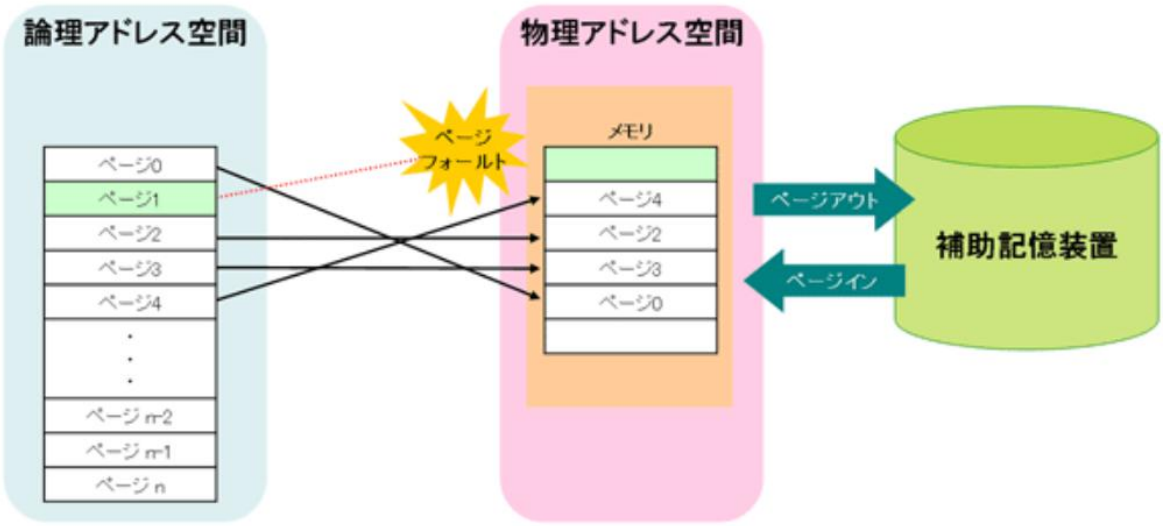
- **ページテーブルにおける仮想ページ番号と物理ページ番号の対応づけ**



02-07. Page fault発生時の処理

◆ Page faultとは

ハードディスクから物理メモリのアドレス空間への割り込み処理のこと。プログラムが、仮想メモリのアドレス空間のページにアクセスし、そのページが物理メモリのアドレス空間にマッピングされていなかった場合に、ハードディスクから物理メモリのアドレス空間にページインが起こる。



◆ Page Replacementアルゴリズム

ページアウトのアルゴリズムのこと。ハードディスクから物理メモリのページフレームにページインを行う時に、アルゴリズムの各方式で、物理メモリからハードディスクにページアウトするページが異なる。

- **LRU方式：Least Recently Used**

  p261-1

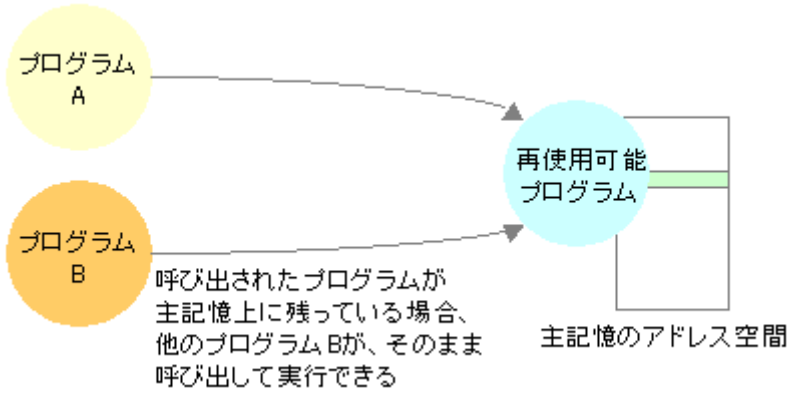
- **FIFO方式：First In First Out**

- **LIFO方式：Last In First Out**
- **LFU方式：Least Frequently Used**

02-08. アドレス空間管理におけるプログラムの種類

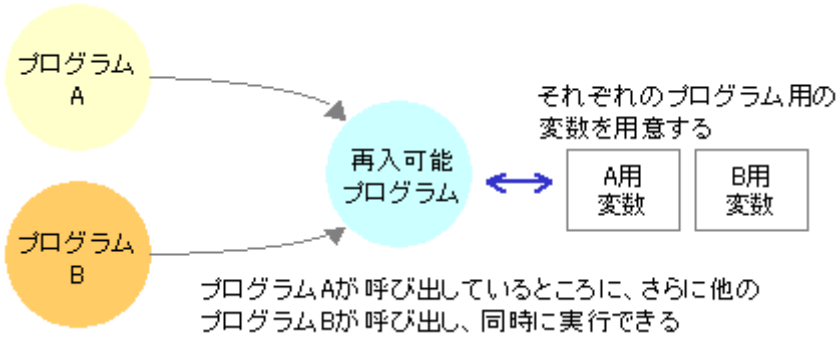
◆ Reusable（再使用可能プログラム）

一度実行すれば、再度、ハードディスクから物理メモリに読み込むことをせずに、実行を繰り返せるプログラムのこと。



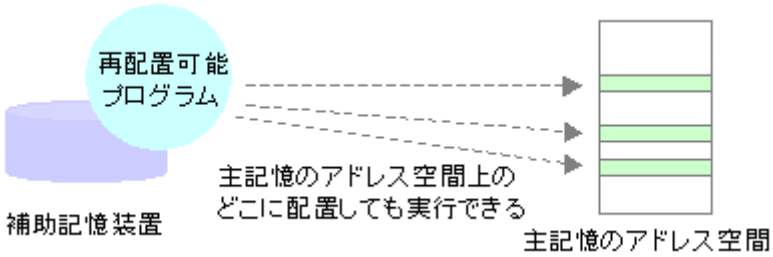
◆ Reentrant（再入可能プログラム）

再使用可能の性質をもち、また複数のプログラムから呼び出されても、互いの呼び出しが干渉せず、同時に実行できるプログラムのこと。



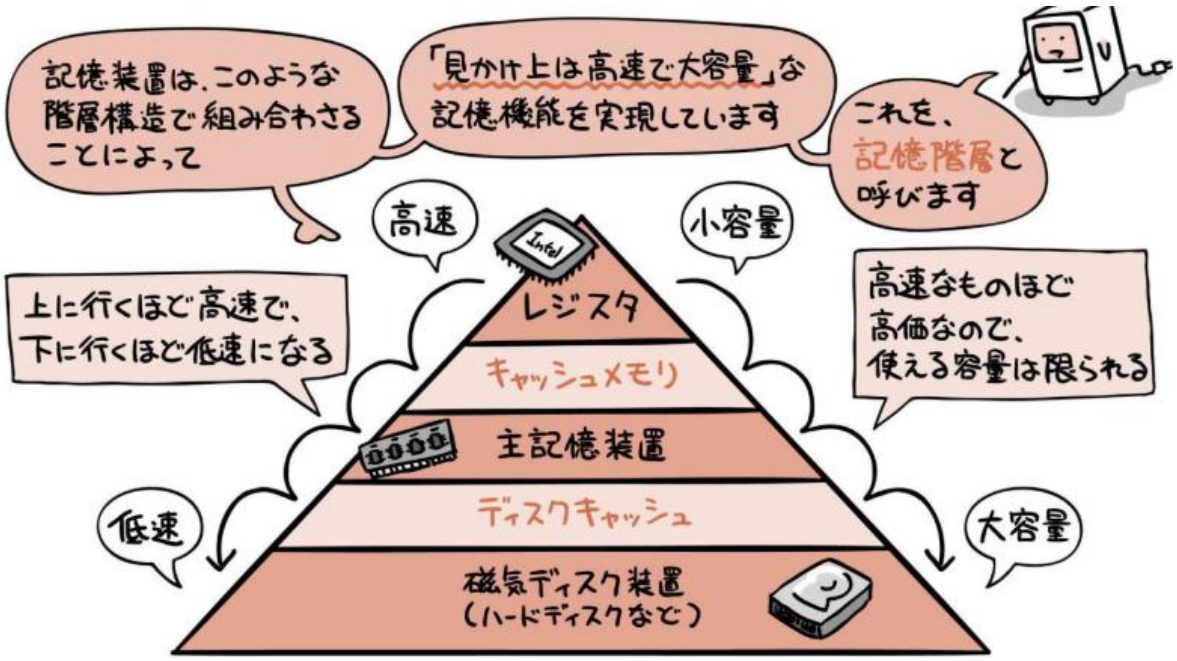
◆ Relocatable（再配置可能プログラム）

ハードディスクから物理メモリへ読み込む時に、アドレス空間上のどこに配置されても実行できるプログラムのこと。



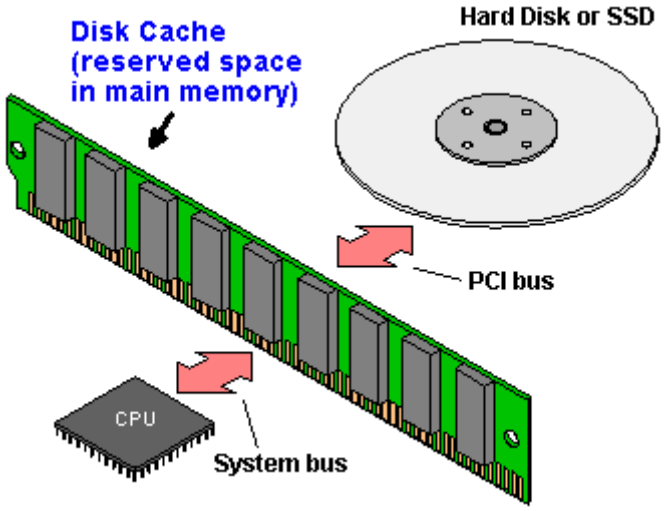
02-09. ディスクメモリ

CPU、メインメモリ、ストレージ間には、読み込みと書き出しの処理速度に差がある。（※再度記載）



◆ ディスクメモリの機能

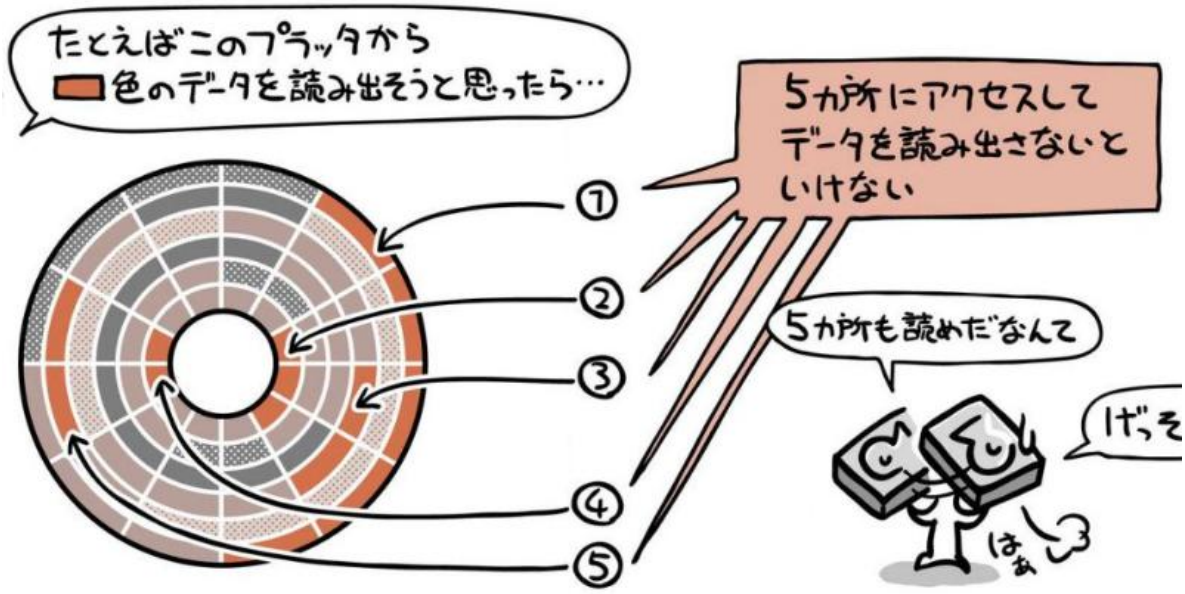
メインメモリとハードディスクの間に、ディスクキャッシュを設置し、読み込みと書き出しの処理速度の差を緩和させる。

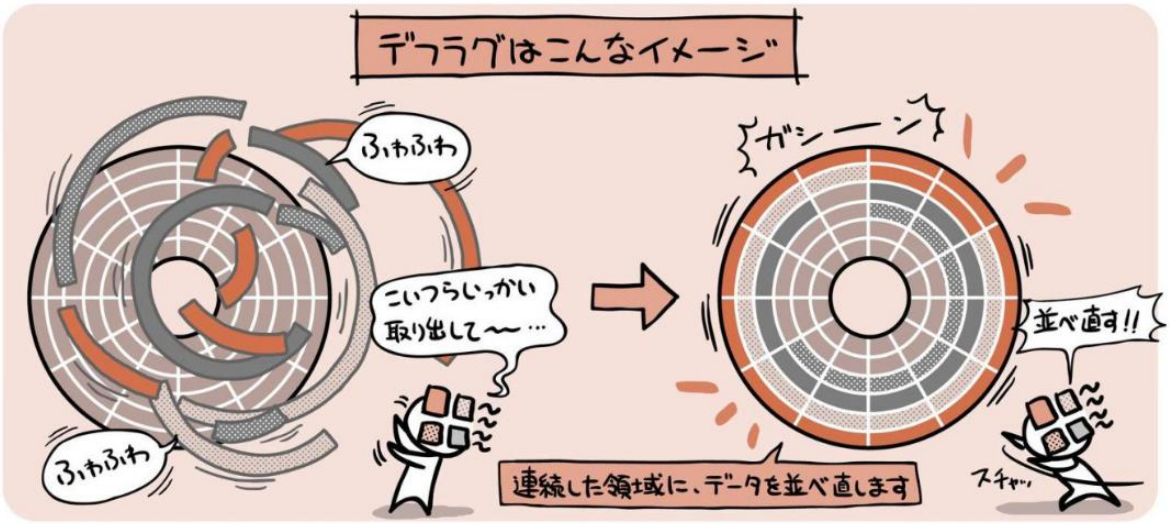


02-10. ハードディスク

◆ Defragmentation

断片化されたデータ領域を整理整頓する。





◆ RAID：Redundant Arrays of Inexpensive Disks

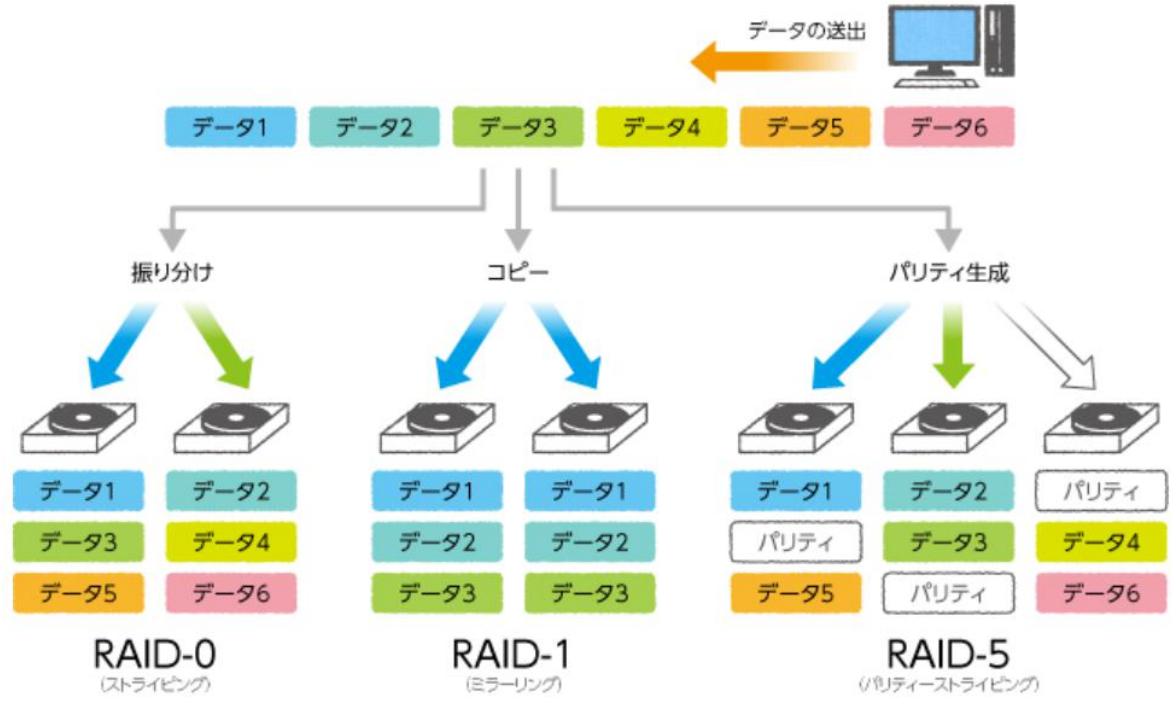
複数のハードディスクを仮想的に一つのハードディスクであるかのようにして、データを管理する技術。

- **RAID0（Striping）**

  データを、複数のハードディスクに分割して書き込む。
- **RAID1（Mirroring）**

  データを、複数のハードディスクに同じように書き込む。
- **RAID5（Striping with parity）**

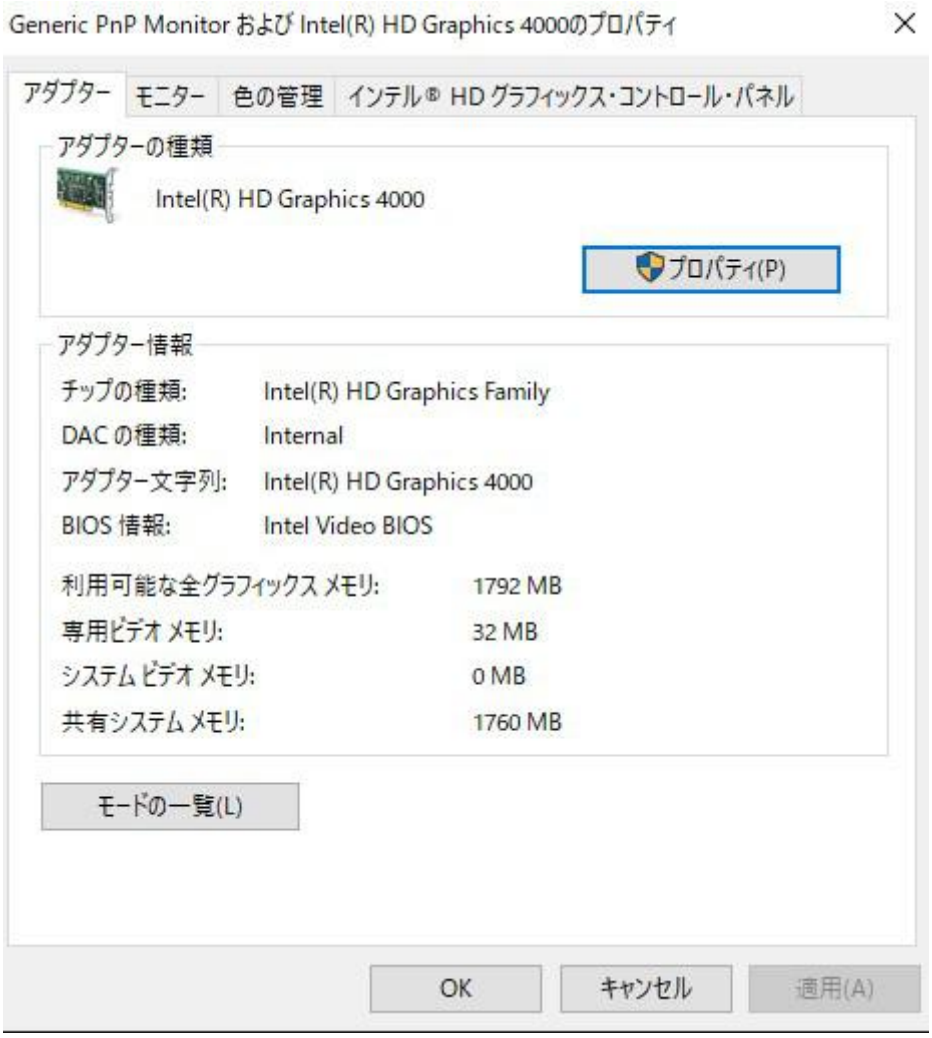
  データとパリティ（誤り訂正符号）を、3つ以上のハードディスクに書き込む。



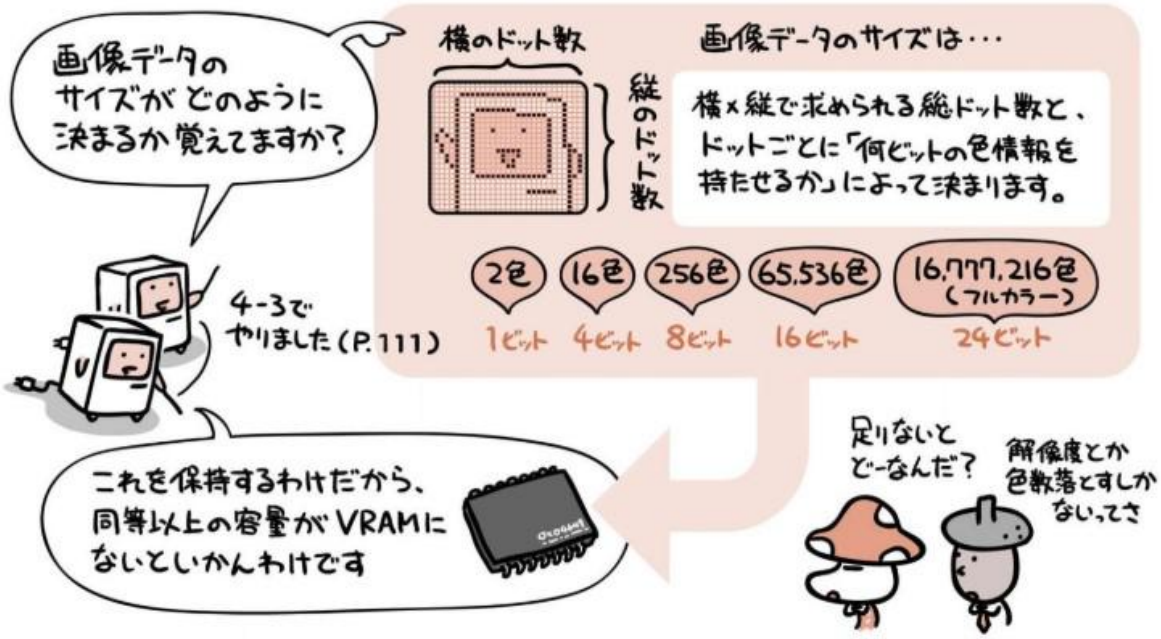
02-11. GPUとVRAM

GPUとVRAMの容量によって、扱うことのできる解像度と色数が決まる。

富士通PCのGPUとVRAMの容量は、以下の通り。



色数によって、１ドット当たり何ビットを要するが異なる。



03-01. 入出力装置

◆ キーボードからポインティングデバイス

- ジョイスティック

◆ 読み取り装置

- **イメージスキャナ**
- **Optical Character Reader**

  紙上の文字を文字データとして読み取る装置。
- **Optical Mark Reader**

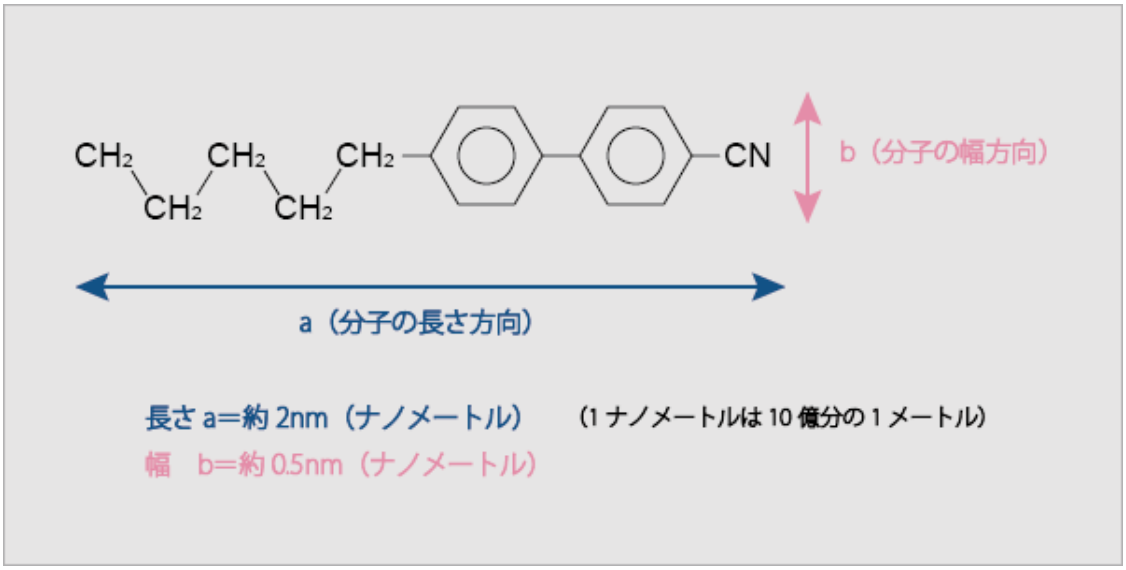
  マークシートの塗り潰し位置を読み取る装置。

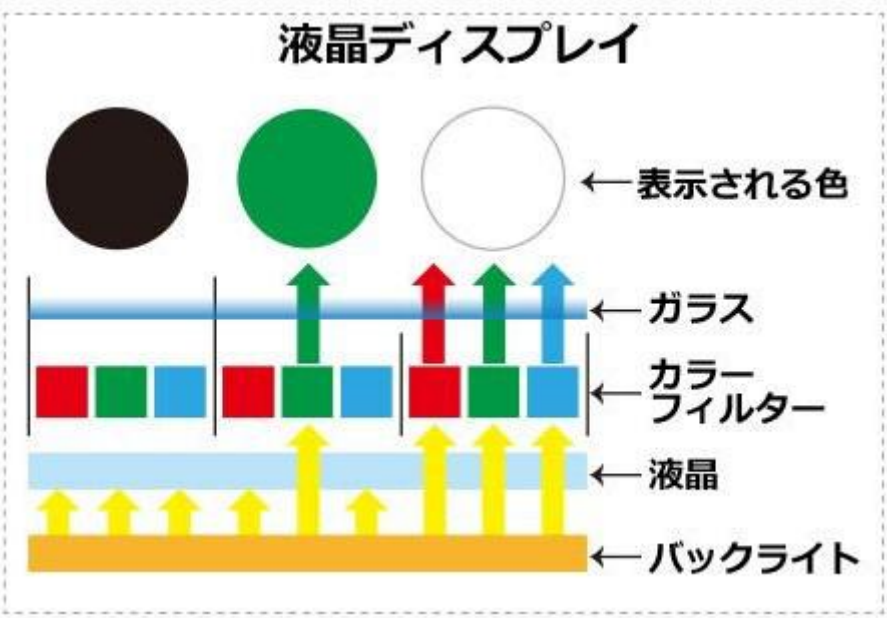
- **キャプチャカード**
- **デジタルカメラ**

◆ ディスプレイ

- **CRTディスプレイ**
- **液晶ディスプレイ**

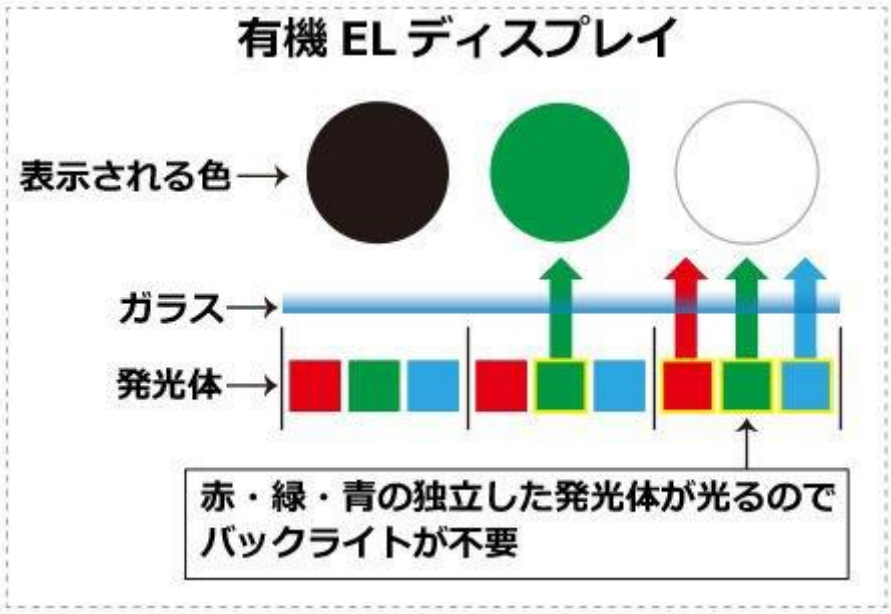
  電圧の有無によって液晶分子を制御。外部からの光によって画面を表示させる。



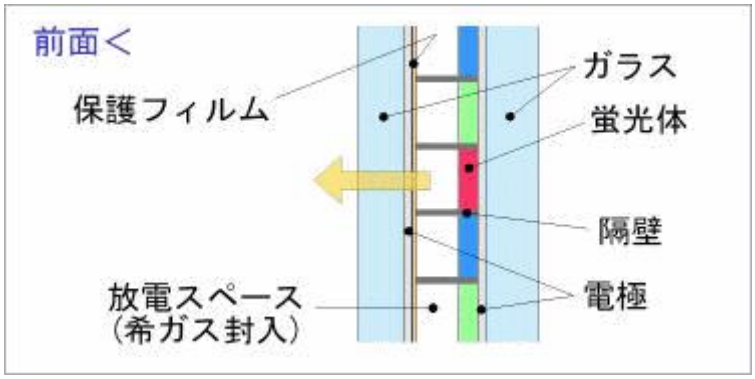


- **有機ELディスプレイ**

  有機化合物に電圧を加えることによって発光させ、画面を表示させる。



- **プラズマディスプレイ**



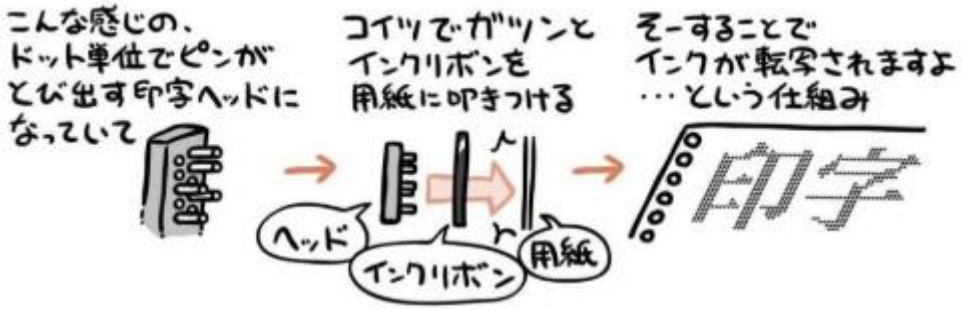
  2枚のガラスの間に、封入された希ガスに電圧をかけると放電し、紫外線が出る。そして、この紫外線が蛍光体を発光させることによって画面を表示する。 液晶ディスプレイとのシェア差が大きくなり、2014年に世界的に生産が終了された。

- **LEDディスプレイ**

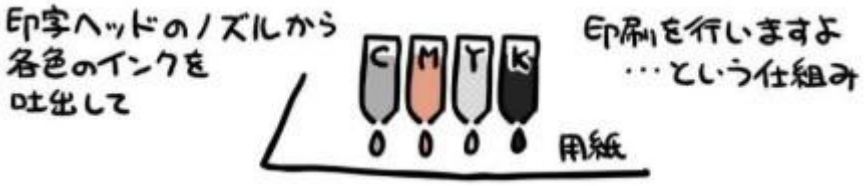
2018年1月に開催された「CES 2018」でサムスンが発表した“マイクロLEDテレビ”「The Wall」は、従来の「液晶」や「有機EL」とは異なる新たな表示方式を採用したテレビとして、大きな話題となった。

◆ プリンタ

- **ドットインパクトプリンタ**



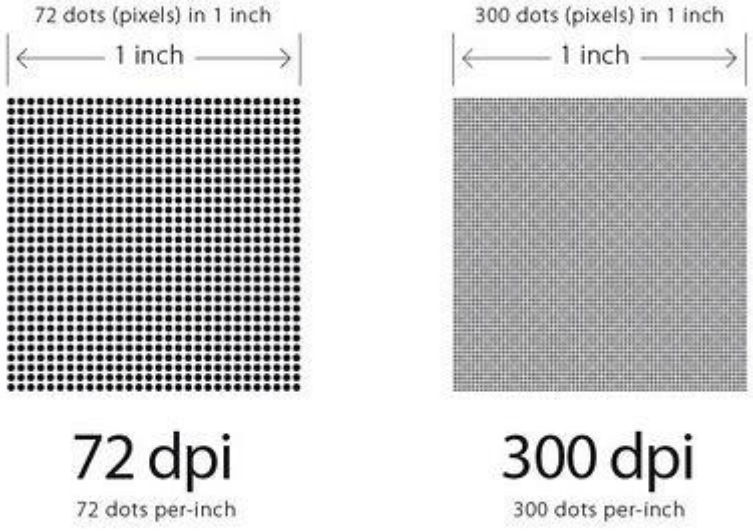
- インクジェットプリンタ



- レーザプリンタ

- **プリンタの解像度**

  １インチ当たりのドット数（dpi）によって、解像度が異なる。復習ではあるが、PC上では、ドット数がどのくらいのビット数を持つかで、解像度が変わる。



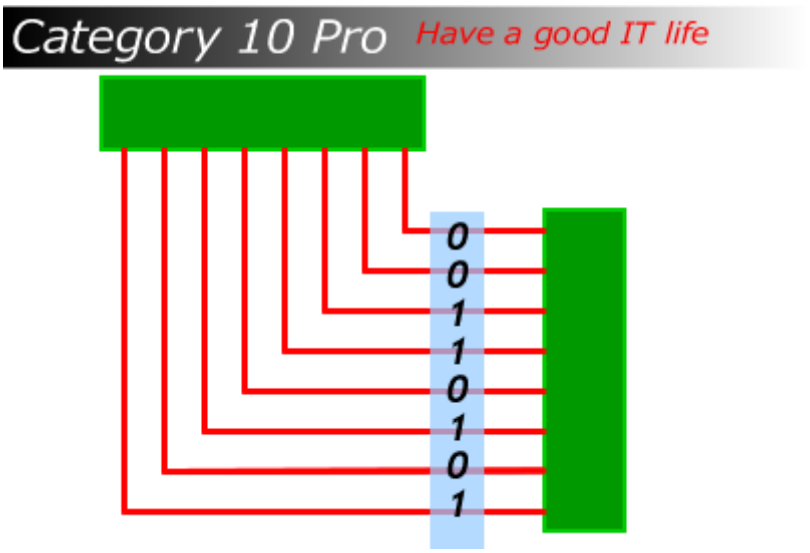
dpiが大きくなるにつれて、解像度は大きくなる。

- **プリンタの印字速度**

03-02 入出力インターフェイス

◆ Serial interface vs. Parallel interface

シリアルインターフェイスは、情報を1bitずつ転送する方式。パラレルインターフェイスは、複数のbitの情報を同時に転送する方式。パラレルインターフェイスは、同時にデータを送信し、同時に受信しなければならない。配線の形状や長さによって、信号の転送時間は異なる。動作クロックが速ければ速いほど、配線間の同時転送に誤差が生じてしまうため、現代の高スペックパソコンには向いていない。



写真出典：@ＩＴ「ケーブル＆コネクタ図鑑」シリアルポート／パラレルポート用コネクタ
https://www.atmarkit.co.jp/ait/articles/0207/10/news003.html

◆ Serial interface が用いられている例

- **USB（Universal Serial Bus）**

- **IEEE1394**

ビデオカメラとの接続に用いられるインターフェイス

◆ Parallel interface が用いられている例

- **IDE（Integrated Drive Electronics）**

ハードディスクとの接続に用いられるインターフェイス。

- **SCSI（Small Computer System Interface）**

ハードディスク、CD-ROM、イメージスキャナなど、様々な周辺機器をデイジーチェーンするために用いるインターフェイス。

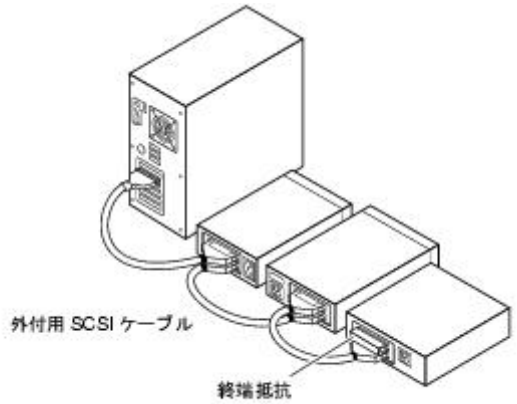


図 2. 外付用 SCSI デバイスのデイジーチェーン接続

◆ 無線インターフェイス

- **IrDA（infrared Data Assoiciation）**
  赤外線を使って、無線通信を行うためのインターフェイス。

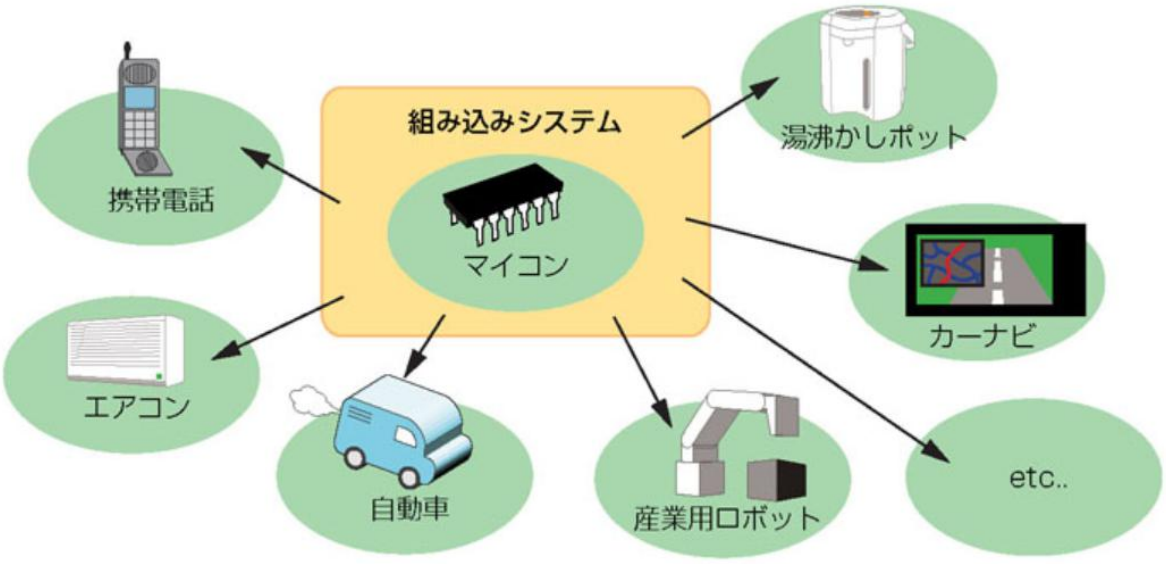
- **Bluetooth**
  2.4GHzの電波を使って無線通信を行うためのインターフェイス。

04-01. 組み込み機器と組み込みシステム

◆ 組み込みシステムとは

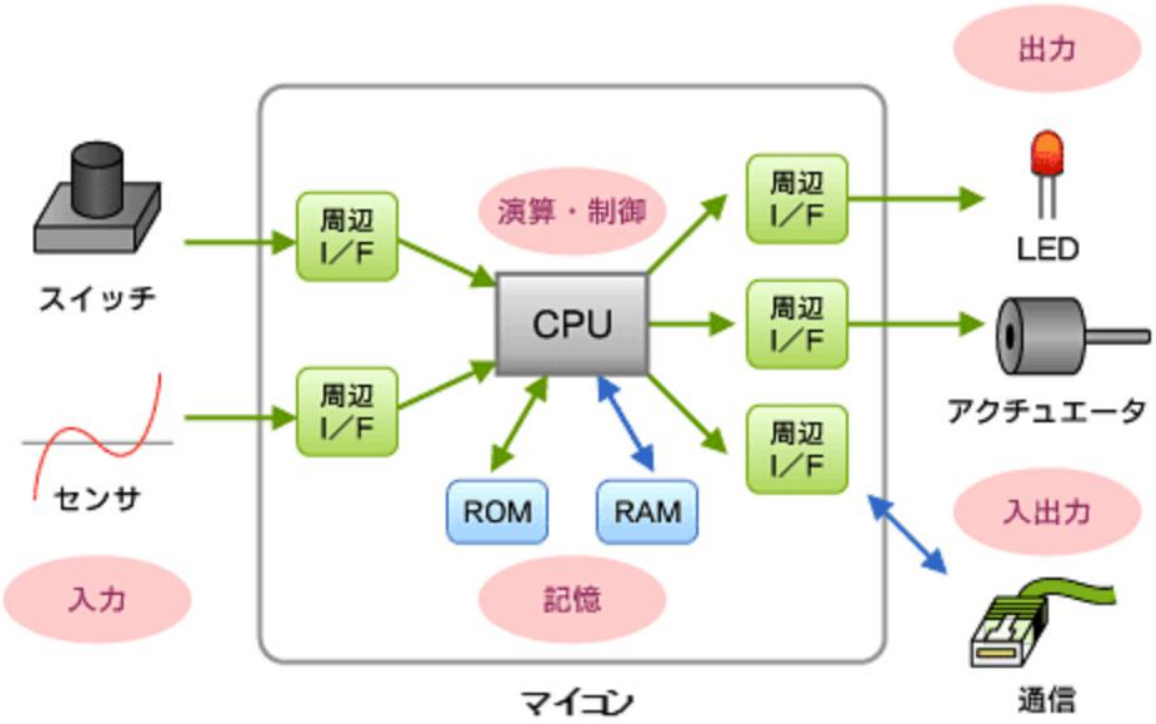
組み込み機器（限定的な用途向けに特定の機能を果たす事を目的とした機器）を制御するシステムのこと。

※パソコンは、汎用機器（汎用的な用途向けに多様な機能を果たす事を目的とした機器）に分類される。



◆ マイクロコンピュータとは

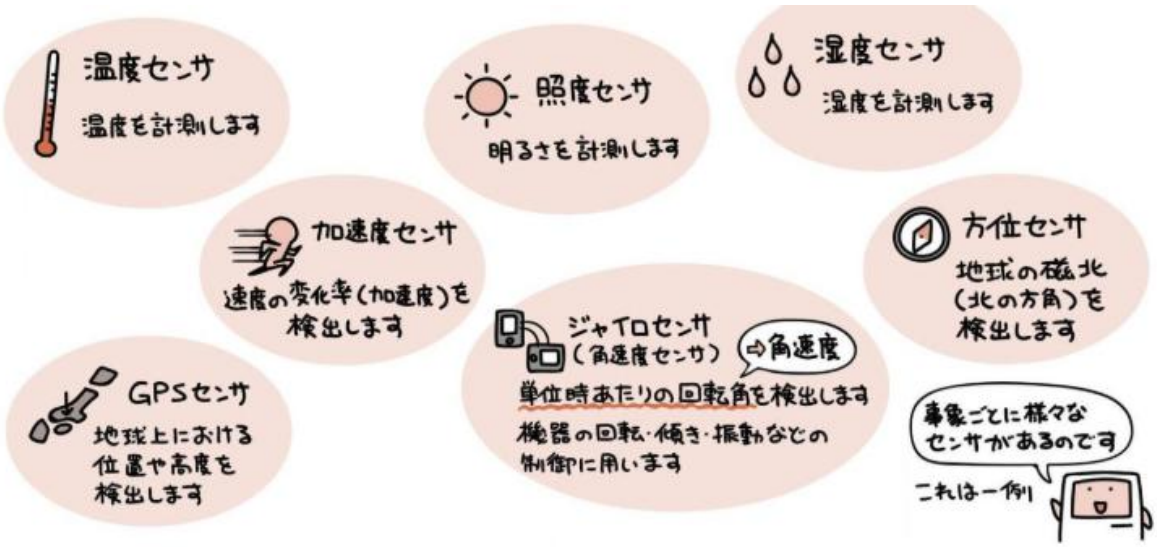
CPUとして、『マイクロプロセッサ』を用いたコンピュータのこと。



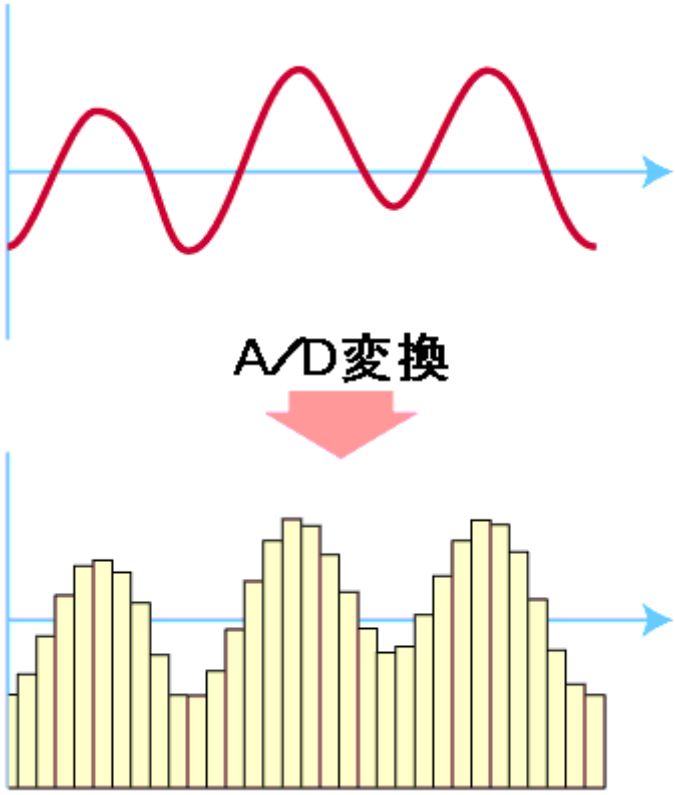
◆ センサによるアナログ情報の入力

外部のアナログ情報を計測し、マイクロコンピュータへ転送する。

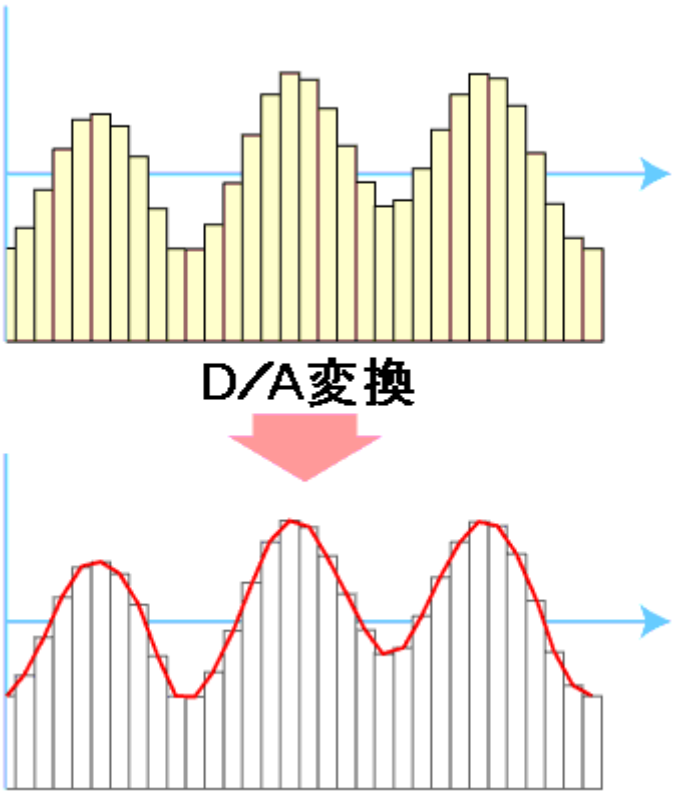
【具体例】：温度センサ、加速度センサ、照度センサ、...



◆ A/D変換器によるアナログ情報からデジタル情報への変換



◆ D/A変換器によるデジタル情報からアナログ情報への変換



◆ Actuater

入力されたエネルギーもしくはコンピュータが出力した電気信号を物理的運動に変換する装置のこと。

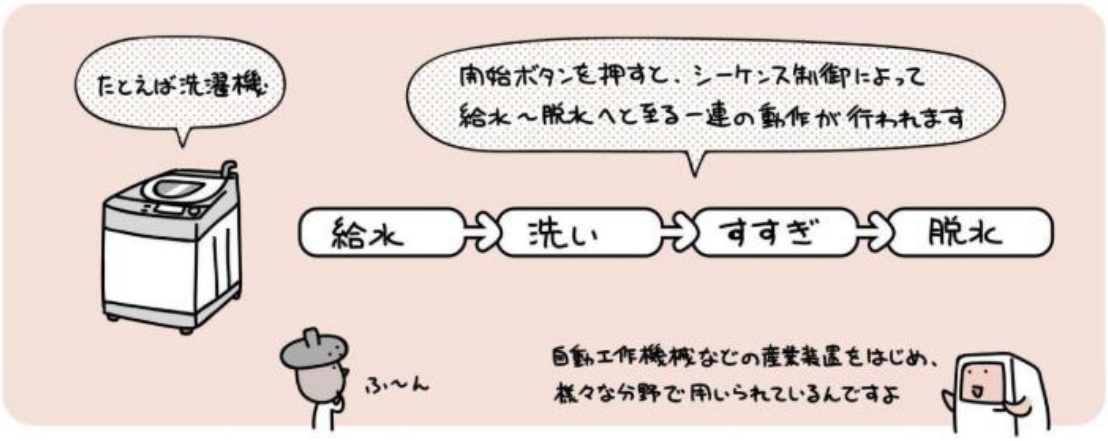
◆ 組み込みシステムの制御方式の種類

- **シーケンス制御**

  決められた順序や条件に従って、制御の各段階を進めていく制御方式。

  【具体例】

  洗濯機



- **フィードバック制御**

  その時の状況を定期的に計測し、目標値との差分を基に、出力を調節する制御方式。

  【具体例】

  エアコン