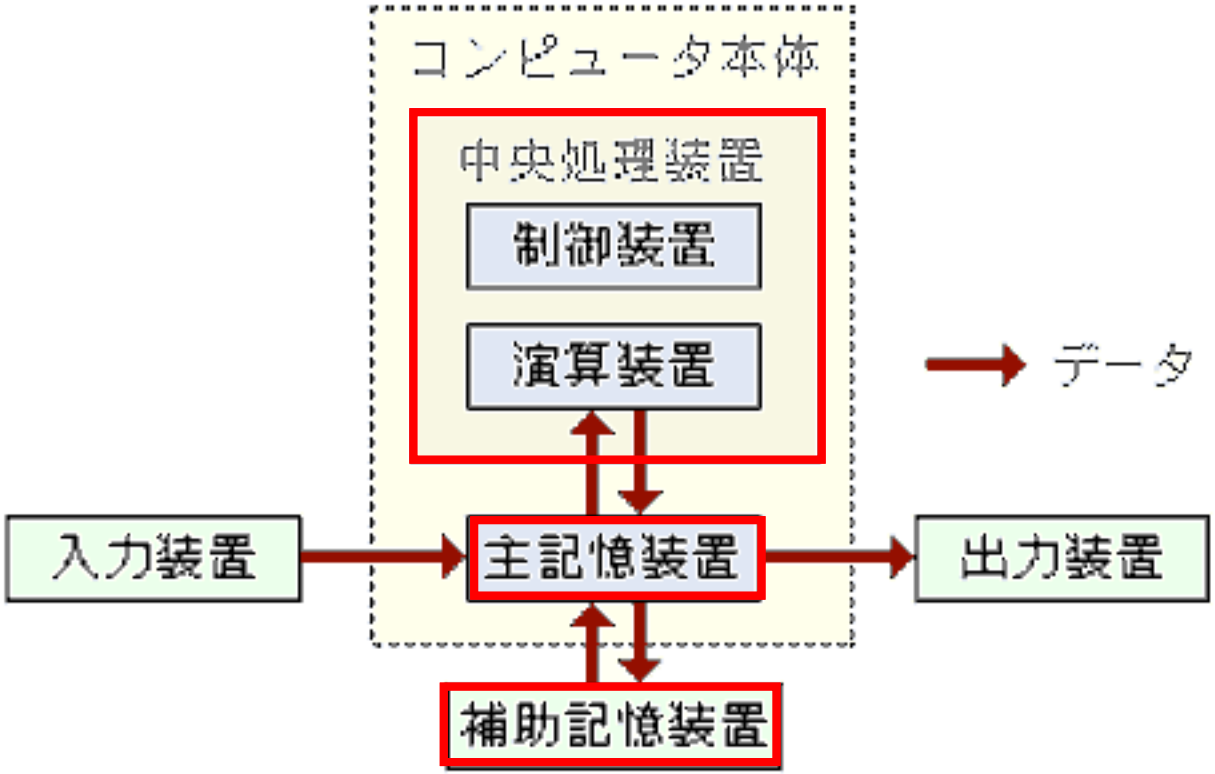
システムソフトウェア講義の概要

1. 計算機システムの復習：中央演算処理装置(CPU)，プログラムの実行，主記憶装置，補助記憶装置
2. オペレーティングシステムとは：CPU，主記憶装置，補助記憶装置などの抽象化
3. CPUの抽象化１：スレッドとプロセス，割り込み
4. CPUの抽象化２：CPUの割り当てアルゴリズム
5. 主記憶の抽象化：アドレス空間と仮想記憶
6. 補助記憶装置の抽象化：ファイルシステム
7. 演習問題
8. プログラミングシステムの概要，文法とそのクラス，字句解析と正規文法
9. 正規表現からの非決定性オートマトンの生成、決定性オートマトンへの変換
10. 字句解析用オートマトン生成ソフトウエアの実際
11. 構文解析と導出，文脈自由文法の構文解析法：LL構文解析
12. 文脈自由文法の構文解析法：LR構文解析
13. コンパイラ-コンパイラと構文解析の実際
14. 演習問題
15. 講義の総括と試験

計算機のリソースとは



1. CPU
2. 主記憶
3. 補助記憶装置

の抽象化

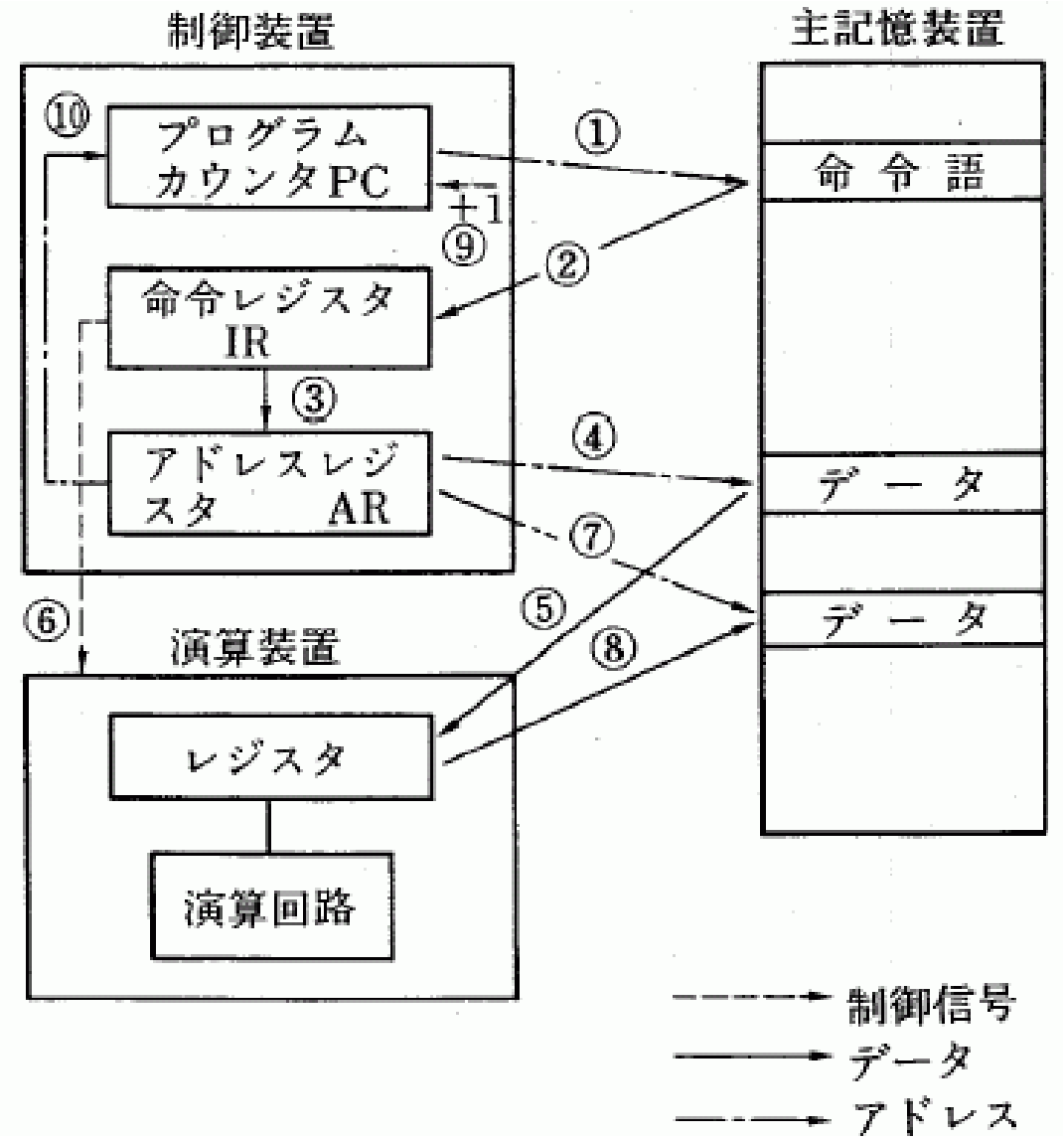
CPUの構造と基本動作（復習）

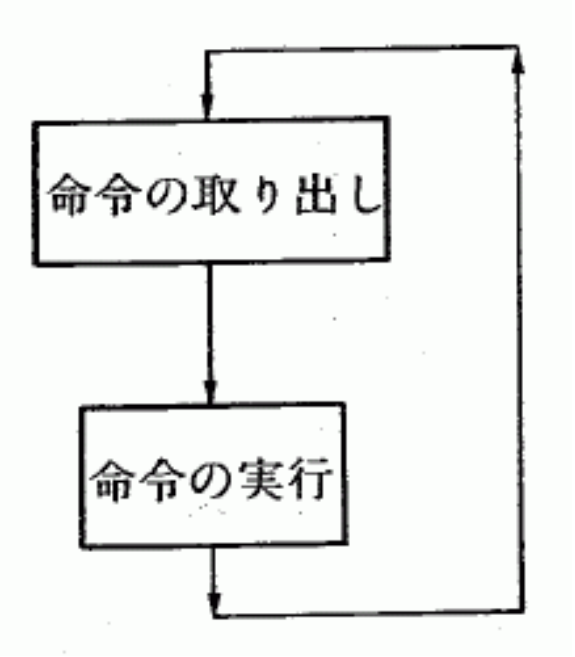
パイプライン

条件分岐

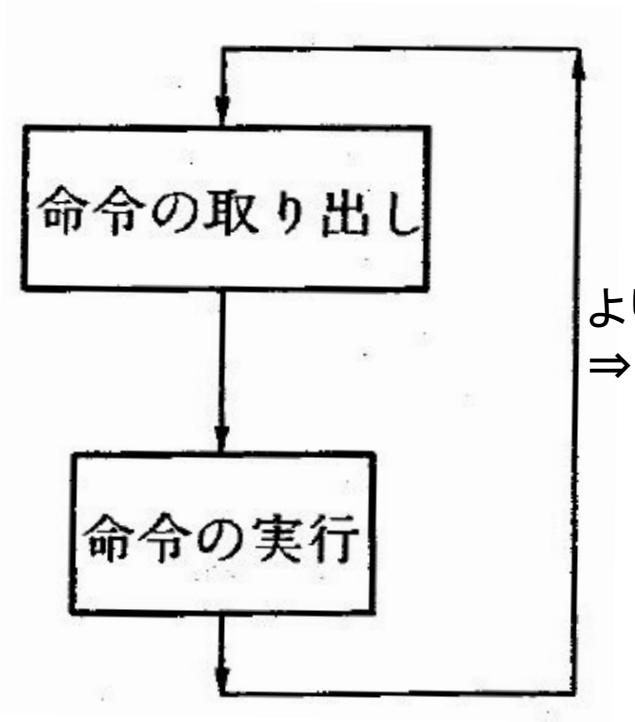
割り込み

命令の実行過程


制御装置
主記憶装置
⑩
プログラム
カウンタPC
+1
⑨
①
②
命令レジスタ
IR
③
アドレスレジスタ
AR
④
⑦
⑤
⑧
⑥
命令語
データ
データ
演算装置
レジスタ
演算回路
制御信号
データ
アドレス


命令の取り出し
命令の実行

命令の実行過程



より詳細には
⇒

命令の読み取り

命令の解読

データの読み取り

命令の実行

結果の書き戻し

各段階で動いている場所は違う

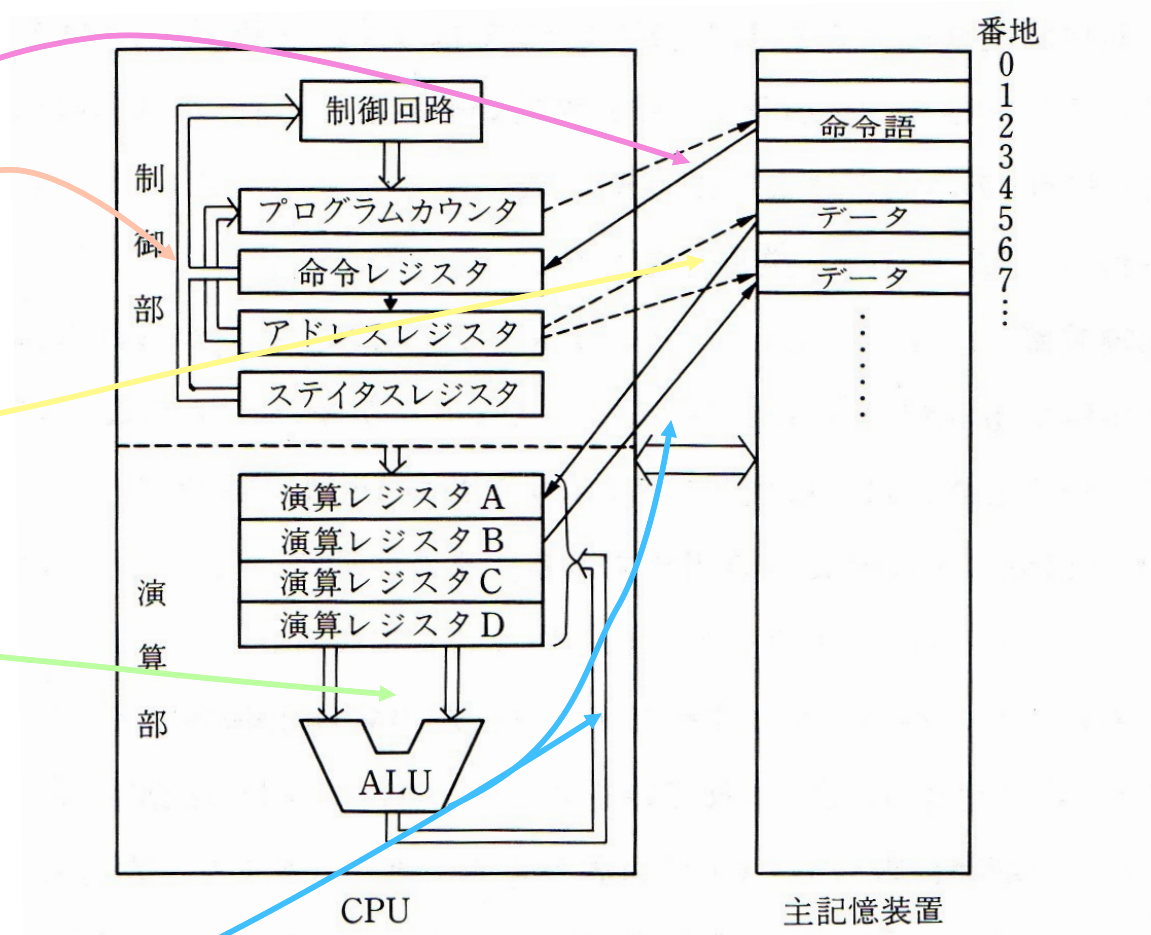
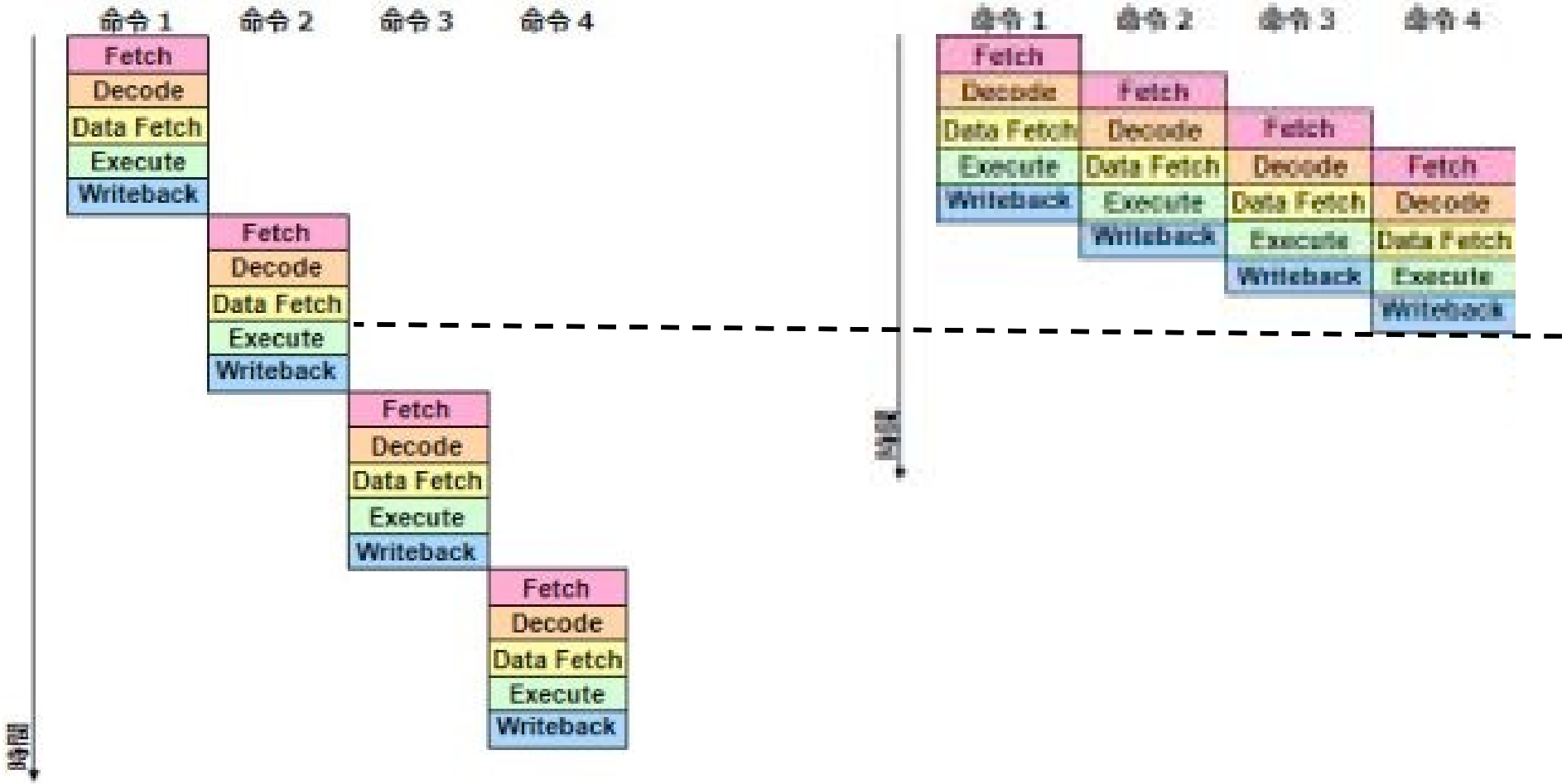


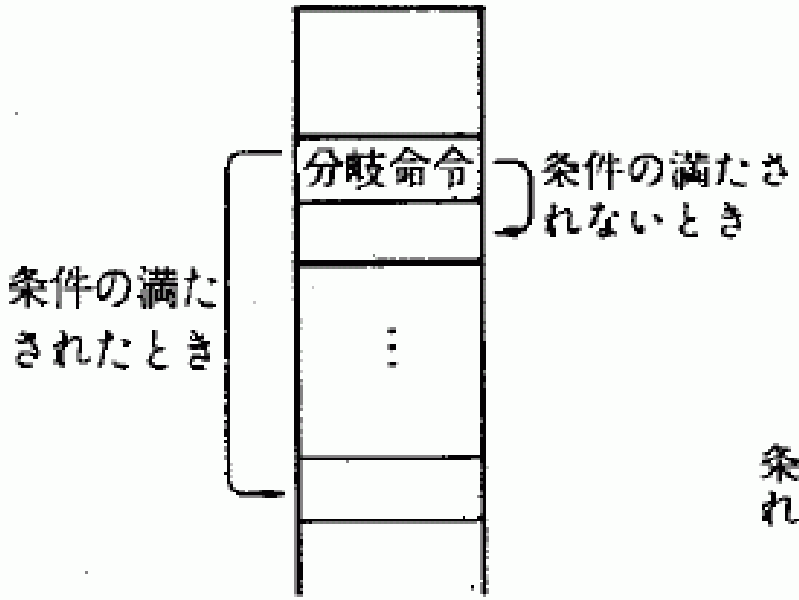
図 3.1　コンピュータの基本構造

パイプライン処理

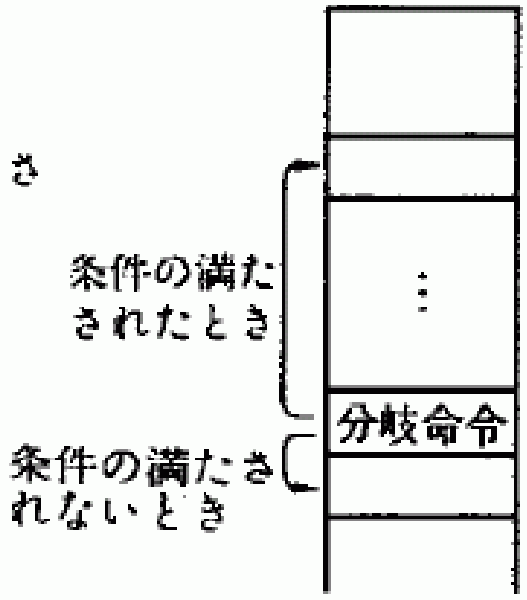


CPUの各部分を並列に動かす．異なる命令に関する処理が同時に進む．

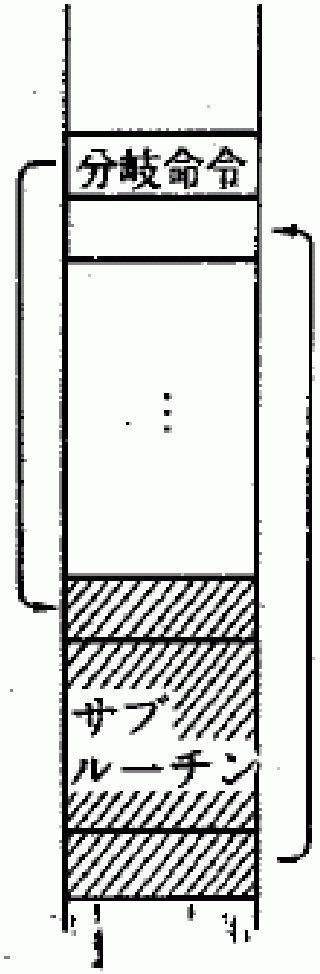
プログラムの制御構造



(a)　ジャンプ



(b)　ループ



(c)　サブルーチン

条件分岐がなければ. . .

- ある計算メカニズムが万能チューリングマシンと同じ計算能力をもつとき、その計算モデルはチューリング完全（Turing-complete）と呼ぶ.
- 条件分岐がなければチューリング完全とはならない.

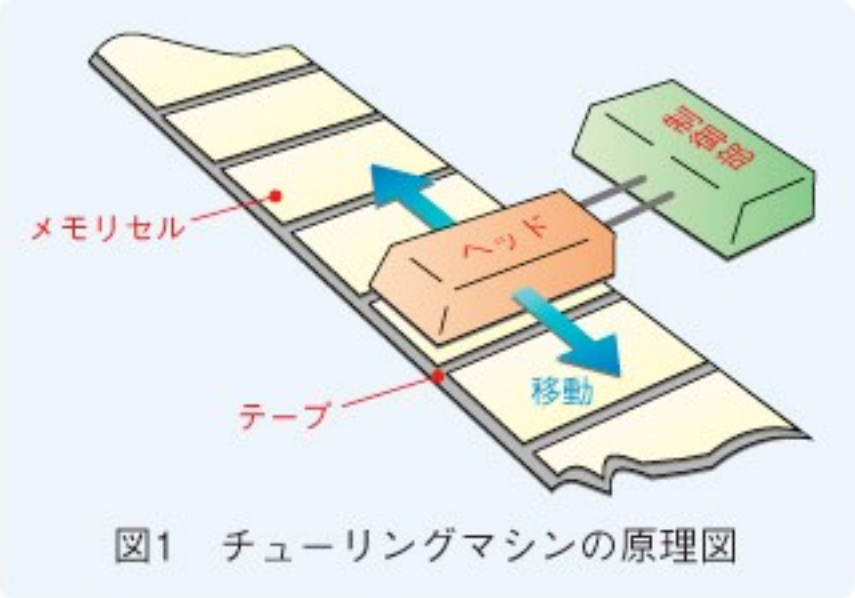
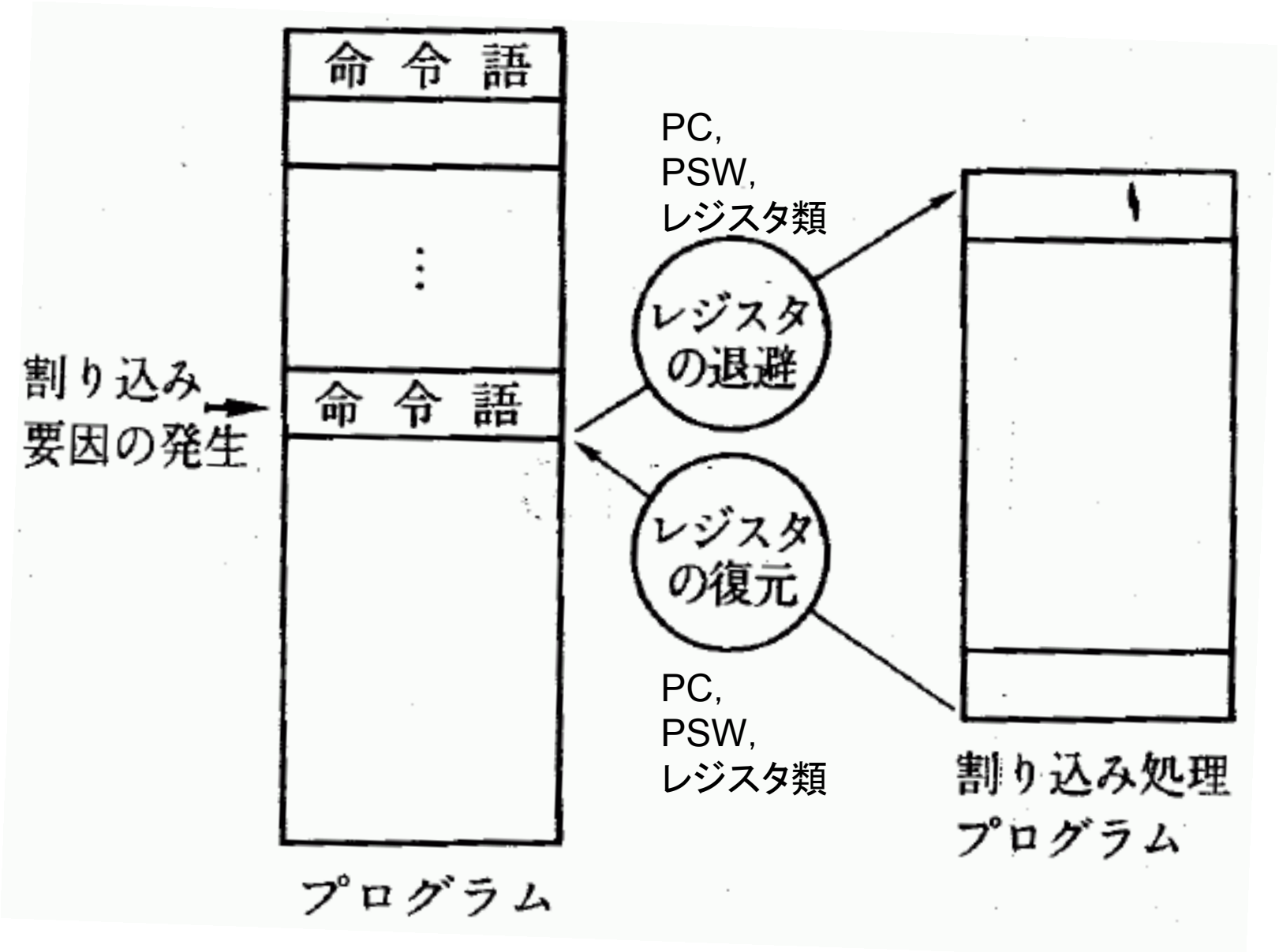


図1　チューリングマシンの原理図

【万能チューリングマシン】
任意のチューリングマシンの動作を再現するもの

割り込み


命令語
命令語
割り込み
要因の発生
プログラム
PC,
PSW,
レジスタ類
レジスタ
の退避
レジスタ
の復元
PC,
PSW,
レジスタ類
割り込み処理
プログラム

割り込みはどの様な時に使われる？

- 外部イベントの検出：

  例：ボタンが押された，HDDに対するデータの転送が完了した，外部から通信メッセージが届いた．等

  外部イベント検出の方式

  –ポーリング　イベントが起きていないかどうかをCPUが常に監視する

  –割り込み　　CPUに対して割り込み信号が入ることにより，通常の処理を中断して割り込み処理に移る．割り込み処理後は元の処理を再開する．

ポーリングv.s.割り込み

- 御用聞きに回る三河屋と，電話で注文を受けて作って配達するPIZZA屋さん．どっちが賢い？
  - ポーリングはCPUの無駄遣い．
  - 割り込みは，発生時までCPUを別用途で使える．

CPUの高度な動作

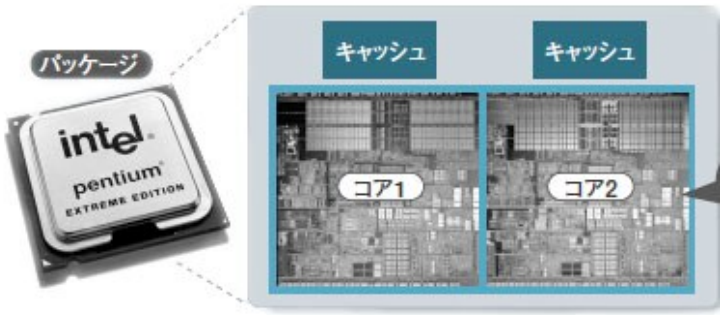
並行処理: Concurrent Processing

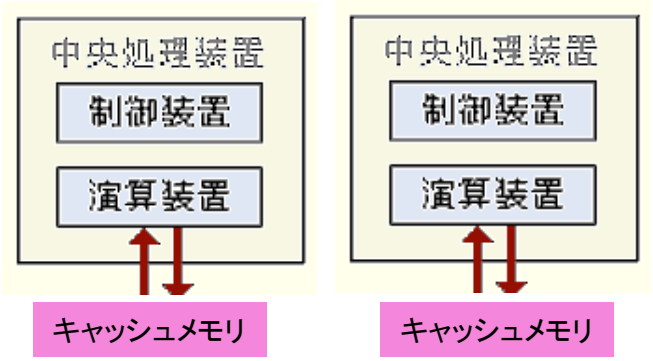
並列処理:Parallel Processing

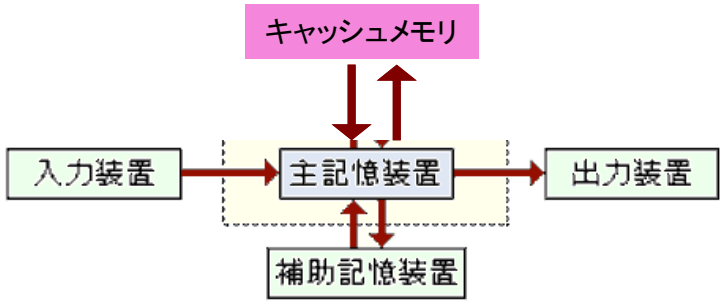
複数の仕事を「同時」にこなす

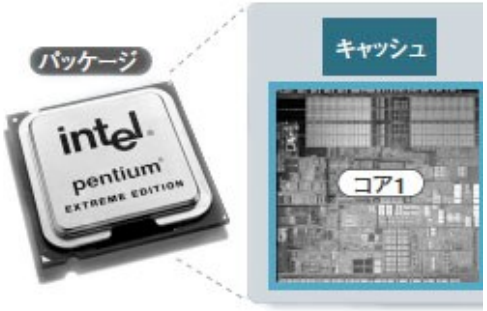
- 並列処理
  - CPUが複数あり，それらが同時に動作する (SMP:対称型並列処理＝メモリ共有型並列計算機による)
    - CPUのコア（制御部と演算部のペア）が複数存在する場合(マルチコア)　後述するキャッシュメモリは共通である
    - 制御部のみが複数存在する場合（Hyper Threading）この場合も，OSは複数のCPUを検出する.
- 並行処理
  - 時分割処理システム(TSS:Time Sharing System)によって一台の計算機で複数の処理が行えるようになっている.

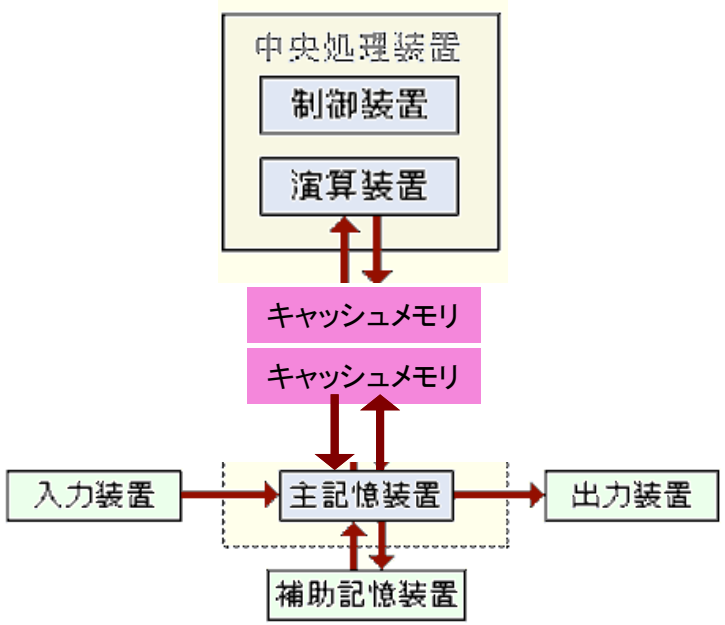
マルチコア


パッケージ
intel
pentium
EXTREME EDITION
キャッシュ
キャッシュ
コア1
コア2


中央処理装置
制御装置
演算装置
中央処理装置
制御装置
演算装置
キャッシュメモリ
キャッシュメモリ

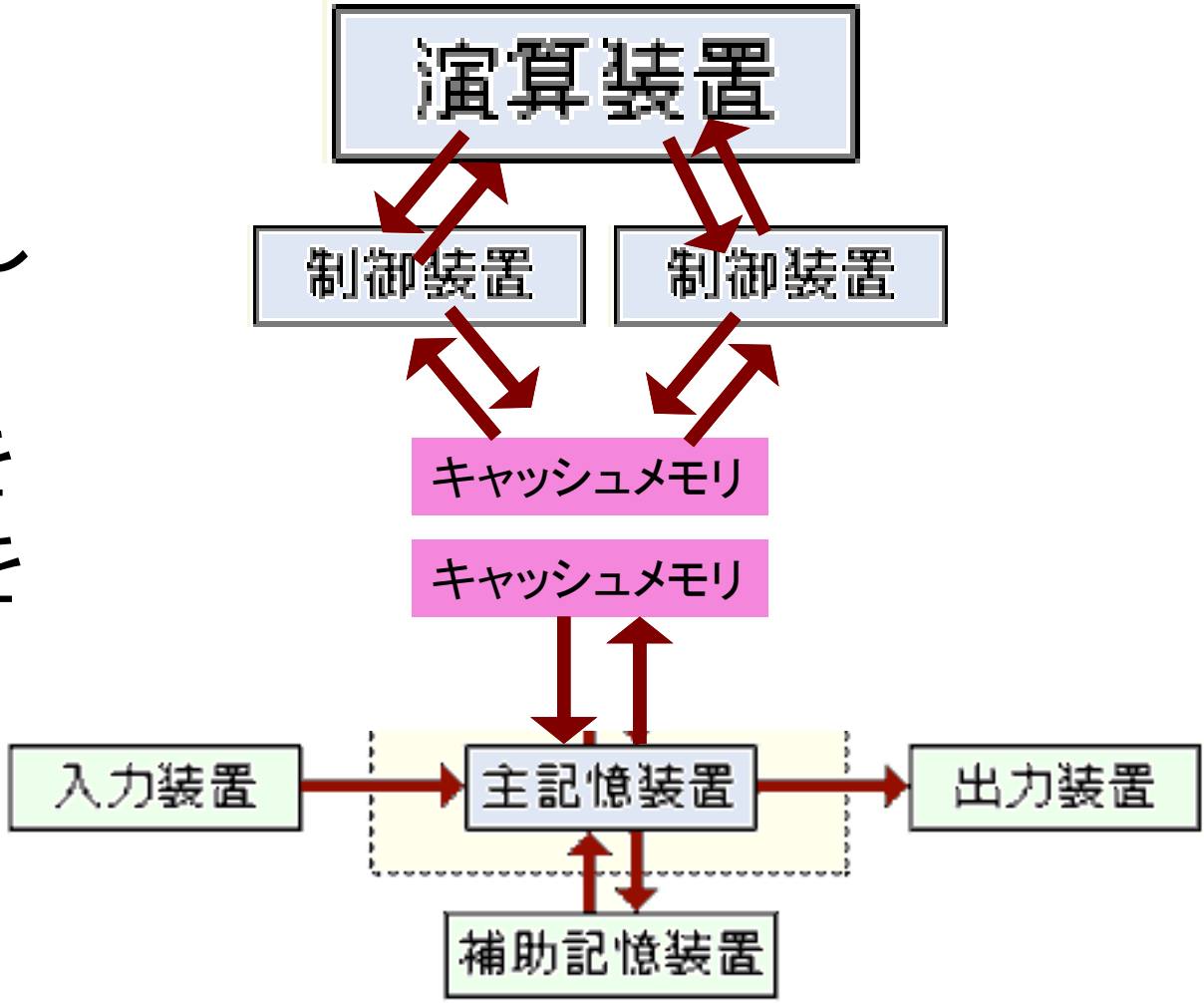

キャッシュメモリ
入力装置
主記憶装置
出力装置
補助記憶装置


パッケージ
intel
pentium
EXTREME EDITION
キャッシュ
コア1

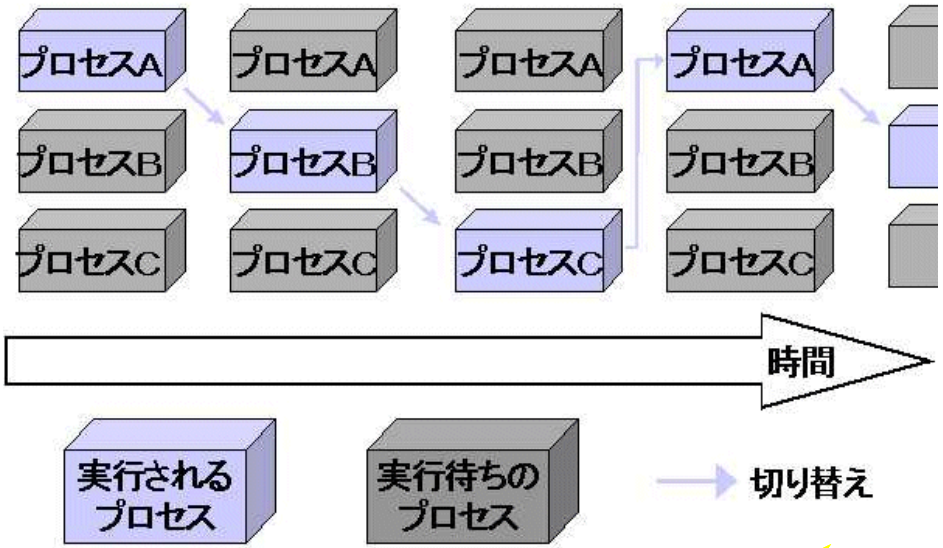

中央処理装置
制御装置
演算装置
キャッシュメモリ
キャッシュメモリ
入力装置
主記憶装置
出力装置
補助記憶装置

Hyper Threading

- 命令読み取りの部分だけ二重化し,
- 空いているALUを使って並列処理を行う.
- OSから見るとCPUは二つに見える



1つのCPUで並行処理を行う.



単一のCPUでも，時間を分割して複数のプログラムを実行させることができる.

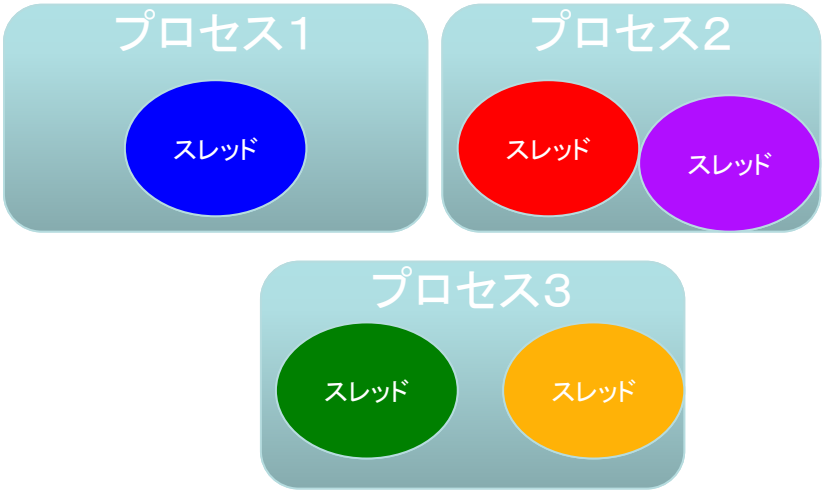
- プロセスとは，実行中のプログラムのこと．タスクとも呼ぶ.
- 同じプログラムでも，異なる実行であれば，プロセスとしては異なる.
- スレッドとは異なり，OSから記憶領域やディスク資源を割り当てられて実行される.
- プロセスが停止状態でも入出力は行える.

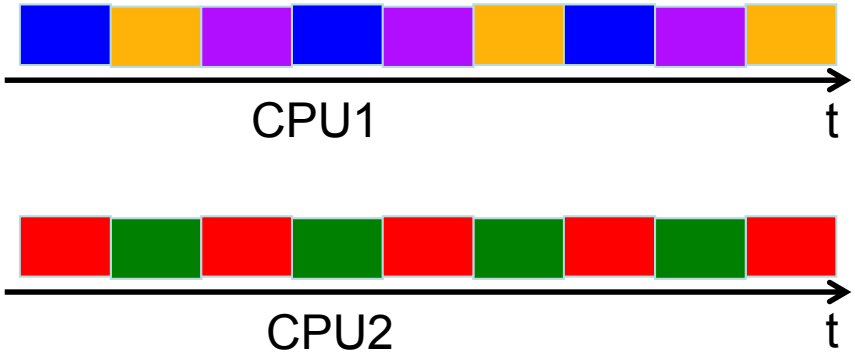
これが時分割処理システム
（Time Sharing System）

ここからOSの話

実際の時分割処理はプロセスではなくスレッド単位で行われる.

- プロセスとスレッドの対応関係は，プログラムによって決められる.
- スレッドとCPUの対応関係はスレッド起動時にOSが決める.
- 各スレッドが時分割で動作する.





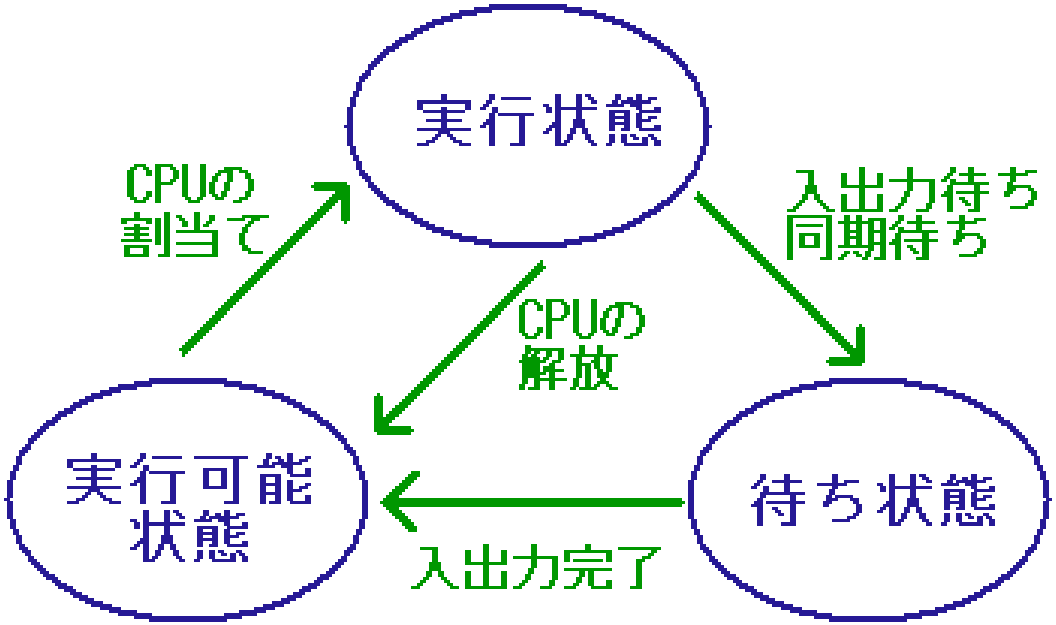
OS内での計算の実行主体は仮想化されたCPU

- 仮想CPU上で実行されるプログラムのことをプロセスまたはタスクと呼ぶ.
  - プロセスには1つ以上のスレッドが対応付けられている.
  - 同じプログラムでも，異なる実行であれば，プロセスとしては異なる.
  - スレッドとは異なり，OSから記憶領域やディスク資源を割り当てられて実行される.
  - スレッドが停止状態でも入出力は行える.

スレッドの状態

1. 実行状態のスレッドは，一定時間CPUを割り当てられるとCPUを解放しなければならない.
2. 実行状態のスレッドが入出力待ちになったときはCPUを解放して待ち状態になる. この場合OSは，実行可能スレッドの中からCPUを割当てる.
3. 入出力待ちのスレッドの入出力が済めば，実行可能状態に遷移し，CPUの割当てを待つ.

CPUの割り当てと解放は，後述するスケジューリングアルゴリズムに従う.



プロセスの特性

- プロセスには負荷の違い，反応の速さが問題となるものと継続的な計算が必要なものなど，様々な種類がある.
  - 反応の速さが問題となるもの：マウスやキーボードのイベントを拾うプロセス
  - 継続性が重要なもの：ファイル検索用インデックスをバックグラウンドで作成するプロセス
- これらの特性にあわせてスレッドのスケジューリングアルゴリズムを考えなければならない.

スケジューリング

- スケジューリングとは，各スレッドに優先順位を与え，優先順位の高い順にCPUへの割り当てを行う処理である．実際には「優先度付きキュー」が用いられる．
- 必ずしもTSS（時分割処理）を前提としない．
- 複数のCPUが存在する場合は，優先順位の高い順にCPUへの割り当てを行う．
  - 先着順
  - 時間順
  - ラウンドロビン
  - 締め切り順
  - レート・モノトニック

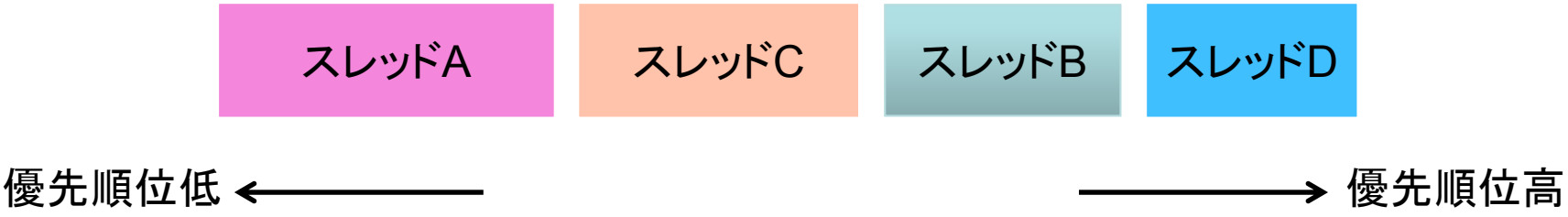
先着順:FCFS
First Come First Serve

- スレッドA,B,C,Dがこの順に実行可能になった場合

こちらから
CPUに割当
られる

時間順：SPTF
Shortest Processing Time First

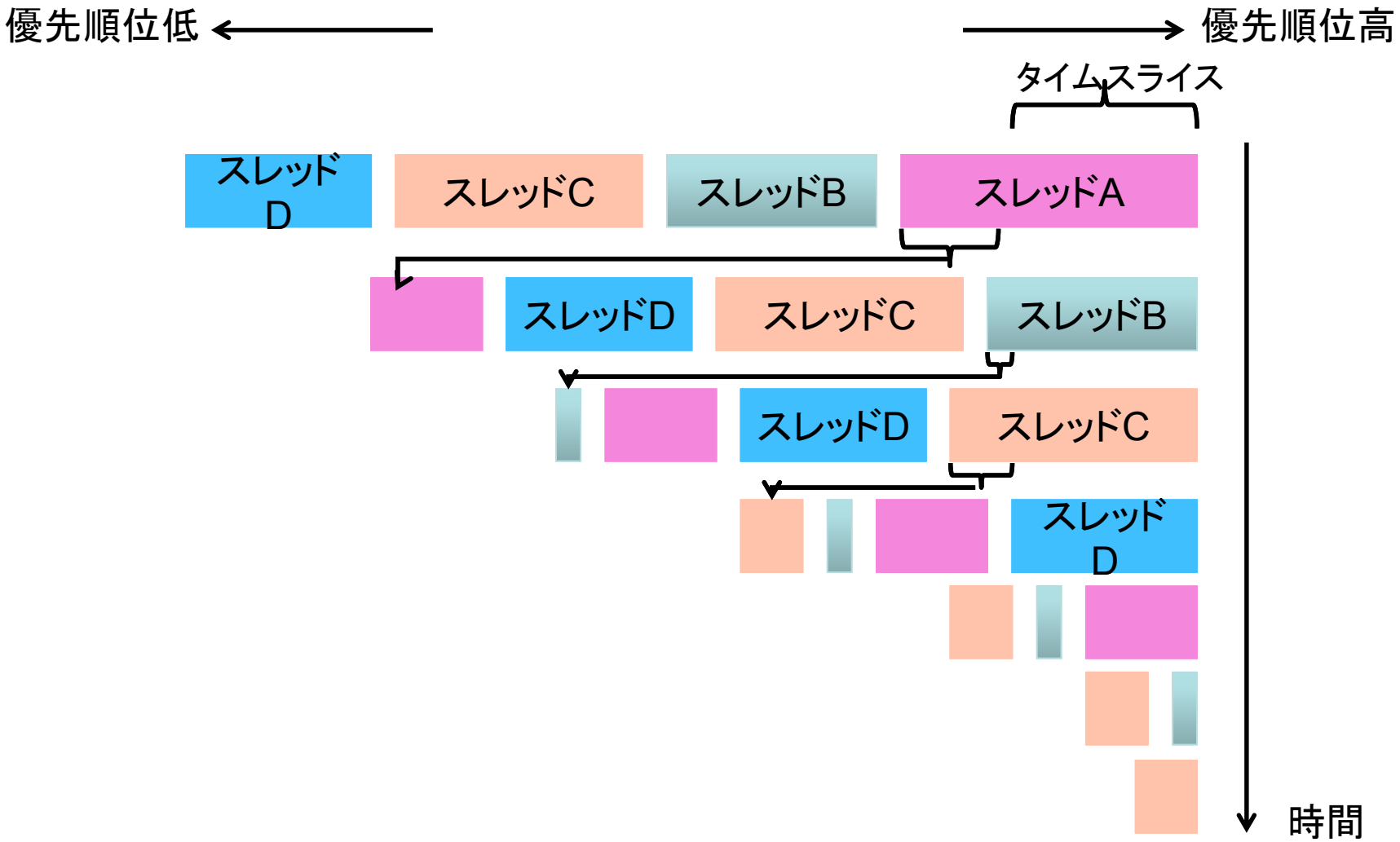
- スレッドD,B,C,Aがこの順に実行時間が短い場合



- スレッドの平均応答時間を最小にする
- 各スレッドの実行時間があらかじめ分かっていなければならない.

ラウンドロビン：RR
Round Robin

- CPUが1つの場合



実時間スケジューリング Real-time Scheduling

- 締め切り順(EDF: Earliest Deadline First)

各スレッドの終了までの時間（締め切り）が短い順に優先順位を上げる.

- レート・モノトニック(RMS: Rate Monotonic Scheduling)

処理が起動される頻度が高いスレッドに高い優先順位を与える.

スレッド・スケジューリングの例

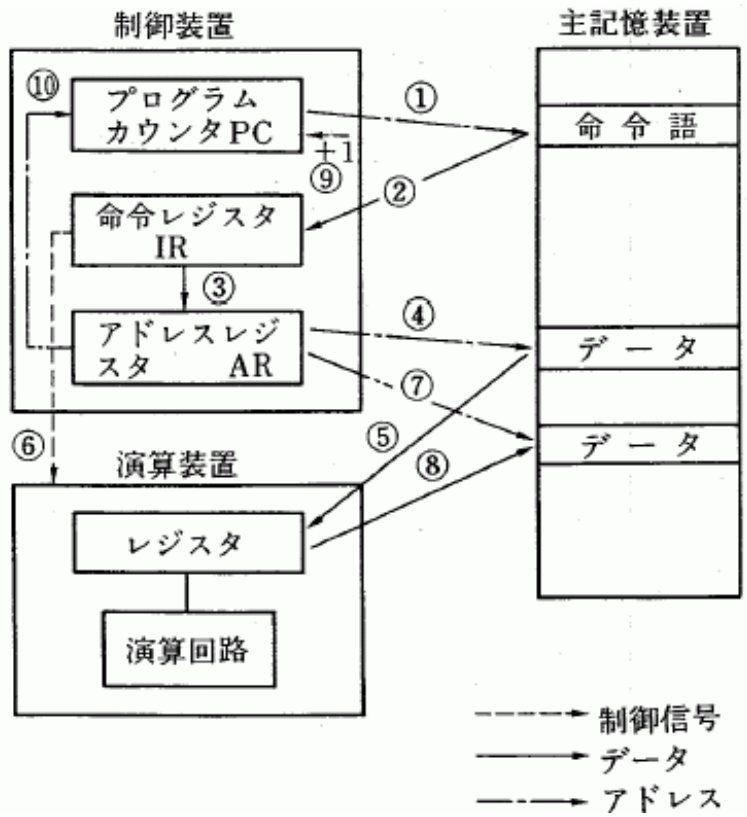
	スレッドの属性			スケジューリング・アルゴリズム			
	実行可能になった時刻	残余実行時間(R)	処理完了締め切り時刻(D)	FCFS	SPTF	RR	EDF（D-R順）
スレッドA	1	6	18	1	4	1	2
スレッドB	3	8	30	2	5	2	4
スレッドC	4	1	35	3	1	3	5
スレッドD	7	2	13	4	2	4	1
スレッドE	9	4	25	5	3	5	3

時刻10でスケジューリングが行われた場合の結果

優先順位：小さいほど優先順位は高い

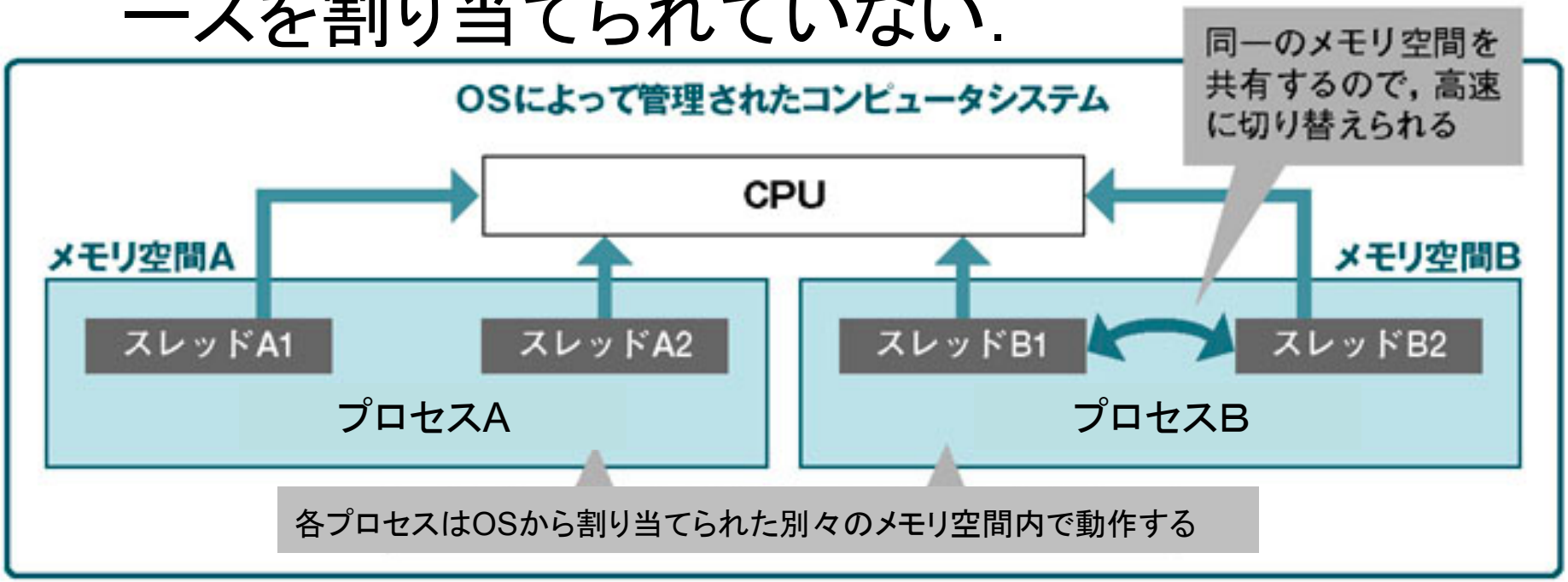
スレッドの切り替え:ディスパッチャ

- スレッドの文脈(コンテクスト)
  - PC: Program Counter
  - PSW: Processor/(Program) Status Word
  - レジスタ類
- 割り込みによってこれらを退避/復帰させる
- これによって1つのCPUが複数のスレッドを時分割実行できる



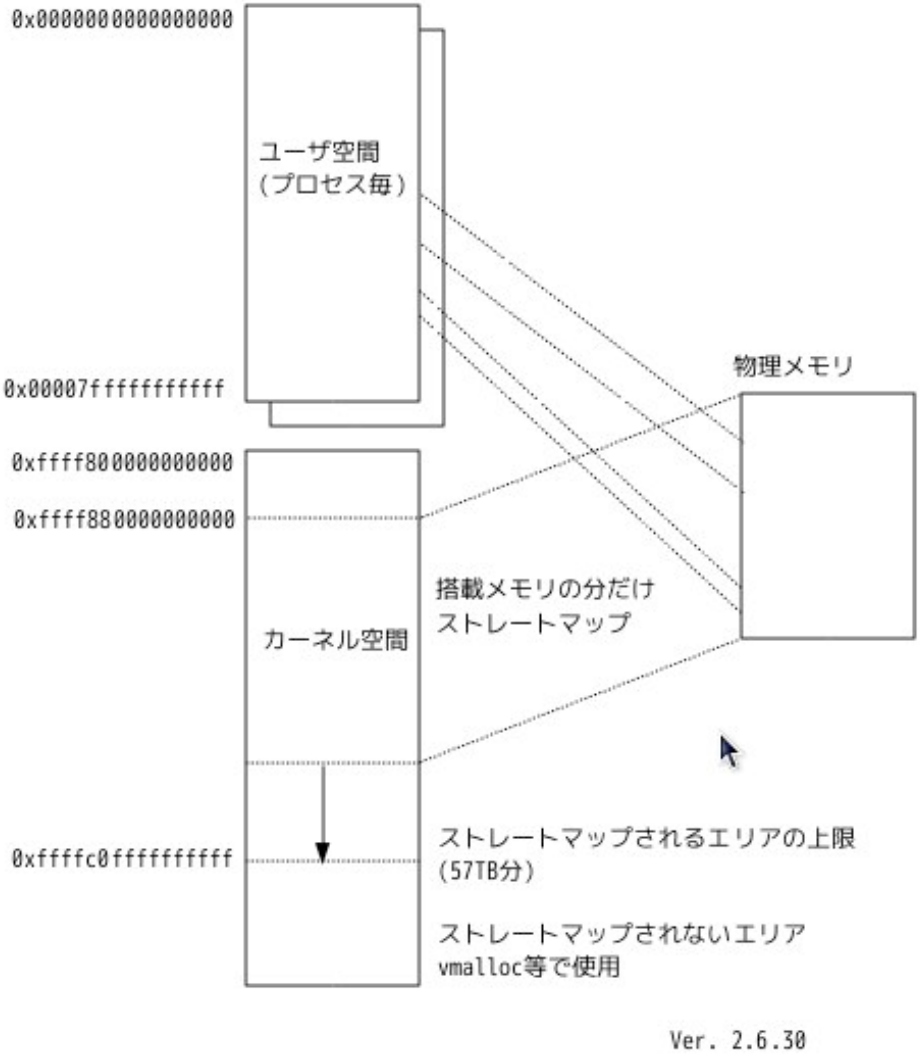
スレッドとプロセスの関係

- プロセスはOSから割り当てられた様々なリソースを持っている.
- スレッドは，プロセス内で動き，OSからはリソースを割り当てられていない.



プロセスに割り当てられたリソースと属性

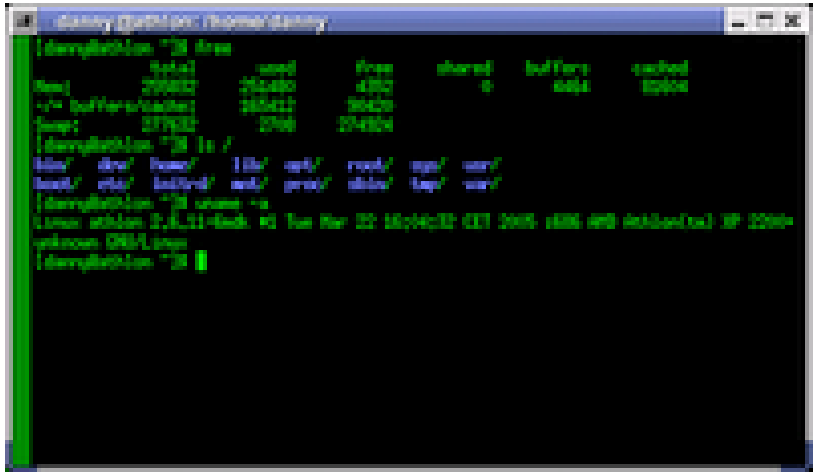
- リソース
  - ユーザ・アドレス空間
  - オープンしたファイル
  - ネットワークソケット
  - 他
- 属性
  - 所有者
  - 優先順位
  - 実行時間
  - 他



UNIXのプロセス

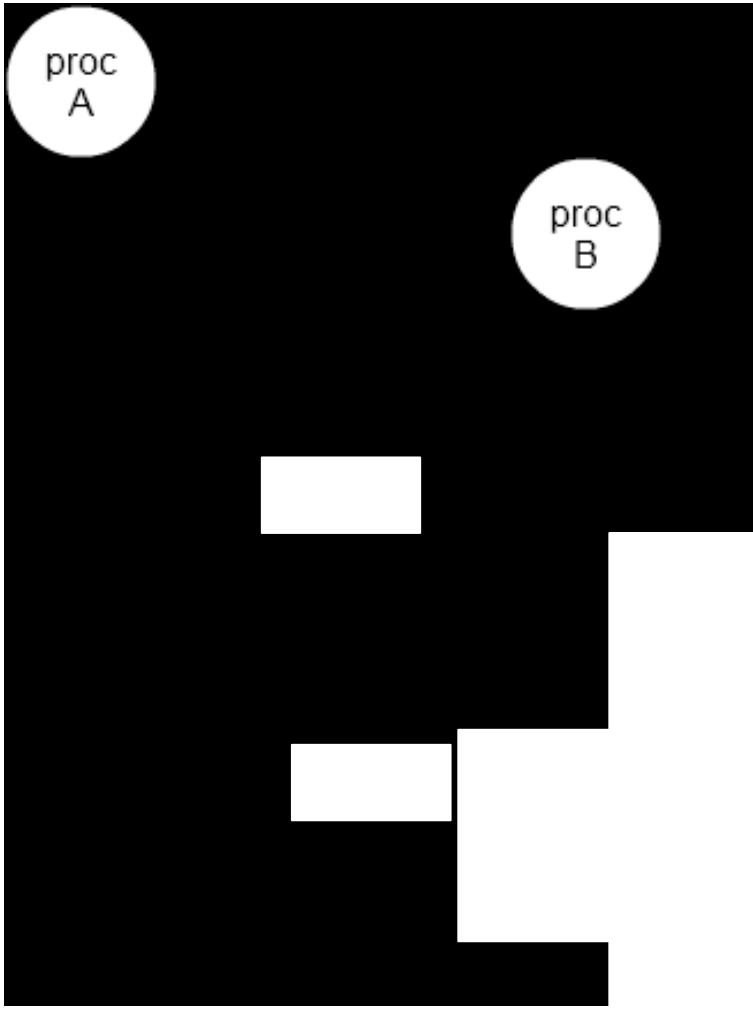
UNIXにおけるプロセスの操作

- 端末のウインドウも，その中で動いているシェルもプロセスである．
- コマンド ’ls’ を打てば，実行可能ファイル/bin/lsを読み込んだプロセスが発生し，シェルのプロセスはこのlsのプロセスの終了を待つ．終了後，lsのプロセスのリソースを開放し，シェルのプロセスが動き出す．



UNIX:プロセスの生成と停止のためのシステムコール

- fork:プロセスの分身を作る
- exec:プロセスの実行イメージを変更する.
- wait:プロセスを休眠させ,子プロセスのexitシステムコールを待つ.
- exit:プロセスのリソースの解放と停止



psで実行中プロセスが見える

- UNIXでは，プロセスには親子関係がある.
- psコマンドで調べると，PIDとPPIDという2つの番号が見える.
- PIDはProcess ID. PPIDはParent Process IDつまり，親のプロセスID

```
 UID   PID  PPID   C     STIME TTY          TIME CMD
……….
   0  2219  2217   0   0:00.37 ttys000    0:01.09 login -pf twada
 501  2220  2219   0   0:00.17 ttys000    0:00.21 -bash
   0  4577  2220   0   0:00.00 ttys000    0:00.00 ps -fU twada
……
```

topコマンドでも実行中プロセスやスレッドを観測することが出来る

```
Processes: 78 total, 2 running, 76 sleeping, 405 threads                13:46:12
Load Avg: 0.50, 0.73, 0.59  CPU usage: 10.0% user, 14.80% sys, 75.20% idle
SharedLibs: 4544K resident, 3540K data, 0B linkedit.
MemRegions: 58428 total, 670M resident, 16M private, 470M shared.
PhysMem: 563M wired, 948M active, 484M inactive, 1996M used, 51M free.
VM: 163G vsize, 1041M framework vsize, 4482822(4) pageins, 551919(0) pageouts.
Networks: packets: 1849247/716M in, 617318/103M out.
Disks: 1347448/28G read, 971221/34G written.

PID   COMMAND      %CPU      TIME     #TH   #WQ   #POR #MREGS RPRVT  RSHRD  RSIZE
4674  screencaptur 0.4       00:00.13 2     1     41   84     504K   6184K  2844K
4589  top          19.8      08:12.90 1/1   0     25   36     1448K  340K   2044K
4450  plugin-conta 0.0       01:04.06 9     1     188  807    18M    38M    35M
4013  Software Upd 0.0       00:14.92 7     2     166  260    18M    7400K  14M
4008  suhelperd    0.0       00:00.16 2     1     38   47     176K   1400K  2700K
3923  iCal         0.0       00:21.92 3     1     126  219    12M    24M    18M
3448- DashboardCli 0.0       00:05.87 8     2     137  293    692K   6628K  1440K
3091  mdworker     0.0       01:10.64 3     1     66   100    25M    4432K  33M
2220  bash         0.0       00:00.21 1     0     17   25     200K   244K   520K
2219  login        0.0       00:01.08 1     0     22   57     0B     340K   0B
2217  Terminal     0.8       00:37.34 5     1     123  139    4480K  25M    20M
1358  AppleSpell   0.0       00:00.79 2     1     34   49     60K    1172K  144K
1355  Dictionary   0.0       00:14.28 8     2     134  312    3908K  17M    7816K
632-  Microsoft Ex 0.3       08:24.41 5     1     135  611    4936K  28M    6276K
```

UNIX:プロセスに対する操作

- プロセスに対する操作（割り込み）
  - 一時停止： SIGSTOP
  - 再開: SIGCONT
  - 停止
    - ハングアップ　SIGHUP
    - 一時割り込み SIGINT
    - 強制停止　　SIGKILL
    - バスエラー　　SIGBUS
    - . . .
- 割り込み後の実行はシグナルの種類で変わる.

実例

prompt$ (sleep 30; echo “End”)& 30秒待って”End”を表示させる

[1] 4695

prompt$ kill -STOP 4695 PID4695にSIGSTOPを送信

[1]+ Stopped (sleep 30; echo "End")

prompt$ kill -CONT 4695 PID4695にSIGCONTを送信

prompt$ End 実行を再開し”End”を表示

[1]+ Done (sleep 30; echo "End")

prompt$

まとめ

- CPUの動作，マルチコアCPU，Hyper Threading,プロセス，スレッド
- 割り込みを使ったスレッドの時分割処理
- スケジューリングアルゴリズムとディスパッチャ
- プロセスのリソース
- UNIXでのプロセスの生成と制御

問題1

	スレッドの属性			スケジューリング・アルゴリズム			
	実行可能になった時刻	残余実行時間(R)	処理完了締め切り時刻(D)	FCFS	SPTF	RR	EDF
スレッドA	1	9	20				
スレッドB	4	7	30				
スレッドC	6	1	25				
スレッドD	7	4	23				
スレッドE	8	3	23				

時刻10でスケジューリングが
行われた場合の結果

問題2

- UNIXのシェルのバックグラウンドジョブでは, fork, exec, wait, exitがどのように動作しているか.

問題3

- なぜプロセスごとに異なるアドレス空間が割り振られるのか考えを述べよ.