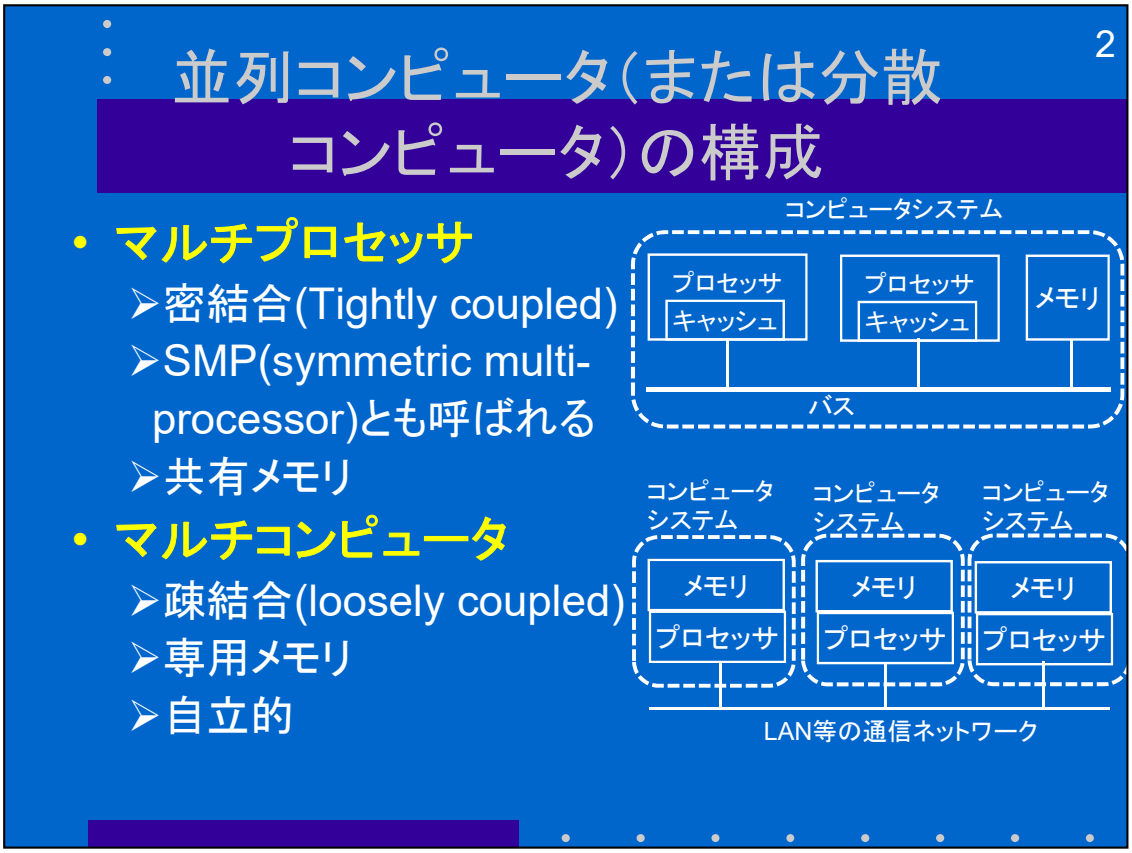
オペレーティングシステム

資料　第3分冊(2021)

村田正幸　(murata@ist.osaka-u.ac.jp)
○松田秀雄(matsuda@ist.osaka-u.ac.jp)

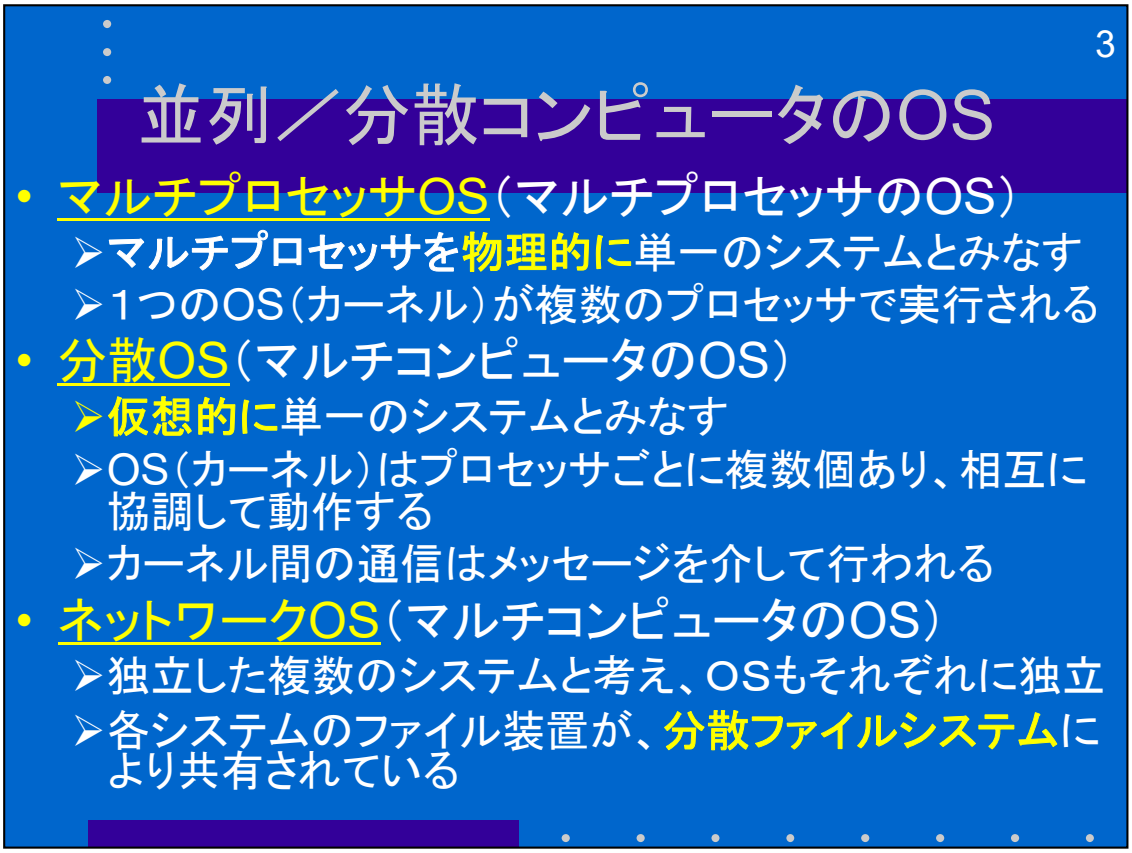


今回は、まず、並列コンピュータと分散コンピュータのオペレーティングシステムについて説明します。

並列コンピュータと分散コンピュータは、コンピュータシステムとしての構成の違いから、それぞれマルチプロセッサ、マルチコンピュータと呼ばれます。

マルチプロセッサとは、その名の通り、複数のプロセッサをバスでつないで一つのコンピュータシステムを構成しているもので、密結合やSMPとも呼ばれます。この構成ではメモリはシステム上で一つにまとめられて、各プロセッサから共有されるため、共有メモリと呼ばれます。

次に、マルチコンピュータとは、複数の独立したコンピュータシステムがLANなどの通信ネットワークでつながれたもので、疎結合とも呼ばれます。メモリは、コンピュータシステムごとに分散しており、専用メモリと呼ばれます。各コンピュータシステムは、単体でも動作する自立した構成となっています。



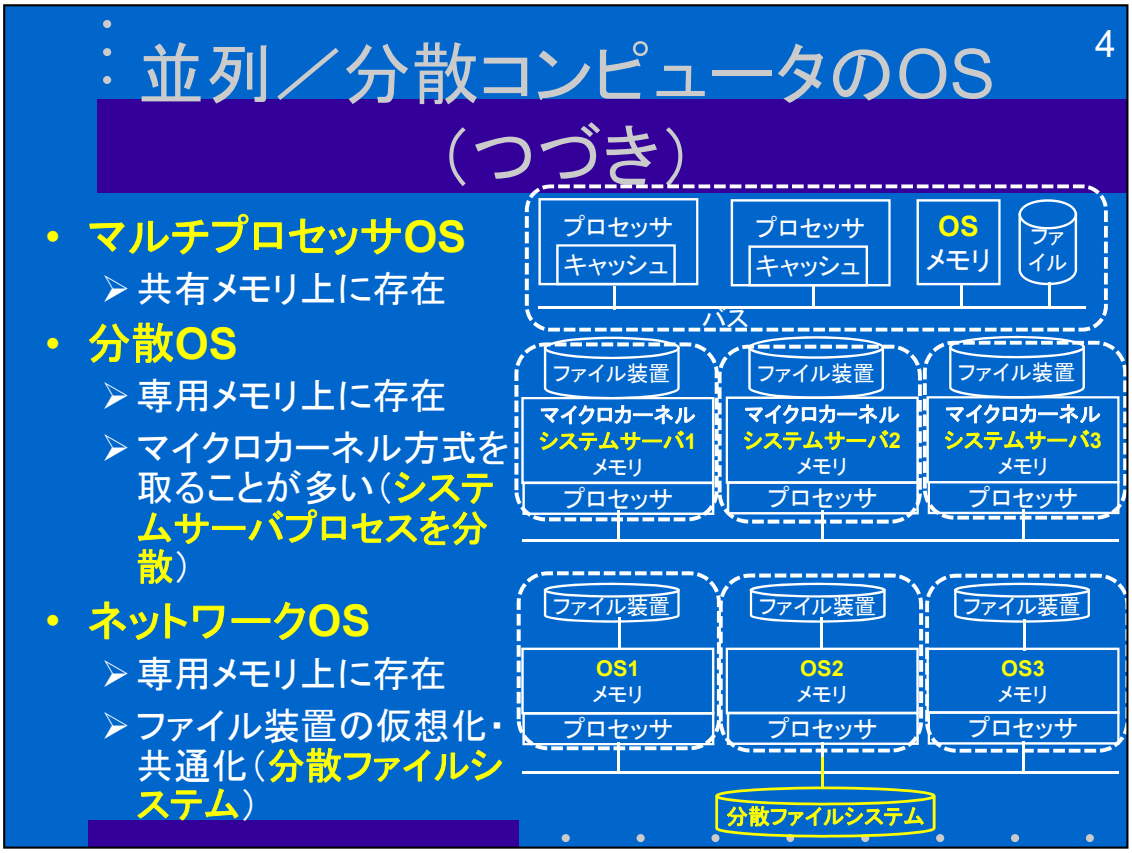
マルチプロセッサとマルチコンピュータのOSはどのように構成されるかを説明します。

マルチプロセッサOSは、マルチプロセッサのOSであり、マルチプロセッサを**物理的に**単一のシステムとみなして管理します。1つのOS（カーネル）が複数のプロセッサで実行されます。

マルチコンピュータのOSには2種類あります。

1つ目は、分散OSで、マルチコンピュータを**仮想的に**単一のシステムとみなして管理します。OS（カーネル）はプロセッサごとに複数個あり、相互に協調して動作します。複数あるカーネル間の通信はメッセージを介して行われます。

2つ目は、ネットワークOSで、マルチコンピュータを独立した複数のシステムと考えて管理します。システムごとにOSは独立して存在します。各システムのファイル装置が、分散ファイルシステムとしてまとめられ共有されています。

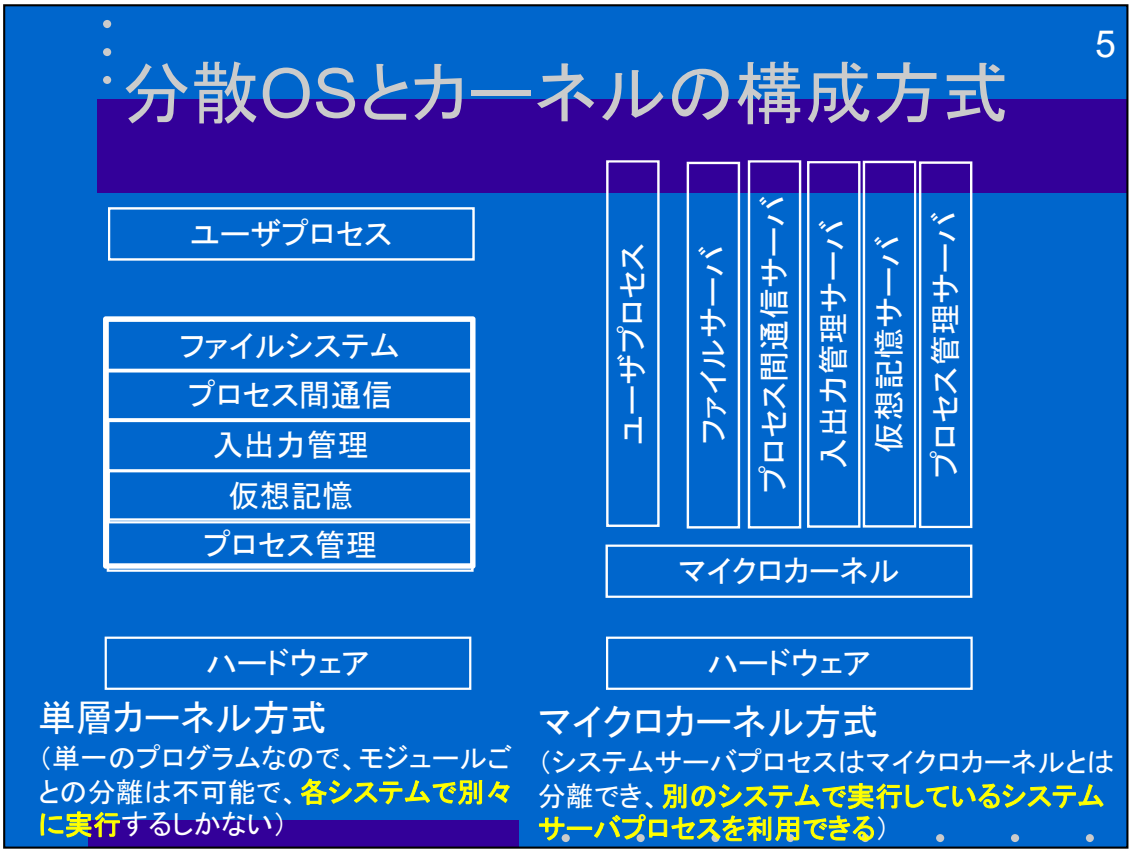


前のスライドの説明を図で表すとこのようになります。

マルチプロセッサOSでは、OSは共有メモリ上に一つだけ存在します。

分散OSでは、各専用メモリ上にそれぞれOS（カーネル）が存在します。この構成では、マイクロカーネル方式を取ることが多くなります。つまり、各専用メモリにそれぞれマイクロカーネルが置かれ、各システムサーバプロセスはいずれかのシステムの専用メモリに置かれます。例えば、ファイル管理はシステム1に、プロセス管理はシステム2にというように置かれます。

ネットワークOSでは、システムごとにそれぞれ独立したOSが置かれます。この構成では、システムごとに異なるOSを置くことができます。各OSはファイル装置を仮想化した分散ファイルシステムでつながっています。



分散OSとカーネルの構成方式との関係を説明します。

単層カーネル方式は、カーネル全体が単一のプログラムなので、モジュールごとに分離することができません。このため、分散OSであっても、各システムで別々にOSを実行することになります。

一方、マイクロカーネル方式では、システムサーバプロセスがマイクロカーネルとは分離しているため、分散OSでは、別のシステムで実行しているシステムサーバプロセスを利用できます。これにより、OSのメモリ上で占める領域を減らすことができます。

6

1.3.2 OSの実際

[1] OSの実際的な動作　ー概要ー

OSの起動時（**ブート**(boot)と呼ばれる）

① ファイル装置からメインメモリに読み出す

② メインメモリに割り付ける

③ メインメモリに保持して、OS自身を起動する

OSによる仮想化

- ハードウェアの仮想化により、OSの機能が同じであれば、ハードウェアに依存せずに、ユーザプログラムがオブジェクトプログラムのレベルで動作する

→ **オブジェクト互換性**（プログラムをコンパイルし直さなくても、別のコンピュータで実行できる）

⇔ **ソース互換性**（ソースプログラムをコンパイルし直せば、別のコンピュータで実行できる）

OSの実際的な動作を説明します。

OSの起動時には、**ブート**(boot)と呼ばれる以下の動作が行われます。

① ファイル装置からメインメモリに読み出す

② メインメモリに割り付ける

③ メインメモリに保持して、OS自身を起動する

OSが起動すると、ハードウェア装置はOSにより仮想化されます。

このハードウェアの仮想化により、OSの機能が同じであれば、ハードウェアに依存せずに、ユーザプログラムがオブジェクトプログラムのレベルで動作します。

つまり、**オブジェクト互換性**（プログラムをコンパイルし直さなくても、別のコンピュータで実行できる）ことになります。

ちなみに、オブジェクトレベル互換性がない場合は、**ソース互換性**（ソースプログラムをコンパイルし直せば、別のコンピュータで実行できる）となりますが、これはOSではなく、コンパイラ等の言語処理プログラムにより可能となります。

7

OSの起動

OSの起動(boot)過程

1. ブートROMにある**ブートストラップローダ**(bootstrap loader)による、ファイル装置から**IPL** (initial program loader)をメインメモリに読み出す
2. IPLに制御権が移り、IPLがOSプログラムをメインメモリへ読み出す（ロードする）
3. OSを初期化する
   - ① ハードウェア装置の認識
   - ② ファイル装置上のファイルの位置を識別
   - ③ メインメモリ上にOSを常駐する領域や空間を確保する

OSの起動は実際には、ここに書かれているような手順で行われます。

8

OSの起動（つづき）

OSの再起動（reboot）

- **ハードリセット**（コールドブートともいう）
  - 電源オンからやり直す
- **ソフトリセット**（ウォームブートともいう）
  - 動手順の2．からやり直す

OSの停止（shutdown）

1. すべてのユーザプロセス（ユーザプログラム）について停止処理を行う
   - オープンしていたファイルはクローズする
2. 電源をオフする

起動されたOSの再起動には、ハードリセットとソフトリセットの2種類があります。
また、OSの停止はここに書かれている手順で行われます。

1.3.3 OSと仮想マシン
ーOS機能の隠ぺいー

9

- ハードウェアだけでなく、**OSそのものも隠ぺい**する
- なぜ、OSを隠ぺいする必要があるのか？

（参考）*ハードウェア機構の隠ぺいの目的*

- *個々のハードウェアが持つ物理的な特性や相違点を隠す*

*→隠ぺいにより、**共通の（論理的な）装置に統一化***

OSの隠ぺいの目的

- 個々のOSが持つ個別の特徴や相違点を隠す

（例：Windows XP/ Vista/ 7/ 8/ 10, macOS, Linux, FreeBSD）

→ 隠ぺいにより、OSを統一化することで**多様なOSに対応できる**（詳細は後述）

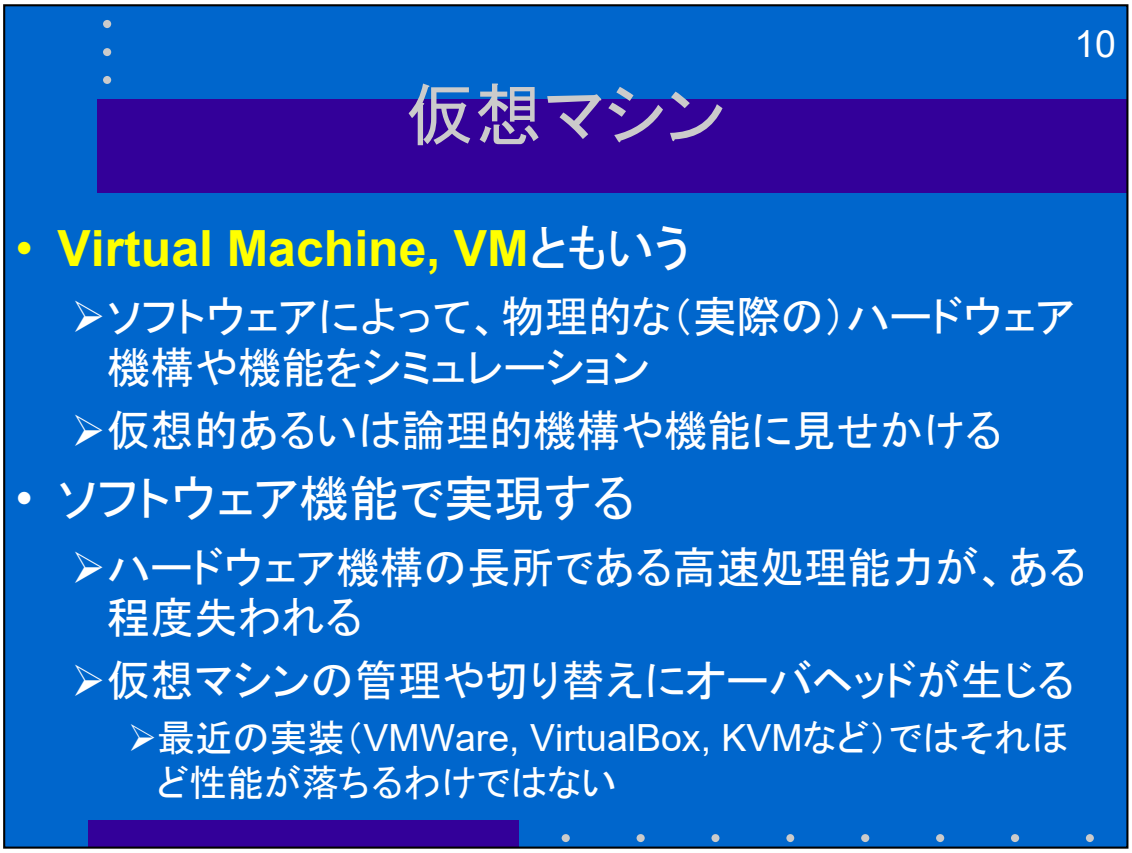
OSの機能の隠ぺいについて説明します。

これは、ハードウェアだけでなく、**OSそのものも隠ぺい**することを指します。

なぜ、OSを隠ぺいする必要があるのか説明します。

ハードウェア機構の隠ぺいの目的は、個々のハードウェアが持つ物理的な特性や相違点を隠すことで、**共通の（論理的な）装置に統一化することにありました。**

OSの隠ぺいの目的は、個々のOSが持つ個別の特徴や相違点を隠すことにあります。例えば、OSには、Windows XP/ Vista/ 7/ 8/ 10, macOS, Linux, FreeBSDと多数の種類がありますが、この隠ぺいにより、OSを統一化することで**多様なOSに対応できます。**



OSの機能の隠ぺいには、仮想マシンが必要です。

仮想マシンは、Virtual Machine, VMとも呼ばれ、ソフトウェアによって、物理的な（実際の）ハードウェア機構や機能をシミュレーション、つまり、仮想的あるいは論理的機構や機能に見せかけます。

仮想マシンは、ソフトウェア機能で実現できます。

一方で、ソフトウェア機能で実現すると、ハードウェア機構の長所である高速処理能力が、ある程度失われることになります。また、仮想マシンの管理や切り替えにオーバヘッドが生じることが考えられます。

ただし、仮想マシンの最近の実装（VMWare, VirtualBox, KVMなど）ではそれほど性能が落ちるわけではありません。

11

エミュレーションとファームウェア

- **エミュレーション**
  - 物理的なハードウェア機構や機能をシミュレーションを、ハードウェアにより行うこと
  - ファームウェアにより実現することが多い
  - 1台の実コンピュータ上で複数の仮想マシンをエミュレーションすることができる（図1.13）
- **ファームウェア**
  - ハードウェアを制御するソフトウェア（マイクロプログラムともいう）
  - FPGA (Field Programmable Gate Array)もファームウェアの一種

仮想マシンによる、ハードウェアのシミュレーションについて説明します。

これには、**エミュレーション**と呼ばれる方式があります。物理的なハードウェア機構や機能をシミュレーションを、ハードウェアにより行うことを指し、ファームウェアにより実現することが多いです。

なお、1台の実コンピュータ上で複数の仮想マシンをエミュレーションすることができることに注意します（教科書の図1.13を参照）

ここで、**ファームウェアとは、**ハードウェアを制御するソフトウェア（マイクロプログラムともいう）を指します。FPGA (Field Programmable Gate Array)もファームウェアの一種です。

12

OSと仮想マシンOS

- 実コンピュータ上で複数の仮想マシンを設定する場合は、次の2種類のOSが機能することになる（図1.14）
  - **ホストOS**（実コンピュータのOS）
    実コンピュータ（ホストコンピュータ）上で稼働し、その実コンピュータを管理・制御するOS
  - **ゲストOS**（仮想マシンのOS）
    実コンピュータ上で稼働する各仮想マシンを管理・制御するOS
- 仮想マシンのそれぞれに異種のゲストOSを搭載し、同時に稼働させるコンピュータシステムを、「**マルチOSコンピュータ**」という

仮想マシンにより、OSをどのように構成するかを説明します。

実コンピュータ上で複数の仮想マシンを設定する場合は、次の2種類のOSが機能することになります（図1.14）

- **ホストOS**（実コンピュータのOS）
  実コンピュータ（ホストコンピュータ）上で稼働し、その実コンピュータを管理・制御するOSを指します。
- **ゲストOS**（仮想マシンのOS）
  実コンピュータ上で稼働する各仮想マシンを管理・制御するOSとなります。

仮想マシンのそれぞれに異種のゲストOSを搭載し、同時に稼働させるコンピュータシステムを、「**マルチOSコンピュータ**」ということがあります。

13

OS機能の隠ぺい

- マルチOSコンピュータでは、次の2段階の隠ぺいが行われている
  1. **ホストOS**による実コンピュータのハードウェア機構の隠ぺい
  2. **ゲストOS**による仮想マシンの隠ぺい
- この2段階の隠ぺいにより、「**仮想マシンによるホストOSの隠ぺい**」が行われる
- より隠ぺいのレベルが高くなっている（図1.15）
  - ➢**ゲストOS： 仮想マシンを隠ぺい**
  - ➢**仮想マシン： ホストOSを隠ぺい**
  - ➢ホストOS: 実ハードウェアを隠ぺい

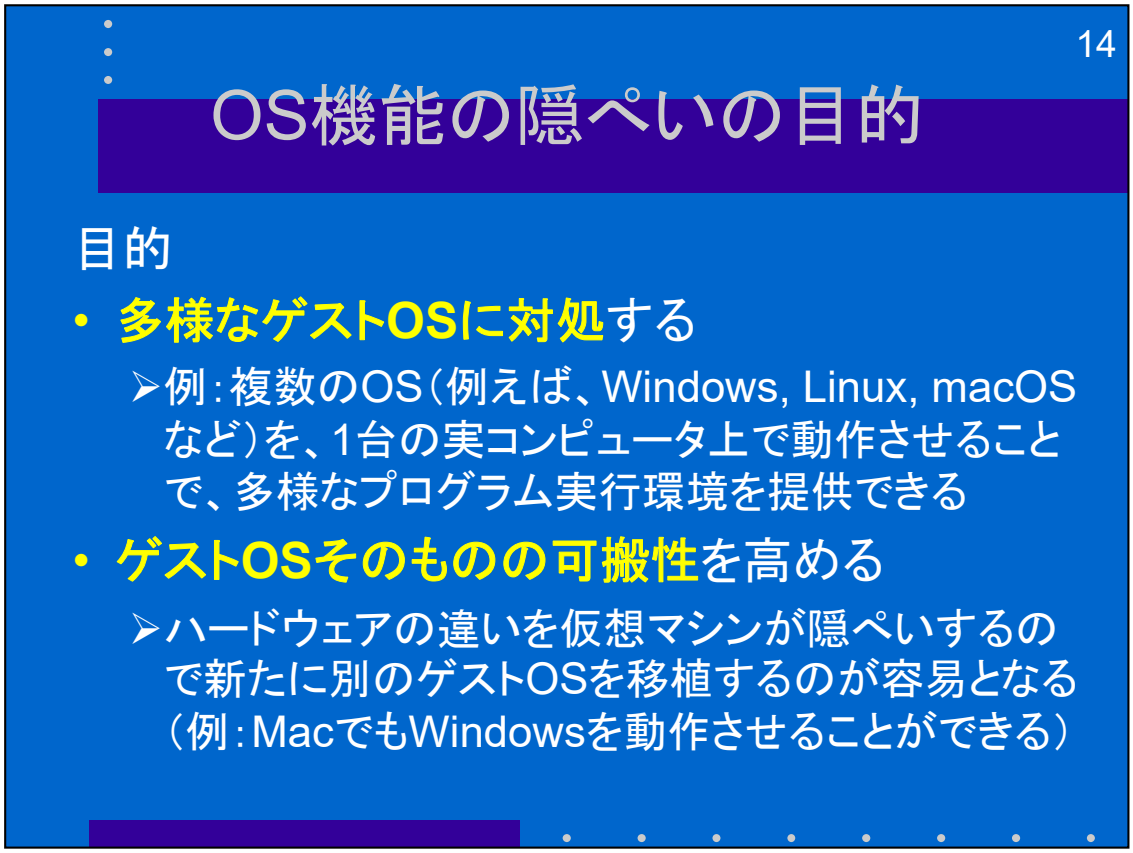
マルチOSコンピュータでは、次の2段階の隠ぺいが行われています。

1. **ホストOS**による実コンピュータのハードウェア機構の隠ぺい
2. **ゲストOS**による仮想マシンの隠ぺい

この2段階の隠ぺいにより、「**仮想マシンによるホストOSの隠ぺい**」が行われます。

ハードウェア機構の隠ぺいと比べて、次のように、より隠ぺいのレベルが高くなっています（教科書の図1.15）

- ➢**ゲストOS： 仮想マシンを隠ぺい**
- ➢**仮想マシン： ホストOSを隠ぺい**
- ➢ホストOS: 実ハードウェアを隠ぺい



OS機能の隠ぺいの目的には、ここであげたようなものがあります。

1つ目は、1台のコンピュータシステムで、多様なゲストOSを動かすことです。例えば、1台のコンピュータシステム上で、Windows, Linux, macOSを動かすことができ、アプリケーションプログラムを多様なOS上で開発することが可能になります。

2つ目は、ゲストOSそのものの可搬性を高めることです。これにより、特定のゲストOSを多様なコンピュータシステムで動作させることが可能になります。

15

OS機能の隠ぺいの効果

- 1台の実コンピュータ上で、多種多数のユーザプログラムの実行環境、開発環境を提供
- 実コンピュータの台数を絞れる
  - ➢**管理コストを削減**（例　情報科学科の演習室）
  - ➢地球温暖化や災害時の**消費電力削減にも有効**（**グリーンIT**）
- 仮想マシンを複数台動作させて別々のプログラムを実行し、冗長性を持たせて**耐故障性を向上**
- 昔のOSを稼働させ、ソフトウェアの寿命を延ばす
  - ➢保守やセキュリティ対策が切れたOSで動作していた、**レガシーソフトウェア**（古いソフトウェア）の寿命の延長

OS機能を隠ぺいするとここで示したような効果が得られます。

実際に、情報2コースの演習室にあるのは、コンピュータシステムではなくディスプレイ端末でそこではプログラムの実行等は行われません。

端末からログインするとサーバ室にあるサーバで仮想マシンが作られ、そこでプログラムが実行されます。

例えば、OSの一部をアップデートしたり、新たなソフトウェアをインストールするときは、仮想マシンのイメージに対してのみ作業すればよくなります（ログインのときに、この仮想計算機のイメージを使って、ユーザごとの仮想計算機が作成されます）。

これにより、管理コストを削減するだけでなく、消費電力の削減や耐故障性の向上が期待できます。

他にも、昔のOSを稼働させ、ソフトウェアの寿命を延ばす、すなわち、保守やセキュリティ対策が切れたOSで動作していた、**レガシーソフトウェア**（古いソフトウェア）の寿命の延長にも有効です。

16

OS機能の隠ぺいの実現

仮想マシンの構成方式としては、次の2種類がある（図1.16）

(A) ホストOSあり・複数分散仮想マシン

- 異種複数の仮想マシンをホストOSとは独立した機能として構成する
- 既存のコンピュータシステム上で、仮想化ソフトウェアを追加インストールするだけで実現可能
- ホストOSと異種複数の仮想マシンが共存する形となり、(B)に比べるとコストが大きい

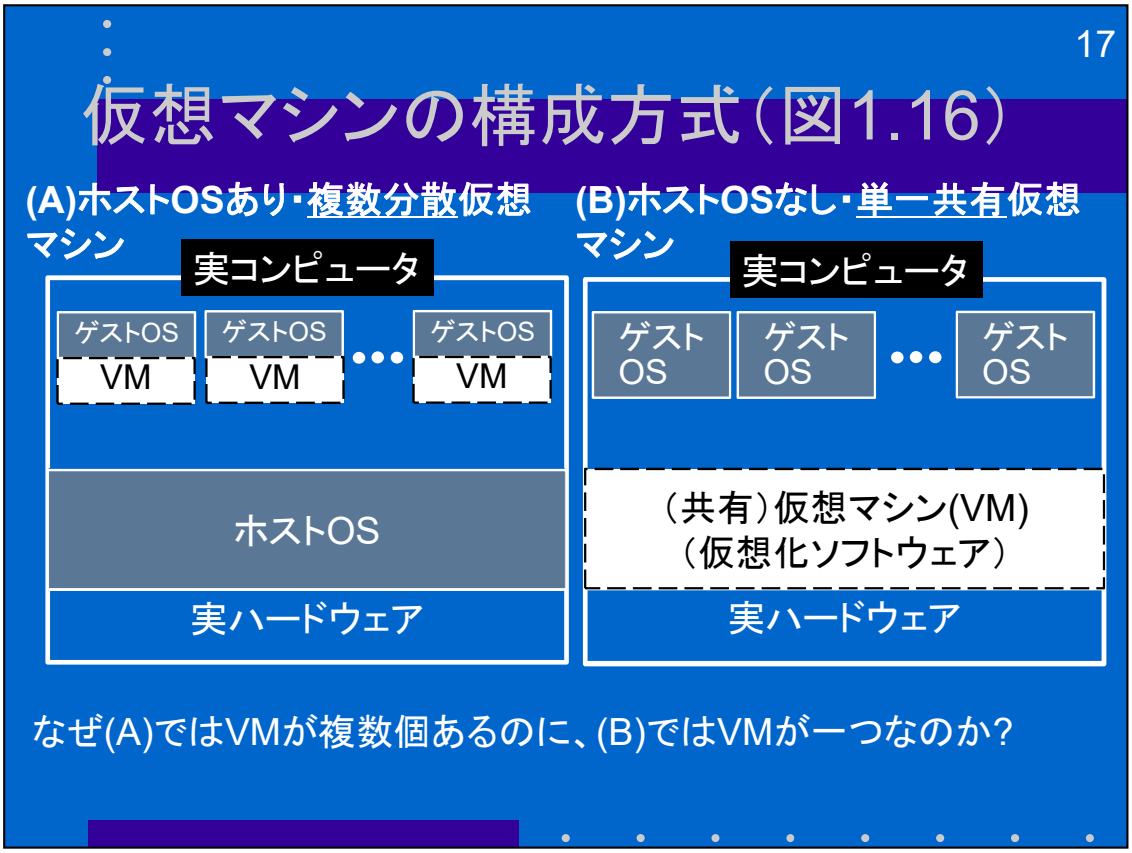
(B) ホストOSなし・単一共通仮想マシン

- 単一の仮想化ソフトウェアをホストOSの代わりにあらかじめ組み込んで構成する
- そのコンピュータシステムに本来あるOSを置き換える必要あり
- 仮想化ソフトウェアを実ハードウェア上で稼働させて共有すればよく、ホストOSがないので、(A)に比べるとコストが小さい

仮想マシンの構成方式としては、従来は、(A)のように、コンピュータシステムの元々のOS（ホストOSと呼びます）の上に、追加で仮想マシンを複数作成する方法が取られています。

近年は別の方式として、(B)のようにホストOSをなくして、その代わりに最初から仮想マシンを作成しておくという方式も使われるようになっています。

(B)の方がホストOSを通さなくてすむので、(A)よりも実行時のコストが小さく、効率がよいという利点があります。



前のスライドで述べた2つの方式を図で表したものがこれです（教科書の図1.16と同じ）。

ここで、注意ですが、（A)では仮想マシン（VMと表記）が複数あるのに、（B)では一つしかないということです、

なぜでしょうか？

なぜ(A)ではVMが複数個あるのに、(B)ではVMが一つなのか？ 18

- (A)の方式
  - ホストOSが実ハードウェアを隠ぺいし、プロセス管理を行う
  - VMは**ホストOS上でプロセスとして実行される**ため、**複数個を実行でき**、ホストOSを隠ぺいする
  - ゲストOSは、VMにより提供される仮想化されたハードウェアの基で、**VMの数だけ動作**する
- (B)の方式
  - **ホストOSがない**ので、VMはプロセスとして実行できず、**1個しか実行できない**
  - ゲストOSは、VMにより提供される仮想化されたハードウェアの基で動作する
  - VMがマルチタスキングのプロセス管理機能を持たない限り、ゲストOSは**1個しか動作できない**（ただし、ユーザ等の指示により、別のゲストOSに切り替えて動作させることは可能）

この理由は、OSの本来の役割がわかれば理解できると思います。

OSは本来は、実際のハードウェア装置を隠ぺい（もしくは、仮想化または共通化）して、ユーザプログラムに統一されたインタフェースを提供するためにあります。

（A)だと、この役割はホストOSが担当してもらえるので、仮想マシンはホストOSから見るとユーザプロセスとなり、いくつでも好きなだけ作成できます。

一方（B）では、ホストOSがないので、仮想マシンがハードウェア装置を直接管理して隠ぺいしなければなりません。複数の仮想マシンを作ってしまうと、実ハードウェアを複数の仮想マシンで共有することになり、仮想マシンとは別に実ハードウェアの利用を決めるソフトウェアが必要になってしまいます。

これではOSをもう一つ別に持たせることになってしまい、意味がなくなるので、仮想マシンは一つにしてそこで実ハードウェアの管理と隠ぺいを行うようにしています。

ただし、この仮想マシンでは、複数のゲストOSを動作させる必要があり、マルチタスキングのプロセス管理が求められるため、（A)の方式の仮想マシンよりも必要とされる機能が多くなります。

OS機能のファームウェア化

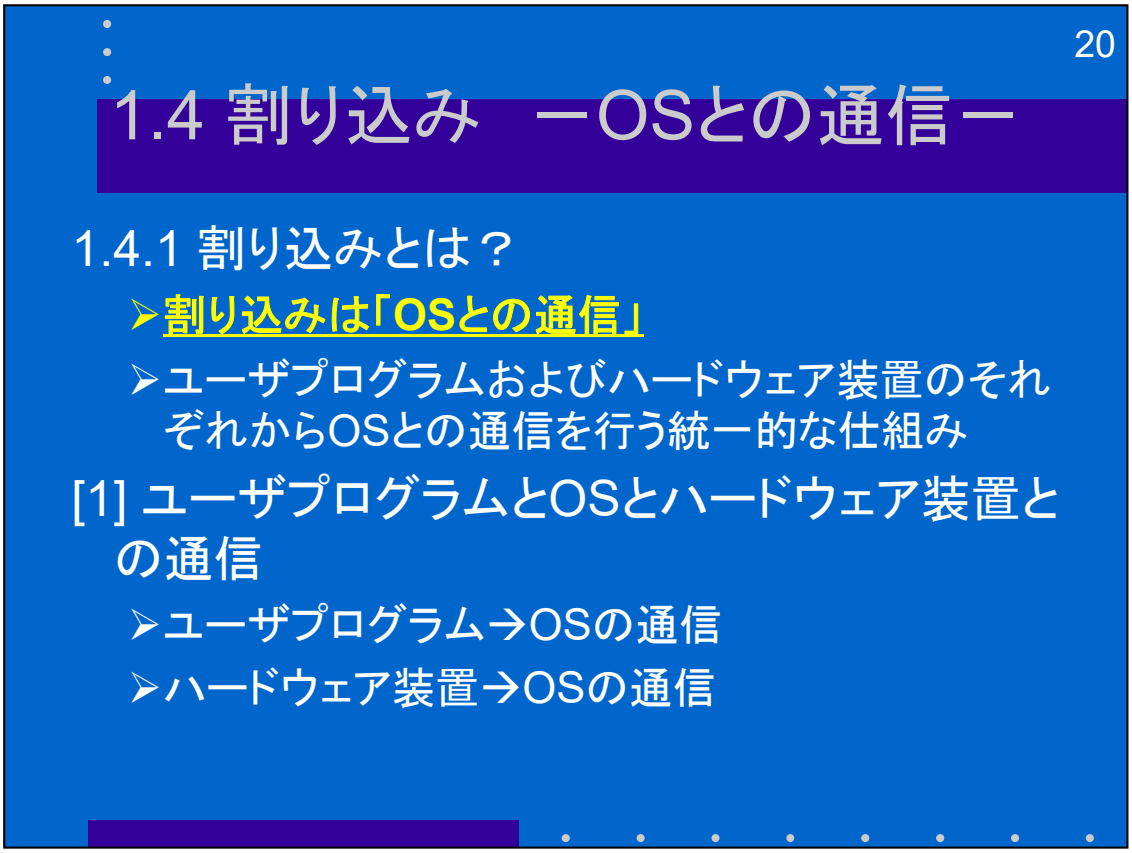
19

- OS機能の一部をソフトウェアではなく、ファームウェアによって実現し、OS機能を改善あるいは強化すること
- 一部の機能で特別な条件が要求される場合（リアルタイムシステムなど）や、利用できるハードウェア装置に制約が強いとき（組み込みシステムなど）に有効
- ファームウェア化の例
  - 仮想マシンや言語処理プログラムのオーバヘッドの発生防止
  - ハードウェア機能のテストや保守での柔軟性確保
  - プロセス管理でのスケジューリングアルゴリズムや、仮想メモリでのページ置換アルゴリズム、通信プロトコルなどの実現
  - 多種多様な割り込み要因に対する柔軟かつ高速な対処

OS機能についての別の改良として、機能の一部をファームウェアで実現することが考えられます。

これは、OSの一部の機能で特別な条件が要求される場合（リアルタイムシステムなど）や、利用できるハードウェア装置に制約が強いとき（組み込みシステムなど）に有効となります。

ファームウェア化の例には、ここであげているようなものがあります。



次に、割り込みについて説明します。

割り込みは、元々は、外部の入出力装置などのハードウェア装置で発生し、コンピュータシステムの動作を中断させて割り込み処理を行わせるためにありました。

しかし、実際には、外部からの割り込み以外に、ユーザプログラムからのOSの機能の呼び出しにも割り込みが使われています。

つまり、割り込みとはOSとの通信の手段であり、これによってユーザプログラムとハードウェア装置のそれぞれについて、OSとの通信を行う統一的な仕組みとなっています。

21

マシン命令の実行と割り込み

- 割り込み機構および機能
  - マシン命令の実行フローを、不測の事態の発生時に強制的かつ動的に変更する手段
  - 割り込みの原因となる不測の事態を「**割り込み要因**」という
  - 割り込みの発生：割り込み要因が生じたことで、実行中のプログラムを一時中断して、割り込み機構の動作を開始（図1.17）
  - 割り込み処理：割り込み要因ごとに決められた処理

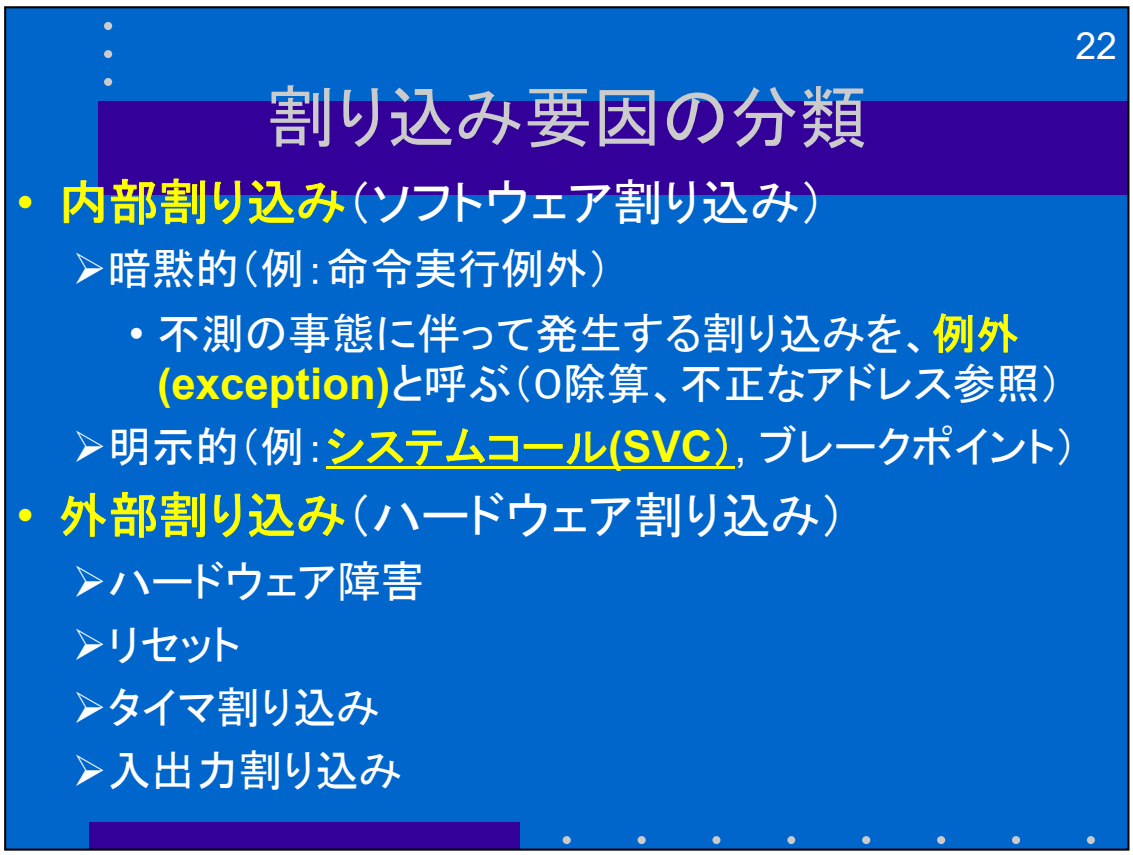
ユーザプログラムが実行するマシン命令と割り込みとの関係について説明します。

割り込みは、マシン命令の実行フローを、不測の事態の発生時に強制的かつ動的に変更する手段となります。

ここで、割り込みの原因となる不測の事態を「割り込み要因」と呼びます。

割り込みの発生では、割り込み要因が生じたことで、実行中のプログラムを一時中断して、割り込み機構の動作を開始します（教科書の図1.17）

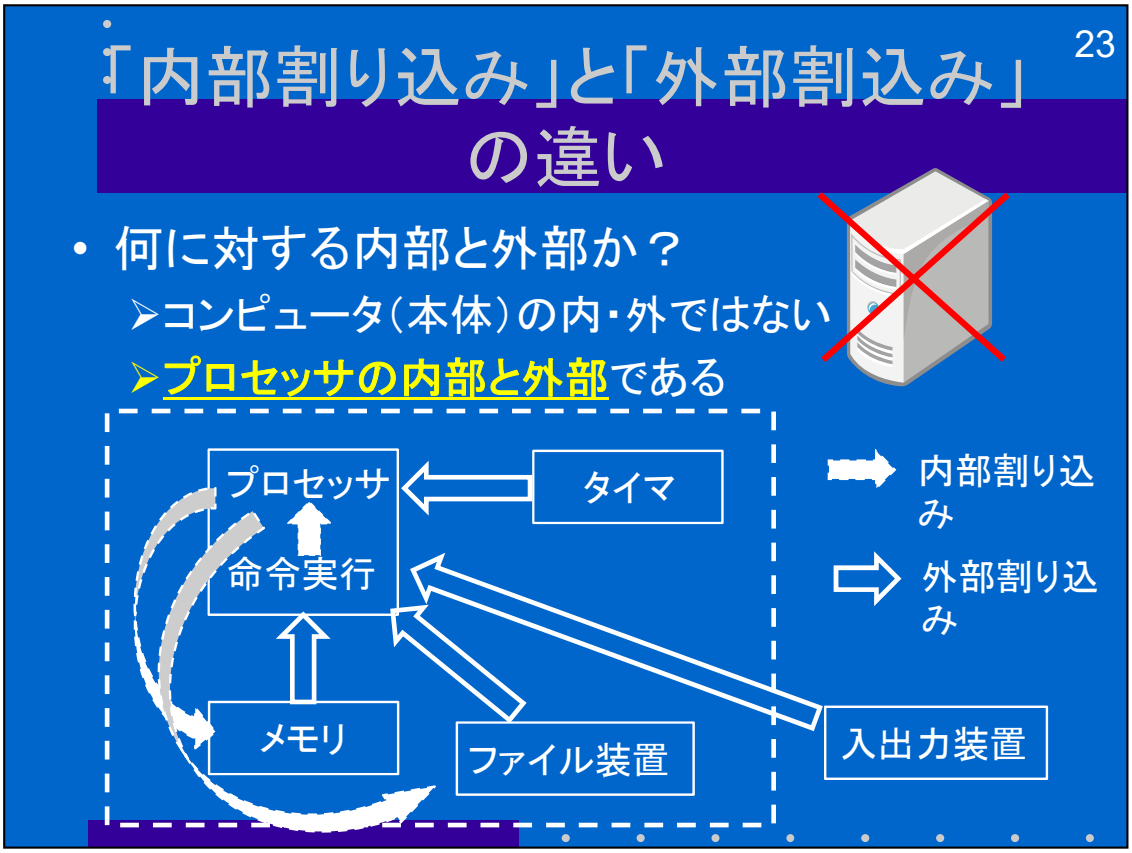
割り込みが発生すると、割り込み要因ごとに決められた処理である「割り込み処理」が行われます。



割り込みは、割り込み要因の違いによって、内部割り込みと外部割り込みに分類することができます。

内部割り込みは、ソフトウェア割り込みとも呼ばれ、ユーザプログラムでの意図とは関係なく暗黙的に、不測の事態に伴って発生する割り込み（例えば、0での除算や不正なアドレスの参照など）と、ユーザプログラムから明示的に呼ばれる割り込み（例えば、システムコールなど）があります。

一方、外部割り込みはハードウェア割り込みとも呼ばれ、外部のハードウェア装置によって割り込み要因が発生するものです。例えば、ハードウェア障害、リセット、タイマ割り込み、入出力割り込みなどがあります。

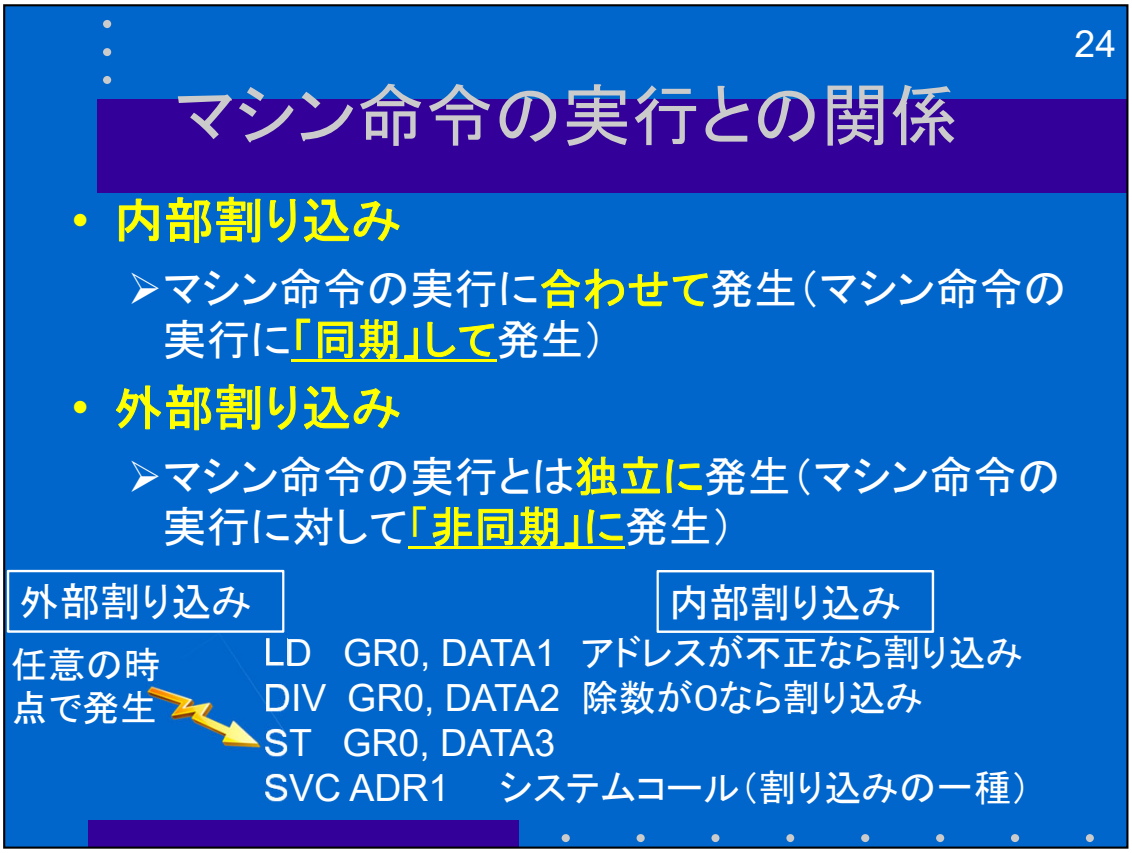


ここで、「内部」と「外部」の意味について注意する必要があります。

外部というと、コンピュータシステムの外部、つまり、例えばキーボードやディスプレイのような入出力ハードウェア装置で発生する割り込みをイメージするかも知れません、

これらは確かに「外部」ではありますが、「外部割り込み」でいう外部とは、システムの外ではなく、**プロセッサの外**を指します。

つまり、プロセッサの外にある、メモリやタイマ、ハードディスクなどはいずれも外部であり、ここで発生する要因による割り込みは全部「外部割り込み」となるので注意が必要です。



割り込みのもう一つの特徴に、マシン命令の実行との関係があります。

内部割り込みは、暗黙的であれ、明示的であれ、ユーザプログラムの実行と関係しますので、マシン命令の実行に合わせて（これを実行に「同期」すると言います）発生します。

これに対して、外部割り込みは、ハードウェアに起因する割り込みなので、マシン命令の実行とは独立に（これを「非同期」と言います）発生します。