# Pertemuan 1: Review Organisasi Komputer

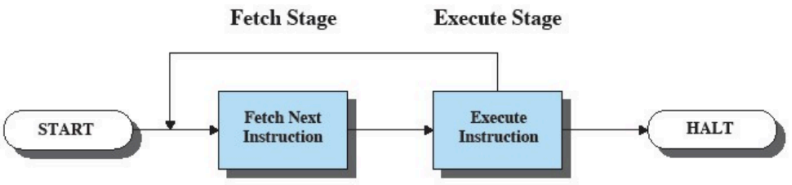
## I. Elemen Dasar Komputer

Ada 4 elemen utama komputer yaitu

1. Processor → CPU
   Processor memiliki beberapa komponen seperti
   a. MAR (Memory Address Register)
      Menyimpan **alamat** dari memori yang akan ditulis atau dibaca selanjutnya
   b. MBR (Memory Buffer Register)
      Menyimpan **data** yang akan ditulis ke memori atau data yang diterima dari memori
   c. I/O AR(Input/Output Address Register)
      Menyimpan **alamat I/O Device** tertentu.
   d. I/O BR (Input/Output Buffer Register)
      Menyimpan **data** dari I/O ke processor dan memory
2. Main Memory/Real Memory/Primary Memory
   - Contohnya RAM
   - Bersifat *volatile* (Ketika komputer mati, data di main memory hilang)
3. I/O Modules
   - Contohnya HDD, SSD, FlashDisk, Keyboard, Camera
   - Penyimpanan I/O modules disebut Secondary Memory Devices dan bersifat *non-volatile*
4. System Bus
   - Menjembatani antara Processor, Main Memory, dan I/O Modules

## II. Instruction Execution

- Instruction Cycle
  Terdiri dari 2 instruksi, yaitu:
  1. Fetch → Fetch adalah proses processor membaca instruksi dari memory
  2. Execute →Execute adalah proses menjalankan perintah instruksi
     Bisa lihat gambar di bawah




- Loop instruksi ini berhenti ketika:
  1. Processor dimatikan
  2. Muncul error yang tidak dapat direcovery
  3. Ada program yang terminate Processor
- Istilah dalam Instruction:

1. Program Counter (PC) → Menyimpan **alamat** dari instruksi yang akan dibaca selanjutnya. PC selalu diincrement
2. Instruction Register (IR) → Menyimpan **instruksi** yang dibaca dari memory
3. Program Status Word (PSW) → Menyimpan **status informasi** seperti interrupt enable/disable, kernel/user mode
4. Condition codes or flags → bit-bit yang diatur oleh perangkat keras prosesor sebagai hasil dari operasi yang dilakukan. Hasilnya dapat berupa positif, negatif, nol, atau overflow
5. Accumulator (AC) → Penyimpanan data sementara pada processor

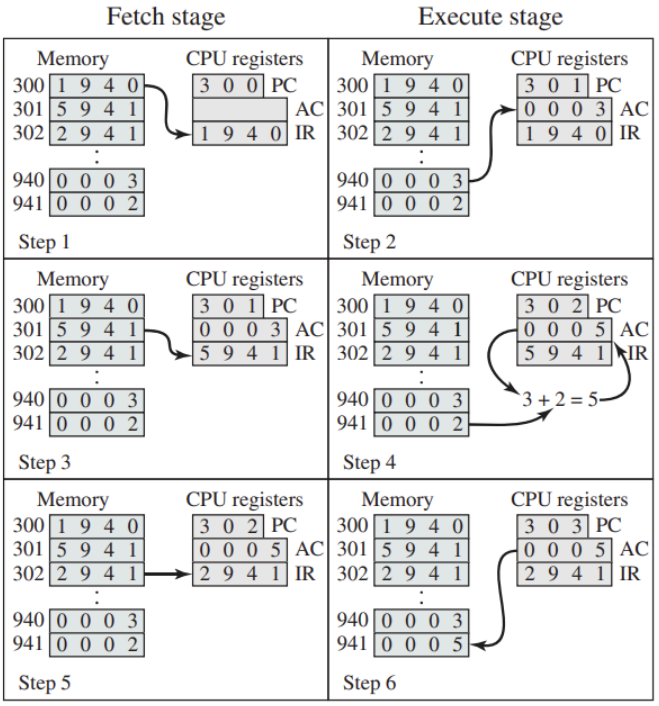
- Instruksi pada IR memiliki 4 kategori yaitu:
  1. Processor - memory → transfer data dari **processor ke memory dan sebaliknya**
  2. Processor - I/O → transfer data dari **processor ke I/O Device dan sebaliknya**
  3. Data Processing → **operasi aritmatika atau operasi logika** terhadap data
  4. Control → Instruksi yang menunjukkan bahwa **urutan eksekusi dapat diubah**. Contohnya instruksi pada alamat 149 menunjukkan bahwa instruksi selanjutnya ada pada alamat 200
- Suatu instruksi memiliki format Opcode dan address. Instruksi ini disimpan pada memory.


**(a) Instruction format**

Opcode → Berupa beberapa bit yang menyimpan instruksi yang akan dilakukan

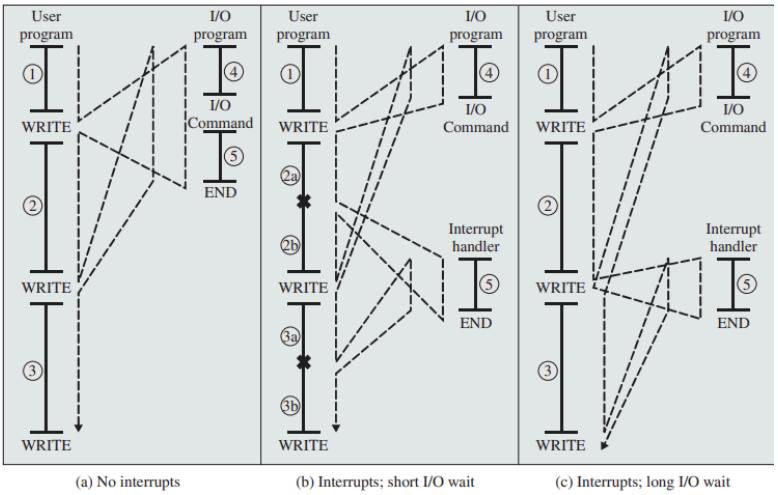
Address → Menyimpan alamat dari tujuan instruksi pada opcode

- Contoh instruksi

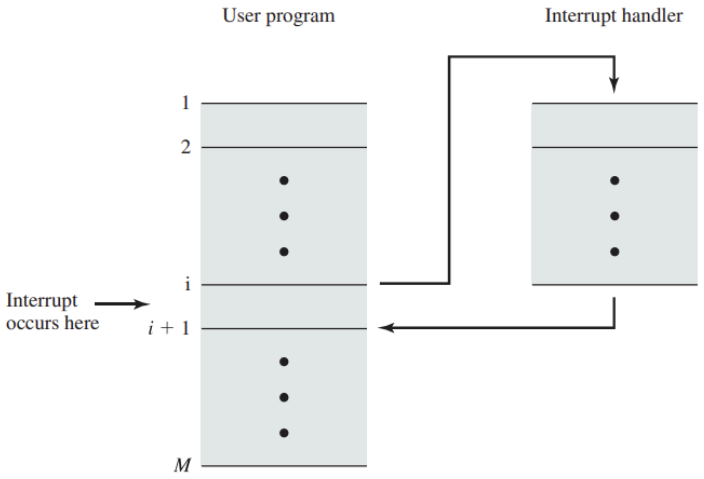



## III. Interrupts

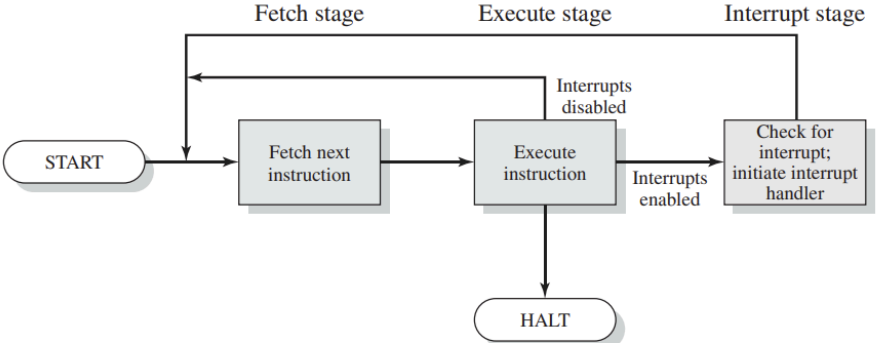
- Interrupts adalah cara untuk meningkatkan kerja processor. Interrupts adalah langkah untuk **mengubah proses atau tugas eksekusi dari satu tugas ke tugas lain**.
- Contoh kasus nyata interrupts adalah pada I/O Device, waktu input lebih lama daripada processor bekerja. Oleh karena itu, agar processor dapat bekerja lebih optimal maka dilakukan interrupts.
- Interrupts berdasarkan penyebabnya dibagi menjadi 4 yaitu
    a. Program → akibat kondisi dari eksekusi instruksi seperti overflow aritmatika, Division by zero, eksekusi hal yang tidak diperbolehkan oleh mesin, dan reference ke memori di luar yang diperbolehkan user
    b. Timer → Timer interrupt terjadi ketika timer internal pada processor menyentuh waktu tertentu. Hal ini untuk menjalankan fungsi pada *operating system*
    c. I/O → interrupt dari I/O device untuk memberi signal terjadi error atau signal operasi yang telah selesai
    d. Hardware Failure → Akibat kegagalan hardware seperti power failure atau memory parity failure
- Contoh interrupt pada I/O Program.



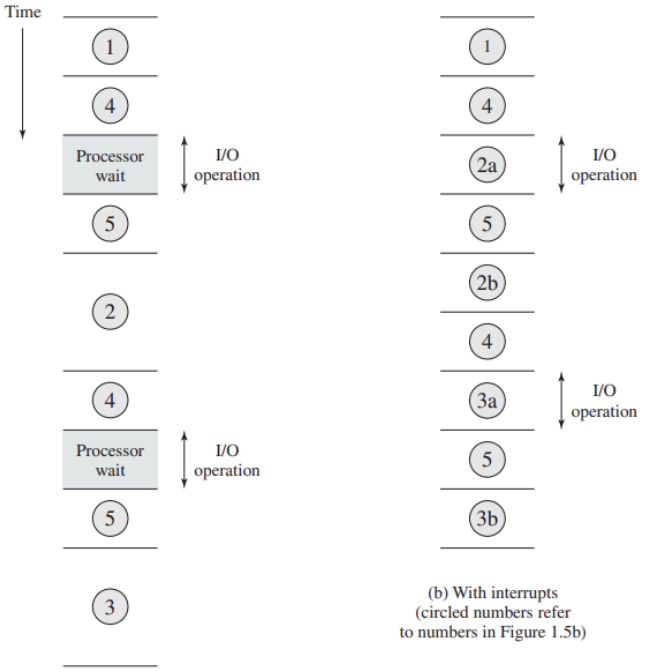

- Gambaran alur interrupt. Perhatikan bagian interrupt disebut **Interrupt handler**




- Gambar Diagram apabila ada interrupt



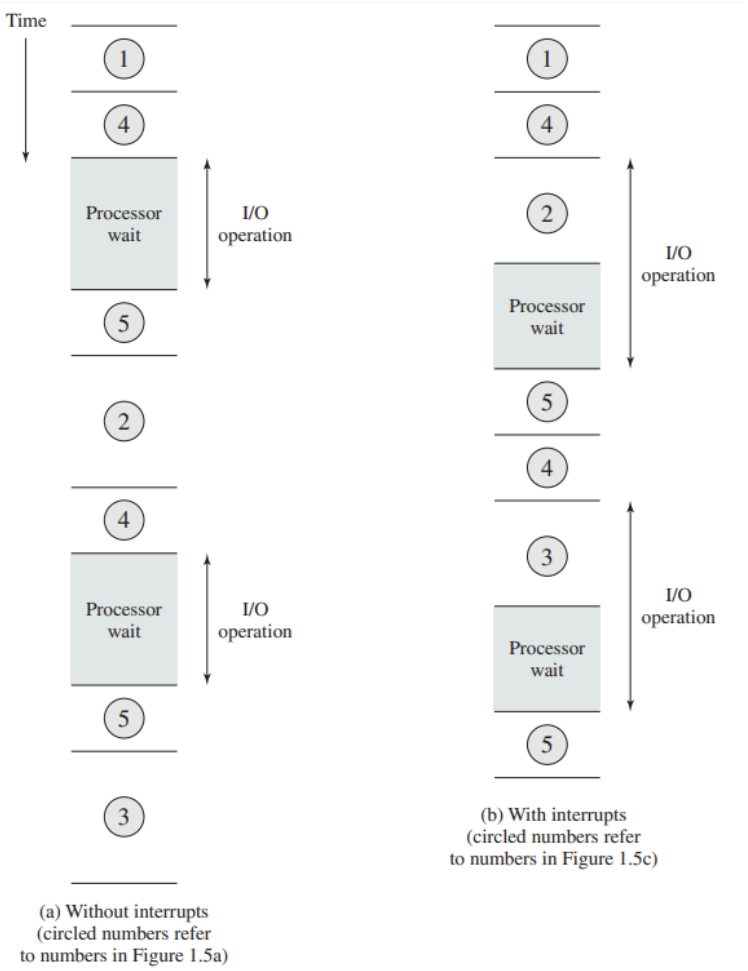

- Interrupt I/O dibagi menjadi 2 tergantung pada I/O dan waktu interrupt handler berjalan
    1. Short I/O Wait
       Short I/O Wait contohnya pada keyboard. Waktu interrupt handler sebentar.



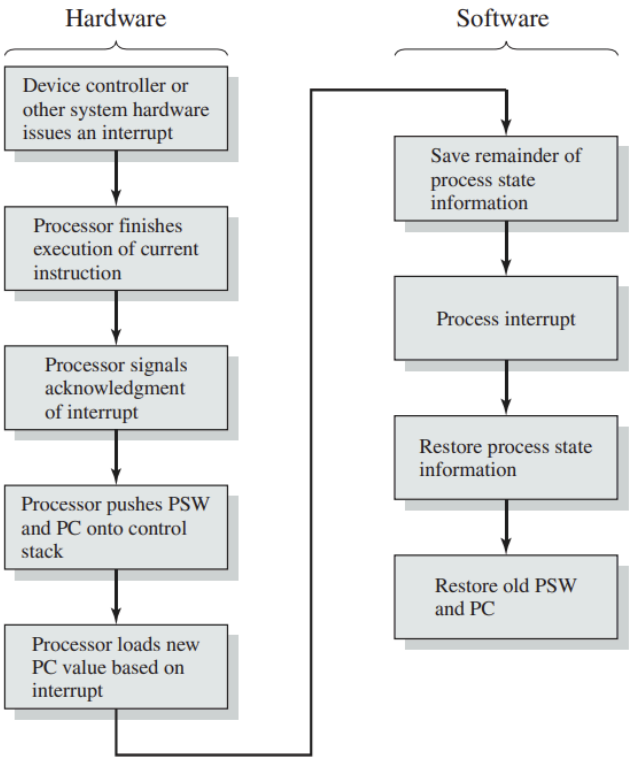

(a) Without interrupts (circled numbers refer to numbers in Figure 1.5a)

(b) With interrupts (circled numbers refer to numbers in Figure 1.5b)

2. Long I/O Wait
   Contohnya pada printer yang merupakan I/O lambat, processor memberi waktu lebih banyak pada interrupt handler ketika melakukan interrupt

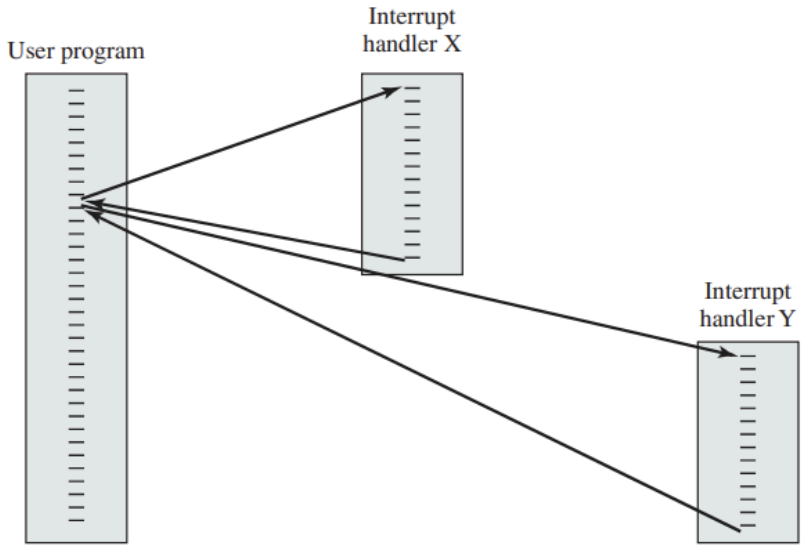



- Proses Interrupt



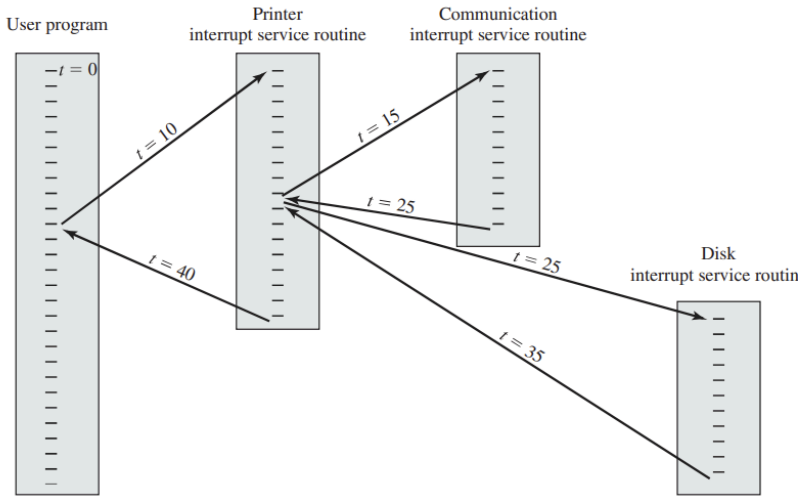
1. Device I/O memberi sinyal interrupt
2. Processor menyelesaikan instruksi yang sedang dijalankan
3. Processor menerima sinyal interrupt
4. Processor mengirim PSW dan PC ke control stack untuk disimpan
5. Processor mengambil nilai PC baru berdasarkan interrupt
6. Sisa program yang terinterupt sebelumnya akan disimpan
7. Interrupt akan diproses
8. Sisa program yang disimpan (pada no 6) dikembalikan
9. PSW dan PC yang lama (pada no 4) dikembalikan

- Multiple interrupts
  Interrupts dapat terjadi lebih dari satu kali. Ada 2 pendekatan untuk menyelesaikan masalah ini.
  1. Sequential Interrupt Processing
     - Menjalankan 1 interrupt pada 1 waktu
     - Apabila ada interrupt baru masuk maka akan di*pending*.
     - Interrupt baru akan langsung dijalankan ketika interrupt yang sebelumnya telah selesai
     - Kekurangannya, program tidak dapat menjalankan sesuai dengan prioritas interrupt. Contohnya dalam komunikasi, ketika komunikasi tidak segera diterima datanya melalui interrupt maka data dikomunikasi dapat mengalami overflow dari data sebelumnya.




  2. Nested Interrupt Processing
     - Nested Interrupt memungkingkan agar terjadi interrupt di dalam interrupt
     - Nested Interrupt mengurutkan interrupt berdasarkan prioritasnya
     - Contohnya pada gambar di bawah, prioritas interrupt diurutkan berdasarkan Communication paling penting (poin 5), Disk

kedua (poin 4), dan Printer (poin 2). Ketika ada interrupt lebih penting dari interrupt yang sedang dijalankan maka akan terjadi interrupt pada interrupt yang sedang dijalankan.




## IV. Memory Hierarchy

- 3 Pedoman utama memory hierarchy:
  a. Faster access time, greater costs per bit
  b. Greater capacity, slower access time
  c. Greater capacity, smaller costs per bit
- Memory hierarchy didesain dengan ketentuan semakin ke bawah:
  a. Kapasitas meningkat
  b. Waktu akses meningkat
  c. Costs per bit berkurang
  d. Frekuensi akses prosesor ke memori berkurang

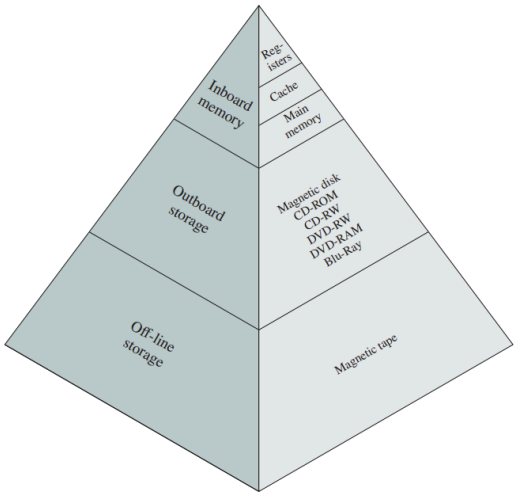



Figure 1.14 The Memory Hierarchy

- Contoh access time, jika memori yang dibutuhkan sebesar 95% berada di cache, dengan access time cache ke processor 0,1 microseconds, dan waktu dari main memory ke cache membutuhkan 1 microseconds. Maka

$$(0.95)\,(0.1\ \mu s) + (0.05)\,(0.1\ \mu s + 1\ \mu s) = 0.095 + 0.055 = 0.15\ \mu s$$

- Sifat Secondary Memory:
    1. Disebut juga auxiliary memory
    2. Bersifat non volatile (data tidak hilang apabila CPU dimatikan)
    3. Bersifat external (terpisah dari CPU)
    4. Digunakan untuk menyimpan program dan data files

## V. Cache Memory

- Cache adalah memori kecil dan cepat yang berada di antara memori dan processor.
- Alasan penggunaan cache:
    1. Processor speed lebih cepat dari pada main memory speed. Hal ini menyebabkan processor tidak dapat bergerak dengan cepat karena kebutuhan memori tidak dipenuhi.
    2. Agar processor dapat bergerak lebih cepat dan biaya pembuatan murah, dibuat cache memory yang memiliki kecepatan transfer ke CPU dengan cepat
- Prinsip cache

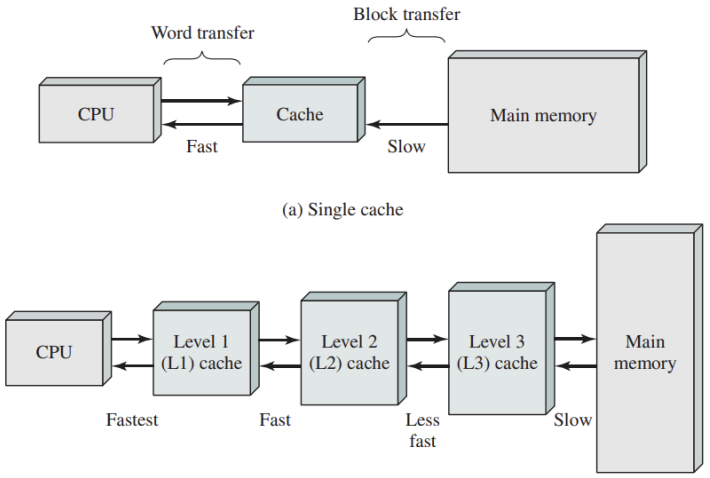
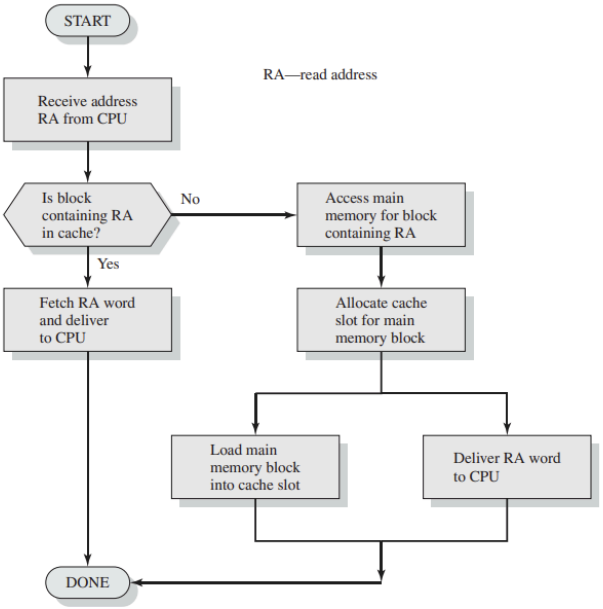



**Figure 1.16** **Cache and Main Memory**

1. Cache menyimpan copy dari sebagian main memory
2. Ketika processor membutuhkan data, processor akan memeriksa terlebih dahulu apakah data ada di cache
3. Jika data tidak ditemukan di cache, cache akan membaca data dari memori
4. Karena prinsip ***Locality of Reference*****,** kemungkinan besar data yang dibutuhkan di masa depan akan berada di cache.




- Design Cache memenuhi sifat-sifat
    1. **<u>Cache size:</u>** Cache kecil pun memiliki dampak besar ke performa
    2. **<u>Block size</u>**

       Block size merupakan besar unit data yang ditukar antara cache dan main memory. Block size dengan ukuran ideal adalah block size yang dapat mencapai **hit ratio probability** terbaik dengan **size terkecil**.

       Apabila block size terlalu kecil: hit ratio probability kecil karena data yang disimpan di cache kemungkinan besar sulit ditemukan

       Apabila block size terlalu besar: Ukuran block size akan jadi terbuang (waste) karena data yang jarang digunakan pun dapat disimpan di sana
    3. **<u>Mapping function:</u>**

       Mapping function adalah fungsi yang berguna untuk menentukan ketika suatu memori masuk cache, dia akan menempati bagian cache yang mana. Mapping function dibatasi 2 hal yaitu

        a. Ketika suatu block data masuk, suatu block data harus digantikan. Mapping function harus mengatur block data yang kemungkinan tidak digunakan lagi di masa depan.
        b. Semakin fleksibel mapping function, biaya yang dibutuhkan juga semakin besar karena semakin kompleks kebutuhan sirkuit
    4. **<u>Replacement Algorithms:</u>**

       Replacement algorithms adalah algoritma yang digunakan untuk menentukan bagian memori mana yang harus direplace ketika suatu memori masuk. Jenis-jenis algoritma

        a. Least-recently-used (LRU) algorithm
        b. First in First Out (FIFO) algorithm
        c. Least Frequently Used (LFU) algorithm
    5. **<u>Write Policy</u>**

Write policy adalah aturan yang menentukan kapan operasi penulisan pada memori dilakukan oleh cache. Ini mengatur kapan data yang diubah dalam cache akan ditulis kembali ke memori utama.
Ada 2 ekstrim write policy:

a. Menulis setiap kali ada perubahan data di cache → efek buruknya operasi penulisan dilakukan terus-menerus (boros)
b. Menulis setiap kali block di cache digantikan → main memory dalam keadaan usang yang dapat mengganggu operasi I/O terutama Direct Memory Access (DMA)

## VI. I/O Operation

Teknik komunikasi I/O dibagi menjadi 3 yaitu

1. Programmed I/O
    - Processor mengecek berulang kali apakah perangkat I/O telah menyelesaikan operasi/tugasnya belum
    - Teknik ini sederhana tetapi memakan banyak waktu CPU karena CPU harus secara aktif terlibat dalam proses transfer data.
2. Interrupt-Driven I/O
    - I/O memberi sinyal interrupt apabila siap melakukan pertukaran data
    - CPU melakukan proses transfer data ketika menangani interrupt
3. DMA I/O
    - DMA dapat melakukan transfer data secara langsung dari I/O ke main memory
    - CPU hanya bertugas menginisialisasi DMA dan tidak terlibat dalam transfer data
    - Beban CPU sangat ringan dalam metode ini
    - Kekurangan DMA:
        a. Kesulitan debugging karena tidak melibatkan CPU dalam memberi pesan bug
        b. Kompleksitas pengaturan untuk mengalokasikan DMA dengan benar
        c. Dapat terjadi konflik antara DMA dengan proses lain.

# Pertemuan 2: Process

## I. Operating System

- Sistem operasi harus dapat:
    1. Menghubungkan eksekusi dari berbagai proses
    2. Mengalokasikan kebutuhan proses serta menjaga kebutuhan tersebut dari proses lain
    3. Membuat proses dapat saling menukar informasi
    4. Sinkronisasi antar proses
- Sistem operasi yang baik harus menjalankan eksekusi seperti
    1. Menyediakan resource bagi banyak aplikasi
    2. Berpindah antar aplikasi oleh processor
    3. Menggunakan processor dan I/O dengan efektif

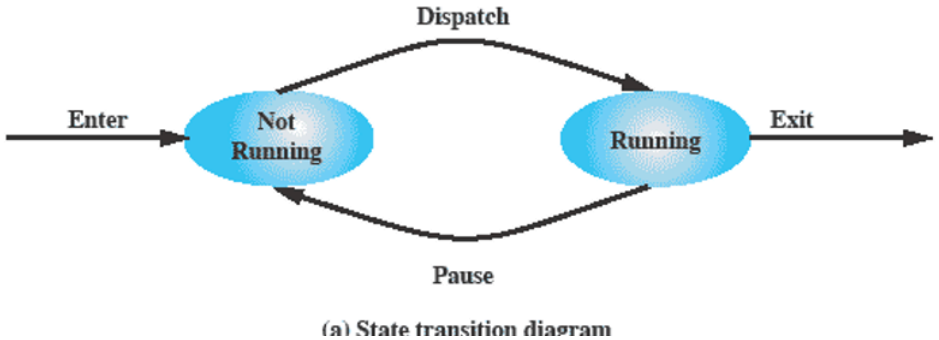
## II. Pengertian Proses

- Proses adalah:
    1. Program yang sedang **dieksekusi**
    2. Salinan dari program yang sedang **dijalankan** oleh komputer
    3. Program yang dapat **dialokasikan dan dieksekusi**
    4. **Serangkaian instruksi berurutan** yang mencerminkan kondisinya pada titik tertentu dalam **eksekusi**, serta kumpulan instruksi sistem yang terkait dengan manajemen proses seperti alokasi sumber daya, penjadwalan, dan komunikasi antar-proses.
- Proses memiliki berbagai elemen seperti (**Process Control Block)**
    1. Identifier: Penanda unik dari setiap Proses
    2. State: Status dari proses, contohnya apabila sedang berjalan maka running state
    3. Priority: Prioritas terhadap proses lain
    4. Program Counter: Alamat dari instruksi selanjutnya dari program yang dieksekusi
    5. Memory Pointers: Menunjuk ke memori yang berhubungan dengan proses
    6. Context Data: Berhubungan dengan data di register ketika proses berjalan
    7. I/O Status Information: Status mengenai I/O yang berhubungan dengan proses
    8. Accounting information: kebutuhan time and clock
- Control block berguna buat nyimpen memori dari proses, kalau ada interrupt dia nanti balik lagi ke proses yang sebelumnya.

## III. Trace of the Process

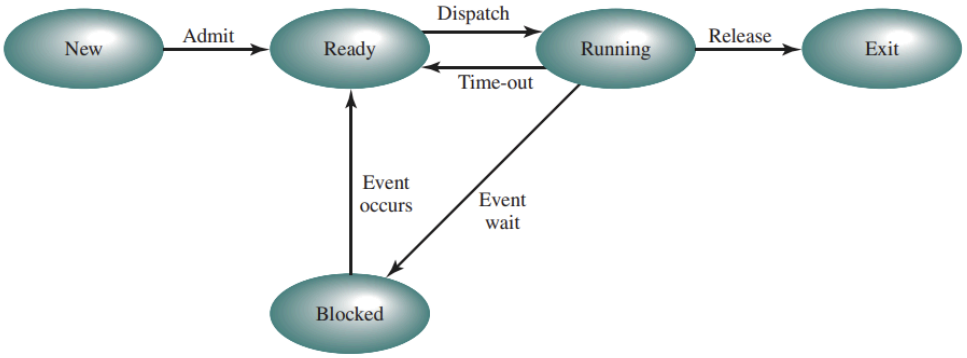
- List dari urutan eksekusi suatu proses selama operasi disebut **Trace**
- Ada sebuah program kecil disebut **Dispatcher** berguna untuk memindah eksekusi dari suatu proses ke proses lain
- **Dispatcher** bermanfaat agar tidak ada 1 program yang menguasai seluruh waktu processor

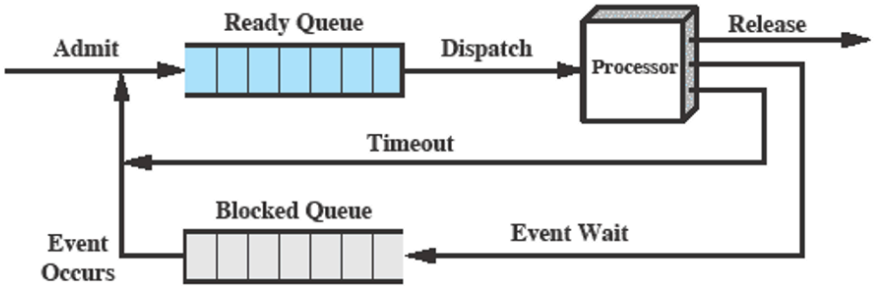
- Two State Process Models:




(a) State transition diagram

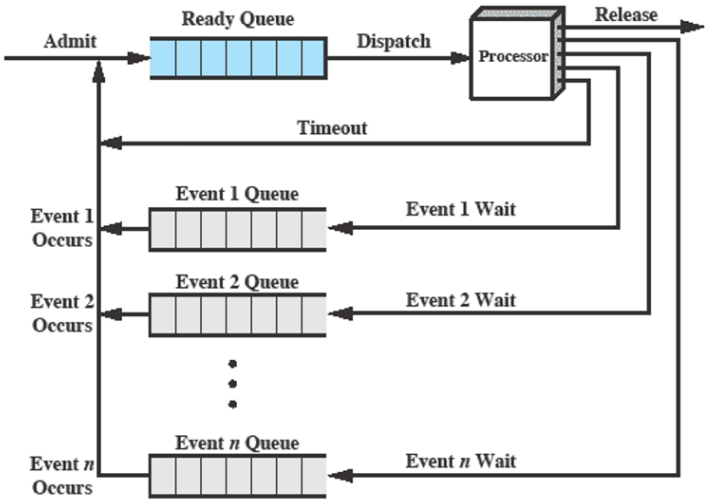
- Process Creation and Termination.
  - Process Creation ketika:
    a. Muncul Pekerjaan baru
    b. User melalui interactive log on
    c. Dibuat oleh OS untuk memberi layanan kepada user
    d. Proses yang sedang berjalan memunculkan proses baru
  - Process Termination ketika:
    a. Normal completion → tugas selesai
    b. Memory unavailable → kebutuhan memori tidak bisa disediakan
    c. Protection error → proses menggunakan resource yang dijaga/diproteksi
    d. Operator atau OS menghentikan proses
    e. Parent process menghentikan child process
- Five State Process Models:



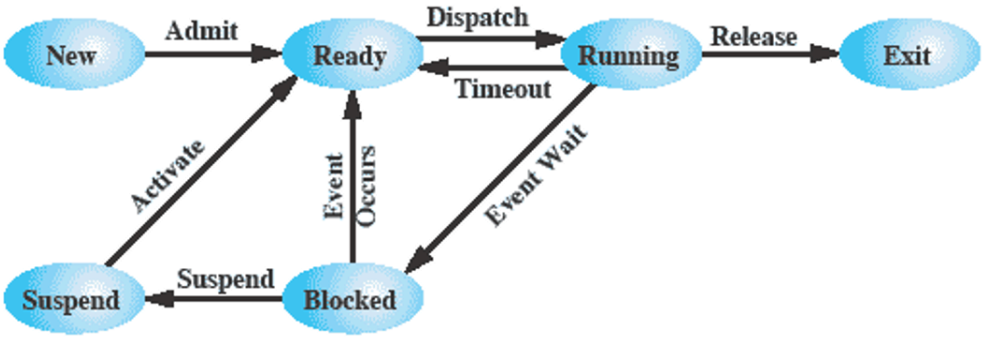

(a) Single blocked queue

# Multiple Blocked Queues



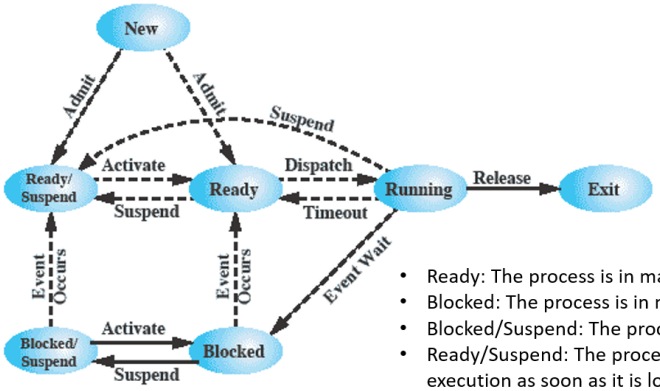

(b) Multiple blocked queues

- Proses Suspend → Kondisi ketika proses dipindah ke secondary memory




(a) With One Suspend State

## Two Suspend States




- Ready: The process is in main memory and available for execution.
- Blocked: The process is in main memory and awaiting an event.
- Blocked/Suspend: The process is in secondary memory and awaiting an event.
- Ready/Suspend: The process is in secondary memory but is available for execution as soon as it is loaded into main memory.

(b) With Two Suspend States

- Alasan ada suspend:
  1. OS perlu mengganti main memory dengan proses yang sudah siap dijalankan
  2. OS merasa proses menyebabkan masalah

3. Permintaan user
4. Waktu berjalan proses lama
5. Permintaan parent process

## IV. OS Control Tables

OS memiliki sistem mengontrol proses dan resource. OS membuat suatu informasi tabel dari setiap entitas yang dikontrol.

**a. Memory Tables**

Keep track dari main memory dan secondary memory. Menyimpan informasi seperti:

1. Alokasi dari main memory terhadap process
2. Alokasi dari secondary memory terhadap process
3. Menjaga akses terhadap memori yang diperbolehkan (izin read, write, execute)
4. Informasi untuk mengolah virtual memory

**b. I/O Tables**

Mengatur I/O Device dan channel dari komputer. OS perlu mengetahui:

1. Apakah I/O Device available atau sedang mengerjakan suatu tugas
2. Status dari operasi I/O
3. Lokasi dari main memory yang jadi sumber serta tujuan I/O

**c. File Tables**

Mengatur file-file. Menyimpan informasi seperti:

1. Apakah suatu file ada atau tidak
2. Lokasi file di secondary memory
3. Status file
4. Atribut lainnya dari file

**d. Process Tables**

Untuk mengatur proses, OS perlu menyimpan informasi seperti:

1. State dari proses
2. ID Process
3. Lokasi proses di memori

- Koleksi dari suatu atribut disebut *process control block* (scroll ke atas mengenai elemen proses)
- Process Control Block menyimpan informasi yang dapat dibagi menjadi 3 kategori, yaitu:
    a. Process Identification, menyimpan:
        - Identifier dari Process (Process ID) (disingkat PID)
        - Identifier dari Parent Process (Process yang membuat process tersebut) (disingkat PPID)
        - User Identifier yang menampilkan user yang menjalankan suatu program (disingkat UID)
    b. Processor State Information, menyimpan:
        - User-Visible Registers → yaitu register yang bisa dilihat oleh user

- Control and Status Registers → register yang mengontrol operasi dari processor, meliputi:
    a. Program counter: menyimpan alamat dari instruksi selanjutnya yang akan diambil
    b. Condition codes: Hasil dari operasi aritmatika atau logika sebelumnya (cthnya: sign (+ atau -), zero/equal, carry, overflow)
    c. Status information: menyimpan informasi seperti interrupt disable/enable, execution mode (user mode/kernel mode/supervisor mode nti ada di bawah)
- Stack Pointer → Points to the top of the stack (stack adalah memori untuk menyimpan secara temporary dari register)

c. Process Control Information
    - Scheduling and State Information
      Menyimpan informasi yang dibutuhkan sistem operasi untuk melakukan penjadwalan seperti:
        a. Process State: status dari proses mengenai kesiapan untuk dieksekusi contohnya running, ready, waiting, halted
        b. Priority: Prioritas dari proses. Contohnya highest-allowable (prioritas tertinggi), default (berdasarkan bawaan proses), current (tingkat prioritas proses yang ditentukan oleh OS saat ini)
        c. Scheduling-related information: Berdasarkan algoritma penjadwalan yang digunakan. Contohnya penggunaan timeout.
        d. Event: Event yang terjadi pada proses sebelum proses tersebut akan dijalankan. Contohnya input events (input pengguna dari I/O), timer events (timeout), communication events (komunikasi), signal events (SIGINT, SIGKILL), error events (adanya error pada proses)
    - Data Structuring
      Penyusunan data yang diakses oleh proses
    - Interprocess Communication (IPC)
      Flag, sinyal, atau pesan komunikasi dari dua proses independen
    - Process Privileges
      Informasi tentang memori yang boleh diakses oleh proses

- Memory Management
  Pointer ke virtual memory yang berhubungan dengan proses
- Resource Ownership and Utilization
  Resource yang digunakan proses seperti file yang dibuka, penggunaan processor, dan resource lainnya.

- Peran Process Control Block:
  a. Data struktur paling penting dalam suatu OS
  b. Menyimpan semua status dari proses yang dibutuhkan oleh OS
  c. Setiap process control block dibaca dan dimodifikasi oleh OS

  Maka PCB:
  d. Process Control Block menentukan state (status) dari OS
- Process Control Block memerlukan proteksi, karena:
  a. sebuah bug pada suatu routine (contohnya interrupt handler) dapat merusak Process Control Block yang berakibat rusaknya sistem yang menangani proses
  b. Perubahan desain di arsitektur pada process control block dapat berakibat kepada beberapa module (barisan kode untuk melakukan fungsi tertentu) pada OS

# V. Process Control

- Modes of Execution
  Mode pada OS dibagi menjadi 2 berdasarkan hak aksesnya. Pembagian ini dilakukan untuk melindungi proses dan juga process control block
  a. User mode (less-privilege mode)
     - Program user biasa dijalankan dalam mode ini
     - Mode ini tidak dapat melakukan beberapa instruksi
  b. System mode, control mode, or kernel mode (more-privilege mode)
     - Beberapa instruksi yang dapat dilakukan di mode ini seperti membaca dan mengubah control register, instruksi I/O yang primitif, instruksi yang berhubungan dengan memory management

  Suatu proses bergantian dalam user mode dan system mode agar dapat berjalan. Processor dapat membedakan user mode dan system mode dengan suatu bit/flag pada Program Status Word. Contohnya dalam mengedit isi suatu file:
  a. User mengetikkan isi file → user mode
  b. User menekan tombol simpan → user mode diubah ke kernel mode dan OS mengakses memori untuk menyimpan perubahan
  c. Perubahan selesai disimpan → kernel mode diubah lagi ke user mode
- Process Creation
  Langkah-langkah pembuatan suatu proses:
  a. Suatu **process identifier** (pid) unik disiapkan

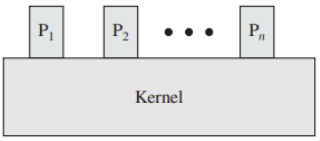
b. **Alokasi ruang untuk proses** (Allocates space for the process). Proses harus menyediakan ruang memori yang diperlukan untuk program, data, dan user stack (stack).
c. **Initializes process control block** (inisialisasi PCB). PCB menyimpan berbagai informasi seperti Parent Process Identifier (PPID), Program counter (menyimpan alamat memori dari awal program), dan sistem stack pointer (menunjuk ke stack). Informasi lainnya dari proses dapat disesuaikan secara default atau sesuai atribut lain yang diminta. Contohnya seperti state yang akan dipasang ke *ready* atau *ready/suspend*, dan prioritas ke prioritas terendah kecuali ada request.
d. **Sets up appropriate linkages** (menyiapkan hubungan). Hubungan proses dengan proses lain disiapkan, contohnya ketika dalam scheduling process, proses akan ditetapkan sebagai Ready atau Ready/Suspend List
e. **Creates or expands other data structures** (membuat atau memperluas struktur data lain). Contohnya jika OS memiliki accounting file pada setiap proses untuk menghitung performa.

- Process Switching
  Interrupting suatu proses hingga mengakibatkan process switching digolongkan dalam 3 sebab yaitu
  1. Interrupt, disebabkan faktor **eksternal** dari proses seperti sinyal I/O. Fungsinya sebagai reaksi terhadap suatu kejadian eksternal yang asinkronus. Contohnya adalah clock interrupt yaitu proses berjalan lebih lama dari waktu yang diperbolehkan. Contoh lainnya adalah I/O interrupt, yaitu I/O memberi sinyal interrupt bahwa tugasnya telah selesai dsb. Contoh lain adalah ketika proses harus mengakses memori di luar main memory, hal ini mengakibatkan OS harus mengambil terlebih dahulu dari secondary memory baru melanjutkan proses.
  2. Trap, disebabkan faktor **internal** dari proses (eksekusi instruksi pada proses), biasanya terjadi error dalam menjalankan proses. Tujuannya adalah handle error atau kondisi tertentu akibat eksekusi suatu instruksi. Contohnya ketika proses berusaha mengakses memori yang tidak diperbolehkan. Trap dapat berakhir antara langsung mengakhiri program (jika error fatal) atau melakukan exception (kondisi tertentu sesuai program untuk menangani error)
  3. Supervisor Call, yaitu **permintaan dari user** untuk memindah proses. Contohnya ketika suatu proses sedang berjalan dan user meminta untuk membuka suatu file, maka proses akan dipindahkan sementara.
- Apa yang terjadi ketika process Switching:
  a. Informasi dari proses sekarang seperti program counter dan register lainnya akan disimpan
  b. Update process control block dari proses yang sedang berjalan sekarang, mengubahnya dari Running State ke state lain (ready;

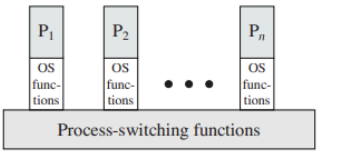
blocked; ready/suspend; atau exit) serta update hal lain seperti alasan state berubah dan accounting information

c. Pindahkan process control block ke barisan sesuai state yang disesuaikan
d. Pilih proses lain yang akan dieksekusi
e. Update process control block ke proses yang dipilih
f. Update memory management data structures
g. Apabila proses yang awal ingin berjalan kembali, informasi yang telah disimpan (pada no a) akan dikembalikan sehingga proses akan melanjutkan dari bagian yang terputus sebelumnya.

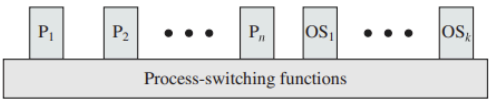
- Execution of the Operating System
  Apakah OS adalah suatu proses? Hal dipengaruhi dari desainnya



(a) Separate kernel



(b) OS functions execute within user processes



(c) OS functions execute as separate processes

a. Nonprocess Kernel
   - Biasa di OS lama
   - Kernel (OS) berjalan di luar proses
   - Yang disebut proses hanyalah user program sedangkan kernel berjalan di privileged mode

b. Execution within User Processes
   - Biasa di komputer kecil (PC, workstation)
   - OS berjalan dalam proses user

c. Process-Based Operating System
   - OS berjalan sebagai kumpulan dari proses, OS berdiri sendiri menjadi suatu proses

## VI. Security Issues

Beberapa masalah keamanan yang dapat mengancam OS:

1. System Access Threat
   Dibagi menjadi 2 yaitu:
   a. Intruders
      Dibagi menjadi 3 jenis, yaitu:
      1) Masquerader (outsider)

Seseorang yang tidak memiliki akses ke komputer dan berusaha untuk menembus sistem akses kontrol untuk exploit ke user account

2) Misfeasor (insider)
Seseorang yang memiliki akses tetapi mengakses data, program atau resource yang tidak diperbolehkan atau menyalahgunakan hak aksesnya

3) Clandestine user (outside or insider)
Seseorang yang memiliki **akses supervisor** kepada sistem dan menggunakannya untuk melakukan tindakan secara **diam-diam** dengan menonaktifkan fitur audit, menghapus log kejadian atau merusak sistem pencatatan

b. Malicious Software (Malware)
Dibagi menjadi 2 jenis berdasarkan kebutuhan program hostnya, yaitu:

1) parasitic (membutuhkan program host), contohnya seperti virus, logic bombs (virus yang jika memenuhi suatu keadaan tertentu akan menyebabkan kerusakan), dan backdoor.
2) Program independen. Contohnya seperti worms atau program bot.

Dibagi menjadi 2 jenis berdasarkan kemampuan membelah diri, yaitu:

1) Tidak bisa mengcopy dirinya sendiri. Biasa program yang memerlukan suatu syarat agar berjalan seperti logic bombs, backdoors, dan bot program
2) Program yang dapat memperbanyak dirinya sendiri seperti virus dan worms.

2. Countermeasures (penanggulangan)
a. Intrusion Detection
yaitu sistem keamanan yang memonitor dan menganalisa kejadian pada sistem untuk mencari dan menyediakan peringatan apabila ada akses yang tidak diperbolehkan. Dibagi menjadi 2:

1) Host-based IDS → memperhatikan karakteristik dari satu host dan kejadian yang dilakukan host tersebut untuk mencari aktivitas yang mencurigakan
2) Network-based IDS → memperhatikan lalu lintas jaringan untuk mendeteksi aktivitas mencurigakan

IDS dibagi menjadi tiga komponen

1) Sensor → mengumpulkan data seperti network packets (aktivitas pengiriman data di jaringan), log files (berkas log aktivitas di komputer), system call traces (jejak panggilan sistem seperti membuka atau menutup file dsb)
2) Analyzers → menerima data dari sensor. Menentukan apakah terdapat intrusion (tindakan mengakses akses yang tidak diperbolehkan) atau tidak, menunjukkan bukti apabila terjadi intrusion, serta saran penanganan yang dapat dilakukan

3) User Interface → user interface agar user bisa melihat output atau kontrol dari sistem

b. Authentication

Dibagi menjadi 2 tahap yaitu:

a. Identifikasi → Menyediakan identifier ke security system
b. Verifikasi → Mencocokkan antara entitas dengan identifier

Ada 4 faktor yang dapat digunakan untuk autentikasi:

a. Something the individual knows → password, pin
b. Something the individual possesses → KTP, kartu, kunci
c. Something the individual is (static biometrics) → fingerprint, face, retina
d. Something the individual does (dynamic biometrics) → voice pattern, handwriting, typing rhythms

c. Access Control

Mengontrol akses secara spesifik agar agar hanya bisa diakses oleh seseorang atau proses tertentu.

- Autentikasi user apakah boleh mengakses sistem tersebut atau tidak
- Menentukan apakah akses tersebut boleh untuk user tersebut
- Administrator keamanan menentukan apakah database boleh diakses oleh user ini
- Suatu fungsi audit memonitor dan terus menjaga record dari user yang akses ke sistem resources

d. Firewalls

Menjaga local sistem atau internet sistem dari ancaman berbasis jaringan. Jadi firewall digunakan untuk menjaga komputer yang mengakses komputer lain melalui jaringan. Firewall didesain untuk:

- Semua traffic keluar dan masuk jaringan harus melalui firewall
- Hanya traffic yang diperbolehkan yang bisa lewat
- Firewall sendiri harus kebal terhadap penetrasi

# Pertemuan 3: Threads

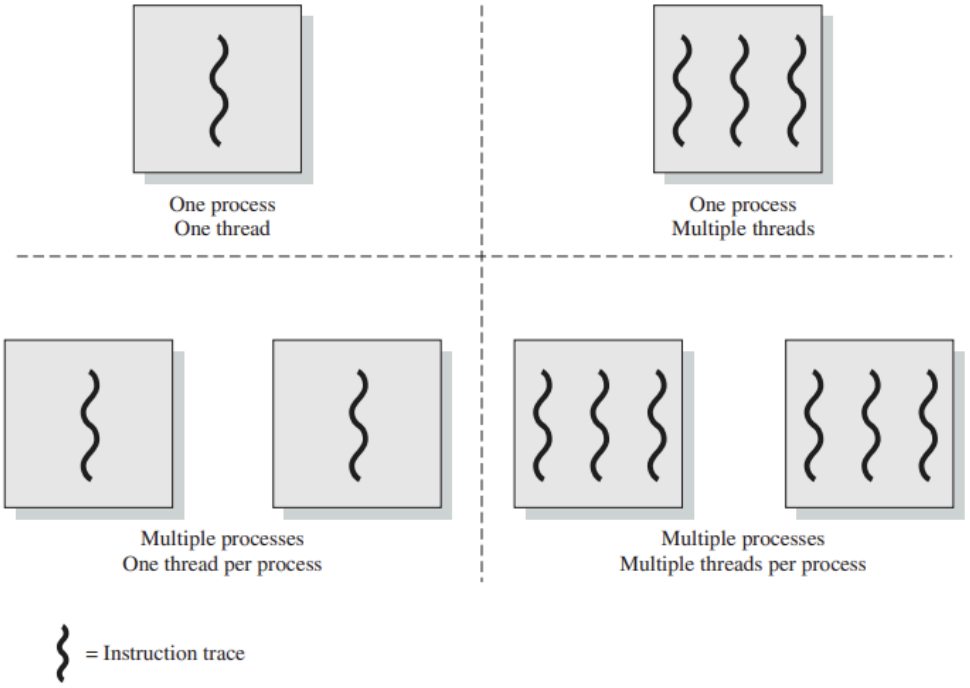
## I. Threads

- Proses vs Thread

| Process | Thread |
|---|---|
| Proses adalah entitas yang **memiliki sumber daya** seperti memori, file, dan area alamat. | Thread adalah **unit eksekusi** yang berjalan **di dalam suatu proses**. |
| Process mengatur Thread | Thread adalah bagian dari Process |
| Process **tidak saling membagi memori** dengan proses lain sehingga | Thread dapat **berbagi memori dengan thread lain dalam 1** |

| | |
|---|---|
| **apabila 1 proses error tidak mempengaruhi proses lain**. | **process. Error pada 1 thread dapat berpengaruh pada keseluruhan suatu process** yang menaunginya. |

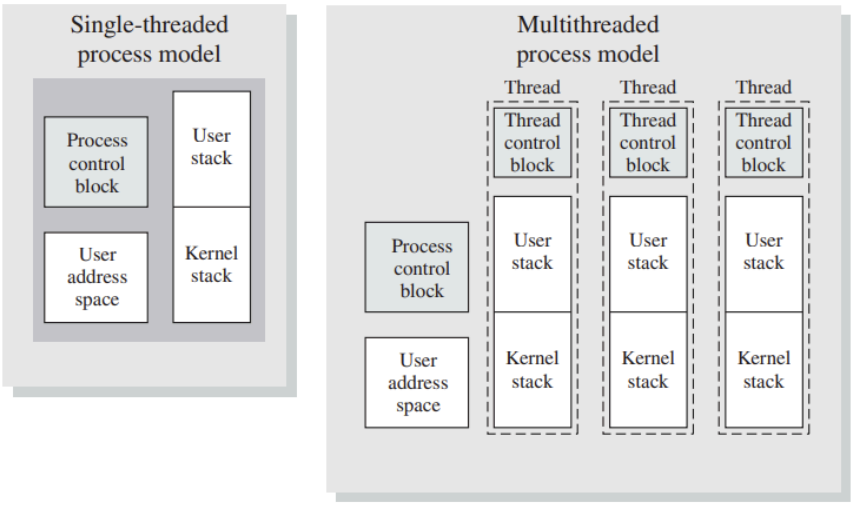
- Multithreading
  Istilah yang dalam suatu process dapat memiliki lebih dari satu thread sehingga suatu process mampu menjalankan beberapa eksekusi bersamaan.




  Penerapannya
  - One Process One Thread → MS-DOS
  - Multiple Processes One Thread → beberapa OS Unix
  - One Process Multiple Threads → Java Runtime Environment (JRE) yaitu aplikasi yang dibuat dengan java
  - Multiple Processes Multiple Threads → Digunakan oleh OS modern seperti Windows, Linux, Solaris, dan Unix lainnya (cth: MacOS)

  Dalam suatu multithreading, process berhubungan dengan:
  - Alamat yang menyimpan process image (Process Control Block)
  - Menjaga akses ke processor dan process lain, file, dan I/O Resource

  Sedangkan threading berhubungan dengan:
  - Thread execution state (Running, Ready, dsb)
  - Thread context yang disimpan ketika tidak berjalan. (Menyimpan Program Counter → **Setiap thread memiliki satu Program Counter**)
  - Execution stack (tumpukan urutan pemanggilan eksekusi)
  - Penyimpanan statis untuk local variabel
  - Akses ke memori dan resource pada prosesnya yang berbagi dengan thread lain dalam proses yang sama




Keuntungan menggunakan thread:

- Waktu membuat thread baru lebih cepat daripada membuat process baru
- Lebih cepat menghentikan (terminate) sebuah thread daripada sebuah process
- Lebih cepat berpindah antara dua thread daripada antara dua process
- Thread dapat berkomunikasi antara satu dengan yang lain (karena berbagi data yang sama) berbeda dengan process yang memerlukan kernel untuk berkomunikasi satu dengan yang lain

Contoh Penggunaan multithreading:

- Foreground and background work → ini kayak fe dan be. Contohnya di spreadsheet, 1 thread bisa bertugas menampilkan data sedangkan thread lain bisa update spreadsheet dari user.
- Asynchronous processing → Contohnya dalam membackup data ke RAM, thread dapat digunakan karena thread dapat di-create dan di-terminate dalam jangka waktu yang pendek serta tidak mengganggu alur program utama
- Speed of execution → multithread dapat bekerja lebih cepat, contohnya suatu kumpulan data dapat diolah bersamaan dengan kumpulan data selanjutnya dibaca
- Modular program structure (program dengan banyak modul/fitur independen) → Program seperti ini lebih mudah dibuat dengan multithreading karena input dan output yang bermacam-macam dan bersamaan dapat dilakukan dengan multithreading.

Thread memiliki states seperti process. Ada 4 jenis states pada thread:

- Spawn → Bisa dimunculkan ketika suatu proses dibuat atau suatu thread spawn thread lain
- Block → Ketika suatu thread perlu untuk menunggu suatu event, thread tersebut akan di block (menyimpan user register, program

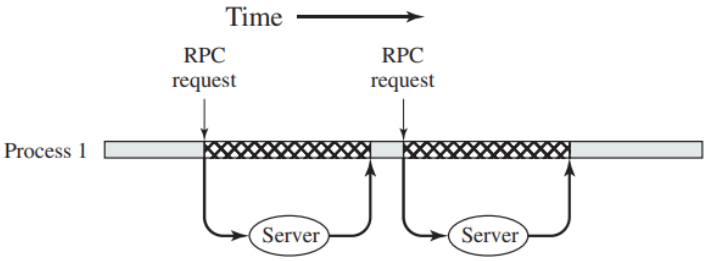
counter, dan stack pointernya). Processor akan menjalankan thread lain yang sudah siap

- Unblock → Memindahkan thread yang siap ke Ready
- Finish → Ketika suatu thread selesai, register context and stacks dipindahkan

Apakah penggunaan jika satu thread dalam kondisi block maka seluruh proses akan dalam kondisi block?

Contoh kasus pada RPC → teknik di mana terdapat dua program yang dieksekusi dalam mesin yang berbeda yang saling berinteraksi dan seakan-akan berjalan dalam mesin yang sama.

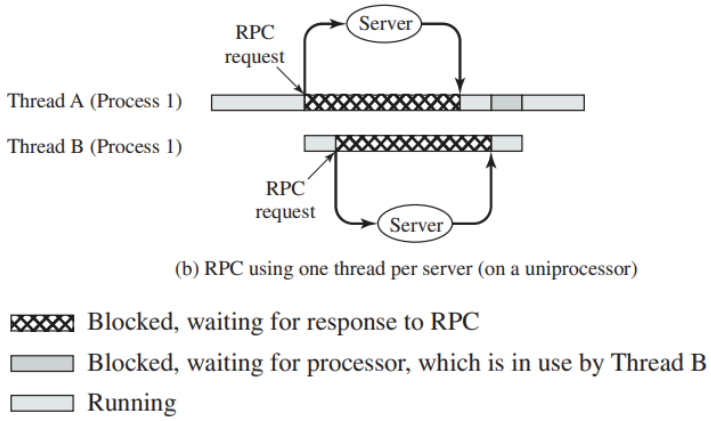
a. Apabila hanya ada 1 thread




(a) RPC using single thread

RPC harus menunggu RPC sebelumnya selesai terlebih dahulu karena dijalankan secara sekuensial

b. Apabila setiap server punya 1 thread dalam komputer uniprocessor (hanya memiliki 1 processor)




(b) RPC using one thread per server (on a uniprocessor)

**Figure 4.3 Remote Procedure Call (RPC) Using Threads**

Apabila 1 komputer hanya memiliki 1 processor maka penggunaan thread akan membantu dalam waiting RPC Request

c. Multithreading dalam 1 processor

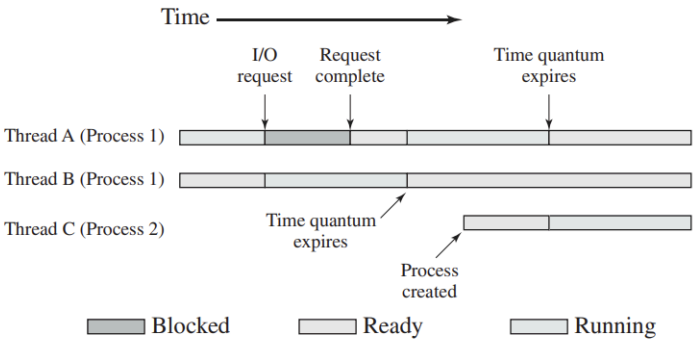
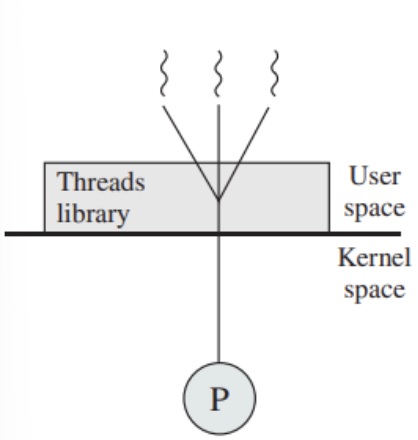



**Figure 4.4 Multithreading Example on a Uniprocessor**

Sinkronisasi Thread:

- Suatu thread berbagi resource dengan thread lain dalam 1 proses sehingga perubahan yang dilakukan suatu thread berefek ke thread lainnya

- Types of Threads
  a. User-Level Threads (ULTs)




(a) Pure user-level

  - Thread yang dibuat dan diatur dari user-level code (aplikasi)
  - Kernel tidak mengetahui adanya thread tersebut
  - Suatu ULTs ditandai oleh **threads library** yaitu suatu tool untuk membuat thread, menjadwalkan thread, sinkronisasi thread, komunikasi antar thread, dan penghentian thread

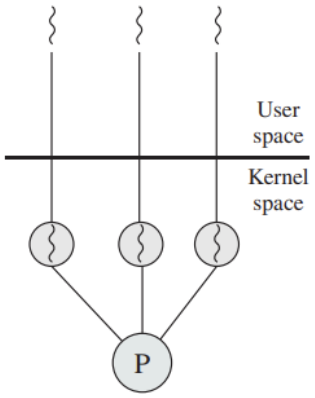
  Kelebihan ULTs:

  - Tidak membutuhkan kernel mode pada process sehingga switching thread lebih cepat (tidak perlu berpindah dari user mode ke kernel mode)
  - Setiap aplikasi dapat mengatur scheduling sesuai keinginannya
  - ULTs dapat berjalan di OS manapun karena tidak dipengaruhi kernel

  Kekurangan ULTs:

- Dalam beberapa OS, banyak system call memblokir yang berakibat ketika suatu thread diblok maka seluruh threads dalam suatu proses akan terblokir (thread itu ga bisa mengakses kebutuhan yang butuh kernel, kalo akses dia keblokir/gabisa jalan). Salah satu cara menanganinya adalah dengan menggunakan teknik **jacketing** yaitu mengubah blocking system call ke non blocking system call
- Jika menggunakan pure ULT, multithreaded tidak dapat menggunakan multiprocessor. Hal ini karena kernel meng-assign 1 proses hanya 1 processor sehingga hanya 1 thread yang dapat bekerja dalam 1 proses

b. Kernel-Level Threads (KLTs) / Kernel-Supported Threads / Lightweight Processes



(b) Pure kernel-level

- Pada aplikasi tidak ada kode yang berhubungan dengan thread
- Kode diatur oleh kernel
- Scheduling dilakukan oleh kernel terhadap thread langsung
- Contohnya Windows

Kelebihan menggunakan KLTs

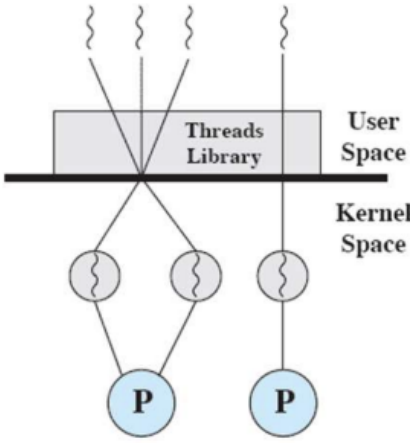
- Kernel dapat menjadwalkan multiple threads pada satu process untuk multiple processors
- Jika suatu process diblokir, kernel dapat menjadwalkan thread lain pada process yang sama
- Bagian kernel sendiri dapat berjalan secara multithreading sehingga meningkatkan kemampuan OS

Kekurangan menggunakan KLTs

- Kecepatan perpindahan thread satu ke thread lain lebih lama karena perlu switching mode process (process harus pindah dari user mode ke kernel mode)

| Criteria | User-Level Threads (ULTs) | Kernel-Level Threads (KLTs) |
|---|---|---|
| Management | Managed entirely by user-level code, without kernel support | Managed by the operating system kernel |
| Scheduling | Scheduled by a user-level thread library or application, without kernel intervention | Scheduled by the operating system kernel, which may provide advanced scheduling algorithms and policies |
| Context Switching | Context switching occurs entirely in user space, without requiring a system call or trap into kernel mode | Context switching requires a system call or trap into kernel mode, which can incur overhead |
| Resources | ULTs share the same process address space and resources as the parent process | KLTs have their own kernel-level data structures, which can result in higher overhead and memory usage |
| Scalability | ULTs are limited to a single processor core and cannot take full advantage of multicore systems | KLTs can be assigned to different processor cores and can take full advantage of multicore systems |
| Synchronization | ULTs must use user-level synchronization mechanisms, which can be more efficient but may be subject to priority inversion and other issues | KLTs can use both user-level and kernel-level synchronization mechanisms, providing more flexibility and better enforcement of policies |
| Portability | ULTs are highly portable across different operating systems, as long as a compatible user-level thread library is available | KLTs may be less portable, as different operating systems may have different thread APIs and scheduling mechanisms |

c. Combined (Kernel dan User)




(c) Combined

- Pembuatan thread dilakukan oleh user (aplikasi)
- User Thread dipetakan/diassign ke berbagai Kernel Thread

- Relationship between Threads and Processes

| Threads:Processes | Description | Example Systems |
|---|---|---|
| 1:1 | Each thread of execution is a unique process with its own address space and resources. | Traditional UNIX implementations |
| M:1 | A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process. | Windows NT, Solaris, Linux, OS/2, OS/390, MACH |
| 1:M | A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems. | Ra (Clouds), Emerald |
| M:N | Combines attributes of M:1 and 1:M cases. | TRIX |

## II. Symmetric Multiprocessing (SMP) (gada di buku)

Komputer pada umumnya dilihat sebagai mesin yang menjalankan perintah secara sekuensial dengan processor menjalankan 1 instruksi pada 1 waktu. Terdapat dua cara yang memungkinkan untuk terjadi parallelism yaitu Symmetric Multi Processors (SMPs) dan Clusters.

- Beberapa kategori dari Computer Systems
    a. Single Instruction Single Data (SISD) stream
       Satu processor hanya menjalankan satu instruksi dalam data yang disimpan di satu memori
    b. Single Instruction Multiple Data (SIMD) stream
       Setiap instruksi dijalankan dalam beberapa set data yang berbeda di processor
    c. Multiple Instruction Single Data (MISD) stream
       Data ditransmisikan ke berbagai processor, setiap processor menjalankan instruksi yang berbeda
    d. Multiple Instruction Multiple Data (MIMD) stream
       Satu set processor dijalankan bersamaan dengan setiap data sets berbeda
- Symmetric Multiprocessing
    a. Kernel dapat mengakses seluruh processor untuk menjalankan proses
    b. Setiap processor memiliki self-scheduling tersendiri untuk menjalankan proses atau thread
    c. Semua processor dapat mengakses sistem bus
- Desain multiprocessor OS
  Desain ini berhubungan dengan:
    a. Menjalankan proses atau thread secara bersamaan
    b. Scheduling
    c. Syncronisasi
    d. Memory management
    e. Reliability and fault tolerance

## III. Microkernel (gada di buku)

Ada 2 jenis arsitektur kernel (microkernel dan monolith).

- Microkernel adalah OS kecil yang menyediakan berbagai service dalam user mode.
- Kernel (microkernel) hanya bertugas sebagai komunikasi antara client process dengan system process
- Komunikasi antar thread dan proses dilakukan dengan **messages** (berupa header untuk status mengirim atau menerima data, dan body yang menyimpan data, pointer block of data, dan beberapa informasi tentang proses)
- I/O dan interrupt pada microkernel → hardware interrupts dan I/O ditangani dengan menggunakan messages
- Benefits dari penggunaan microkernel:
    - Uniform interfaces on requests made by a process (Permintaan yang dibuat oleh proses seragam interfacenya) → karena setiap proses berjalan secara terpisah dan dihubungkan dengan message
    - Extensibility (Extensibilitas) → karena layanan ada di user mode sehingga menambahkan layanan bisa mudah
    - Flexibility (Fleksibilitas) → karena layanan ada di user mode sehingga mengubah layanan bisa mudah
    - Portability (Portabilitas) → Memindahkan kernel ke platform lain lebih mudah karena tugas kernel hanya sederhana
    - Reliability → karena kernel lebih simpel sehingga kodenya lebih minim, kemungkinan terjadi error lebih kecil dan kegagalan pada layanan yang non-kritis tidak akan mengganggu kernel dan layanan lainnya.
    - Distributed System Support → Microkernel mendukung arsitektur sistem terdistribusi dengan baik karena komunikasi antara prosesnya mirip dengan komunikasi antar proses dalam sistem terdistribusi (jaringan komputer yang terhubung dan membagi penyimpanan, perangkat keras, dan perangkat lunak)
    - Object Oriented Operating Systems → Pembuatan microkernel menggunakan OOP yang memudahkan pengembangan, pemeliharaan, dan skalabilitas

## Comparison Microkernel vs Monolitic kernel

| Criteria | Microkernel | Monolithic Kernel |
|---|---|---|
| Architecture | Minimalistic kernel that provides basic services, with additional services implemented as user-space processes | Entire kernel including device drivers, system calls, and other services are executed in kernel space |
| Security | More secure due to smaller attack surface, as most operating system services are run in user space | Exposes entire kernel to potential security vulnerabilities |
| Flexibility | More flexible and easier to customize and maintain as new services can be added without modifying the kernel itself | Requires modifications to kernel code to add new functionality |
| Performance | Generally slower due to increased reliance on interprocess communication and message passing | Better performance due to all system services being executed in kernel space, avoiding the overhead of interprocess communication |
| Debugging and Maintenance | Easier to debug and maintain due to modular design, where different services run as independent processes | Debugging and maintenance can be more difficult due to all services running in the same kernel space |

Di PPT ada tambahan use case nya

# Pertemuan 4: Mutual Exclusion

## I. Principles of Concurrency

- Poin utama dari desain OS modern adalah me-manage multiple process:
    a. Multiprogramming → manage multiple process dalam uniprocessor system (banyak process, hanya ada 1 processor)
    b. Multiprocessing → manage multiple process dalam multiprocessor (banyak proses, banyak processor)
    c. Distributed processing → management dari banyak process yang dijalankan dalam multiple dan distributed computer systems (banyak proses banyak komputer)

  Masalah dari desain ini adalah **konkurensi (concurrency)** → persengketaan antar proses:
    a. Masalah komunikasi antar proses
    b. Sharing dan berebut resource (memori, files, I/O access)
    c. Sinkronisasi aktivitas dari multiple processes
    d. Alokasi dari processor time terhadap banyak process

  Concurrency muncul dalam tiga konteks permasalahan berbeda:
    a. Multiple applications (banyak aplikasi jalan bersamaan) → multiprogramming ditemukan agar **processing time dapat dibagi** secara dinamis dari beberapa aplikasi yang aktif
    b. Structured applications → **Extension dari modular design** (sistem aplikasi dibagi menjadi modul yang terpisah), suatu aplikasi dapat didesain memiliki beberapa proses (1 aplikasi bisa punya banyak proses).
    c. Operating system structure → OS sendiri diimplementasikan sebagai set process dan thread
- Istilah yang berhubungan dengan Concurrency:
    a. atomic operation → Rangkaian instruksi yang tidak dapat dibagi sehingga tidak dapat diinterrupt. Pilihannya hanya menjalankan seluruh instruksi bersamaan atau tidak dijalankan sama sekali

b. critical section → bagian kode yang tidak boleh dijalankan ketika ada proses lain yang mengakses bagian tersebut
c. deadlock → situasi ketika 2 proses atau lebih tidak dapat berjalan karena saling menunggu salah satu proses melakukan sesuatu
d. livelock → situasi ketika 2 proses atau lebih terus mengganti states mereka sebagai respon dari process lain tanpa melakukan pekerjaan yang berguna
e. mutual exclusion → kebutuhan suatu process memasuki critical section yang mengakses shared resources, tidak boleh ada proses lain yang juga mengakses critical section tersebut
f. race condition → situasi ketika banyak thread atau proses membaca dan menulis ke data yang sama dan hasilnya dipengaruhi kecepatan eksekusi mereka
g. starvation → situasi suatu proses yang dapat dijalankan tetapi diabaikan oleh scheduler sehingga tidak pernah berjalan

- Process pada multiprocessing dapat berjalan secara:
  a. Interleaving (menyisipkan/saling bergantian)

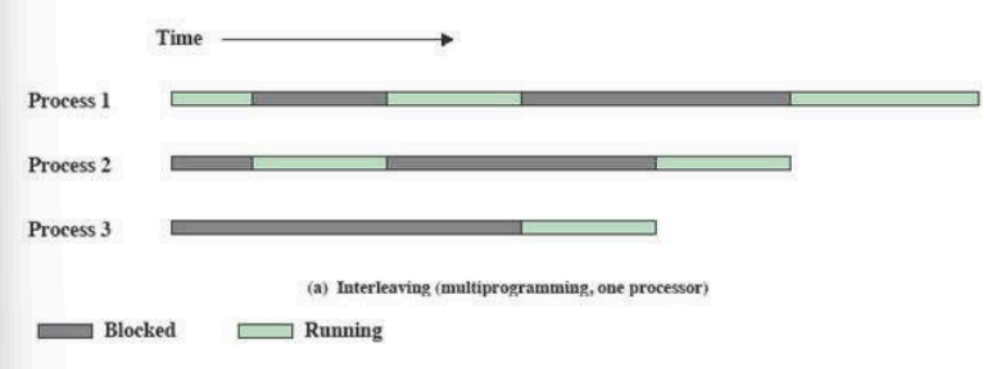



Figure 2.12 Multiprogramming and Multiprocessing

  b. Overlapping (tumpang tindih)

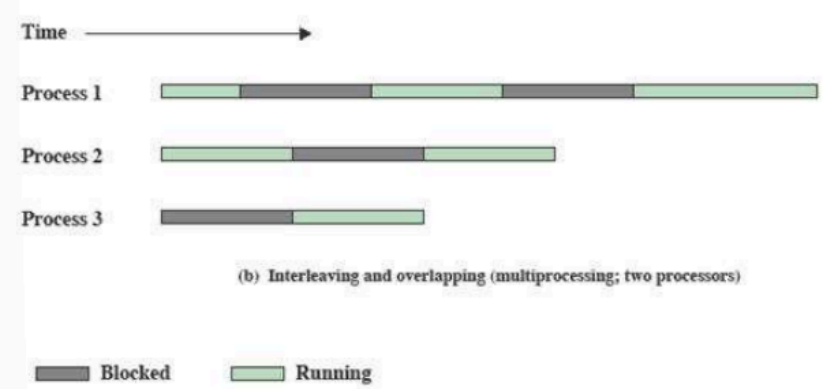



Figure 2.12 Multiprogramming and Multiprocessing

- Baik secara interleaving maupun secara overlapping keduanya menimbulkan masalah yaitu:

a. Sharing of global resource → Apabila 2 proses atau lebih menggunakan global variabel yang sama dan keduanya melakukan proses read dan write pada variabel tersebut. Maka urutan berjalan proses berpengaruh terhadap variabel itu
b. Optimally managing the allocation of resources → OS sulit untuk mengalokasikan resource secara optimal. Sebagai contoh apabila proses A meminta akses ke I/O channel maka proses lain tidak dapat mengakses channel itu, hal ini dapat menyebabkan kondisi deadlock.
c. Difficult to locate programming errors as results are not deterministic and reproducible (sulit untuk menemukan dan memperbaiki kesalahan dalam program karena hasilnya tidak selalu bisa diprediksi atau direproduksi) → karena banyak proses yang bisa saling berinteraksi hal ini sulit untuk menentukan bagian yang error dan memunculkan error itu kembali

- Contoh multiprocessor (di buku hal pdf 222 ada contoh kalau uniprocessor)

| Process P1 | Process P2 |
|---|---|
| • | • |
| `chin = getchar();` | • |
| • | `chin = getchar()` |
| `chout = chin;` | `chout = chin;` |
| `putchar(chout);` | • |
| • | `putchar(chout);` |
| • | • |

Intinya karena chin itu variabel global yang diakses process 1 dan process 2, maka waktu diprint P1 maupun P2 hasilnya adalah karakter dari P2, karena P2 baca input terakhir.

- Salah satu cara mengatasinya adalah dengan menetapkan aturan **hanya 1 process yang boleh mengakses fungsi dalam 1 waktu**:
  a. P1 & P2 berjalan dalam processor yang berbeda
  b. P1 memasuki fungsi echo terlebih dahulu
  c. P2 berusaha memasuki fungsi echo tetapi diblokir → P2 pindah ke states suspend
  d. P1 melanjutkan hingga eksekusi selesai
  e. P2 melanjutkan dan eksekusi hingga selesai
- Race condition
  a. Kondisi ketika multiple process atau thread membaca dan menulis pada data yang sama
  b. Hasilnya bergantung pada urutan eksekusi proses atau thread
  c. Output bergantung ke siapa yang menyelesaikan race lebih dahulu
- Fokus desain OS untuk menghindari masalah concurrency:

a. OS harus bisa keep track dari berbagai macam proses. Hal ini diselesaikan dengan menggunakan process control block di bab sebelumnya.
b. OS harus bisa allocate dan deallocate resource pada proses yang aktif, resource tersebut meliputi:
    - Processor time → diselesaikan dengan scheduling function
    - Memory → menggunakan virtual memory scheme
    - Files
    - I/O Devices
c. OS harus bisa melindungi data dan resource setiap proses dari intervensi proses lain
d. Memastikan bahwa proses dan hasilnya **tidak bergantung pada kecepatan pemrosesan**

- Interaksi antar proses

| Degree of Awareness | Relationship | Influence that One Process Has on the Other | Potential Control Problems |
|---|---|---|---|
| Processes unaware of each other | Competition | • Results of one process independent of the action of others<br>• Timing of process may be affected | • Mutual exclusion<br>• Deadlock (renewable resource)<br>• Starvation |
| Processes indirectly aware of each other (e.g., shared object)<br>contohnya I/O buffer | Cooperation by sharing | • Results of one process may depend on information obtained from others<br>• Timing of process may be affected | • Mutual exclusion<br>• Deadlock (renewable resource)<br>• Starvation<br>• Data coherence |
| Processes directly aware of each other (have communication primitives available to them) | Cooperation by communication | • Results of one process may depend on information obtained from others<br>• Timing of process may be affected | • Deadlock (consumable resource)<br>• Starvation |

- Kompetisi antar proses untuk mendapatkan resource memiliki tiga masalah utama:
    a. Kebutuhan Mutual Exclusion (critical resource)
    Masalah ini muncul jika banyak proses mengakses ke satu I/O Device yang tidak dapat dibagi, contohnya printer. Resource seperti ini disebut **critical resource** dan program yang menggunakannya disebut **critical section**. Hanya 1 program dalam 1 waktu yang harus dapat mengakses critical section.
    b. Deadlock
    Contohnya ketika ada 2 proses: P1 dan P2 dan 2 resource: R1 dan R2. Setiap proses membutuhkan akses ke kedua resource. OS memberikan R1 ke P1 (artinya P1 masih butuh R2) dan R2 ke P2 (masih butuh R1). Kedua proses tidak dapat berjalan dan saling menunggu karena butuh 2 resource. Kedua proses tersebut deadlocked
    c. Starvation

Sebagai contoh ada 3 proses, P1, P2, dan P3. Seluruh proses harus mengakses critical resource R. OS memberi ke P1 terlebih dahulu, setelah itu memberi akses ke P3 sebelum P1 selesai. Setelah itu, OS memberi lagi ke P1, dan berlanjut ke P3. OS terus memberi bergantian ke P1 dan P3 sehingga P2 tidak berjalan dan tidak mendapat resource tersebut.

- Suatu mutual exclusion harus berjalan dengan syarat berikut:
  a. Hanya satu proses yang boleh mengakses critical section
  b. Suatu proses yang berhenti/selesai di luar critical section tidak boleh mengganggu proses lain
  c. Tidak boleh ada deadlock atau starvation
  d. Ketika tidak ada process di dalam critical section, setiap process harus yang ingin mengakses critical section harus masuk tanpa delay.
  e. Tidak boleh ada asumsi tentang kecepatan proses atau jumlah processor
  f. Suatu proses di dalam critical section hanya untuk waktu tertentu (tidak boleh gajelas waktunya berapa lama)

## II. Mutual Exclusion: Hardware Support

Pendekatan mutex dengan hardware support

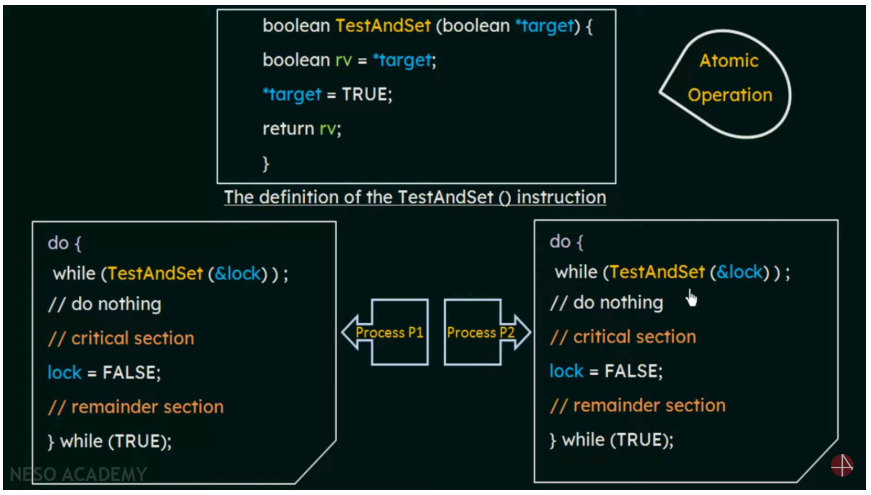
A. Interrupt Disabling
   - Digunakan hanya pada **<u>uniprocessor system</u>** karena tidak mungkin terjadi overlapped dan hanya mungkin interleaved.
   - Suatu proses pada uniprocessor akan berjalan terus hingga diinterrupt atau memanggil OS service
   - Mutex bisa terjadi jika proses tidak diinterupt selama memasuki critical section
   - Menghilangkan interrupt menjamin terjadinya mutex karena process lain tidak mungkin dijalankan
   - Pseudo-code:

```
while (true) {
    /* disable interrupts */;
    /* critical section */;
    /* enable interrupts */;
    /* remainder */;
}
```

B. Special Machine Instructions
   Cara agar multiprocessor bisa berjalan
   Ada 3 cara:
   1. Compare & Swap Instruction / Test & Set Lock
      Bedanya compare & swap sama test & set lock ini kalo compare & swap return alamat, kalau test & set lock return boolean

Test & Set Lock:

- Proses yang mengakses critical section akan berbagi variabel boolean lock yang sama (bernilai true/1 atau false/0)
- Jika suatu proses memasuki critical section, nilainya akan menjadi true/1 (locked)
- Ketika nilai true dari lock maka proses lain tidak akan dapat memasuki critical section
- Ketika proses di dalam critical section keluar, nilai dari variabel akan menjadi false/0 kembali sehingga proses lain bisa masuk
- Pseudo-code:




- Di sini itu, gampangnya kalau TestAndSet return nilai 0 artinya unlocked, proses bisa masuk ke critical section. Waktu masuk ini, dari fungsi TestAndSet nilai dari locknya dibuat true (locked). Nah waktu dia udah selesai dari critical section, nilai lock dikembalikan ke false (unlocked)

Compare & Swap Instruction

- Compare and swap ini mirip kayak test & set lock cuma beda di nilainya bukan boolean
  https://www.youtube.com/watch?v=5oZYS5dTrmk

```
int compare_and_swap (int *word, int testval, int newval)
{
    int oldval;
    oldval = *word
    if (oldval == testval) *word = newval;
    return oldval;
}
```

```
/* program mutualexclusion */
const int n = /* number of processes */;
int bolt;
void P(int i)
{
   while (true) {
     while (compare_and_swap(bolt, 0, 1) == 1)
         /* do nothing */;
     /* critical section */;
     bolt = 0;
     /* remainder */;
   }
}
void main()
{
   bolt = 0;
   parbegin (P(1), P(2), ... ,P(n));

}
```

- Ini tu anggapannya bolt itu sama kayak variabel lock di atas.
- Kalau nilai dari bolt harus sama kayak testval (argumen pertama == argumen kedua) maka berarti proses bisa masuk dan nilai dari bolt akan digantikan sama newval.
- Setelah critical section berjalan, nilai bolt bakal dikembalikan sesuai testval

2. Exchange Instruction

```
void exchange (int *register, int *memory)
{
   int temp;
   temp = *memory;
   *memory = *register;
   *register = temp;
}
```

```
/* program mutualexclusion */
int const n = /* number of processes */;
int bolt;
void P(int i)
{
   int keyi = 1;
   while (true) {
      do exchange (&keyi, &bolt)
      while (keyi != 0);
      /* critical section */;
      bolt = 0;
      /* remainder */;
   }
}
void main()
{
   bolt = 0;
   parbegin (P(1), P(2), ..., P(n));
}
```

- Exchange ini juga mirip sama yang sebelumnya, jadi nilai keyi ini bakal selalu diganti sama nilai bolt. Nah nilai keyi harus sesuai sama permintaan (kalo contoh di atas itu 0)

# Pertemuan 5: Semaphore

Semaphore/counting semaphore/general semaphore adalah suatu nilai integer yang digunakan untuk memberi sinyal pada proses.
Ada 3 operasi yang bisa dilakukan pada semaphore dan semuanya bersifat atomic (tidak dapat dipisahkan):

a. initialize
b. increment (semSignal) → unblocking process
c. decrement (semWait) → blocking process

https://chat.openai.com/share/772a83bd-559d-475e-9b8c-2160e1d2e59a (ini pseudocode kek di buku)
https://www.youtube.com/watch?v=XDIOC2EY5JE
https://www.youtube.com/watch?v=2cGo2HdA0dM