



オペレーティングシステム

資料 第3分冊(H27)

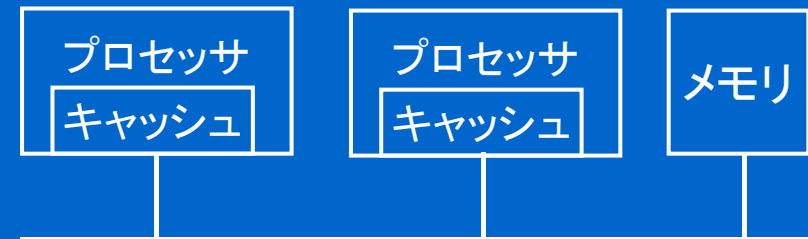


村田正幸 (murata@ist.osaka-u.ac.jp)
○松田秀雄(matsuda@ist.osaka-u.ac.jp)

並列コンピュータ(または分散コンピュータ)の構成

• マルチプロセッサ

- 密結合(Tightly coupled)
- SMP(symmetric multi-processor)とも呼ばれる
- 共有メモリ



• マルチコンピュータ

- 疎結合(loosely coupled)
- 専用メモリ
- 自立的



並列／分散コンピュータのOS

- マルチプロセッサOS

- 物理的に単一計算機のOS(プロセッサだけが複数個ある)
- 1つのOS(カーネル)が複数のプロセッサで実行される

- 分散OS

- 仮想的に単一計算機のように見ることができる(プロセッサだけでなく、メモリもファイル装置も複数個ある)
- 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する
- 通信はメッセージを介して行われる

- ネットワークOS

- 単一計算機ではなく、独立した複数の計算機
- OSは各計算機ごとに独立して存在している
- 各計算機のファイル装置が、分散ファイルシステムにより共有されている

・ 並列／分散コンピュータのOS (つづき)

4

- マルチプロセッサOS

- ー 共有メモリ上に存在



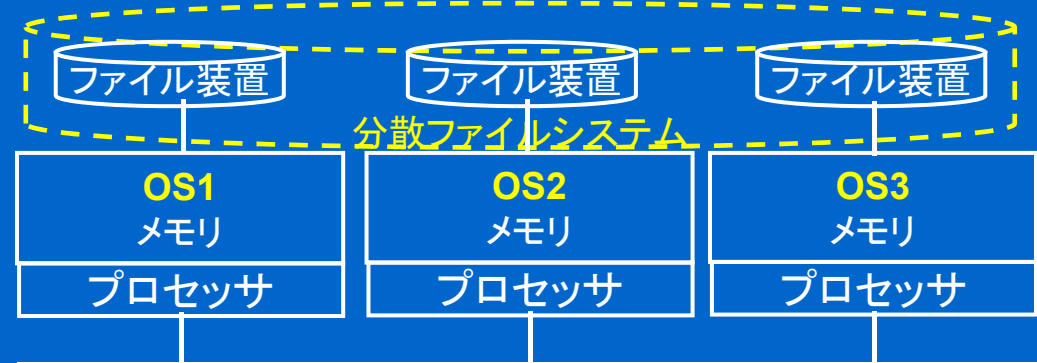
- 分散OS

- ー 専用メモリ上に存在
- ー マイクロカーネル方式を取ることが多い(システムサーバを分散)

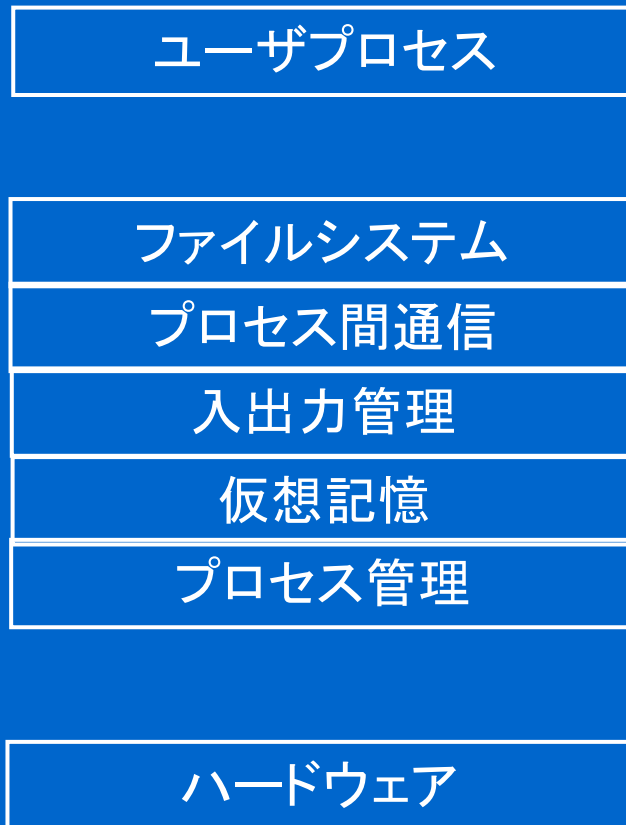


- ネットワークOS

- ー 専用メモリ上に存在
- ー ファイル装置の仮想化・共通化(分散ファイルシステム)



参考 マイクロカーネル方式



単一カーネル方式

(単一のプログラムなので、モジュールごとの分離は不可能で、**同じマシンで実行**するしかない)



マイクロカーネル方式

(システムサーバプロセスはマイクロカーネルとは分離でき、**別のマシンでも実行可能**)

1.3.2 OSの実際

[1] OSの実際的な動作 —概要—

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(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)
 - 起動手順の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

1.3.3 OSと仮想マシン

—OS機能の隠ぺい—

- 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)ハードウェアの隠ぺいの目的

— 同じ種類のハードウェア装置でも、構成や機能が多少異なる装置が多数存在している

→ 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

— 同じハードウェア上で動作するOSが複数存在している
(例: Windows XP/ Vista/ 7/ 8, Mac OS X, Linux, FreeBSD)

→ 隠ぺいにより、多様なOSに対応できる

仮想マシン

- Virtual Machine, VMともいう
 - 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
 - 仮想的あるいは論理的機構や機能に見せかける
- ソフトウェア機能で実現する
 - ハードウェアの特長である高速処理能力が、ある程度失われる
 - 仮想マシンの管理や切り替えにオーバヘッドが生じる
 - ただし、最近の実装ではそれほど性能が落ちるわけではない(VMWare, Xen, KVMなど)

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

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

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

OS機能の隠ぺいの目的

目的

- **多様なゲストOSに対処する**
 - 例：複数のOS（例えば、Windows, Linux, Mac OSなど）を、1台の実コンピュータ上で動作させることで、多様なプログラム実行環境を提供できる
- **ゲストOSそのものの可搬性を高める**
 - ハードウェアの違いを仮想マシンが隠ぺいするので、新たに別のゲストOSを移植するのが容易となる（例：Macでなくても、Mac OSを動作させることができる）

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**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
- 共有する仮想マシンを1台だけ実ハードウェア上で実現し稼働させればよいので、(A)に比べるとコストが小さい

： OS機能のファームウェア化

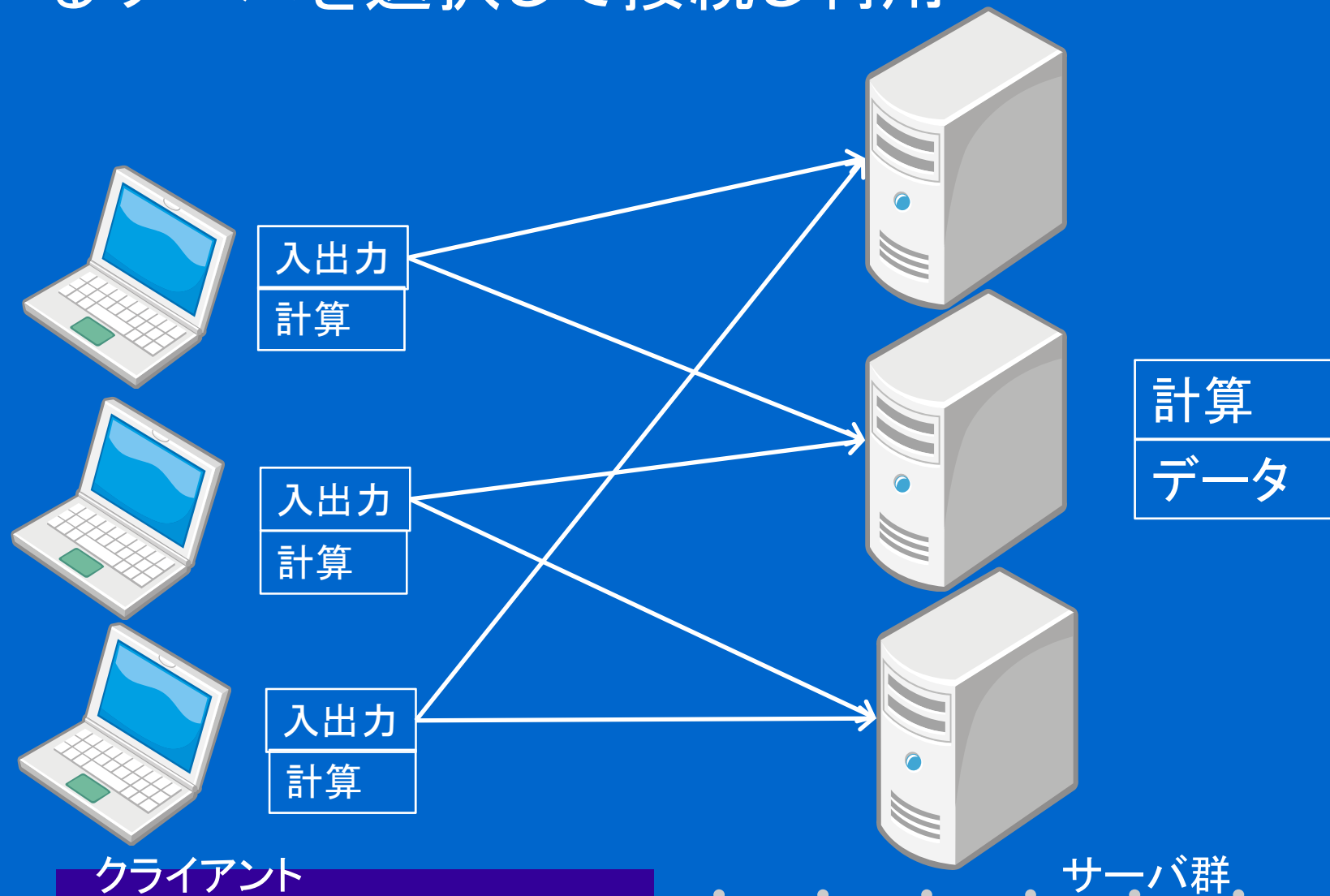
- OS機能の一部をソフトウェアではなく、**ファームウェアによって実現**し、OS機能を改善あるいは強化すること
- 一部の機能で特別な条件が要求される場合（リアルタイムシステムなど）や、利用できるハードウェア装置に制約が強いとき（組み込みシステムなど）に有効
- ファームウェア化の例
 - 仮想マシンや言語処理プログラムのオーバヘッドの発生防止
 - ハードウェア機能のテストや保守での柔軟性確保
 - プロセス管理でのスケジューリングアルゴリズムや、仮想メモリでのページ置換アルゴリズム、通信プロトコルなどの実現
 - 多種多様な割り込み要因に対する柔軟かつ高速な対処

（参考）仮想化の効果の別の例

- （従来の方式）クライアント・サーバ方式
 - － 1台のクライアントで多様な機能が利用できる（複数のサーバを切り替えて接続）
 - 計算サーバ
 - ファイルサーバ
 - データベースサーバ
- （仮想化を利用した方式）クラウドコンピューティング
 - － ユーザ側には大きな計算機設備を置かずに、ユーザが希望する処理を、ネットワークを通じてサービスとして提供する

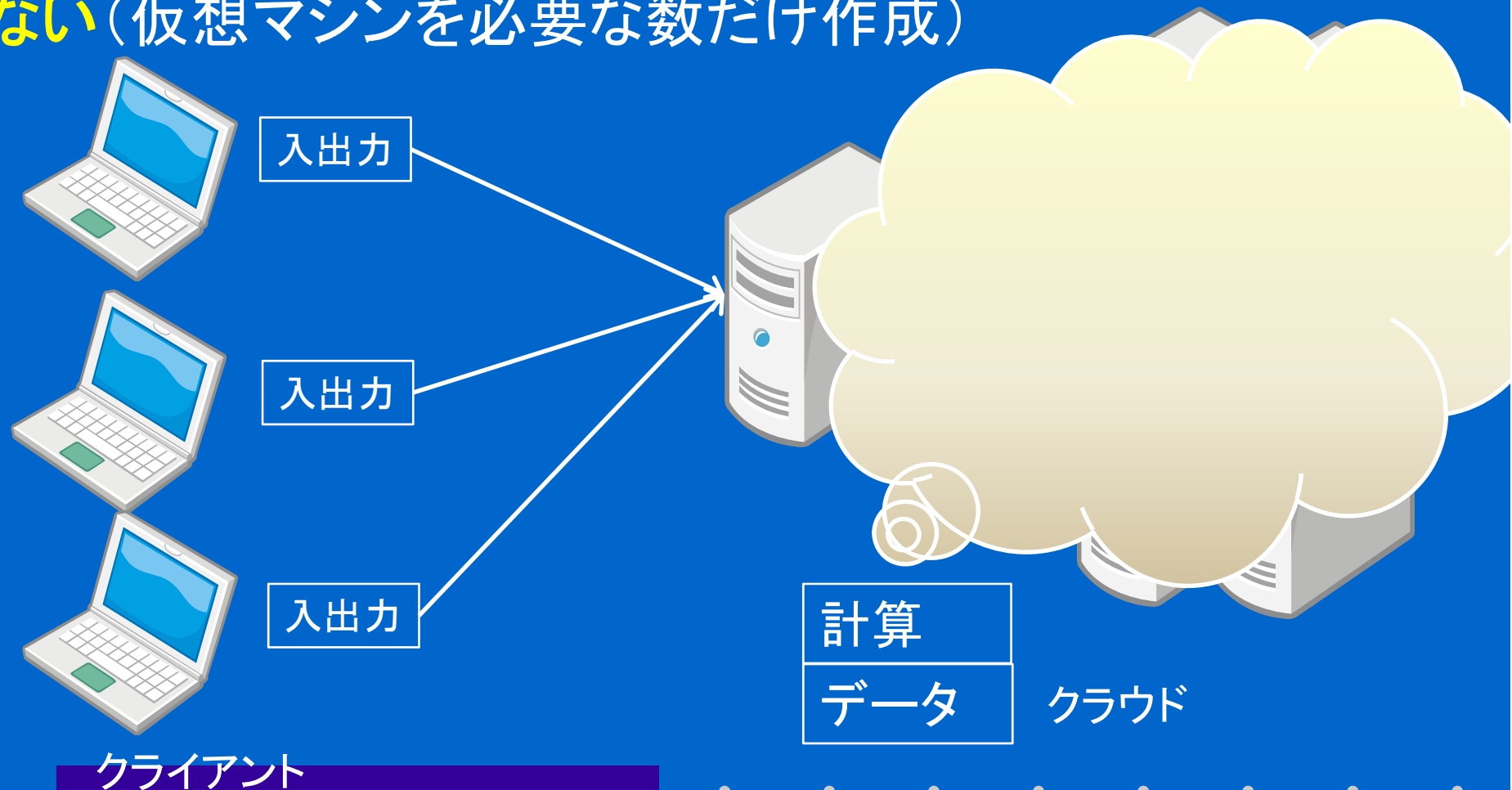
クライアント・サーバ方式

- 入出力と計算機能を持つクライアントが、複数あるサーバを選択して接続し利用



： クラウドコンピューティング

- 2006年に、Google社のエリック・シュミットが提唱。処理をネットワーク経由でサービスとして利用（Google apps, Gmail）。サーバ群の規模や構成は**仮想化されていて見えない**（仮想マシンを必要な数だけ作成）



クラウドコンピューティングの特徴

- サービスが規格化・標準化されている
 - ユーザのニーズに合わせて、最も適当な規模のサービスを選択できる
- サービスの規模を変更できる
 - クライアント・プログラムの変更なしに、その時の状況に応じてサービスの規模を増やしたり、減らしたりできる（仮想マシンの数を変更することで対応）

1.4 割り込み — OSとの通信 —

1.4.1 割り込みとは？

— 割り込みは「OSとの通信」 —

— ユーザプログラムおよびハードウェア装置のそれぞれからOSへの通信を統一的に行う仕組み

[1] ユーザプログラムとOSとハードウェア装置との通信

— ユーザプログラム→OSの通信

— ハードウェア装置→OSの通信

マシン命令の実行と割り込み

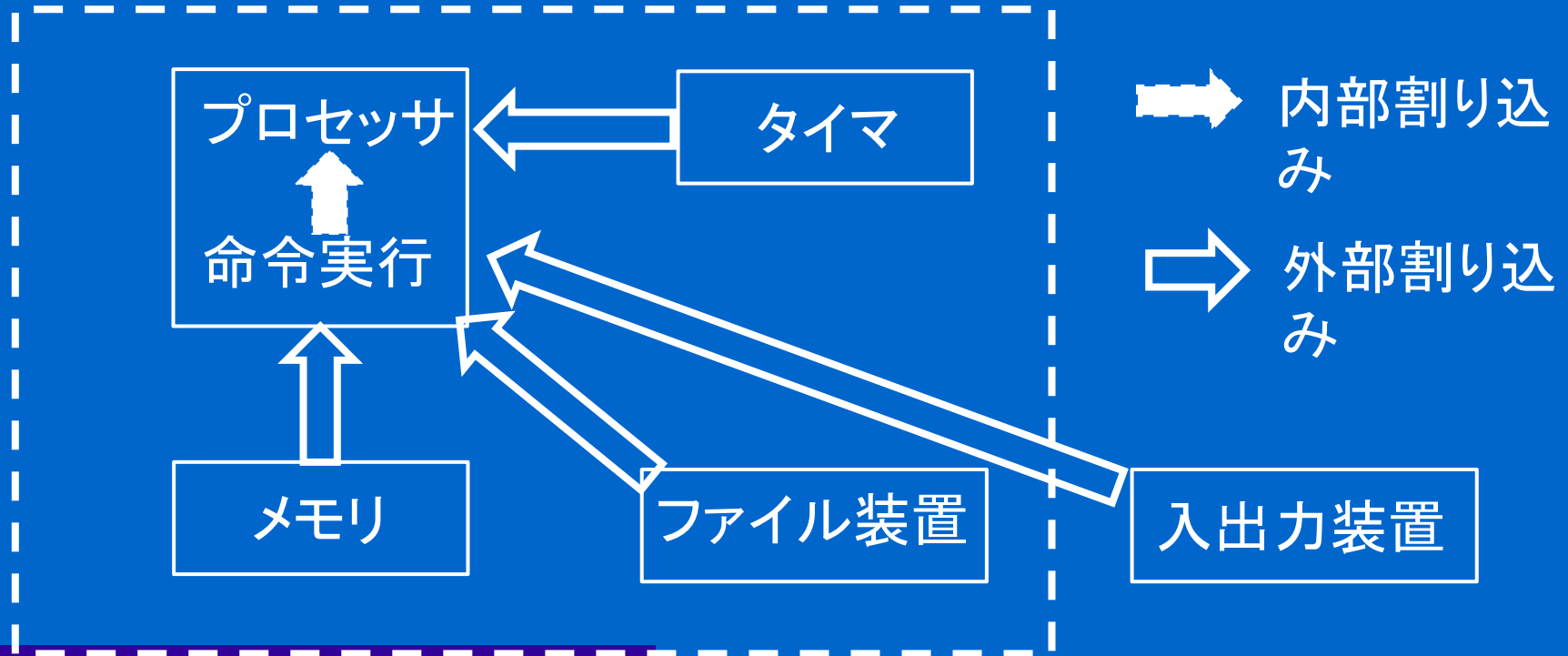
- 割り込み機構および機能
 - マシン命令の実行フローを、不測の事態の発生時に強制的かつ動的に変更する手段
 - 割り込みの原因となる不測の事態を「**割り込み要因**」という
 - 割り込みの発生：割り込み要因が生じたことで、実行中のプログラムを一時中断して、割り込み機構の動作を開始(図1.17)
 - 割り込み処理：割り込み要因ごとに決められた処理

割り込み要因の分類

- **内部割り込み**(ソフトウェア割り込み)
 - 暗黙的(例: 命令実行例外)
 - 不測の事態に伴って発生する割り込みを、**例外(exception)**と呼ぶ(0除算、不正なアドレス参照)
 - 明示的(例: **システムコール(SVC)**, ブレークポイント)
- **外部割り込み**(ハードウェア割り込み)
 - 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

「内部割り込み」と「外部割り込み」 の違い

- 何に対する内部と外部か？
 - コンピュータ(本体)の内・外ではない
 - プロセッサの内部と外部である



マシン命令の実行との関係

- 内部割り込み

- マシン命令の実行に合わせて発生(実行に「同期」して発生)

- 外部割り込み

- マシン命令の実行とは独立に発生(実行に対して「非同期」に発生)

外部割り込み

任意の時
点で発生

LD GR0, DATA1 アドレスが不正なら割り込み
DIV GR0, DATA2 除数が0なら割り込み
ST GR0, DATA3
SVC ADR1 システムコール(割り込みの一種)

内部割り込み