

# CPU の動作

図 1●CPU 内部では 4 つのステップで命令を実行する

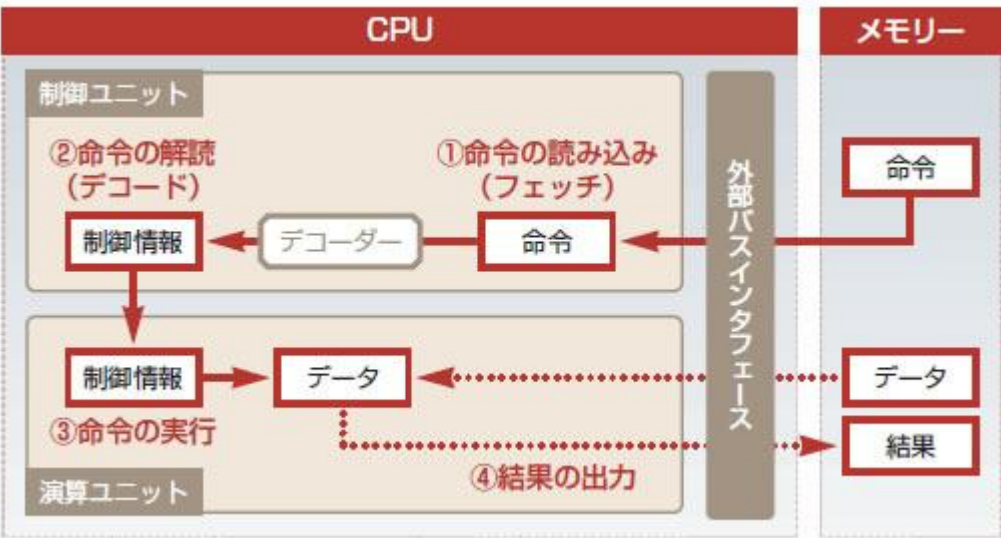


図 2●命令の実行は演算ユニットの中の演算器が行う

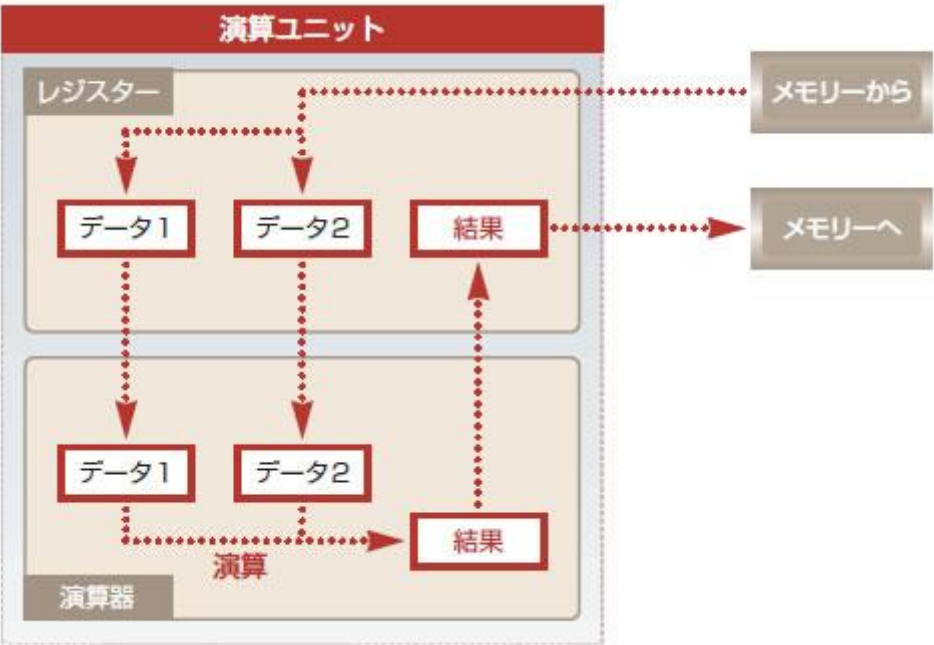
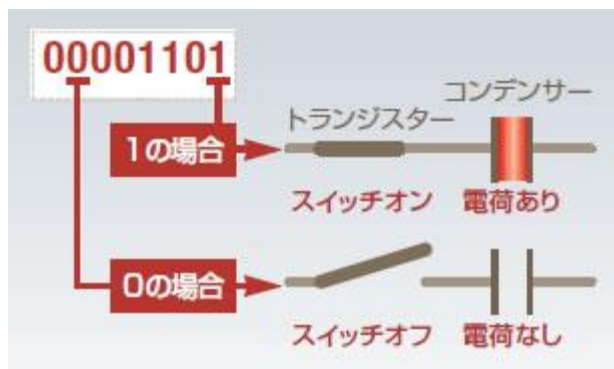


図 3●演算は 0 と 1 だけを使って行われる



図 4●0 と 1 は電荷で表現する



ワード (英: word) は、

コンピュータで扱うデータ量の単位。バイト同様、コンピュータやプロセッサによってワードの定義は異なるが、現在の多くの CPU では 32 ビットや 64 ビット。バイトと同様にデータ量の単位として、また 1 ワードのデータ自体を表す言葉として使用される。

演算等の処理は 8 ビット CPU では 8 ビット単位で行われる。16 ビット、32 ビットなどの CPU ではそれに加えて 16 ビットや 32 ビットなどを単位としても処理を行える。そのような場合にその処理単位をワードと表現した。

そのため、ワードの大きさはコンピュータが一括して取り扱える情報量を基準にしている場合がある。これは CPU のレジスタ長などによって決まるが、ワードの大きさに明確な基準があるわけではなく、CPU 設計者の定義による。

一方、コンピュータによっては、ワードが演算等の最小単位になっているものもある。このようなコンピュータの場合、1 ワードが 12 ビット、16 ビット、36 ビットなどで構成され、そのビット単位でレジスタ演算やメモリとのやり取りが行なえる。メモリアドレスもワード単位である。

## CPU の基本動作

プログラムは、命令の実行手順が記述されたものです。そして命令の実行は、**命令取り出し段**

階と命令実行段階の2つの動作を繰り返します。この過程を命令サイクルといいます。

命令取り出し段階と命令実行段階による一連の流れのうち、個々の過程をステージといいます。最も基本的な命令実行過程は、次の4つの過程(4ステージ)に分けられます。

1. F: 命令の取り出し
  2. D: 命令の解読
  3. E: 有効アドレスの計算と命令の実行
  4. WB: 演算結果をメインメモリに書き戻す
- 

## ●CPU 命令の構成

命令部	アドレス部(オペランド部)
-----	---------------

「命令部」は「実行する操作」で「アドレス部」は「操作の対象となるデータのアドレスやレジスタ」を示します。以下のようにアドレス部の個数により、4種類の形式があります。

1 アドレス方式	
命令部	アドレス部
「アキュムレータ」(計算結果一時保存場所)へ、ある番地のデータを足し込む場合に使用	

2 アドレス方式		
命令部	アドレス部 1	アドレス部 2
これは、1 番地と 2 番地の内容を加算して 1 番地へ書き込むような場合に使用		

3 アドレス方式			
命令部	アドレス部 1	アドレス部 2	アドレス部 3
1 番地と 2 番地の内容を加算して 3 番地へ書き込むような場合に使用			

## ●アドレス修飾

命令の対象となるデータが格納されているアドレスを「有効アドレス」と言います。通常は何らかの操作を加えて有効アドレスとします。このように命令のアドレス部(アドレス番地のデータ)にある操作を追加で行って決定づけることを「アドレス修飾」と言います。

## ●アドレス指定方式 補足

- ・インデックスアドレス指定方式(命令語のアドレスに添え字の値を表すレジスタ値を加えて参照)
- ・ベースアドレス指定方式(命令語でベースレジスタからの相対位置を指定してアドレスを指定)

再配置可能には2つの意味があります。

1. メモリのどこにでも入れられるプログラム
2. メモリに読み込んだあと、場所を動かしても動くプログラム

まず前提がいくつかあります。

1. プログラムはメモリに読み込まれた時にどこの位置に入るか分かりません。
2. 読み込まれた時、先頭アドレスがベースレジスタ(基準番地)に入られます。アドレス部に先頭を0番地としたときのデータのアドレス(変位)を入れておきます。

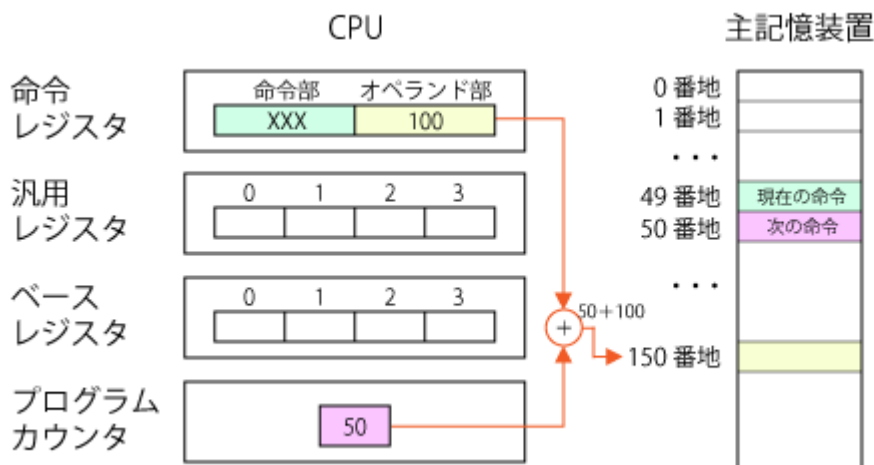
### ・自己相対アドレス指定方式

プログラムカウンタの値を加算

特徴:ベースアドレス指定とほぼ同じ

プログラムカウンタの値と、オペランド部で指定した値の合計が目的の番地となる指定方式。

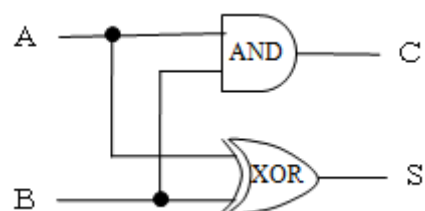
プログラムカウンタは、次に実行する命令の番地を記憶しているレジスタです。ベースレジスタや汎用レジスタを持たないコンピュータ(ミニコンやマイコンで利用されている)で、再配置可能なプログラムを作成する場合に利用されます。



## 半加算器と全加算器

1ビットの数AとBを加算して、その結果を1ビットの数Sに入れ、繰り上がり(キャリーという)を1ビットの数Cに入れる回路を半加算器といいます。

入力		出力	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



### 機械語のプログラムの例

番地	命令(16進数で表現)	ニーモニック
D000	5E	LD E,(HL)
D001	23	INC HL
D002	56	LD D,(HL)
D003	F3	DI
D004	3E FE	LD A,FE
D006	D3 71	OUT (71),A
D008	CD 7261	CALL 7261
D00B	3E FF	LD A,FF
D00D	D3 71	OUT (71),A
D00F	FB	EI
D01C	13	INC DE
D01D	13	INC DE
D01E	13	INC DE
D01F	13	INC DE
D020	D9	EXX

D021	0E 02	LD C,02
D023	06 08	LD B,08
D025	D9	EXX
D026	0E 02	LD C,02
D028	D5	PUSH DE
D029	E1	POP HL
D02A	06 08	LD B,08
D02C	CB	EXX
D02D	06 1F	LD B,1F
D02F	23	INC HL
D030	23	INC HL
D031	10 F9	DJNZ D02C
D033	CD 3ED4	CALL 3ED4
D036	0D	DEC C
D037	20 F1	JR NZ,D02A
D039	D9	EXX
D03A	10 E9	DJNZ D025
D03C	D9	EXX
D03D	13	INC DE
D03E	D9	EXX
D03F	0D	DEC C
D040	20 E1	JR NZ,D023
D042	D9	EXX
D043	C9	RET
D045	00	NOP

### ニーモニック

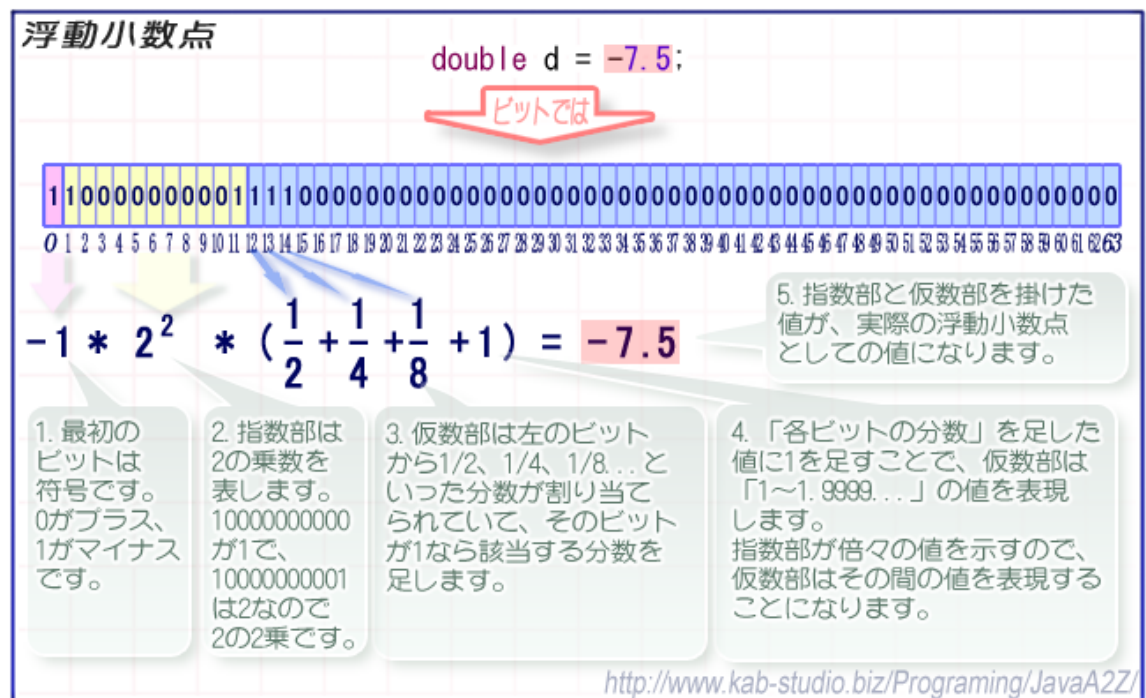
CPU は命令と書いてあるパターンで作業するが、人間にはこれはきついで、それぞれどのような命令かを短く書いたものがニーモニックと呼ばれるものである。これならば少し楽

になる。ニーモニックはプログラミング言語ではありません。単にわかりやすい記号にしたものです。

LD	load	データの転送
INC	increase	1増やす
OUT	out put	出力
JR	jump relative	相対ジャンプ
RET	return	戻る
NOP	no operation	なにもしない

メモリ上に展開された機械語のプログラム

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
D000 5E 23 56 F3 3E FE D3 71 CD 72 61 3E FF D3 71 FB
D010 13 13 13 13 D9 0E 02 06 08 D9 0E 02 D5 E1 06 08
D020 CB 06 1F 23 23 10 F9 CD 3E D4 0D 20 F1 D9 10 E9
D030 D9 13 D9 0D 20 E1 D9 C9 00
```



「浮動小数点演算というのは、指数形式の数値を使う計算のこと。エクセルでも表示形式の中に『指数』というのがあって、例えば『1230』は『1.23E+03』って表示されるよね。これは $2.3 \times 10$  の 3 乗』という意味。こういう指数形式で表現した数値のことを浮動小数点数 (floating-point number) って言うの」

「浮動小数点というのは指数形式のことなんですね」

『1.23E+03』は見た目の小数点の位置と本当の小数点の位置が違っているよね。小数点位置が動くから浮動小数点って言うわけ！

「なるほど」

「浮動小数点の反対は固定小数点。例えば『123.1』のようにそのままの小数点位置で表現したり、位置を移動するとしても必ず一定の桁数だけ移動するの」

「固定小数点の方がわかりやすいですね。どうして浮動小数点を使うんですか？」

「浮動小数点の方が少ない桁で広い範囲の数を表現できるの。例えば、1000兆分の1を表すには、固定小数点では『0.000000000000001』になって、たくさんの桁が必要だけど、浮動小数点なら『1.0E-15』と書くだけで済む。コンピュータでは桁数が少ない方が速く計算できるから、広い範囲の数を高速に計算するには固定小数点より浮動小数点の方が都合がいいの」

「そうなんですか」



固定小数点	浮動小数点
123	1.23E+02
1230000000000000	1.23E+15
0.0000000000000123	1.23E-13

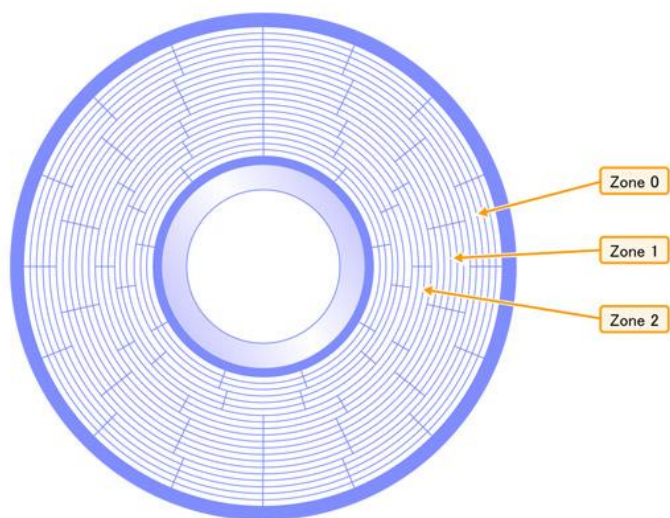
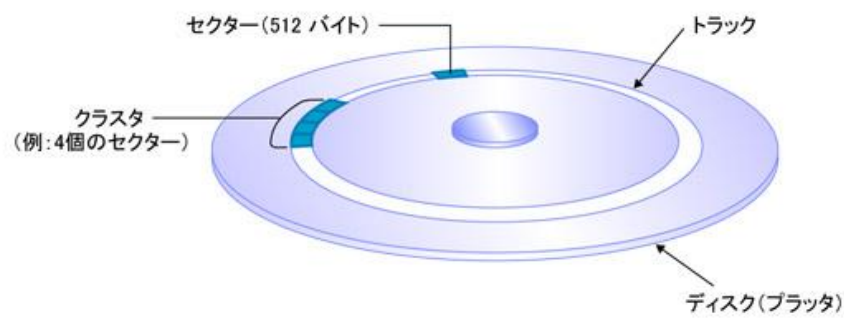
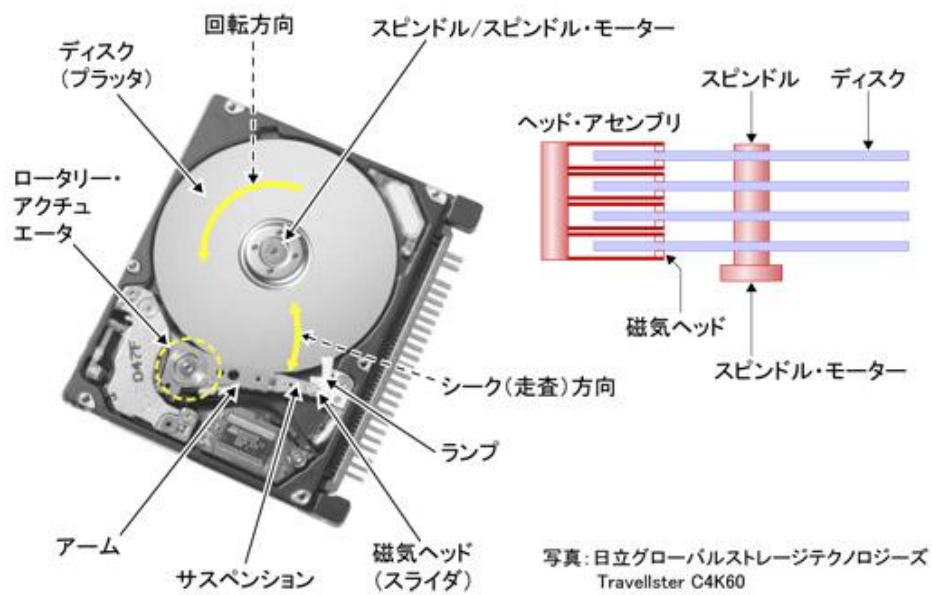
固定小数点では数の大きさによって、表現するために必要な桁数が増減する。  
浮動小数点では指数が増減するだけで、桁数は変わらない。

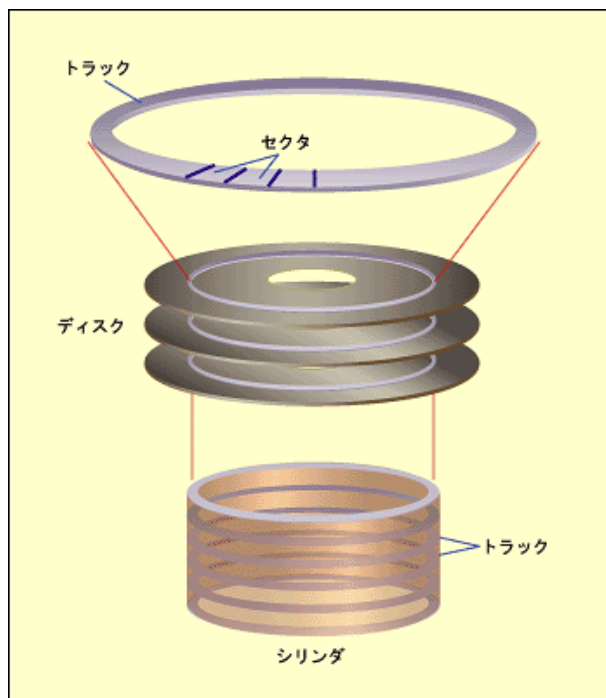
## ハードディスクドライブの内部構造

データは回転する「ディスク」の上に記録され、それを読み取る（あるいは書き込む）のは「磁気ヘッド」である。磁気ヘッドは「アーム」の先に取り付けられ、アームには磁気ヘッドを適度にディスク面に押し当てる「サスペンション」が付いている。

磁気ヘッド、サスペンション、アームの三つが一体化された「ヘッド・アセンブリ」が円弧を描くように走査（シーク）して、先端の磁気ヘッドがディスク上のデータにアクセスする。

磁気信号をディスクに記録する。ディスクの両面には磁性層膜が形成され、この層の上にヘッドがデータに応じた磁化のパターン（N 極と S 極の配列・後に 0 と 1 に置き換え）を記録する。記録されたデータを読み込むには、ヘッドで磁性層膜上に記録された磁化のパターンから磁界を検出し、データを再生する。





### トラックとセクタとシリンダ

トラックとは、ディスク・ドライブやテープ・ドライブ、光ディスクなどの記憶装置において、データを記録する、線状に連なった部分のこと。トラックの中にはいくつかのセクタがあり、データはセクタ単位で読み書きされる。トラックは同心円状に多数配置されている場合と、連続した、らせん状に配置されている場合がある。

同心円状のトラックの場合は、同じ位置にある各記録面のトラックをひとまとめにして、シリンダと呼ぶ。同じシリンダ上にあるトラックはヘッドをシークしなくてもよいので高速にアクセスできる。そのため連続してデータを記録する場合は、同一シリンダ上のすべてのトラックへ書き込んでから次のトラックへ移動するのが普通である。

---

### ●フリップフロップ回路

「high」と「low」の二つの安定状態を持つ電子回路。二つの状態を「0」と「1」に対応させることで、1ビットの情報を保持できる。加える信号によって二つの状態が交互に変化するようにできている。大規模な電子回路を構成する基本的な素子で、SRAM や、マイクロプロセッサ内部のレジスタ等の記憶回路に使われる。

---

### ●割り込み

#### ・ 実現方法

- ソフトウェア:ソフトウェア割り込み

- ・ 例:ユーザプログラムからシステムコールの呼び出し

- ハードウェア:ハードウェア割り込み(単に,「割り込み」と言うと,通常,ハードウェア割り

込みを指すことが多い)

- ・ どこから割り込まれる

- 計算機内部:内部割込み(割り出しとも言う);例:オーバ／アンダフロー, アクセス違反

- 周辺装置:外部割込み

- ・ 外部割込み事象の例:

- キーボードからの入力

- マウス操作通知:移動, クリックなど

- ディスクからの入出力完了連絡

- ・ ディスクへのファイル書き出し完了通知

- ・ ディスクからメモリへのページのロード完了通知

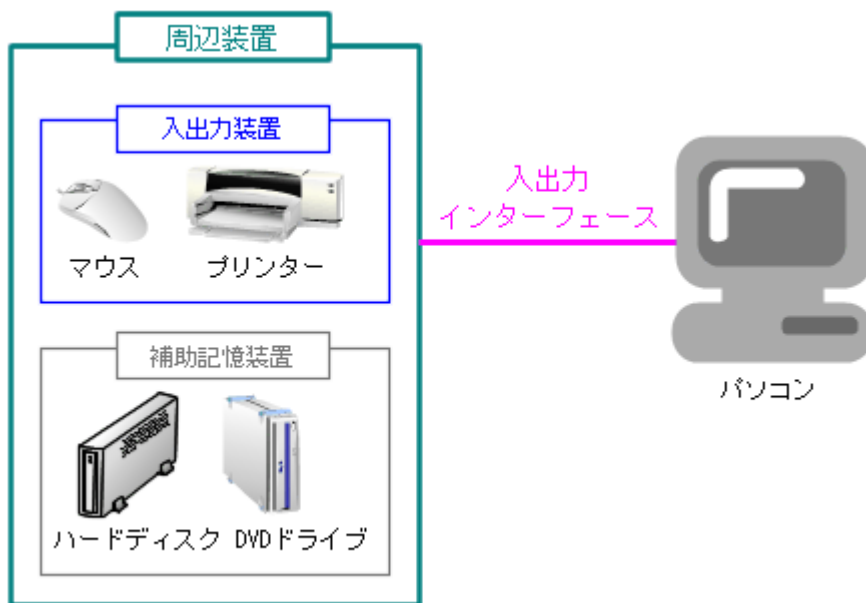
- ネットワークからのメッセージ

- タイマからの時刻通知

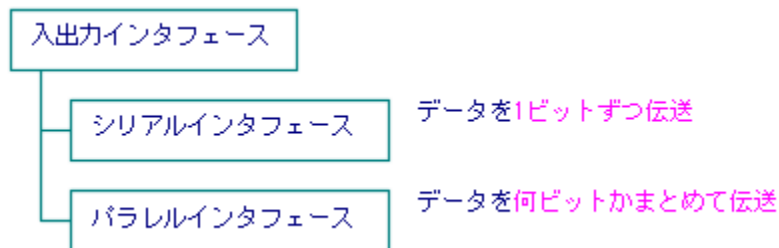
---

## 入出カインタフェース

入出力装置と補助記憶装置を併せて周辺装置という。周辺装置とパソコンの接続方式のことを入出カインタフェースという。



入出力インターフェースは、データの転送単位により、1ビットずつ転送する**シリアルインターフェース**とデータを何ビットかまとめて転送する**パラレルインターフェース**の2つの方式がある。



Point 入出力インターフェースは2種類の方式がある

- 1ビットずつ転送する**シリアルインターフェース**
- 何ビットかまとめて転送する**パラレルインターフェース**

## シリアルインターフェース

シリアルインターフェースは以下のようなものがある。

種類	特徴	伝送速度
RS-232C	モデムなどを接続するときに使用される	
USB	USBはUniversal Serial Busの略で、パソコンと周辺機器全般の接合に使用される。接続可能な機器の数は、USBハブを用いることで最大で <b>127台</b> である。	USB1.1 12Mビット/秒 USB2.0 最大 480Mビット/秒

	USB の規格には USB1.1 とさらに高速転送可能な USB2.0 がある。	
IEEE1394	パソコンと周辺機器全般の接続に使用される。接続可能な機器の数は、最大で <b>63 台</b> である。	100M ビット/秒 200M ビット/秒 400M ビット/秒

## パラレルインターフェース

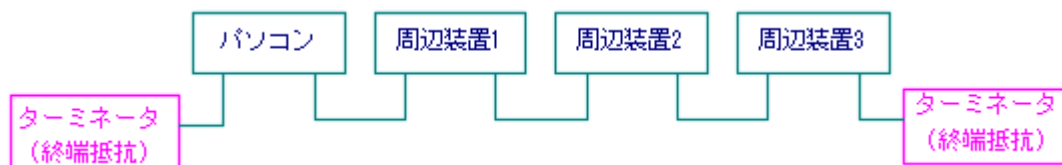
パラレルインタフェースは以下ようなものがある。

種類	特徴
IDE/E-IDE	ハードディスクや CD-ROM 装置を接続するときに使用する。接続台数は、IDE で 2 台、E-IDE で 4 台まで接続可能である。
SCSI	パソコンと周辺装置を接続するときに使用される。8 ビットのパラレルインタフェースである。装置から装置へ芋づる式に接続する <b>ディジーチェーン接続</b> によって <b>最大 7 台</b> の装置が接続可能である。ケーブルの最大長は規格によって異なるが 1.5m から 25m である。
セントロニクス	パソコンと <b>プリンタ</b> の接続に使用される。8 ビットのパラレルインタフェース。
GPIB	<b>計測機器</b> を接続するときに使用される。接続可能台数は、最大で 15 台である。

## ディジーチェーン接続

SCSI で用いられるディジーチェーン接続は次のように接続する。両端の装置には、ターミネータと呼ばれる終端抵抗が必要である。

### ディジーチェーン接続



### ディジーチェーン接続

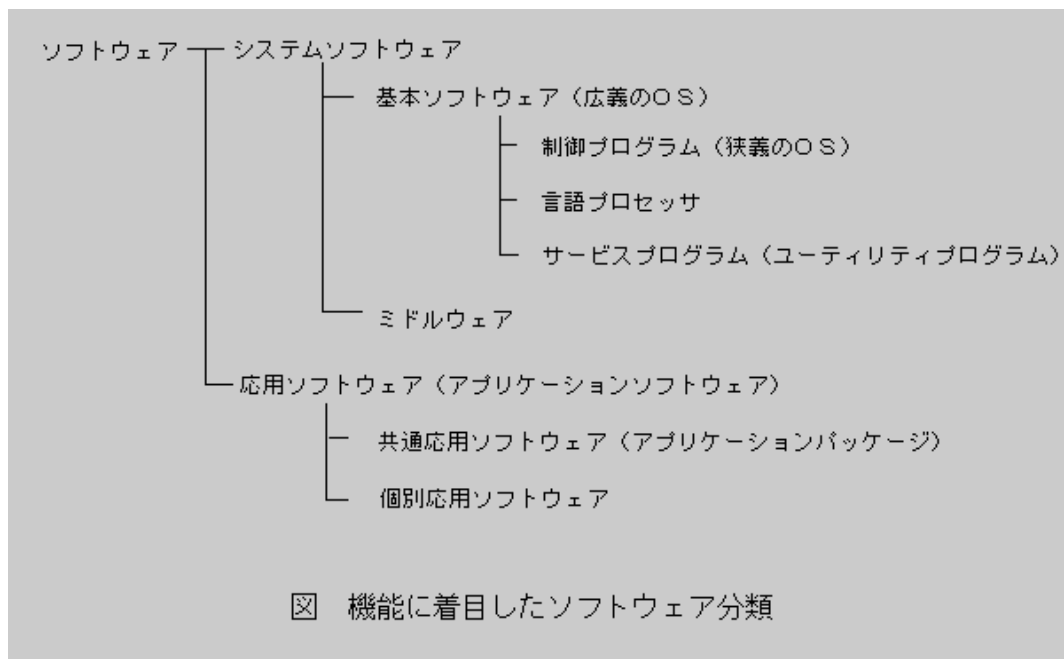
- 装置間を芋づる式に接続する
- 両端の装置には、ターミネータ(終端抵抗)が必要である

## 装置間をコードレスで接続する

シリアルインタフェースおよびパラレルインタフェースは、その規格のケーブルでパソコンと周辺装置を接続する。パソコンと周辺装置を接続するのにコードレスで接続する方法がある。赤外線を使った IrDA(Infrared Data Association)と無線を使った Bluetooth がある。

### IrDA と Bluetooth

種類	特徴
IrDA	赤外線を使って装置間を接続する。ノートパソコン、PDA(Personal Digital Assistants)などで使われる。通信可能距離は最大で 1m である。装置間に遮へい物があると通信できない。
Bluetooth	無線を使って装置間を接続する。ノートパソコン、PDA、デジタルカメラ、携帯電話などで使われる。通信可能距離は機器間の障害物の有無にかかわらず 10m である。



機能に着目してソフトウェアを分類した図です。

まずソフトウェアは、システムソフトウェアと応用ソフトウェアに分類することができます。

**システムソフトウェア**とは、ハードウェアの機能を効率的に活用し、コンピュータの利用を容易にさせる機能を持つソフトウェアの総称です。

この説明だけを見ると、先週説明したオペレーティングシステムとあまり変わりはないように思え

ます。確かに、オペレーティングシステムはシステムソフトウェアの一部です。

しかしシステムソフトウェアはミドルウェアを含み、もう少し広い範囲のソフトウェアを指します。ミドルウェアについてはこのページの後の方で説明しましょう。ここでは、システムソフトウェアは基本ソフトウェアとミドルソフトウェアの総称である、と覚えておいてください。

次に**基本ソフトウェア(広義のOS)**ですが、これは**制御プログラム(狭義のOS)**と言語**プロセッサ**と**サービスプログラム(ユーティリティプログラム)**を総称する用語です(合わせた機能を持つのではなく、あくまで総称する用語ですよ)。

サービスプログラム(ユーティリティプログラム)とは、様々な分野に共通した基本的な処理を支援するプログラムです。これは主にコンピュータメーカーが提供します。とは言っても何のことやら分かりませんね。ここでは、パソコンのことは忘れてください。

汎用コンピュータでは一般に、コンピュータの直接のユーザ(使う人)はソフトウェア開発者(というかアプリケーションソフトの開発者)です。そして、ソフトウェア開発を行うに当たって定型的な処理や作業があるのですが、これを支援するのがサービスプログラムです。

例えば、プログラムを組むには絶対と言ってよいほど**(テキスト)エディタ**が必要です。これはそのコンピュータのユーザ(アプリケーションソフト開発者)にとって(分野に関係なく)共通的に必要なプログラムです。

パソコンの世界とは異なり、汎用コンピュータはコンピュータが異なれば同じソフトは動作しません。だからそのエディタは当該コンピュータでしか動作しないのです。だから、予めコンピュータメーカーがエディタを作っておき、コンピュータを提供する時に(OSらと含め)それと込みで提供します。このようなソフトウェアがサービスプログラム、と言えばちょっとは分かってもらえたでしょうか。

サービスプログラムはエディタの他に、**分類併合プログラム(ソート/マージプログラム)**、**システム生成プログラム**、**ファイル変換プログラム**、**ライブラリ管理プログラム**、**連携編集プログラム**等があります。これを一つ一つ説明する訳にはいかないなので、キーワードだけざっと眺めておいてください。

次に**応用ソフトウェア**の説明に移ります。

**応用ソフトウェア(アプリケーションソフトウェア)**とは、利用者の目的に応じた処理を行うソフトウェアの総称です。ここでの利用者とは開発者ではなく、一般のユーザを指します。つまり、コンピュータを使って実際の業務を処理するソフトウェアが応用ソフトウェアです。

応用ソフトウェアはさらに、**個別応用ソフトウェア**と**共通応用ソフトウェア**とに分類することができます。

**個別応用ソフトウェア**とは、その適用業務が限定され、各業務固有の処理を行う**応用ソフトウェア**のことをいいます(オーダーメイド的な性格を持つ)。例えば、JRのみどりの窓口で動作する予約システムや、あるコンビニで動作する**POSシステム(販売時点情報システム)**がこれにあたりま



す。

対する共通応用ソフトウェア(アプリケーションパッケージ)とは、多様な業種や業務に適用できる応用ソフトウェアです(既製品の性格を持つ)。例えば、CADシステムなどがこれにあたります。

---

インターレース PNG



インターレース GIF

## ■プログラムにおける静的・動的リンクの特徴

コンピュータのプログラム作成時においては、一般に大規模なプログラムをモジュールに分割して、コンパイル後に、オブジェクトファイルを汎用ライブラリと共につなぎ合わせて実行可能形式のバイナリを作成する。これを**静的リンク**と呼ぶ。

それとは異なり、プログラムを実行する時に初めて他のモジュールやライブラリと結合される方式を**動的リンク**と呼ぶ。この動的リンクを使ったライブラリを、共有ライブラリあるいはダイナミックリンクライブラリ(DLL)と呼ぶ。動的リンクの利点として、実行可能形式のプログラムサイズを小さくできること、共有ライブラリをバージョンアップしたときにプログラムを再コンパイルする必要がないことがあげられる。

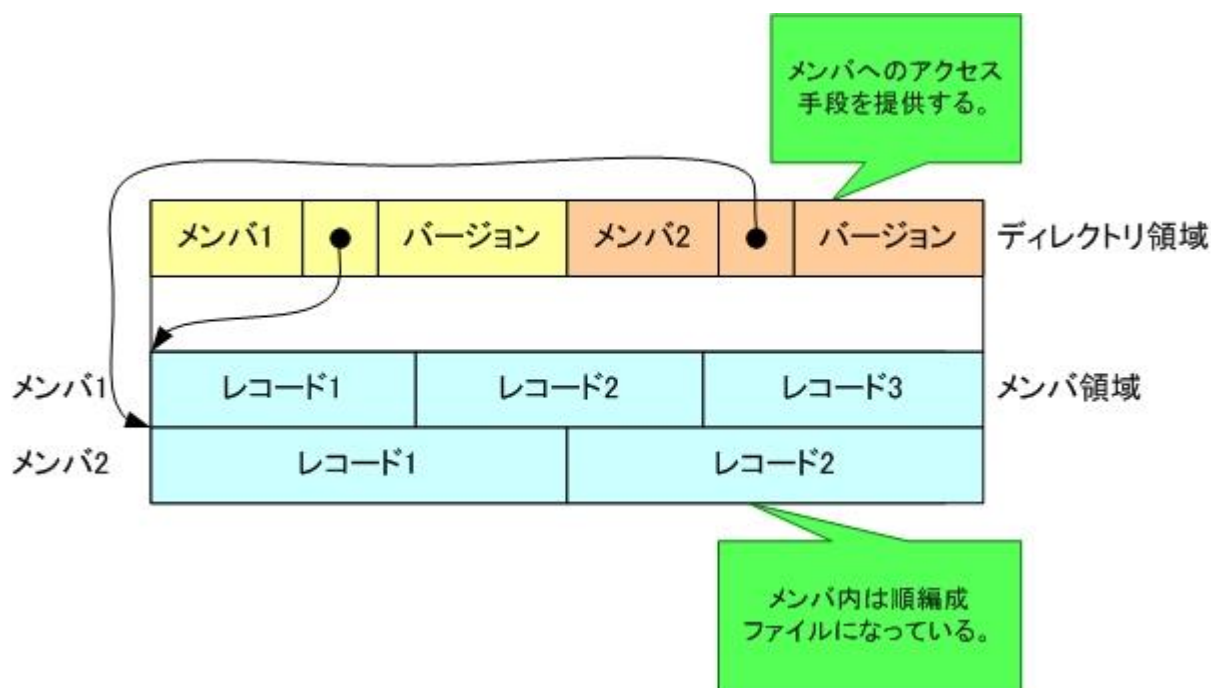
欠点としては、暗黙的に特定のバージョンの共有ライブラリの内部処理や仕様に依存していたプログラムがライブラリのバージョンアップによって動作しなくなる場合があること。

---

## ■区分編成ファイル

順編成法と索引編成法の両方の特徴を併せ持つファイルのこと。ファイル領域をディレクトリ領域とメンバ領域に分ける。レコードはメンバ領域に格納され、その索引がディレクトリ索引に格納される。

メンバ領域が順編成ファイルであることと、ディレクトリ領域に名前・格納位置・メンバ長・登録日・バージョンなどの様々な情報が格納されるという点で索引編成ファイルと異なる。



区分編成ファイルはバージョン管理が容易なので、プログラムライブラリの格納などに利用される。

メンバは順編成ファイルと同じ編成で、メンバの名前や格納位置の情報はディレクトリ（登録簿）に登録される。

#### ■VSAM(Virtual Storage Access Method)ファイル

汎用コンピュータの仮想記憶OSで用意されているファイル編成のことである。

既出の編成ファイルはトラックサイズやシリンダサイズなどの媒体の物理的特性に大きく依存していた。これにより、アクセス性能を向上させていたが、同時に**移植性を大きく低下させる要因**にもなっていた。もしレコードを別の媒体に移行する際には、別媒体の物理特性に合わせてレコードを格納し直さなければならない。

VSAM ファイルは IBM 社の System/370 (汎用コンピュータ) で最初に採用された。仮想記憶を実現したときに同時に発表したため、この名で呼ばれるようになった (記憶管理の仮想記憶とは関係がない)。レコード全体をバイトストリームとして扱い、**媒体の物理的特性に依存したパラメータ (トラック番号やシリンダ番号など) はすべて相対値として扱う**。これにより、媒体の物理的特性から独立した編成が可能になった。

## ■データベース

“「データベース」の種類”についてご説明します。

### 第 3 回 1 章 ..... 階層型データベース



データベースは、おもに「階層型データベース」「ネットワーク型データベース」「リレーショナル型データベース」の 3 つに分類できます。

階層型データベースは、名前が示すとおりデータを階層型に格納/整理する仕組みをもったデータベースです。もっともイメージしやすい例は、会社の組織図です。階層型データベースでは、データはツリー構造で表します。そして、ある 1 つのデータが他の複数のデータに対して、親子の関係をもちます。したがって、データにアクセスするためのルートは一通りしかありません。

また、データを階層型でもつため、データの冗長が発生しやすくなります。つまり、データは常に親子 (1 対多) の関係とは限らず、「多対 1」の場合や「多対多」の場合もあり得、同じデータがあちこちの親データに所属する形になります。先ほど例としてあげた会社の組織図だと、ある一人の人物が複数の部門の仕事を兼任している、と考えればわかり易いかもしれません。兼任していた場合、組織図上は複数の部門に所属 (つまりデータベース内には複数存在) していますが、実際には一人だけです。



階層構造だと、どうしても冗長化が起こりやすくなってしまいますね。



それに、階層型データベースでデータにアクセスするには、プログラマーがあらかじめ階層構造を理解しておく必要があります。また、階層構造に変更があった場合には、それに合わせたプログラムの改変も必要になります。



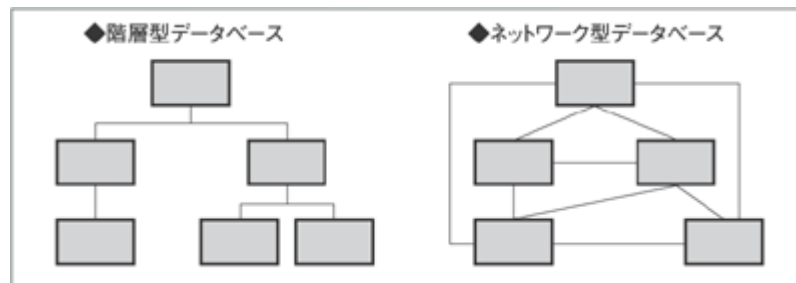
ネットワーク型データベースでは、データは網の目の形で表現されます。それぞれのデータ単位(ノード)が繋がっていることから、ネットワーク型と呼ばれています。ネットワーク型データベースでは、複数の親データへのアクセスが可能になりました。つまり、階層型で問題となっていた冗長性を排除する仕組みになっているのです。



ネットワーク型データベースも、データ構造が変更されるとプログラムに影響があるのでしょうか？



はい、影響します。階層型データベースと同じく、ネットワーク型でもデータ構造を理解してプログラム開発をする必要があります。



データベースの種類



階層型データベースとネットワーク型データベースでは、それぞれ欠点がありました。階層型では、データの冗長/重複が発生し、プログラムはデータ構造に強く依存します。ネットワーク型では、データ冗長/重複は解消されましたが、依然としてデータ構造への依存性を強くもっています。これではデータベース本来の目的である「データの整理・一元化」は難しくなります。また、データ構造を意識したプログラムの必要性や、データ構造の変更に伴うプログラム改変の必要性は、多くのユーザやプログラムがデータを使用していた場合にメンテナンスに大きな工数と負荷をかけることにもなります。これでは、ファイルシステムにおける問題を引き継いでしまい、解決になりません。

また、階層型データベースとネットワーク型データベースは、データベースの構築やデータの検索/更新のためにきわめて複雑なプログラム言語が必要になり、かなりの熟練者でないと扱うのが困難でした。



階層型データベースとネットワーク型データベースの登場により、単なるファイルシステムに比べるとデータアクセスは確かに容易になりましたが、データとプログラムの独立性(データ構造の変更がプログラムに影響を与えないこと)やデータの扱いの容易性に関しては、まだ大きな問題が残っていたのですね。

#### 階層型データベースの場合

- データの中身は同じだが、親の位置が異なることにより、重複するデータ所有が発生
- 階層型データベース構造が修正される場合、プログラムの変更も必要

#### ネットワーク型データベースの場合

- 1つのデータに対し、複数の親を持つような構造
- ネットワーク型データベース構造が修正される場合、プログラムの変更も必要

階層型データベース、ネットワーク型データベースの問題点

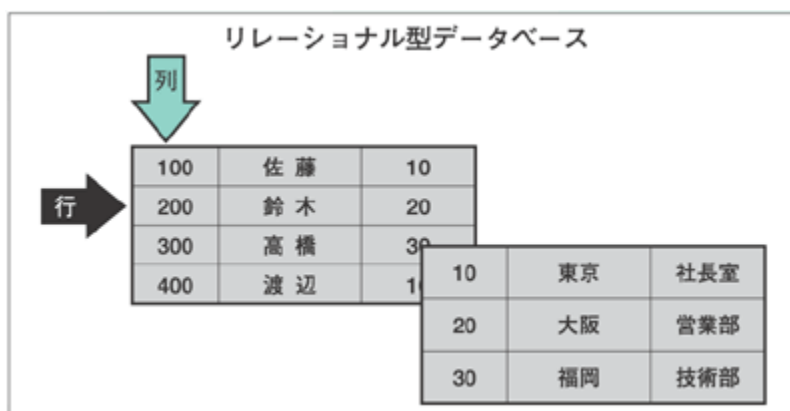
第 3 回

4 章

リレーショナル型データベース



リレーショナル型データベース(以下、リレーショナル・データベース)は、データを行と列から構成される 2 次元の表形式で表します。列は各項目を表し、行はデータのエントリー(レコード)を表します。データ同士は複数の表と表の関係によって関連付けられ、SQL(構造化問い合わせ言語)によりユーザの目的に応じて自由な形式で簡単に操作できます。そして、リレーショナル・データベースは、重複排除や一元管理の為のルールももっているのです。



リレーショナル型データベースのイメージ



階層型データベースやネットワーク型データベースと比べると、随分画期的なデータベースが開発されたように思えるのですが、データ構造への依存は解消されたのでしょうか？



ほとんど解消されています。まずはリレーショナル・データベースの利点について簡単に説明しましょう。

リレーショナル・データベースは階層型・ネットワーク型に比べると以下の点で大きなアドバンテージを誇っています。

- プログラムとデータの分離  
プログラムとデータの独立性が高いため、データ構造に修正が入ったとしてもプログラムへの影響は極めて小さい。
- 柔軟かつ容易なデータの取り出しが可能
- データベース操作の簡略化

SQLにより、データベースの構築や問合せが簡単になりました。



なるほど。リレーショナル・データベースの登場により、そもそもデータベースによって実現したかった「データを整理/蓄積し活用する」ことが容易になったんですね。



そのとおり。このような優位性により、リレーショナル・データベースは今日ではデータベース市場の主流を占めることになったのです。

## 主キーと代替キーの例

---

・生徒名簿（生徒番号、生徒名、クラス）というリレーションの場合、生徒番号が主キーになり得る。同姓同名を考慮すると、生徒番号は唯一の候補キーであるから、代理キー はない。

・市町村（市町村 ID、市町村名、都道府県名）というリレーションの場合、市町村 ID と {都道府県名、市町村名} が候補キーであり、いずれかが主キーになり得る。例えば、市町村 ID を主キーにした場合、{都道府県名、市町村名} は代替キーとなる。

## ■3 層スキーマ

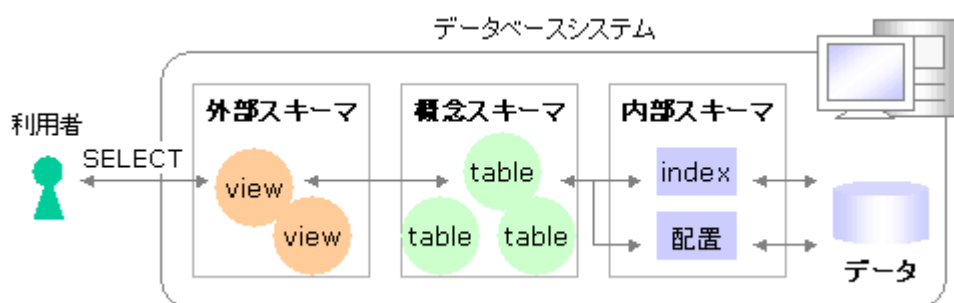
データベースシステムの基本的な構成を 3 つの構造により定義したもので、現在ほとんどの DBMS 製品で取り入れられています。3 層スキーマでは、各層が以下のように定義されています。

**外部スキーマ** データベース利用者に必要なデータの定義。ビュー(テーブルの一部のデータ)など。

**概念スキーマ** データベースで管理する対象の定義。テーブル(全データ)など。

**内部スキーマ** データの物理的な格納方法。インデックスやデータファイル配置など。

利用者がデータベースへデータ検索などを要求する場合、下図のように処理されます。



---

#### ・NDL(Network Data Language)

ネットワーク型データベースを定義・操作する為のプログラム言語。3 つの言語の構文と意味を定義している。

- ・スキーマ定義言語(スキーマ DDL):DB の構造などを定義するための言語
  - ・サブスキーマ定義言語(サブスキーマ DDL):DB 利用者のためのビューを定義する言語
  - ・データ操作言語(DML):DB を利用するための言語
- 

#### ■OS とは何か

コンピュータやパソコン関係の話をしていると、必ずと言っていいほど「OS」という言葉が出てきます。この OS とは何なのでしょう？

OS(オーエス)というのは、Operating System(オペレーティングシステム)の略であり、基本ソフトとも呼ばれています。ハードウェア(パソコン本体や周辺機器)とソフトウェア(プログラム)の動作を仲介して、コンピュータを利用できるようにするための必要不可欠なソフトです。

OS がコンピュータにインストールされていないと、そのコンピュータを動かすことはできないのです。

#### ・ソフトのインストールや起動を行う

OS はアプリケーションソフトをハードディスクにインストールして、ユーザが使えるようにします。ソフトの起動は、ハードディスクからメモリに呼び出されて行われます。

#### ・ファイルの管理

デスクトップにあるアイコンをダブルクリックすると、ソフトが起動できるように管理しています。また、アプリケーションで作成したファイルの保存場所を指定します。

#### ・ソフトに共通の機能を提供

OS はアプリケーションに共通の機能を提供しています。例えば、起動や終了、データの入力・印刷、記憶媒体に保存するなど、ソフトは違っても機能は同じになります。

共通する機能をソフトごとに搭載していくと、プログラムが長く複雑になるため、開発効率が悪くなります。OS は、アプリケーションに共通するプログラムを搭載し、要求に応じて提供しています。

#### ・ハードウェアの管理

CPU の処理時間や実行するプログラムの順序を管理しています。メモリの使用状況も管理しています。

#### ・周辺機器の制御



パソコンの周辺機器には、キーボード、マウス、ディスプレイ、プリンタなどがありますが、OS はそれらを正しく使えるようにデバイスドライバをインストールしています。

## ■スループット

単位時間あたりの処理能力。**コンピュータ**が単位時間内に処理できる命令の数や、通信回線の単位時間あたりのデータ実効転送量などを意味する。

### 【タスク】

いつ、何を、どのように実行するのか、予め定義された処理のことを「タスク」と呼びます。パソコンを起動したら必ず実行される処理はタスクです。

### 【ジョブ】

ジョブという言葉はずいぶん懐かしい用語ですね。汎用機(メインフレーム)などの大型コンピュータではこの言葉をよく使っていましたが最近聞かなくなりました。

いくつかの処理(プログラム)を組み合わせることでコンピュータに大きな仕事を実行させるときの命令と考えてよいかと思います。PC では“バッチ”(まとまった一括処理)に相当する用語です。

### 【プロセス】

タスクの一種ですが、メモリ上に常駐して電源が入っている限り、常に動いているプログラムのことをプロセスと呼びます。Unix 系 OS の用語です。Windows では「サービス」と呼ばれています。

---

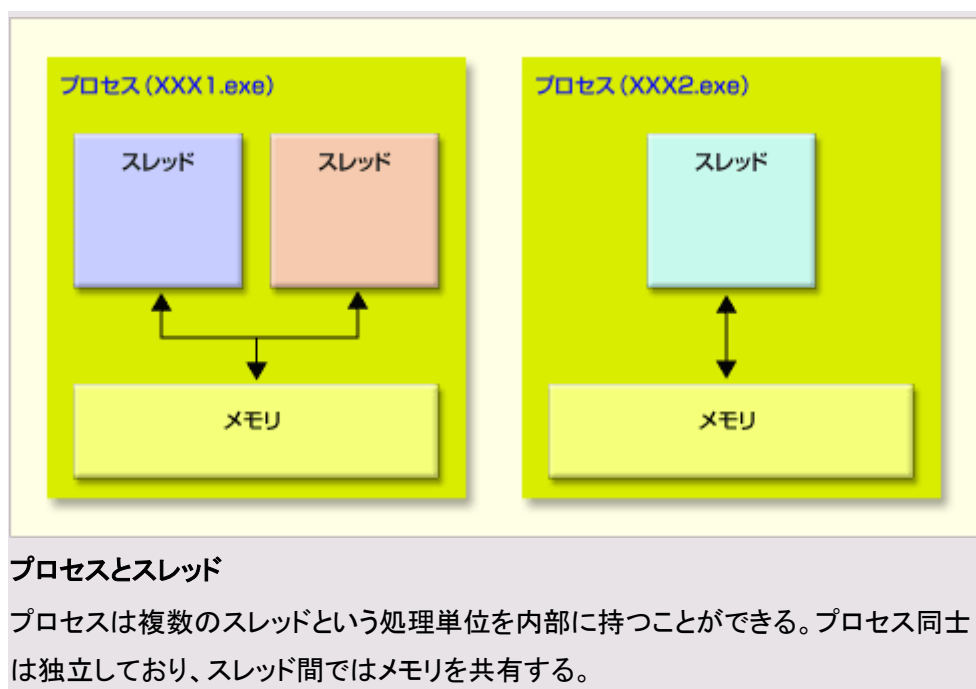
## ■バッファ(Buffer)

処理速度に差がある装置間に中継として設置されるメモリ。処理の速い装置が処理の遅い装置の速度に合わせてしまうと、全体の速度が落ちるため、データをいったんバッファに蓄えておくことで、CPU など処理の速い装置の待ち時間を減らす。バッファには、高速なメモリが使用される。たとえば、CPU とディスク装置やプリンタなどの周辺装置との間にバッファが設けられる。

---

## ■プロセスとスレッドの違い

複数のプロセスが並行動作するマルチプロセスと、プロセス内で複数のスレッドが並行動作するマルチスレッドではどこが違うのだろうか。一見すると、どちらも処理を並行して行う技術であるが、その違いは、プロセス間では基本的にメモリは共有されず、スレッド間ではメモリが共有されるという点である。つまり、異なるプロセスが同じメモリ上のデータにアクセスすることは基本的にはないが、スレッド間では同じデータに簡単にアクセスできるということである。



このため、同じデータにアクセスしながら並行動作するような複数の処理には、マルチスレッドを使った方がプログラミングは断然楽になる。

また、プロセスに比べて、スレッドの方が消費するリソース(プロセスやスレッドを管理するために必要なデータ)が少なく、新しいプロセスやスレッドを作成する場合には、スレッドの方が起動の手間やコストが小さい。

## ●ヒープ領域(ヒープりょういき)とは

コンピュータープログラミングにおいて、動的に確保可能なメモリの領域。ヒープ (heap) とは、『山積み』という言葉の中の『山』をさす英単語である。