ソフトとハードとその応用

（シスアド受験より）

基礎の基礎

16進法

16進法は2進数を4桁づつ纏めて1桁と表したものです。

対応は次の表で与えられます。

2進法	0000	0001	0010	0011	0100	0101	0110	0111
16進法	0	1	2	3	4	5	6	7
2進法	1000	1001	1010	1011	1100	1101	1110	1111
16進法	8	9	A	B	C	D	E	F

例えば，

1001101011（2進法）＝0010　0110　1011　（2進法）

＝　2　　6　　B　（16進法）

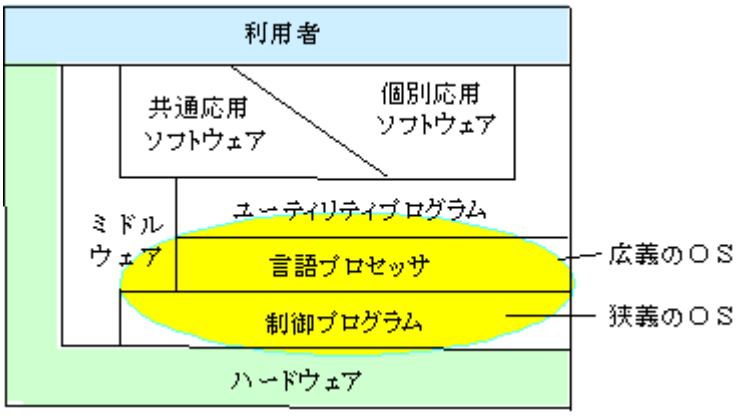
＝　26B（16進法）

となります。

現在のコンピュータは8ビットを基本単位として扱っています。ですから，この8ビットを2桁の16進数で表すのは便利で，16進法表記は良く使われます。

ソフトウェアの体系

ソフトウェアを区分するのには多様な観点がありますが，ハードウェアに密着しているか，それとも人間の利用分野に近いかにより区分すると次のようになります。また，これらの体系も厳密なものではなく，人により解釈も異なりますし，どちらにも属するようなソフトウェアもあります。



ソフトウェア
├ システムソフトウェア
│　├ 基本ソフトウェア
│　│　│（オペレーティングシステム：ＯＳ）
│　│　├ 制御プログラム
│　│　└ 言語プロセッサ
│　├ ミドルウェア
│　│　（ユーザインタフェース，
│　│　　通信管理ソフトウェア，
│　│　　データベース管理ソフトウェアなど）
│　└ ユーティリティプログラム
│
└ 応用ソフトウェア（アプリケーションプログラム）
　├ 共通応用ソフトウェア
　│　（特定業務パッケージ，プログラムコンポーネントなど）
　└ 個別応用ソフトウェア

OS（オペレーティングシステム）

システム全体を管理しているソフトウェアを**OS**といいます。日本語では**基本ソフト**と訳しています。パソコンの起動から終了まで常駐して、以下のソフトウェアを起動します。パソコンの代表的なOSには、Windows，Linux，MacOSなどのことです。複数のOSをインストールしておき、起動時に選択することができます。

ミドルウェア

基本ソフトウェアと応用ソフトウェアとの中間的な存在だということで，このような名称がつけられました。それで，通信管理ソフトウェアをネットワークOSともいうように，この区分はあいまいです。

ユーティリティプログラム

エディタやディスク圧縮ソフトなど，多くの人が共通して用いる基本的な処理をするプログラムです。

言語プロセッサ

COBOLやCなどのプログラミング言語で記述されたプログラム（ソースプログラム）をコンピュータで処理できる機械語のプログラム（ロードモジュール）に翻訳するプログラムです。

応用ソフトウェア

利用者の利用目的のために作成されたソフトウェアで**アプリケーション**システムともいいます。販売システムや会計システムなどのように個別の業務のために構築されたソフトウェアで，一般に利用する企業ごとに構築するのが通常ですが，Office ツールのように共通利用を目的としたものもあります。

OSの機能

OSは次のような多様な機能をもっています。それぞれの中核部分あるいはその集合を**カーネル**といいます。製品としてのOSは、各カーネルを統合して使いやすくする機能（**シェル**）を加えたものです。

- 構成管理
  CPU、メモリ、磁気ディスク、ディスプレイ、プリンタなどコンピュータの内蔵機器や周辺機器の管理。
- タスク管理
  行うべきタスクに必要なプログラムやデータを呼び出して実行し終了時に解放する機能。
- メモリ管理
  タスク実行に必要なメモリを確保すること、OSや他のタスクのメモリ領域の保護を行う機能。
- 入出力管理
  周辺装置の入出力の管理。通信管理もこれに含まれる。
- ファイル管理
  ファイルを物理的にどう配置するか、それを論理的にどう管理するか、デフラグなどを行う。

OSの目的と機能

ジョブ，プロセス（タスク），スレッド

ジョブとタスク

コンピュータに処理を行わせる単位で、利用者の立場での単位を**ジョブ**といい、コンピュータの内部での単位を**タスク**といいます。

例えば、「売上一覧表の出力処理」はジョブです。その処理を行うには、コンピュータの内部では、「売上ファイルからデータを読み込む」「CPUで売上の計算をする」「プリンタに印刷する」などの処理を繰り返して行っています。それぞれの処理単位がタスクです。

タスクは、ジョブ管理の**イニシエータ**により生成され、タスク管理に引き渡されます。また、タスクの実行が終了して消滅すると、ジョブ管理の**ターミネータ**に引き渡されます。

タスク、プロセス、スレッド

これらの関係はあいまいです。厳密性にこだわらなければ、タスク≒プロセスであり、特に並行処理を重視してプロセスを細分化したものがスレッドであると理解しておけばよいでしょう。

タスクをさらに細分化して、細分化のレベルにより、スレッド ⊆ プロセス ⊆ タスク とする場合もありますし、汎用コンピュータではタスクといっていたのを、オープン系ではプロセスといっていることもあります。また、スレッドもタスクと似ていますが、共有メモリを利用しながら複数の処理を行なうように、プロセスを細分化したものがスレッドといいます。

マルチタスク

音楽のダウンロードをしている間に表計算ソフトを使うというように，A、Bのタスクが発生したとき、Aが入出力を行っておりCPUを使っていない場合には、BがCPUを使うようにすれば、全体の効率が向上します。このように、複数のタスクを並行して行うことができる機能を**マルチタスク**（マルチプロ

セス、マルチプログラミング）といいます（同時に一つのタスクしか実行できない方式を**シングルタスク**といいます）。現在のパソコンOSはマルチタスク機能を持っています。

スプーリング

特にプリンタへの出力は時間がかかります。それで、プリンタへの出力を直接行わずに，出力データをディスクに書き出しておき，他の処理と並行して専用の出力プログラムでプリントします。

キーバッファ（キーボードバッファリング）

CPUが負荷の高い処理を行うと、キーボードから入力できなくなります。それを防ぐために、CPU処理とキーボード入力をマルチタスクにします。キーボードからの入力データを主記憶のキュー（キーバッファ）に一旦保存しておき、CPUからの要求により逐次取り出す方法です。

マルチユーザ

Webサーバのように，同時に複数の人（ユーザ）からの仕事を処理できる機能をマルチユーザといいます。ログオフせずに他人が利用できる機能でもあります。現在のパソコンOSは、マルチユーザ機能を持っています。

RAS, MTBF, MTTR

Reliability（信頼性）

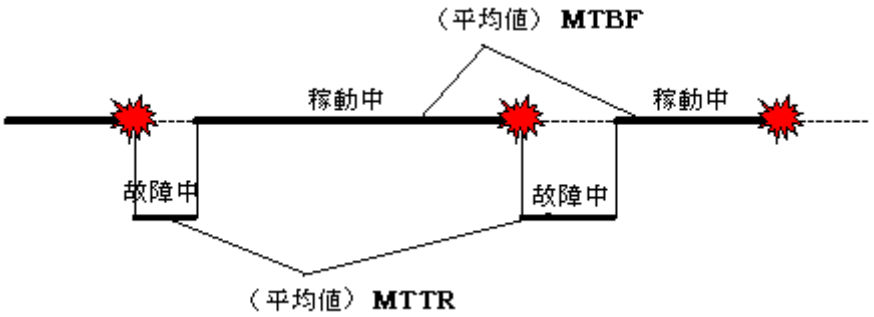
ハードウェアやソフトウェアが故障しないようにすること。故障が回復してから次の故障が起こるまでの平均時間を**MTBF**（mean time between failures）というが，それを大きくすること。

Availability（可用性）

システムが利用できる割合のこと。稼働率ともいいます。上のMTBFと下のMTTRを使うと**MTBF／(MTBF＋MTTR)** で表せます。

Serviceability（保守性）

故障してから回復するまでの平均時間である**MTTR**（mean time to repair）を小さくすること。それには原因を短時間に見つけることや故障した機器を切り離したり正常な機器と取り替える時間を短縮することが必要になります。



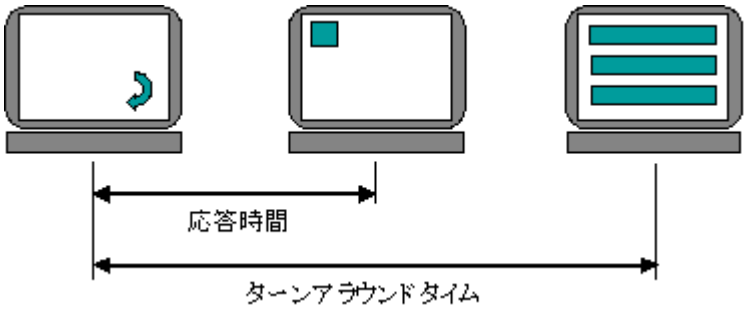
すなわち，コンピュータの稼働率をあげるためには，MTBFを長く，MTTRを短くすることが必要です。これがOSの目的の一つです。

システムの評価尺度

応答時間（レスポンスタイム）
　コンピュータに指示を出してから，最初の出力が出るまでの時間

ターンアラウンドタイム
　コンピュータに指示を出してから，出力が完了するまでの時間

スループット
　単位時間あたりに処理できるデータ件数

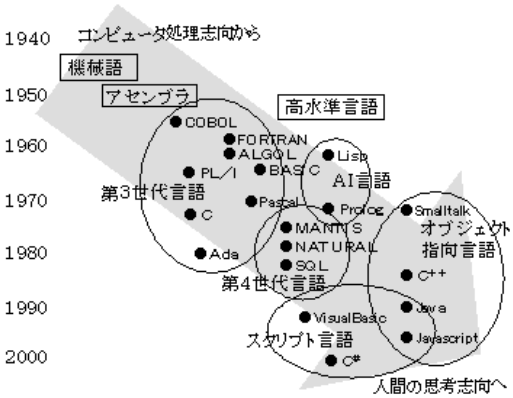


　応答時間とターンアラウンドタイムの単位は[時間]ですから，これらの値は小さいことが望ましく，スループットは[件/時間]ですから大きいほうが性能がよいことになります。
　これらの値は，CPUだけでなく，入出力装置の速度，マルチタスク／マルチユーザ機能の性能にも関係する総合的な評価尺度になります。それで，コンピュータシステムを比較検討するときには，使用目的に合うプログラムを実行させて，これらの性能尺度を比較検討することが必要です。それを**ベンチマーキング**といいます。

プログラミング言語

プログラミング言語の発展

　プログラミング言語の発展は，コンピュータの動作に依存した言語から人間の発想に近い言語への発展であるといえます。



プログラミング言語の出現時期

第１世代言語	機械語
第２世代言語	アセンブラ
第３世代言語	コンパイラ・手続き型言語(コボル、フォートランなど)
第４世代言語	非手続き型言語(SQL など)
最近の動向	オブジェクト指向言語（Ｃ＋＋，Ｊａｖａなど）
	スクリプト言語（JavaScript など）

SQL

以前は多様な第4世代言語がありましたが，SQLがRDB（リレーショナル・データベース）の国際標準言語になってからは，SQLの独壇場です。DB/2，Oracle，Accessなどほとんどの RDB言語はこれを用いています。

スクリプト言語

オブジェクト指向言語のような部品やGUI環境が整備されると，エンドユーザでも簡単にプログラムを作成することができます。それを**スクリプト言語**といいます。それにはこのようなビジュアルな環境で構築できるVisualBasicや，Webページの言語であるHTMLに組み込んで記述できるJavascriptなどがあります。

Java

○Javaの特徴

・コンパイラ言語，手続き型言語

・オブジェクト指向言語（特にインターネット環境に適す）

・移植性が高い（機種やOSに関係なく互換性あり）

ほとんどのOSがJavaをサポート

バイトコード（中間言語）はOSに無関係

・開発環境が整備されている

○Java関連

・Java アプレット：Java の小さなプログラム。ブラウザで実行

・Java サーブレット：Java の小さなプログラム。サーバで実行

・J2EE：Javaのシステム開発環境

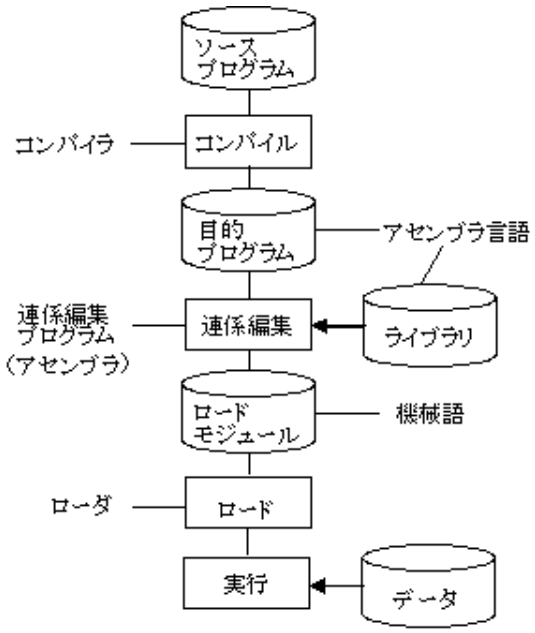
・JDBC：Java プログラムから SQL命令を発行して RDBにアクセスするための API

・JavaScript：HTML内に記述できる。Java とは異なる言語

言語プロセッサによる翻訳

翻訳の手順

プログラミング言語の文法に従って記述されたソースプログラムは，コンパイルされて目的プログラムになり，連係編集されてロードモジュールになります。



コンパイル

プログラミング言語の文法に従って記述されたプログラムを**ソースプログラム**あるいは**原始プログラム**といいます通常の高水準言語では，ソースプログラムをいったんアセンブラ言語に翻訳する必要があります。それを**コンパ**

イルといい，コンパイルするソフトウェアを**コンパイラ**といいます。それで，高水準言語のことをコンパイラということもあります。

コンパイルされた結果を**目的プログラム**あるいは**オブジェクトモジュール**といいます。これは通常はアセンブラ言語になっています。ですから，アセンブラ言語で記述されたソースプログラムは，それ自体が目的プログラムになります。

連係編集（リンカー）

高水準言語では，全体の処理を一つのプログラムとして記述するのではなく，ロジックをいくつかの**サブプログラム**に分解し，それらをつなぎ合わせてプログラムにする方法が取られます。それらのサブプログラムは部品として他のプログラムにも利用されることがあります。また，言語プロセッサは多様な共通利用サブプログラムを多く持っています。それらは**ライブラリ**として保管されています。

連係編集（リンカー）とは，作成した目的プログラムとライブラリにある既存の目的プログラム群を組み合わせて，一つの機械語プログラムに翻訳します。この機械語プログラムのことを**実行可能プログラム**あるいは**ロードモジュール**といいます。すなわち，機械語で作成したソースプログラムは，それ自体がロードモジュールになっています。

この翻訳プログラムを**連係編集プログラム**あるいは**リンケージ・エディタ**といいます。言語プロセッサとしてのアセンブラはこの機能を持っています。なお，目的プログラムでは高水準言語の特性は失われますので，COBOLで作成した目的プログラムとFORTRANで作成したライブラリを連係することもできます。

静的リンキングと動的リンキング

プログラムの実行前に，リンケージ・エディタを用いてリンクするのを静的リンキング（static linking）といいます。それに対して動的リンキング（dynamic linking）とは、プログラムの実行時に必要に応じて，主記憶上に変数領域やデータ領域をロードする機能です。これにより、実行前にすべてのモジュールを主記憶にロードする必要がなく，主記憶の無駄な使用が削減されます。

Javaでの用語

Javaコンパイラでは、コンパイルした目的プログラムに相当するものを**バイトコード**といいます。バイトコードは機種やOSに限定しない言語になっており、移植性を高めています。

バイトコードは、それをインタプリタとして実行するには、Java VM（仮想マシン）というソフトウェアを用います。このソフトウェアは通常ブラウザに標準機能として搭載されています。また、コンパイルするには**JITコンパイラ**が用意されており、機種やOSに固有の機械語プログラムに翻訳されます。

実行による言語プロセッサの区分

翻訳の方法にはコンパイラ以外にインタプリタやジェネレータなどもあります。

コンパイラ

多くの高水準言語はコンパイラ方式です。この特徴は，ソースプログラムの全体を一つのかたまりとして翻訳しロードモジュールとして実行します。

- プログラム全体を作成しないと実行させてチェックすることができないし，プログラムの一部を修正したときも，全体を翻訳する必要があります。そのために，1回しか使わないようなプログラムを作成するのには非効率です。

- 逆に，1回ロードモジュールを作成しておけば，実行時に毎回コンパイルする必要がありませんので，何度も利用するプログラムに適しています。

インタプリタ

インタプリタは，ソースプログラムを1行づつ翻訳して実行する方式です。代表的なインタプリタ言語にBASICがあります（コンパイラのBASICもあります）。

プログラムを一部作成して実行し逐次追加修正するのが容易ですから，プログラムを作成する段階では便利な方式です。

しかし，実行のたびに翻訳をする必要があるので，何度も利用するプログラムでは非効率です。また，大量のデータを用いるプログラムでは，一つのデータを処理するたびに翻訳することが起こりますので，処理が非効率になります。

ジェネレータ

プログラムの一部をパラメタとして与えることにより，完成したプログラムを出力する方式です。これにはソースプログラムを出力してコンパイルするものと，既に実行可能プログラムがあり最初にパラメタを読み込み，それを用いてデータを読み込み帳票作成などをするようなものがあります。