オペレーティングシステム
（2024年 第１回）

授業の概要と進め方について

授業のねらいと達成目標

- 情報システムの実行環境や開発環境について基礎的な知識を得るとともに、オペレーティングシステム（OS）を構成する要素技術やそれらの考え方の理解を深める
  - OSの役割と基本構成を理解する
  - OSの基本機能（プロセス管理、メモリ管理、ファイル管理）を理解する
  - ユーザとOSの係りに関する機能を理解する

OSを学ぶ狙いと各回の実施項目について、ここでもう一度みておいてください。

「情報システム」としていますが、プログラムのことです。
プログラムを動作させるための基盤やプログラムを開発する道具として、OSの役割や基本的な構成、ユーザとOSとのかかわりについて理解するため、
・プロセス管理
・メモリ管理
・ファイル管理
の三つについてを中心に学んで行きます。

授業の計画（予定）

[第1回]　講義のイントロダクション、OSの概要
　　本講義の内容や進め方などの説明とともに、OSの役割や位置づけの概要を学ぶ

[第2回]　OSの基本構造と機能
　　コンピュータの利用やプログラムの開発・実行とOSとの関連について学ぶ

[第3回]　OSとハードウェアアーキテクチャ
　　OSの動作を支援するために必要となるハードウェアアーキテクチャについて学ぶ

[第4回]　プロセス管理（1）
　　プログラム実行の実体であるプロセスの基本概念とその機構について学ぶ

[第5回]　メモリ管理（1）
　　主記憶を効率的に利用するためのメモリ管理の課題、アドレス空間について学ぶ

[第6回]　メモリ管理（2）－ 仮想記憶
　　前回で述べた課題を解決するページングによる仮想記憶の概念と機構について学ぶ

[第7回]　I/Oシステム
　　入出力装置とOSでの管理について概要を学ぶ

シラバスにある講義内容を挙げています。

今回と次回でOSの概要、およびOSの基本構造と機能について紹介し、第3回でOSが動作するためのハードウェアからの支援機構について、おさらいする形で確認します。

それらを基礎として、
第4回でプロセスの管理について、その前半の部分を、
第5回と6回でメモリの管理について、
第7回で入出力について、
を学ぶことにします。

授業の計画（つづき）

[第8回]　ファイルシステム
　　データの保存や交換のために二次記憶装置を管理し、効率的に利用する方式について学ぶ

[第9回]　プロセス管理（2）－ 多重プロセス
　　複数のプロセスが並行して実行される環境での制御方式、プロセス間の通信について学ぶ

[第10回]　ネットワークと分散処理
　　コンピュータネットワークや複数のコンピュータが協調する技法について学ぶ

[第11回]　アクセス制御とデータ保護、セキュリティ
　　コンピュータの資源を脅威から保護するための対策についての概要を学ぶ

[第12回]　OSのユーザインタフェース
　　ユーザからみたOSについて、コンピュータシステムを操作するための機能について学ぶ

[第13回]　OSの構成法と仮想計算機
　　OS構成のアプローチや仮想計算機について概要を学ぶ

[第14回]　a：まとめ
　　b：期間前試験

第8回でファイルシステムについて、
第9回でプロセス管理の後半、
を学びます。

以降、ネットワーク、アクセス制御、コマンド、OSの構成法について、
と進めます。

第14回でまとめと期間前試験を行います。

評価について

- 期間前試験（60%）、授業での課題提出と参加度（40%）で評価する
  - 授業各回での課題はそれぞれ、クラスウェブのレポートで提出してください

今回の内容

- 授業の進め方について
  - スライド資料で説明する
    講義で使用するスライドは通常、前日の午前中までに電子資料として置いておく
    - ✓ 次回の内容に関して、講義資料、教科書や参考書に目を通しておくこと
    - ✓ 講義資料を振り返り、教科書や参考書などの該当箇所を読むこと
    - ✓ 小課題を出すことがあるので、解答をクラスウェブで提出すること
  - 質問はメール ktoyama_cs@meiji.ac.jp でも受けます
  - 使い方よりは考え方を示すようにする
  - 今回は用語を挙げることが中心
- オペレーティングシステム（Operating System：OS）とは・・・
  - 一定の機能を提供するもの
  - 特別なプログラム

今回の講義では、主として用語をあげることにしています。
そして、OSとはどのようなものかについて概要を示します。

教科書

「IT Text オペレーティングシステム（改訂2版）」，
野口 健一郎、光来 健一、品川 高廣 共著，オーム社，2018，
ISBN-13: 978-4-274-22156-9

参考書

[1] 吉澤 康文 著「オペレーティングシステムの基礎 −ネットワークと融合する現代OS− 」, オーム社, 2015

[2] 電子情報通信学会 知識ベース「知識の森」7群 コンピュータ - ソフトウェア 3編 オペレーティングシステム
https://www.ieice-hbkb.org/portal/doc_588.html (サイトマップ)

[3] 清水 謙多郎 著「オペレーティングシステム」, 情報処理入門コース 2, 岩波書店, 1992

A. S. Tanenbaum 著, 水野 忠則 他 訳「モダンオペレーティングシステム 原書 第2版」, ピアソンエデュケーション, 2004

A. S. Tanenbaum他 著, 吉澤 康文 他 訳「オペレーティングシステム 設計と実装 第3版」, ピアソンエデュケーション, 2007

Abraham Silberschatz他 著, 土居範久 監訳:「オペレーティングシステムの概念」, 共立出版, 2010 (第7版の邦訳)

和書の参考書として、[1]と[3]を挙げます。

[2]は電子情報通信学会のウェブサイトで公開されているもので、[1]の吉澤先生が執筆されているものです。

なお、大部になりますが、翻訳された洋書の参考書も挙げておきます。

参考書[2]について

これは、参考書[2]の目次ページになります。
ここから辿っていくことで、それぞれの内容を見ることができます。

オペレーティングシステムとは

- システムソフトウェアの一種、「基本ソフトウェア」
- プログラム（アプリケーション）を実行する基盤
- 用途
  - プログラムを実行する
  - プログラムを開発するための環境を提供する
  - サーバを構築する
  - …
- 目的
  - ハードウェアの抽象化を行う
    - 抽象化したインタフェース（Application Programming Interface：API）をアプリケーションに提供するソフトウェア
  - 資源（リソース）の効率的管理を行う
  - 保護とセキュリティを行う

システムソフトウェア … コンピュータなどの基本的な制御や管理、使用のサポートを行うためのソフトウェア

まず、OSとは何かについてですが、コンピュータのソフトウェア（プログラム）であって、他のプログラムを動かすために、コンピュータの基本的な制御や管理 ― これは、メモリを確保したり、ある地点からCPUを動作させたりするといったことです ― が、それを行うもののことです。

また、プログラムを作成して動作するようになるまで、テキストエディタやコンパイラ、リンカ、デバッガなどといったいろいろの道具を用いますが、それらがうまく使えるような環境を提供したりします。
このような用途であるため、OSは「基本」ソフトウェアと呼ばれたりします。
OSによって動かすプログラムは、応用プログラムとかアプリケーションと呼ばれます。

今述べた、他のプログラムを動かすとか開発するために・・・ということがOSの用途であって、そのために目指す基本事項が目的になります。
その目的は、ハードウェアの抽象化、資源の効率的な管理、そして保護、です。

オペレーティングシステムの役割

- 使いやすさの提供
  - 操作しやすい（入力、出力）
  - プログラムの格納場所やロード方法は意識不要
  - 共通の機能を提供
    - 機器が多少違っても同じように操作でき、プログラムは共通に利用可能
  - ハードウェアを直接駆動する必要がない
  - 複数のプログラムを安全・同時に実行可能
- 効率性の提供
  - 高価な資源を効率よく動作させる
- 資源の共用
- 信頼性の提供
  - ハードウェアやソフトウェアの障害が起きてもシステムダウンさせない
  - データが破損、漏洩しないようにする

前ページの抽象化、効率的管理、保護といった目的を実現することで、OSがユーザに提供するものが、
　使いやすさ、効率性、共用、信頼性
であり、それらを提供することがOSの役割だといえます。

ここで、使いやすさとは、個々のハードウェアが違っていても使う側ではそれを意識しなくてよく、共通の機能を指示するだけで動作させられるというようなことです。
これはたとえば、入力と出力がどのようなものであるかが明確に決まっていて、処理の中身がどのようになっているかは意識する必要がないということなどで実現されるものです。

オペレーティングシステムの例

Windows系	Windows10, Windows11 など
macOS	Monterey, Ventura など BSD UNIXベース
UNIX系	Solaris, Linux, FreeBSD など UNIXのAPI に準拠するOS全般
Android	Linuxベース
iOS, iPadOS	macOSベース
メインフレームのOS	OS/360, MVSなど
組込みシステムのOS	iTRON, VxWorks など

OSにはいろいろなものがあります。ここでは名前だけ挙げます。

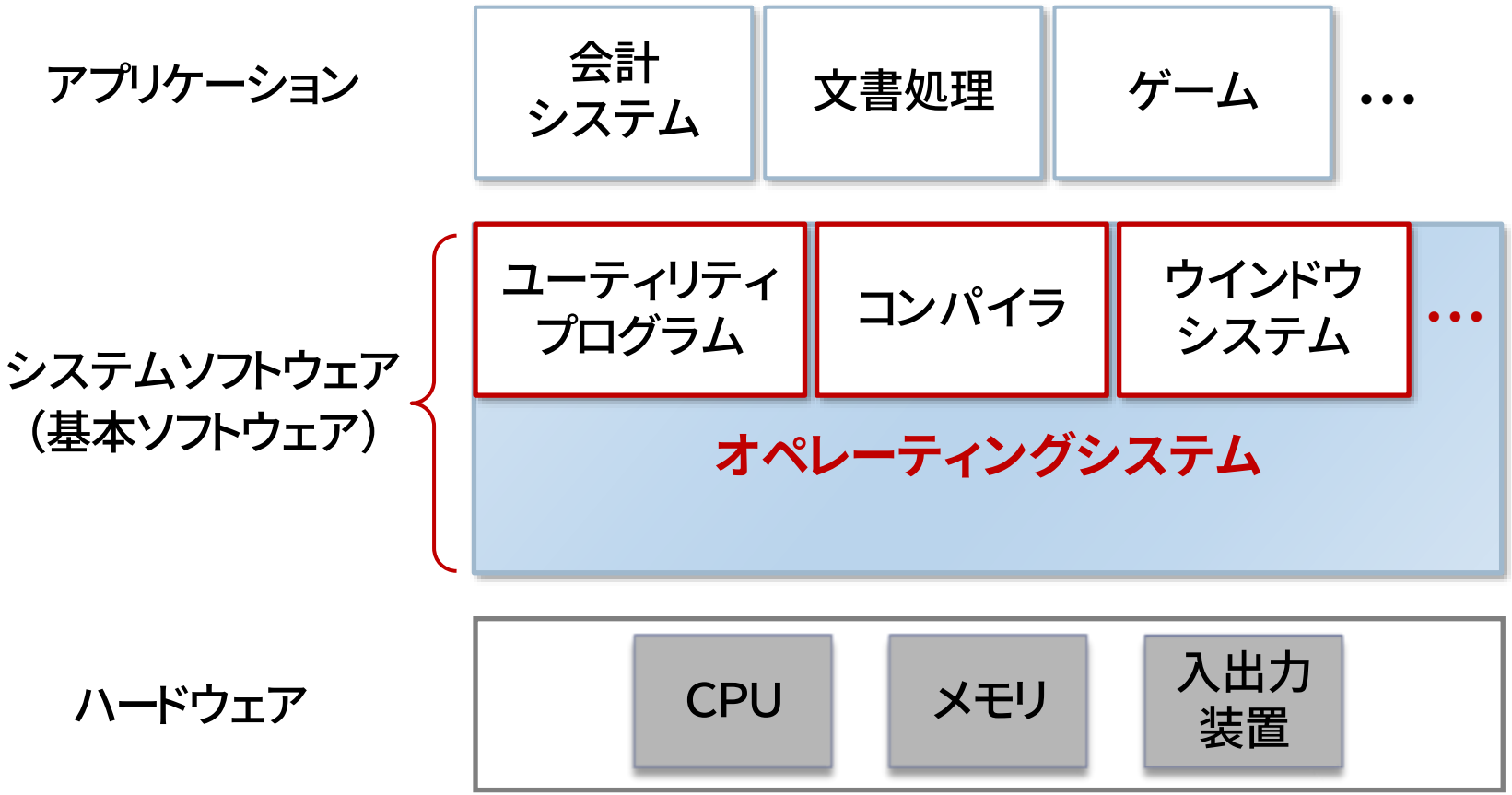
PCのOSとしては、WindowsやmacOSが多く利用されているものです。皆さんが最もよく使っているものは、おそらくスマートフォンのOSであるAndroidやiOSではないかと思いますが、OSを使っていることを意識することはあまりないかもしれません。

組込みシステムのOSでは、それが使用されていることを意識することはほとんどないかもしれませんが、ゲーム機のコントローラにiTRON系のOSが使用されていたりします。

なお、「組込みシステム」とは、特定の機能を実現するために機械や装置などに内蔵されるコンピュータシステムのことで、テレビや複写機、医療機器、エンジン、・・・などを制御するものです。

コンピュータの構成とオペレーティングシステム

- コンピュータシステムの階層構成

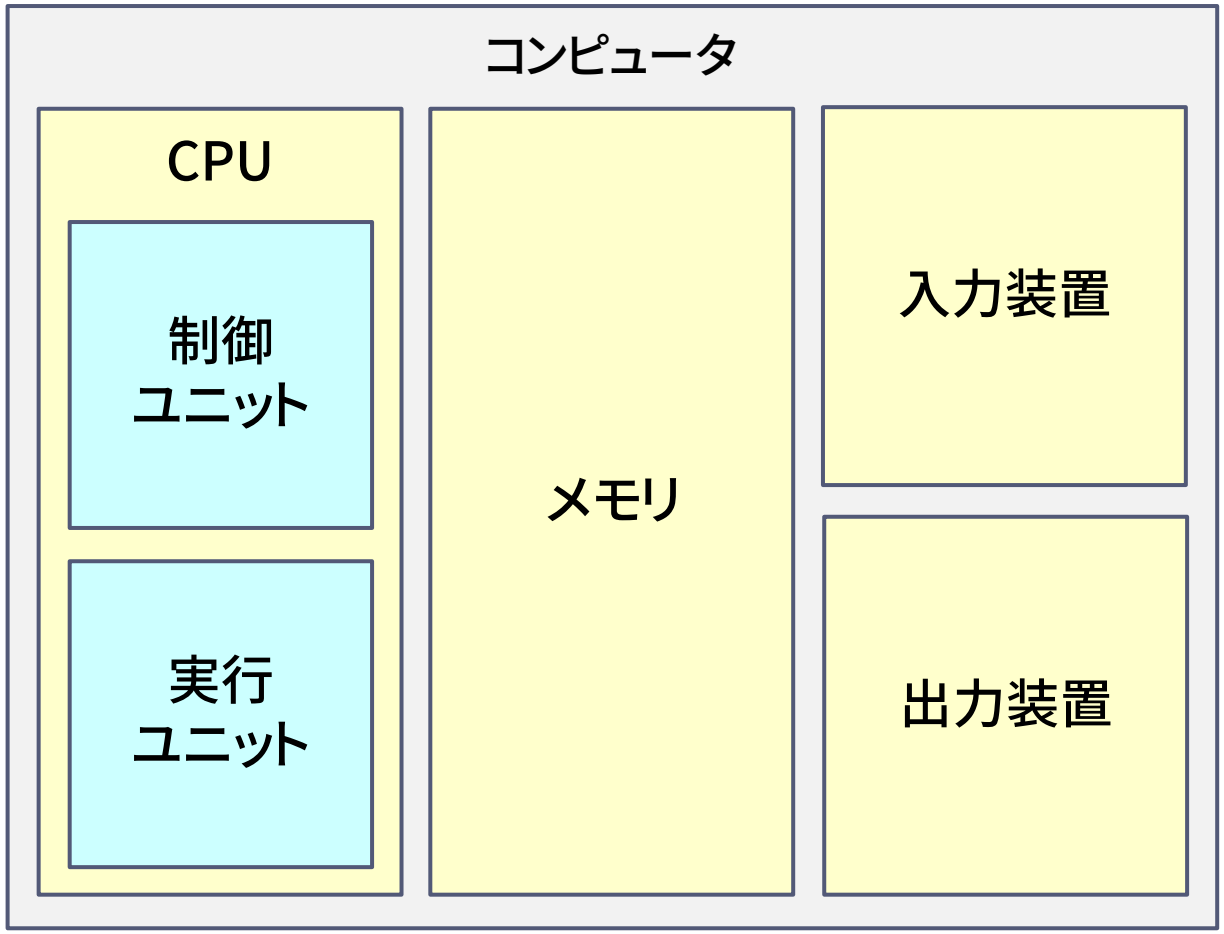


非常に大まかな形でコンピュータとOS、アプリケーションの関係を示します。
アプリケーションはある特定の機能や目的のために開発・使用されるソフトウェアで、ここにあるゲームやたとえばWordといった文書処理などで、それが、コンピュータ ― ここではハードウェアとしています ― が、それで実行されます。

OSはこの両者の間にあって、アプリケーション（プログラム）に対してサービスを提供して、実行させるためのものです。

「オペレーティングシステム」の箱内の上の方の部分にユーティリティプログラム、コンパイラといったものが書いてありますが、それらについては次回以降で説明します。
ハードウェアは、CPUとメモリ、入出力装置で構成されます。

コンピュータの構成要素

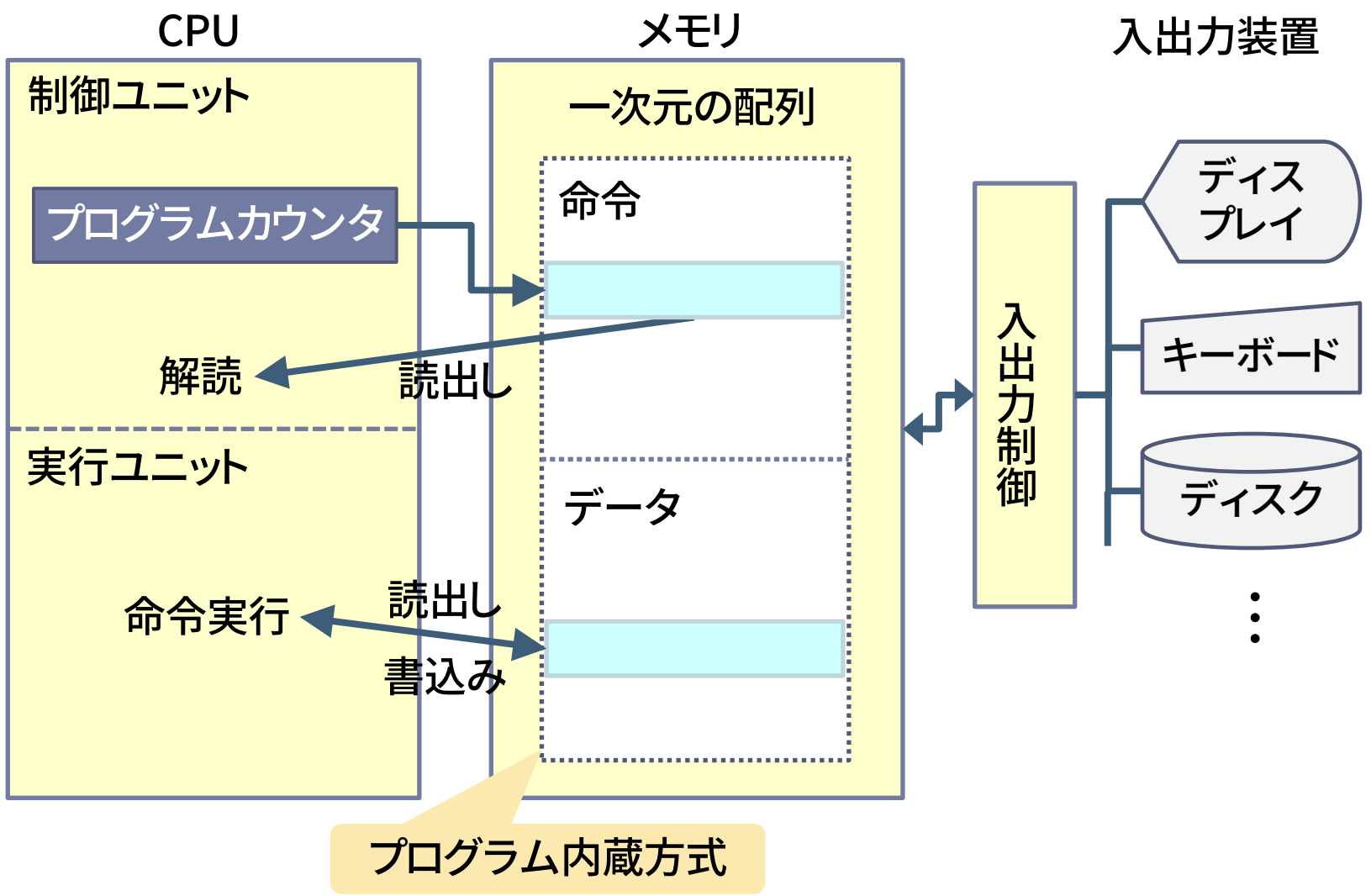


以下、コンピュータの構成とプログラムの実行について簡単におさらいして行きます。

前ページで、ハードウェアは、CPU、メモリ、入出力装置から構成されているとしました。

CPUの中には制御ユニットと実行ユニットがあります。

ハードウェア：コンピュータ構成の概要



プログラムが動作する仕組みはこのようなものです。メモリには、足し算や移動などといったCPUに対する命令とその命令操作の対象となるデータが一次元の配列のように並べて配置されています。

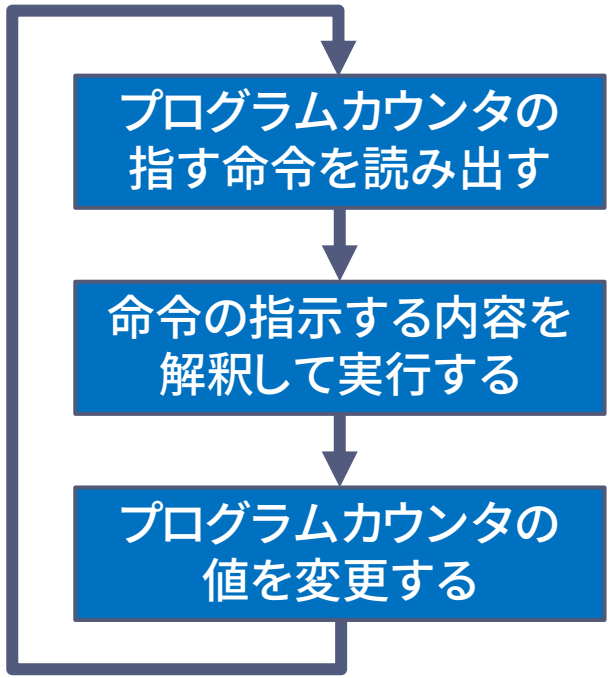
制御ユニットは命令を次々に読み出して、それが足し算であるなど、何であるか解読します。メモリのどの位置から命令をもってくるかを管理するために「プログラムカウンタ」とよばれる機構があり、メモリの配列での位置（番地）を指しています。

実行ユニットでは足し算などの命令を実際に実行します。その際、メモリからデータを読み、結果をメモリに書くなどを行います。

なお、命令（プログラム）もメモリに格納される方式をプログラム内蔵方式と呼びます。

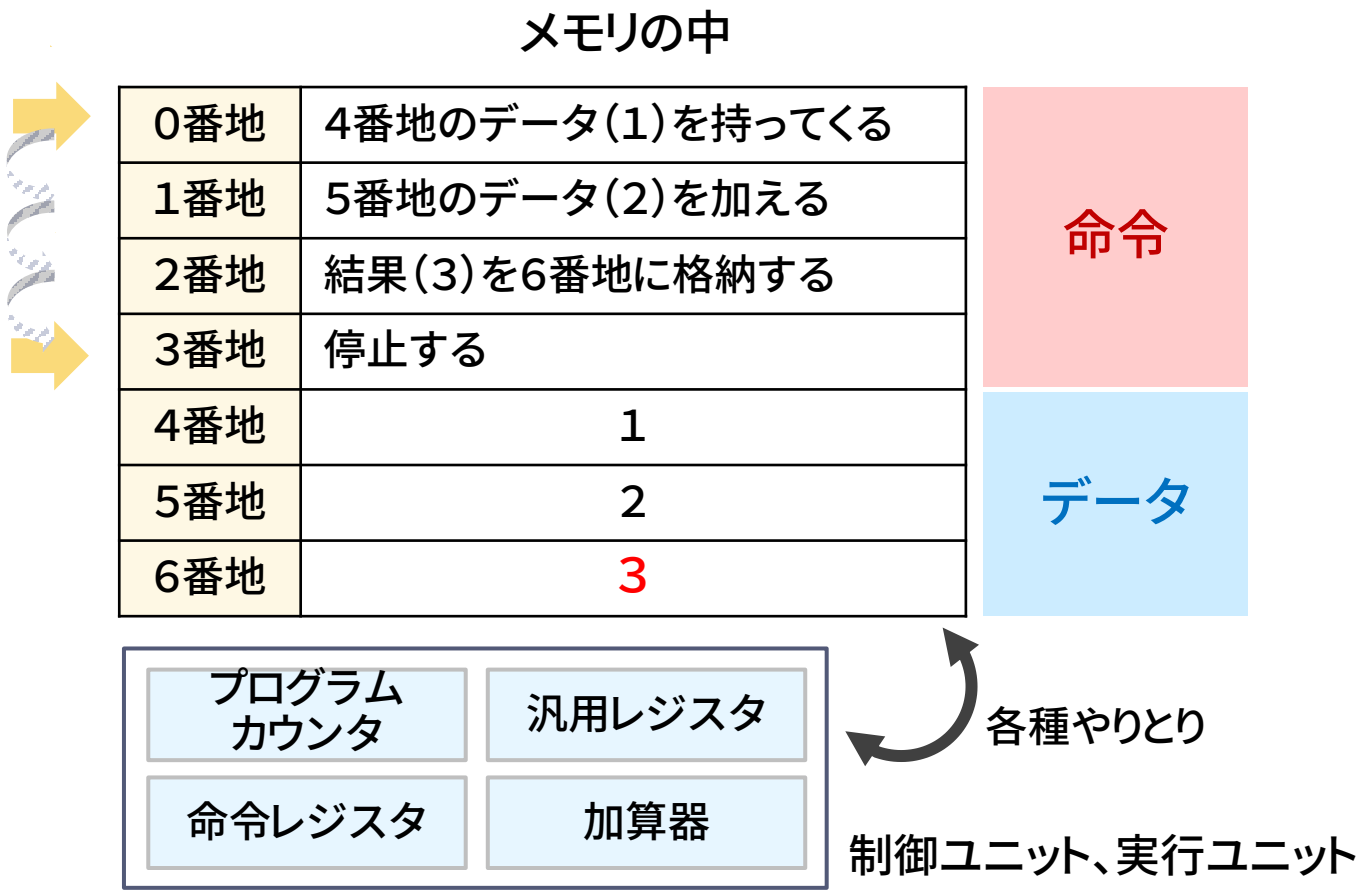
プログラムの実行について

- メモリ内にある「命令」が順に実行される
  - プログラムカウンタの指す命令が表す操作内容に従って、
    操作（演算）：加算・乗算・転送・分岐 ・・・ を、
    データ：メモリ内にあるものや専用の記憶回路（レジスタ）にある
    に対して実施し、
    結果をメモリやレジスタに入れる
  - 次の命令を実行する



プログラムが実行される過程は、プログラムカウンタが指す場所にある命令を読み出し、それが示す内容に従って演算操作を行い、次の命令に行く、すなわちプログラムカウンタが次の命令を指すように更新して、また同じように繰り返す、・・・となります。

制御・実行ユニットとメモリのイメージ



簡単な例を示しますと、
最初、プログラムカウンタが0で開始され、0番地の命令「4番地のデータを持ってくる」が読み出されて実行され、4番地に入っている値「1」がレジスタに入れられます。
プログラムカウンタは+1されて1番地を指し、「5番地のデータ「2」を加える」が読み出されて実行され、5番地に入っている値「2」が先のレジスタに加えられて演算結果が「3」となります。

プログラムカウンタは+1されて2番地を指し、「結果をもつレジスタの値を6番地に格納する」が読みだされて実行され、値「3」が6番地に書き込まれます。

プログラムカウンタは+1されて3番地を指し、「停止」命令が読みだされて実行され、CPUは停止します。

参考：コンピュータアーキテクチャ

コンピュータシステムのハードウェア全体の、ユーザ（プログラマ：OSを設計するプログラマも含む）から見たインタフェースの定義

- 命令セットアーキテクチャ（ISA：Instruction Set Architecture）
  機械語/アセンブリ言語プログラマ（OSに対するものも含む）から見たプロセッサの抽象化されたイメージ
  - 命令セット、アドレッシングモード、レジスタ、アドレスとデータの形式など（プロセッサをソフトウェア側から見たときのインタフェース定義）
- マイクロアーキテクチャ
  さらに下位でのより具体的なシステムに関する記述
- アーキテクチャの例
  - IA-32 … 80386以降、80486などの32ビットのもの。AMDもある（x86とも呼ばれる）
  - x64 … IA-32の64ビット拡張（Core、Ryzenなど）
  - RISC（Reduced Instruction Set Computer）… ARM、PowerPCなど

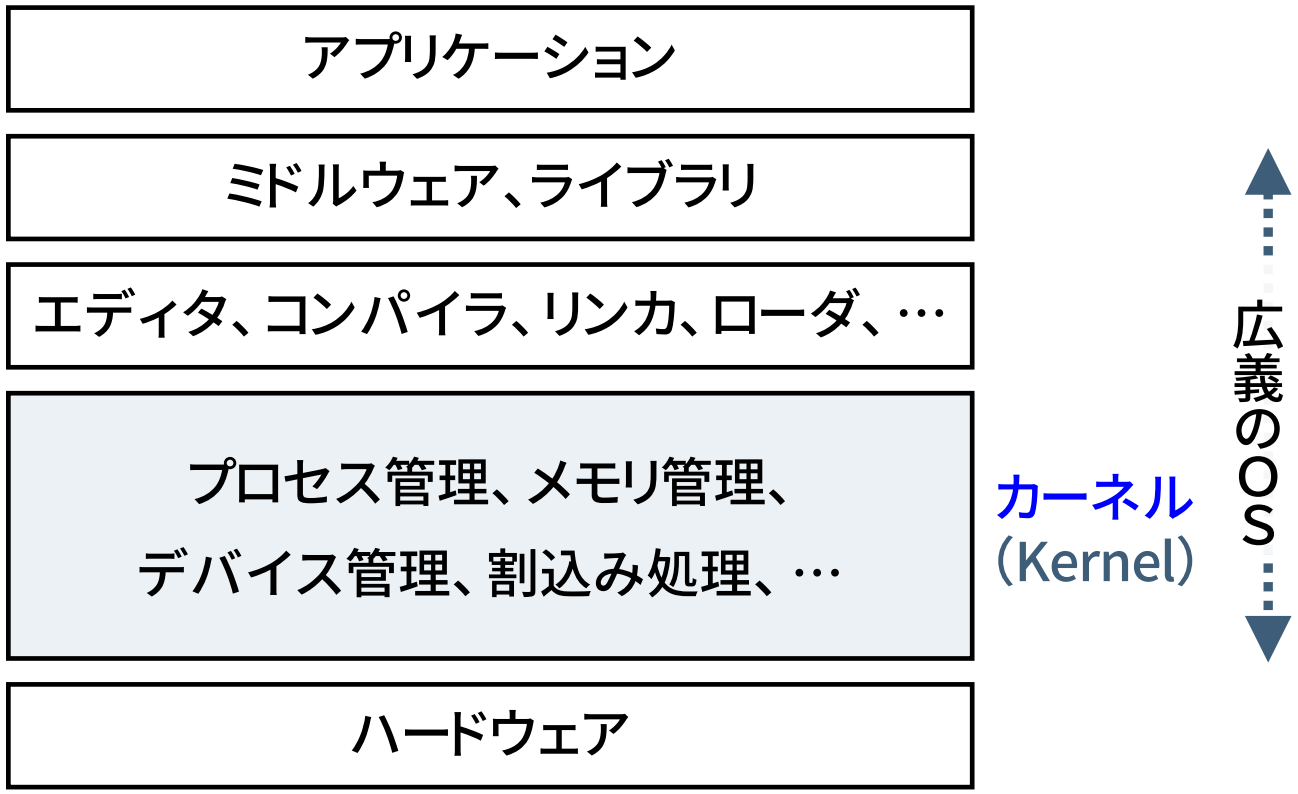
参考までに、コンピュータの構成や命令がどのようになっているかを定義するものがコンピュータアーキテクチャです。
IA-32、x86やx64とか言った名前を聞いた方も多いと思います。

講義は特定のアーキテクチャに依存したものではありませんが、IA-32を例とした説明を行うことがあります。

カーネル

- 基本的サービスを行うOSの根幹部分

プログラムの起動の準備や終了後の後始末、必要とするメモリ領域の割り当て、CPUで実行するプロ グラムの切替え、などを行う



「OSとは、」に戻りまして、まず、カーネルについてです。

先ほど、OSはアプリケーションとハードウェアの間にあるものとしました。

カーネルはOSの根幹部分で、これから学んで行くプロセスやメモリを管理するなど、基本的なサービスを行う部分です。

カーネルに加えて、それよりもアプリケーション寄りにあってツールや共通的に使用されるプログラムを含めてOSと考えることもあり、それも含めたものは、広義のOSとされます。p.13の図で、「オペレーティングシステム」の箱の上の方の部分にユーティリティプログラム、コンパイラといったものが書かれていましたが、それらを含めたものです。

この授業では主にカーネルについて学んでいきます。

ハードウェアの抽象化、仮想化

- ハードウェアの抽象化とは
  - 異なる構成のハードウェアにおいても、同じ操作、使い勝手を提供する
  - ハードウェアを直接制御するのは難しいので、簡単に使えるようにOSが、関数群（抽象概念に対するアクセス手段：API、システムコール）を提供
- 資源の仮想化
  - 複数（多数）のユーザやプログラムで資源を共有しているが、それを意識しなくてよい
    - CPUはどのプログラムも実行し、メモリも十分に使用可能に見える

実際の装置	仮想化された装置
CPU	プロセス（タスク、スレッド）
メモリ	仮想記憶
ディスク	ファイル、ディレクトリ（フォルダ）

それでは、p.10で挙げたOSの目的について、少し詳しく説明します。
まず、OSを考える上で重要な考え方として抽象化と仮想化があります。

抽象化とは、個々の具体的な機能などから共通した概念をとりだすことで、ハードウェアの違いを隠して同じ操作で扱えるようにするために行われます。
抽象化された機能がAPIやシステムコールと呼ばれる関数を呼び出すことで簡単に使えることになります。

仮想化は、資源の物理的な制約を意識することなく利用できるようにすることであり、例えば、CPUの実体が1個であっても多数のプログラムが同時に実行されたり、メモリも実際に設置されているサイズの制約を気にせず、十分に大きなものがいくつも使えたりするということです。

CPUに対しては、プロセスなどと呼ぶ論理的なものを考え、ディスク装置に対しては、同じくファイルを対応付けたりすることになります。
これらが実際にどのようなものであるかは、今後説明して行くことになります。

資源の管理、保護

- 資源の管理
  - 高価な資源をできるだけ有効利用する
    CPUの遊休時間を減らす、使用しないメモリを割り当てない　など
    - CPUと入出力装置との並列実行
  - マルチプログラミング、TSS（Time Sharing System）
    - 時分割で複数の異なるプログラムを見かけ上同時に実行
  - 効率的かどうかは評価尺度（利用形態など）に依存する
- 保護とセキュリティ
  - アプリケーションのバグなどから他のプログラムを保護
    - アプリケーションからの操作を制限する
    - 不当なアクセスを検知し防御する

TSS：大型コンピュータに複数の端末を接続した環境で用いられたコンピュータの処理形態の一つ
1台のコンピュータが短い間隔で処理を順次切り替えてゆくことで、複数のユーザが1台のコンピュータを共有して利用することを可能にする方式

資源の管理は、高価な資源をできるだけ遊ばせることなく使うようにするということで、たとえば、
CPU（初期のコンピュータシステムでは、CPUやメモリは非常に高価なものでした）と入出力装置を並列に実行する（入出力装置が動作しているときにCPUが別の仕事をして遊ばないようにする）ことです。
複数のプログラムを見掛け上同時に実行するマルチプログラミング、複数のユーザが1台のコンピュータシステムを見掛け上同時に使用するTSSなどがあります。

マルチプログラミングで複数のプログラムが同時に実行されていると、あるプログラムのバグが、別のプログラムを破壊するなどして問題が発生することがあり、メモリが知らないうちに書き換えられてしまうということを防ぐ必要があります。
このため、書き込むという操作を制限する、また、アクセスしてはならない領域に対するアクセスが検知できるようにすることが必要です。

資源管理の着想

- 時分割
  - 資源の割当を時間軸上で分割し、その単位ごとに割当てる
  - TSSが代表例
    CPUが複数あるかのように見せかける
- 空間分割
  - 空間（記憶域スペース）を細かい単位に分割し、その単位ごとに割当てる
  - メモリ（主記憶）、二次記憶装置が代表例
    連続した領域という制約などの解消
- オンデマンド
  - 必要になったときに、必要な量を割当てる
    （プログラムが実際に必要とする資源の量は、実行されるまで分からない）

資源管理において基本的な考え方が、次の三つです。

時分割は、CPUなどの資源をある時間ごとに区切って、その区画ごとに別のプログラムに割り当てるものです。それにより、TSSではあたかもCPUが複数あって、複数のプログラムが同時に動いているように見えます。

また、空間分割はメモリのような空間をある大きさの単位で分割して、その単位を用いて割り当てを行うというもので、この考え方によって領域が連続していなければならないという制約を解消できます。

オンデマンドは、必要になった時に必要な量を割り当てるということです。
空間分割と組合わせて、必要になった時点で分割された単位を割り当てるというような使い方が行われます。

オペレーティングシステムの歴史と発展（1）

- 1940年代：
  - OSは存在しない
    ユーザは機械語によってプログラムを作成し、それを計算機本体のスイッチからメモリにロードして実行
- 1950年代：
  プロセッサの利用効率（単位時間当たりの処理能力）の向上が主目的
  - 複数のプログラムを連続して処理する「バッチ処理」用OSの出現
    - 処理手順や入力データを予め用意して一括処理
    - 多くのユーザで資源を共有でき、資源の空き時間帯に実行
- 1960年代前半：
  人間との対話性の重視
  - マルチプログラミング
    複数のプログラムをメモリ上に置き、切り替えて見かけ上同時に実行
  - TSS（タイムシェアリングシステム）
    複数のユーザで同時にコンピュータを利用（対話的な処理）

OSの歴史と発展については、教科書では1.5、1.6にあります。また、参考書[2]に比較的詳細な記述があります。OSについてひととおり見たまとめのところで確認します。

手短に言うと、1940年代のOSがない時代から、資源とくにプロセッサの効率的な利用を目指したバッチ処理による管理を行う時代、
次いで、マルチプログラミングやTSSによって複数のユーザが同時に利用できるようにする時代と発展し、

オペレーティングシステムの歴史と発展（2）

- 1960年代後半から80年代：
  仮想記憶の本格的な採用
  - 汎用大型計算機（汎用コンピュータ、メインフレーム）
    - 科学技術計算、商用計算、バッチ処理、タイムシェアリングシステム
    - IBM System/360 の OS/360 など
    - Multics（Multiplexed information and computing service）
      MIT・GE・AT&Tベル研究所の共同プロジェクト
  - ワークステーション … UNIX
  - マイクロコンピュータ → PC … PCのOS（MS/DOS）
- 1990年代から：
  パソコンの急速の進歩、Windows系のOS、MacのOS
  - GUI
  - UNIX系のOS
  - 組込みOS
  - ネットワーク

仮想記憶が実現され、ワークステーション、マイクロコンピュータの出現となって、1990年以降のPCの時代となり、さらに、モバイル機器のOSへと進みます。

エポックメーキングなOS

- OS/360

  IBM社がSystem/360シリーズのために開発したオペレーティングシステム

  - コンピュータを制御するために必要な機能を体系化
    高機能・大規模、世界初の商用OS
  - 「オペレーティングシステム」という名称が一般化

- Multics

  タイムシェアリング処理をベースとしたシステム
  非常に先進的、高機能（実用化しなかった）

  - 斬新なアイデア
    - 単一レベル仮想記憶、動的リンク、オンライン構成変更
  - マルチプロセッサ、階層型ファイルシステム
  - 開発言語に高級言語を使用

エポックメーキングなOSとして名前の挙がるものでは、世界初の商用OSで、オペレーティングシステムという名称を一般のものとしたOS/360がありました。

また、非常に先進的・高機能なOSであるMulticsが開発されました。

UNIX

- Multicsプロジェクトで仕事をしていたAT&Tベル研究所の K. ThompsonとD. Ritchieらが開発
- Multicsに影響を受け、単純明快で軽量なOSを目指す
  UNICS : Uniplexed Information and Computing Service
- 小さく、単純な機能の組合せで複雑で巨大なものを構成する
  - ASCIIテキストのファイル
  - バイト単位の入出力
  - 正規表現
- 1969年から開発
- System V系とBSD（Berkeley Software Distribution）系の2系統

  Sytem V系 … AT&T社が製品化。ベル研究所が開発したオリジナル版の流れをくむ。商用OSに多く採用

  BSD系 … UCB（カリフォルニア大学バークレー校）のCSRG（Computer Systems Research Group）が開発、配布
  オープンソース開発プロジェクトの基盤となる

Multicsでの経験から、軽量なOSを目指したUNIXが開発されました。

Linux

- MINIX
  A. Tanenbaum 教授が教科書「オペレーティングシステム　設計と実装」の中で開発したUNIX系のOS
  - OSの教材用に UNIXの互換システムを再設計
  - 機能上新しさはないが、モダンな洗練が行われている
  - 1987年のリリース当初からすべてのコードは公開
- Linux
  L. Torvalds が MINIX に触発され、新たにOS（カーネル）を作り始める
  - 1991年 初バージョンリリース
  - 1994年 バージョン 1.0 リリース
  - 1996年 バージョン 2.0
  - 2011年 バージョン 3.0
  - 2019年 バージョン 5.0
  - …

UNIX系のOSではLinuxが多く利用されています。

PCのOS

- Windows系　　x86系とx64系のCPUを搭載したコンピュータで動作
  - 1985年にMS/DOS上で稼働するGUI環境として登場
  - 1994年 Windows NT
  - 1995年 Windows 95
  - 2001年 Windows XP
  - 2015年 Windows 10
  - …
- macOS
  - ビットマップディスプレイとマウスの利用、マルチウインドウ、メニュー操作など、Xerox社で1970年代に研究開発されていた Alto から多くの影響
  - ドラッグ・アンド・ドロップのファイル操作、クリップボード、プルダウンメニュー、ゴミ箱などの発明
  - 1999年 Mac OS 9
  - 2001年 Mac OS X v10.0（Cheetah）
  - 2012年 OS X Mountain Lion
  - 2016年 macOS Sierra
  - …

PCのOSとしてWindows、macOSなどがあります。

事後学習・事前学習

- 今回の講義資料に基づいて内容を振り返り、教科書などの該当箇所を読む
  「オペレーティングシステムの歴史と発展」に関して、教科書 1.5「オペレーティングシステムの利 用形態」をみてください
- コンピュータアーキテクチャとプログラムについて概要を確認しておく
- 教科書第1章（1.6）、第3章（3.1～3.3）に目を通しておく

今回の講義内容の振り返りと次回の準備をお願いします。