
# Arsitektur dan Organisasi Komputer
## COM 60011

### Bab #8 – Dukungan Sistem Operasi

# Bab 8

## Dukungan Sistem Operasi

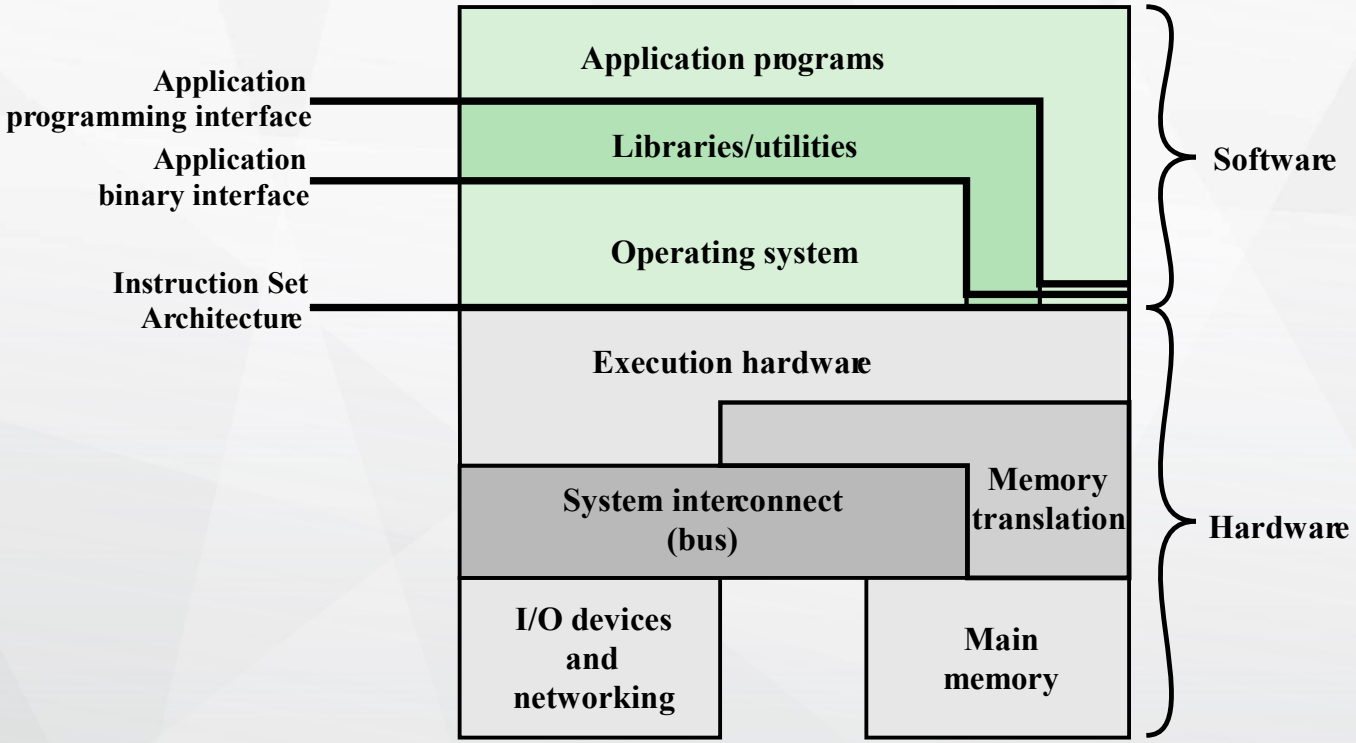



**Figure 8.1 Computer Hardware and Software Structure**

# Layanan Sistem Operasi (OS)

- **Program sistem yang paling penting**
- **Menyamarkan detail perangkat keras dari pemrogram dan menyediakan pemrogram antarmuka yang nyaman untuk menggunakan sistem**
- **OS biasanya menyediakan layanan di bidang berikut:**
  - Pembuatan program
  - Eksekusi program
  - Akses ke perangkat I / O
  - Akses terkontrol ke file
  - Akses sistem
  - Deteksi dan respons kesalahan
  - Akuntansi

# Antarmuka

## Antarmuka utama dalam sistem komputer biasa:

### Instruction set architecture (ISA)

- Menentukan instruksi bahasa mesin yang bisa diikuti komputer
- Batas antara perangkat keras dan perangkat lunak

### Application binary interface (ABI)

- Mendefinisikan standar untuk portabilitas biner di seluruh program
- Mendefinisikan antarmuka panggilan sistem ke sistem operasi dan sumber daya dan layanan perangkat keras yang tersedia dalam sistem melalui ISA pengguna

### Application programming interface (API)

- Memberikan akses program ke sumber daya dan layanan perangkat keras yang tersedia dalam sistem melalui ISA pengguna yang dilengkapi dengan panggilan pustaka bahasa tingkat tinggi (HLL)
- Memberikan akses program ke sumber daya dan layanan perangkat keras yang tersedia dalam sistem melalui ISA pengguna yang dilengkapi dengan panggilan pustaka bahasa tingkat tinggi (HLL)

# Sistem operasi sebagai Manajer Sumber Daya

**Komputer adalah sekumpulan sumber daya untuk pergerakan, penyimpanan, dan pemrosesan data dan untuk mengontrol fungsi-fungsi ini**

- OS bertanggung jawab untuk mengelola sumber daya ini

**OS sebagai mekanisme kontrol tidak biasa dalam dua hal:**

- OS berfungsi dengan cara yang sama seperti perangkat lunak komputer biasa – ini adalah program yang dijalankan oleh prosesor
- OS sering kali melepaskan kendali dan harus bergantung pada prosesor untuk memungkinkannya mendapatkan kembali kendali

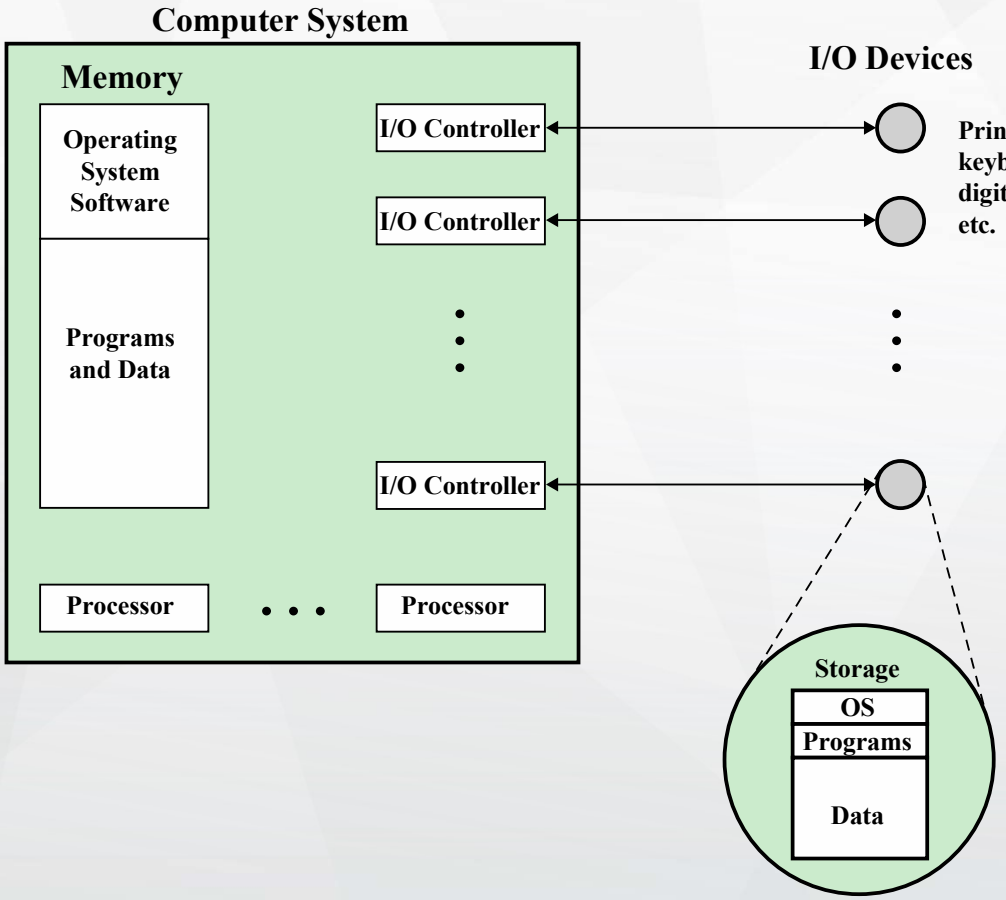



**Figure 8.2 The Operating System as Resource Manager**

# Jenis Sistem Operasi

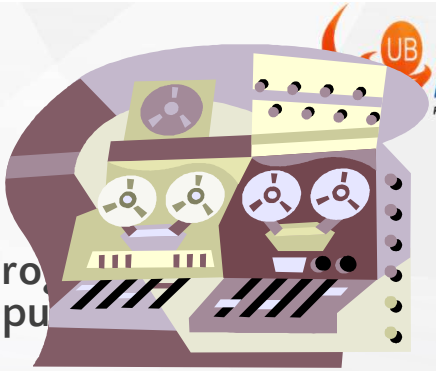
## Sistem interaktif

- Pengguna / pemrogram berinteraksi langsung dengan komputer untuk meminta pelaksanaan suatu pekerjaan atau untuk melakukan transaksi
- Pengguna dapat, tergantung pada sifat aplikasi, berkomunikasi dengan komputer selama pelaksanaan pekerjaan

## Sistem batch

- Kebalikan dari interaktif
- Program pengguna digabungkan dengan program dari pengguna lain dan dikirimkan oleh operator komputer
- Setelah program selesai, hasil dicetak untuk pengguna

# Sistem Komputer Generasi Awal



**Dari akhir 1940-an hingga pertengahan 1950-an pemro berinteraksi langsung dengan perangkat keras kompu ada OS**

Prosesor dijalankan dari konsol yang terdiri dari lampu layar, sakelar sakelar, beberapa bentuk perangkat input dan printer

**Masalah:**

Penjadwalan

- Lembar pendaftaran digunakan untuk memesan waktu prosesor
  - Hal ini dapat mengakibatkan waktu komputer menganggur terbuang percuma jika pengguna selesai lebih awal
  - Jika masalah terjadi, pengguna dapat dipaksa untuk berhenti sebelum menyelesaikan masalah

Waktu penyetelan

- Satu program bisa melibatkan
  - Memuat kompilator plus program sumber ke dalam memori
  - Menyimpan program yang telah dikompilasi
  - Memuat dan menghubungkan bersama program objek dan fungsi umum

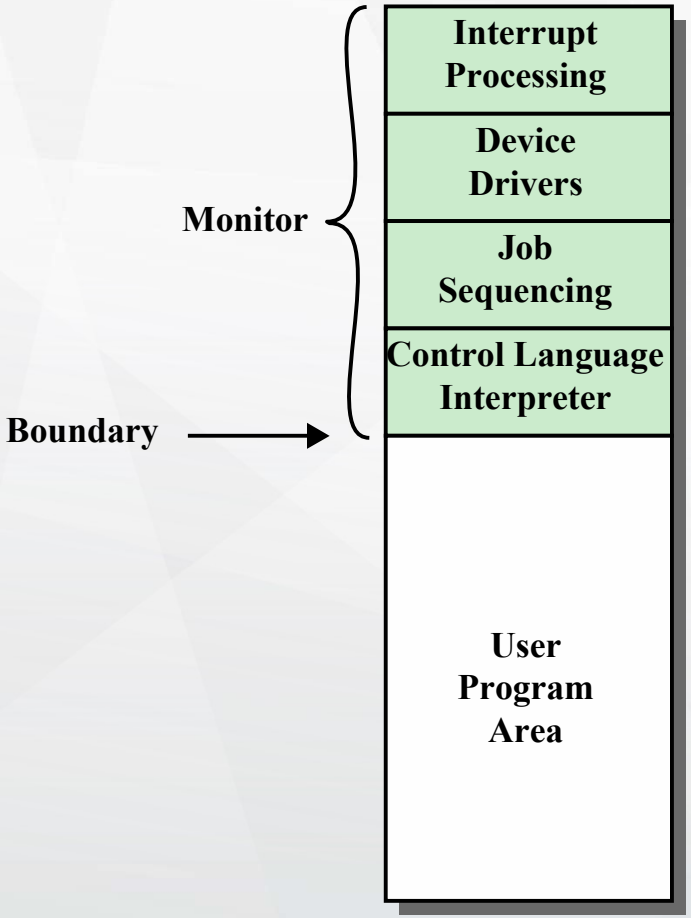



**Figure 8.3 Memory Layout for a Resident Monitor**

# Dari Tampilan Prosesor. . .

**Prosesor menjalankan instruksi dari bagian memori utama yang berisi monitor**

- Instruksi ini menyebabkan pekerjaan berikutnya dibaca di bagian lain dari memori utama
- Prosesor menjalankan instruksi dalam program pengguna hingga menemui kondisi akhir atau kesalahan
- Salah satu peristiwa menyebabkan prosesor mengambil instruksi berikutnya dari program monitor

**Monitor menangani pengaturan dan penjadwalan**

- Sekumpulan pekerjaan diantrekan dan dieksekusi secepat mungkin tanpa waktu idle

**Bahasa kontrol pekerjaan (JCL)**

- Jenis bahasa pemrograman khusus digunakan untuk memberikan instruksi ke monitor

**Contoh:**

```
$ PEKERJAAN
$ FTN
... Beberapa instruksi Fortran
$ BEBAN
$ RUN
... Beberapa data
$ AKHIR
```

** Setiap instruksi FORTRAN dan setiap item data ada pada kartu berlubang terpisah atau rekaman terpisah pada kaset. Selain FORTRAN dan jalur data, pekerjaan tersebut mencakup instruksi kontrol pekerjaan, yaitu dilambangkan dengan awal "$".

Monitor, atau batch OS, hanyalah sebuah program komputer

- Ini bergantung pada kemampuan prosesor untuk mengambil instruksi dari berbagai bagian memori utama untuk mengambil dan melepaskan kendali secara bergantian

# Fitur Perangkat Keras yang Diinginkan

## Perlindungan memori

- Program pengguna tidak boleh mengubah area memori yang berisi monitor
- Perangkat keras prosesor harus mendeteksi kesalahan dan mentransfer kontrol ke monitor
- Monitor membatalkan pekerjaan, mencetak pesan kesalahan, dan memuat pekerjaan berikutnya

## Instruksi yang diistimewakan

- Hanya bisa dijalankan oleh monitor
- Jika prosesor menemukan instruksi seperti itu saat menjalankan program pengguna, terjadi interupsi kesalahan
- Instruksi I/O memiliki hak istimewa sehingga monitor tetap mengontrol semua perangkat I/O

## Timer

- Digunakan untuk mencegah pekerjaan memonopoli sistem
- Jika pengatur waktu berakhir, terjadi interupsi dan kontrol kembali ke monitor

## Interupsi

- Memberi OS lebih banyak fleksibilitas dalam melepaskan kendali dan mendapatkan kembali kendali dari program pengguna

| | |
|---|---|
| Read one record from file | 15 $\mu$s |
| Execute 100 instructions | 1 $\mu$s |
| Write one record to file | 15 $\mu$s |
| TOTAL | 31 $\mu$s |

$$\text{Percent CPU utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

**Figure 8.4 System Utilization Example**

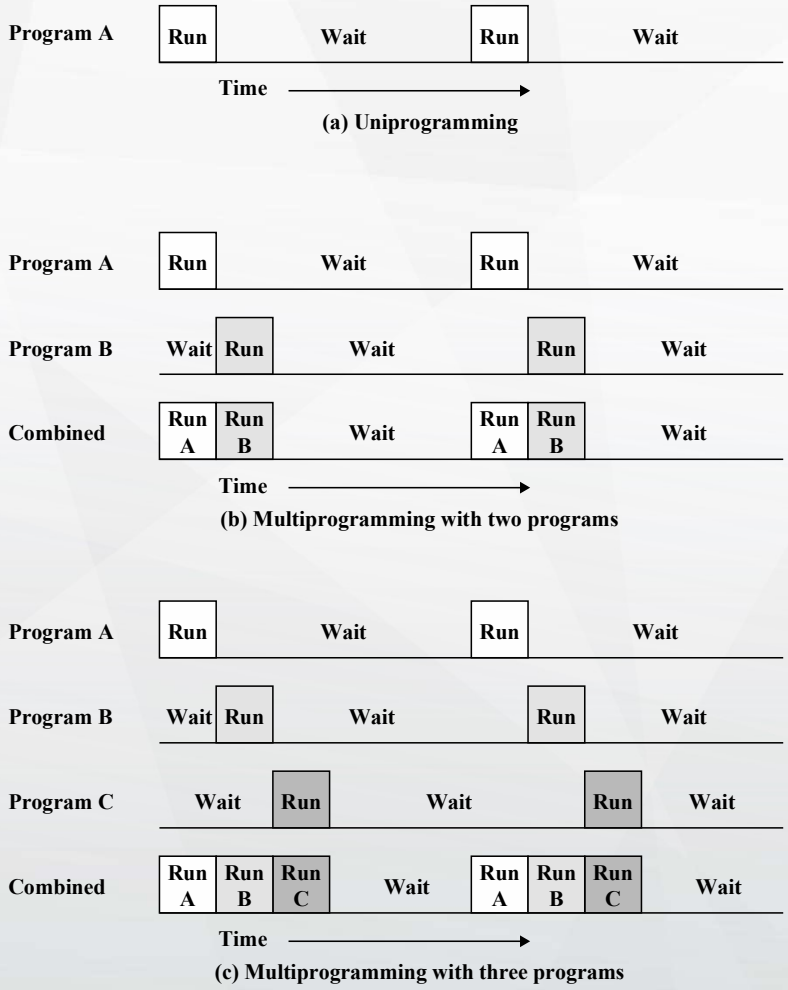



**Figure 8.5 Multiprogramming Example**

# Tabel 8.1
# Contoh Atribut Eksekusi Program

| | JOB1 | JOB2 | JOB3 |
|---|---|---|---|
| **Type of job** | Heavy compute | Heavy I/O | Heavy I/O |
| **Duration** | 5 min | 15 min | 10 min |
| **Memory required** | 50 M | 100 M | 80 M |
| **Need disk?** | No | No | Yes |
| **Need terminal?** | No | Yes | No |
| **Need printer?** | No | No | Yes |

# Tabel 8.2
# Pengaruh Multiprogramming pada Pemanfaatan Sumber Daya

| | Uniprogramming | Multiprogramming |
|---|---|---|
| **Processor use** | 20% | 40% |
| **Memory use** | 33% | 67% |
| **Disk use** | 33% | 67% |
| **Printer use** | 33% | 67% |
| **Elapsed time** | 30 min | 15 min |
| **Throughput rate** | 6 jobs/hr | 12 jobs/hr |
| **Mean response time** | 18 min | 10 min |

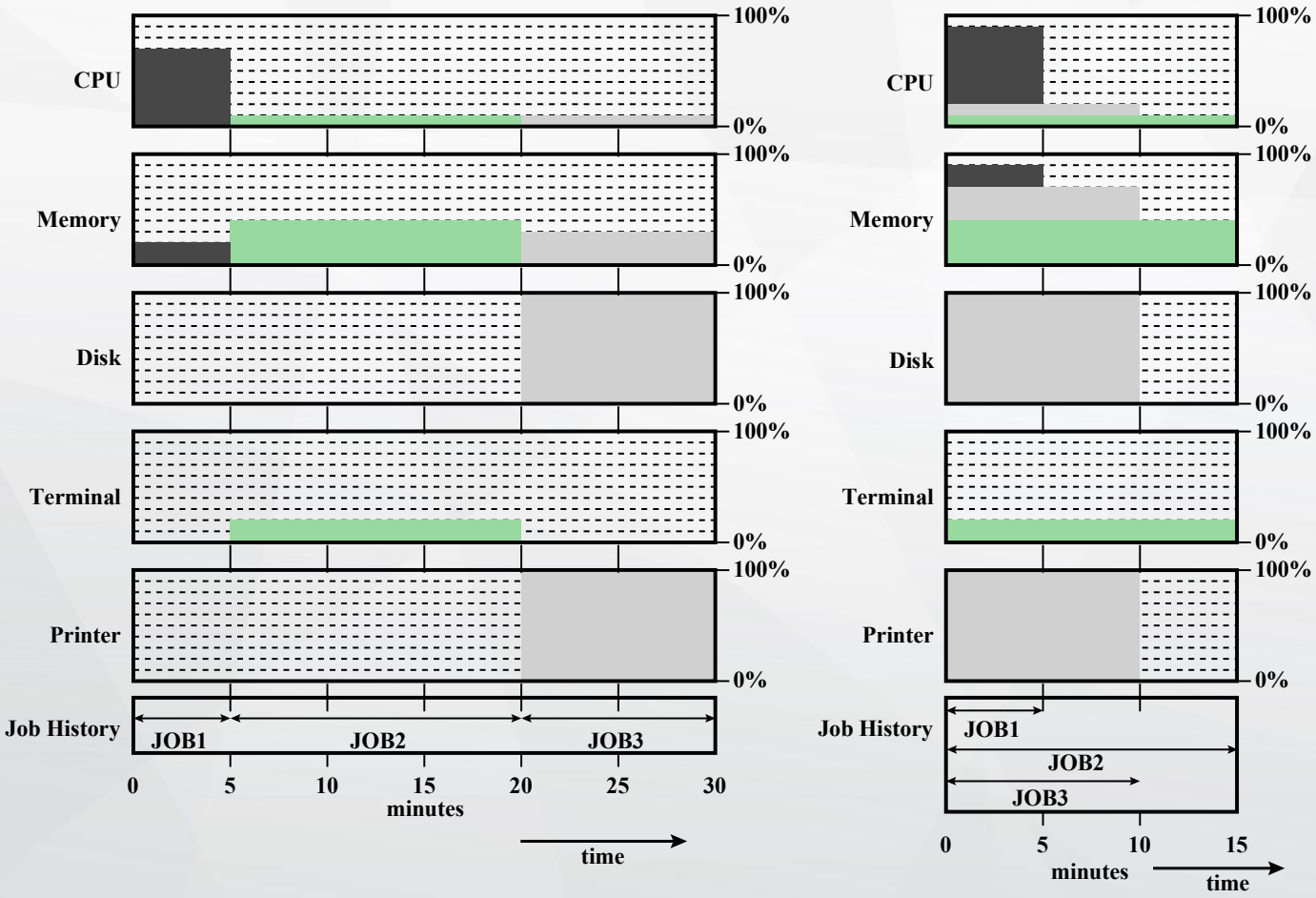



**Figure 8.6 Utilization Histograms**

# Sistem Pembagian Waktu (Time Sharing)

- **Digunakan saat pengguna berinteraksi langsung dengan komputer**
- **Waktu prosesor dibagi di antara banyak pengguna**
- **Beberapa pengguna secara bersamaan mengakses sistem melalui terminal, dengan OS menyisipkan eksekusi setiap program pengguna dalam ledakan singkat atau komputasi kuantum**
- **Contoh:**
  - Jika ada *n* pengguna aktif meminta layanan pada satu waktu, setiap pengguna hanya akan melihat rata-rata 1 /*n* kecepatan komputer efektif

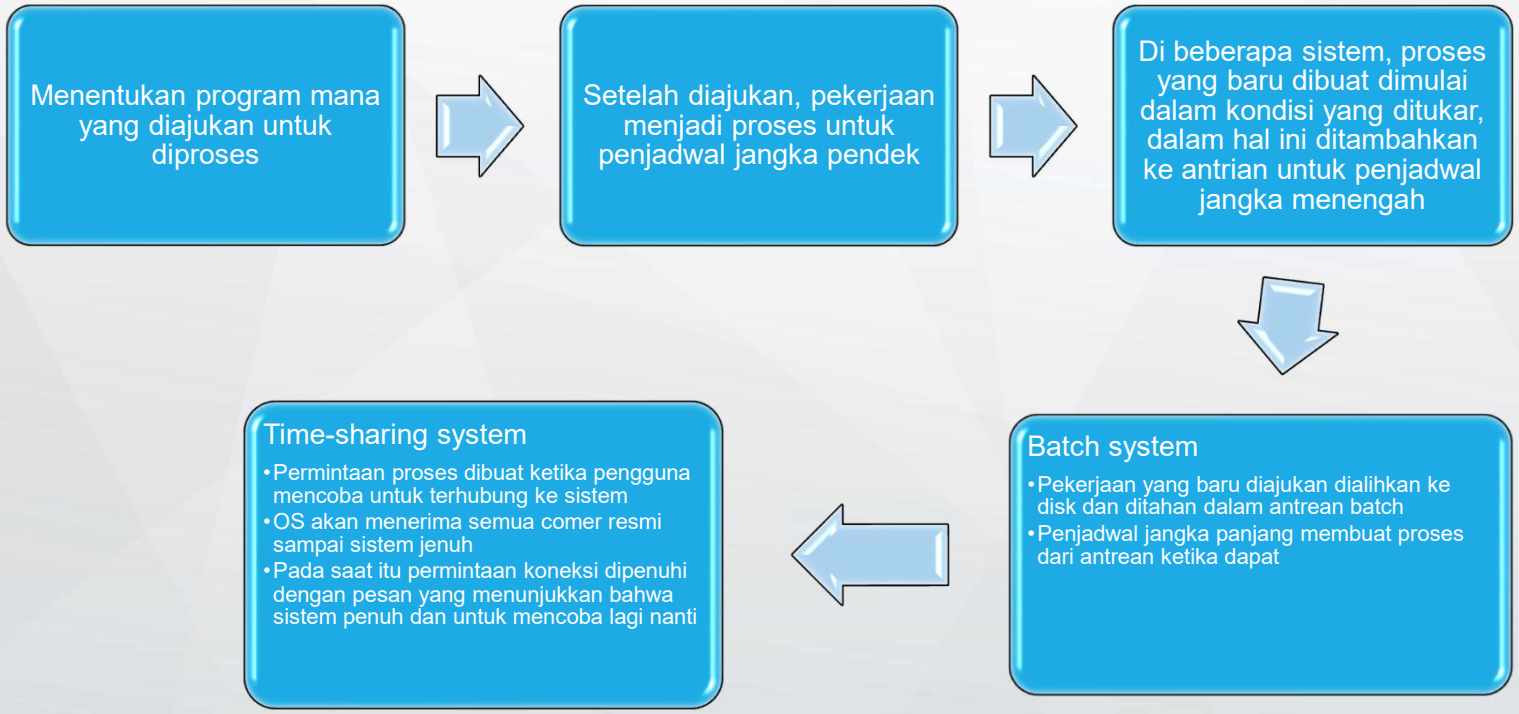
# Tabel 8.3
# Batch Multiprogramming versus Time Sharing

| | Batch Multiprogramming | Time Sharing |
|---|---|---|
| Principal objective | Maximize processor use | Minimize response time |
| Source of directives to operating system | Job control language commands provided with the job | Commands entered at the terminal |

## Table 8.4 Types of Scheduling

| | |
|---|---|
| **Long-term scheduling** | The decision to add to the pool of processes to be executed |
| **Medium-term scheduling** | The decision to add to the number of processes that are partially or fully in main memory |
| **Short-term scheduling** | The decision as to which available process will be executed by the processor |
| **I/O scheduling** | The decision as to which process's pending I/O request shall be handled by an available I/O device |

# Penjadwalan Jangka Panjang



# Penjadwalan Jangka Menengah dan Penjadwalan Jangka Pendek

Medium-Term

Bagian dari fungsi swapping

Keputusan swap-in didasarkan pada kebutuhan untuk mengelola derajat multiprogramming

Keputusan swap-in akan mempertimbangkan persyaratan memori dari proses yang ditukar

## Jangka pendek

**Juga dikenal sebagai dispatcher**

**Sering mengeksekusi dan membuat keputusan mendetail tentang tugas mana yang akan dijalankan berikutnya**

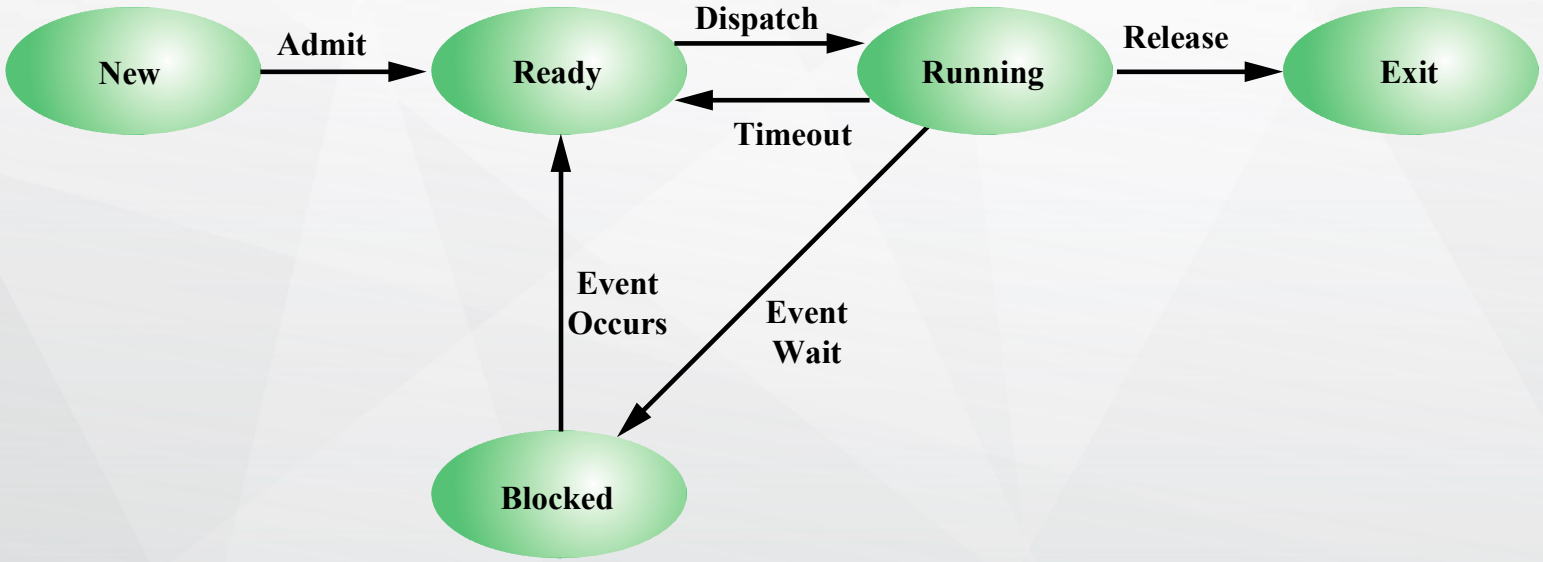



**Figure 8.7 Five-State Process Model**

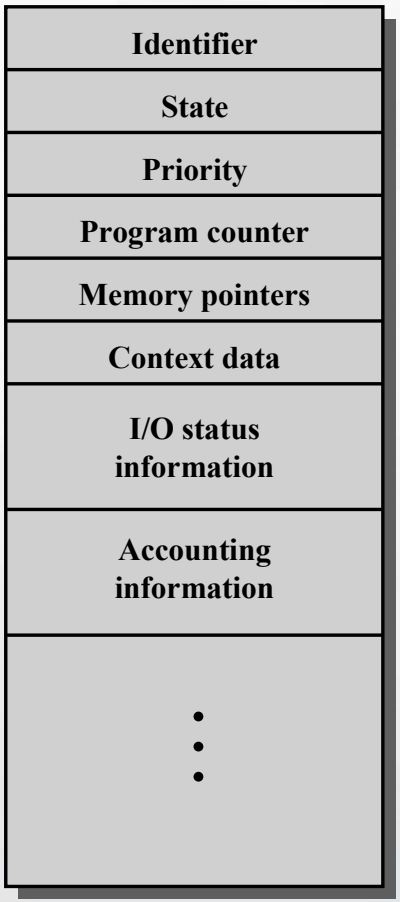



**Figure 8.8 Process Control Block**

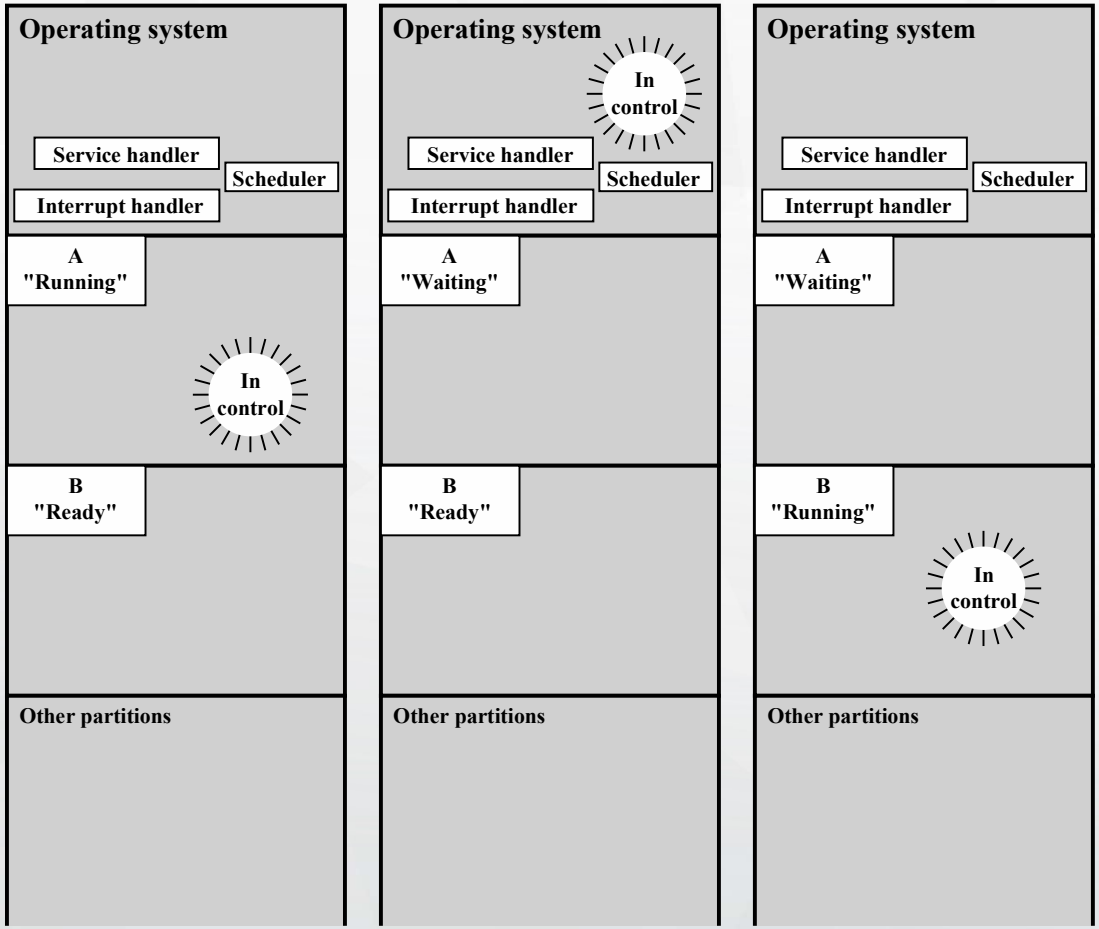



**Figure 8.9 Scheduling Example**

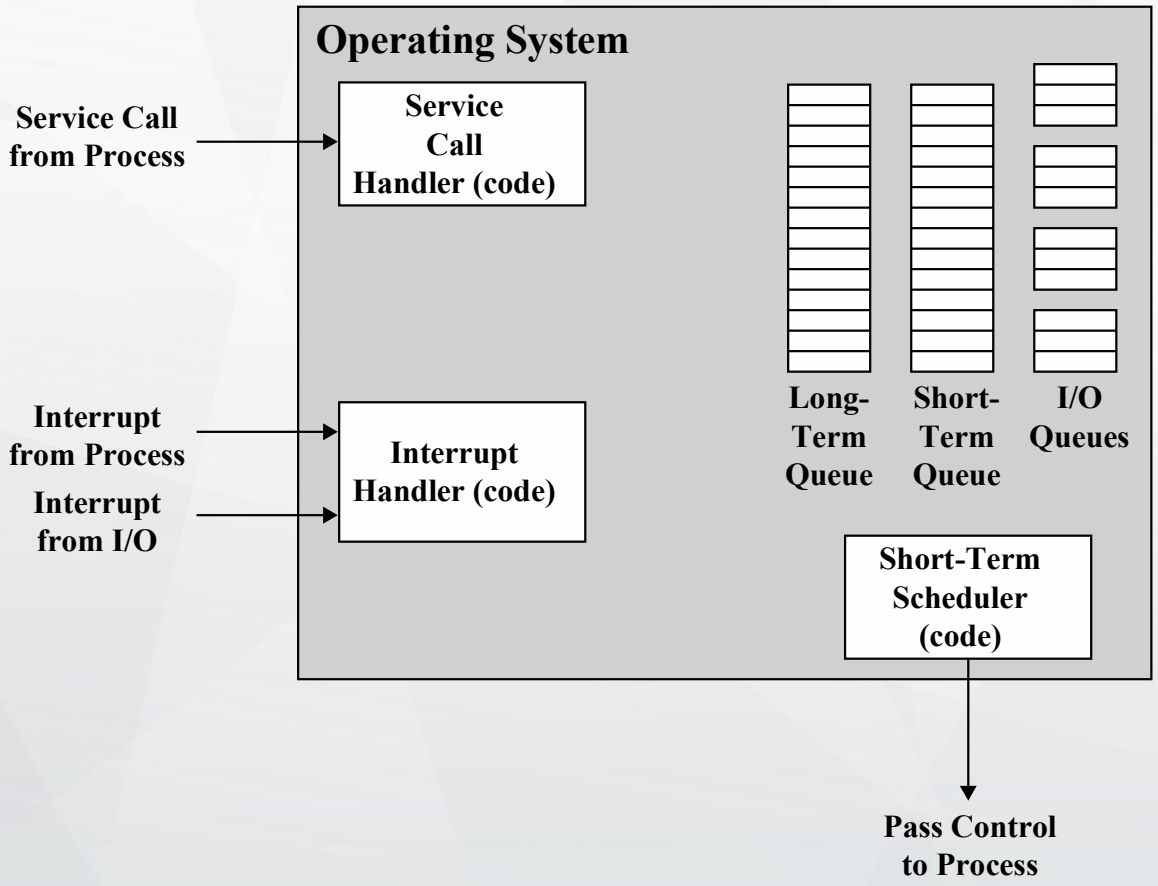



**Figure 8.10 Key Elements of an Operating System for Multiprogramming**

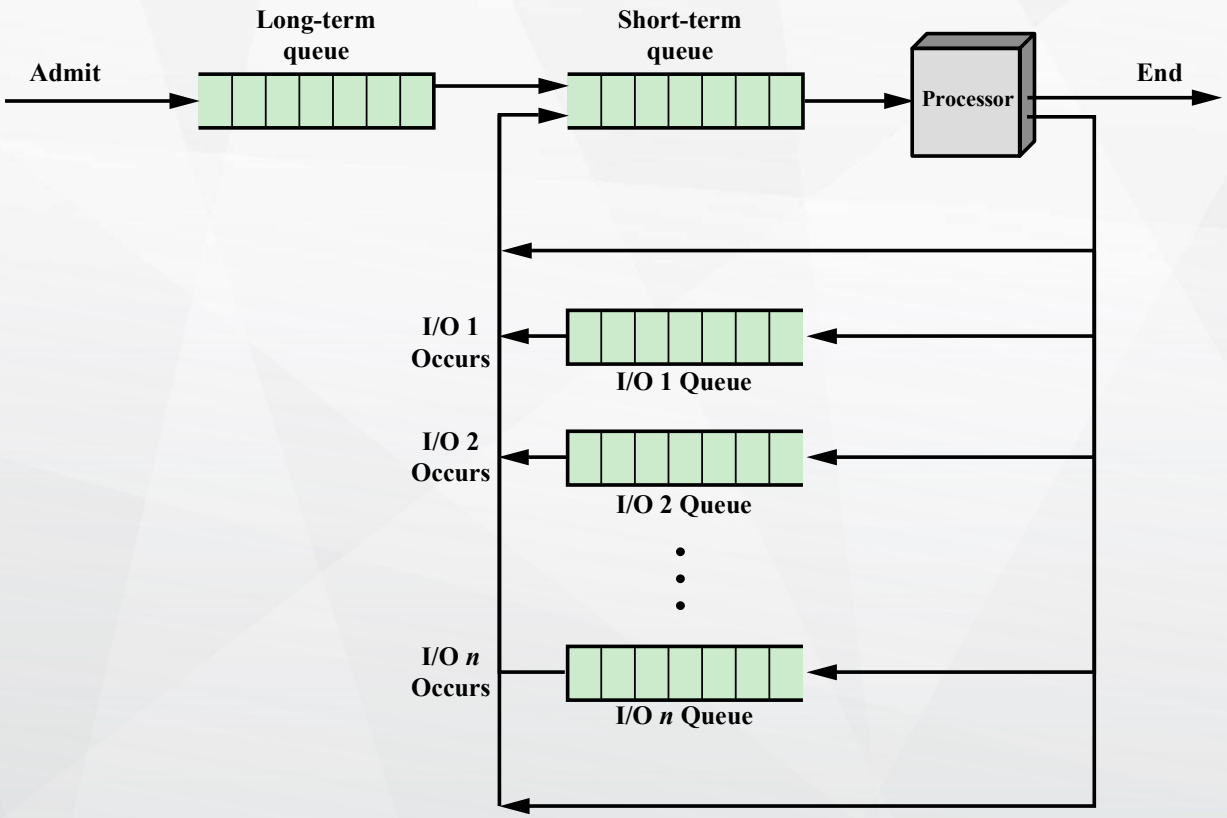



**Figure 8.11 Queuing Diagram Representation of Processor Scheduling**

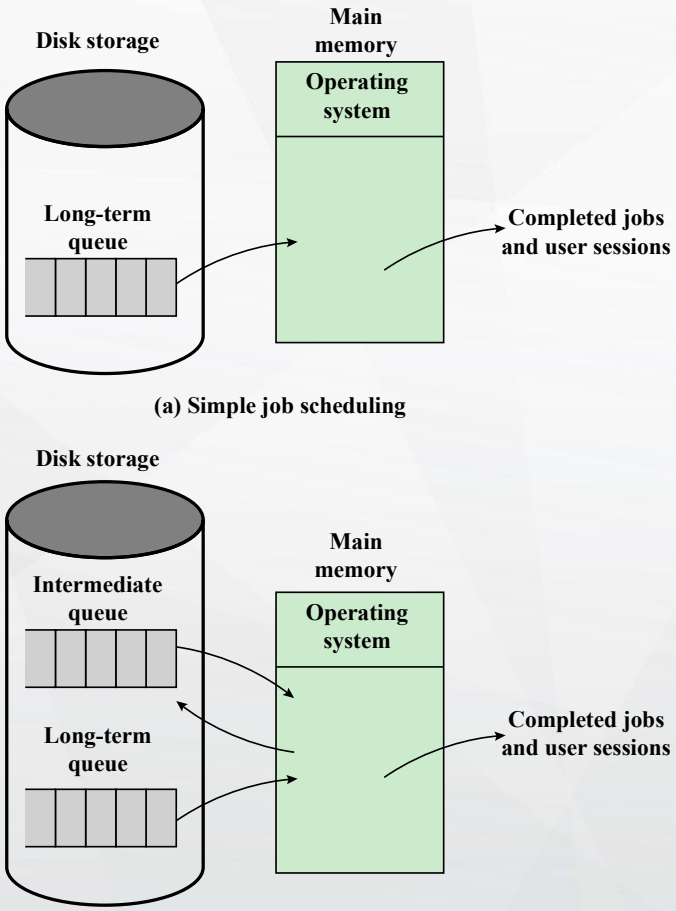
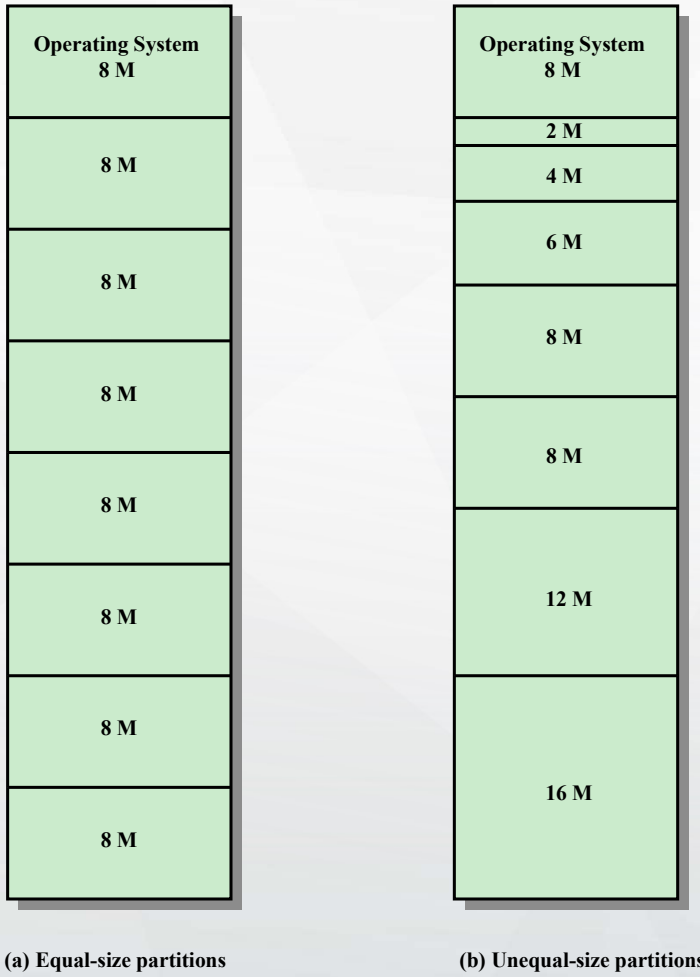



**Figure 8.12 The Use of Swapping**




(a) Equal-size partitions

(b) Unequal-size partitions

**Figure 8.13 Example of Fixed Partitioning of a 64-Mbyte Memory**

**Alamat logis**
- Dinyatakan sebagai lokasi relatif terhadap awal program

**Alamat fisik**
- lokasi sebenarnya di memori utama

**Alamat dasar**
- lokasi awal proses saat ini

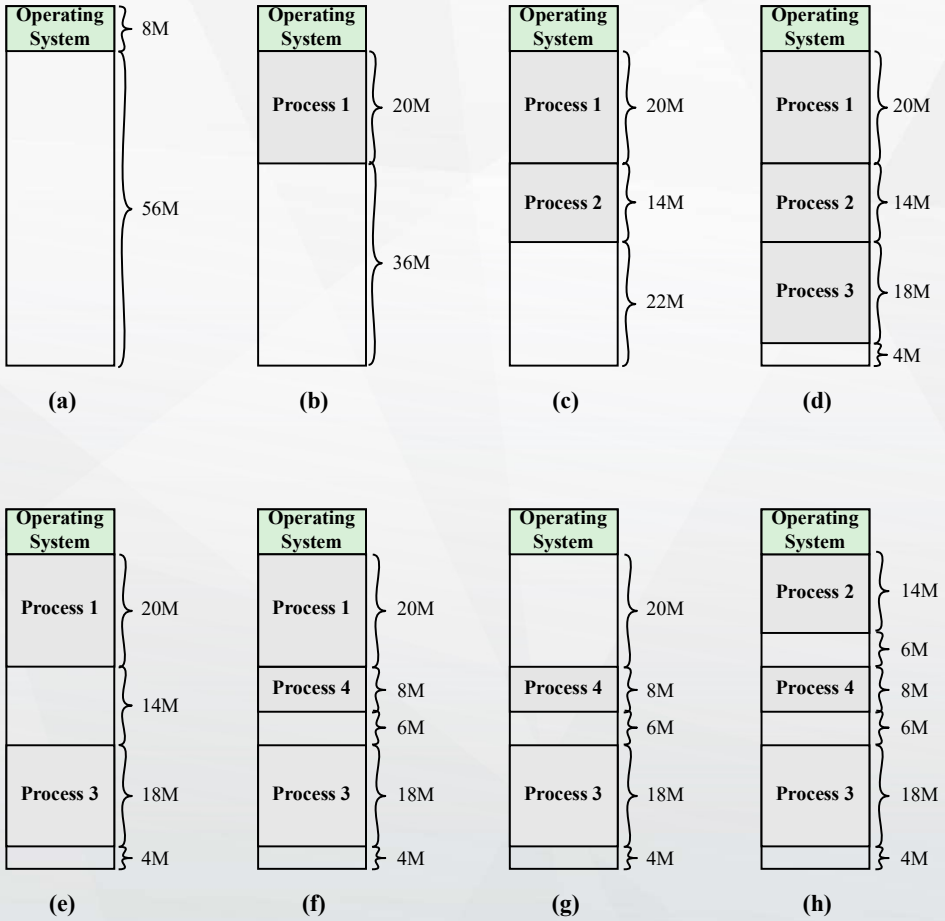



**Figure 8.14 The Effect of Dynamic Partitioning**

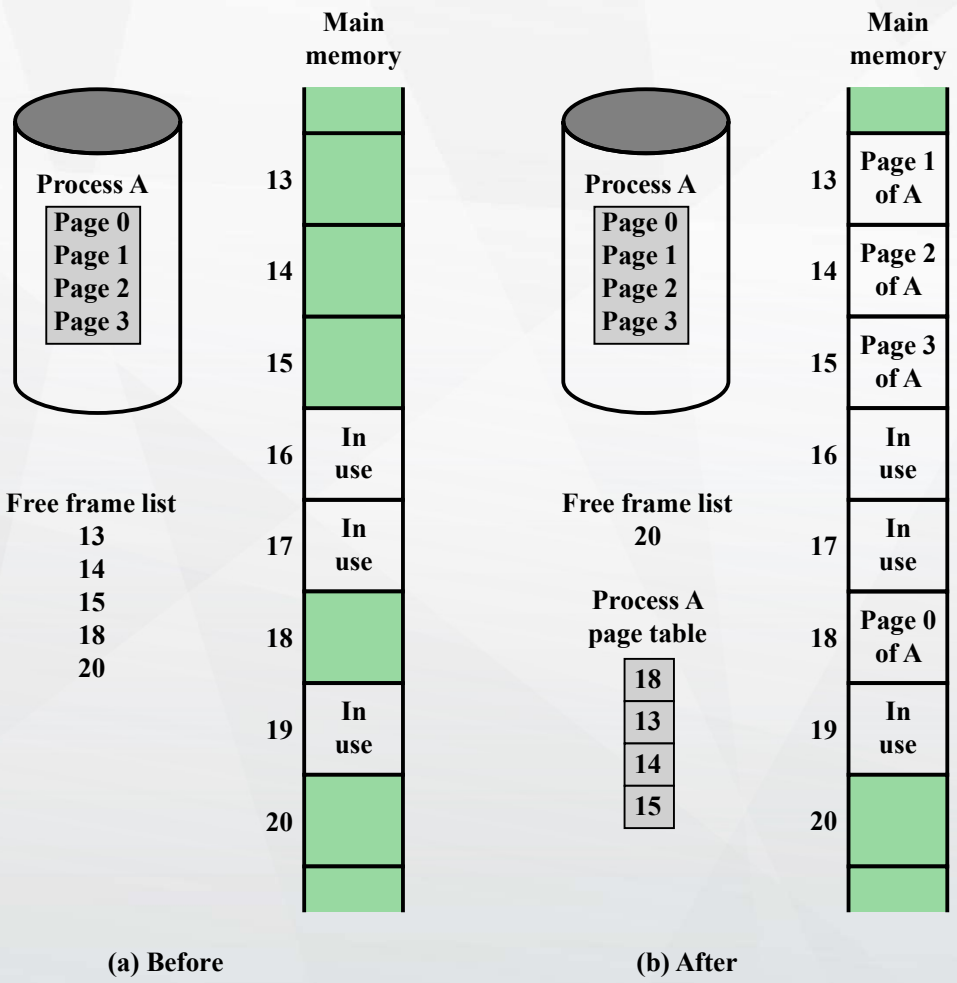



**Figure 8.15 Allocation of Free Frames**

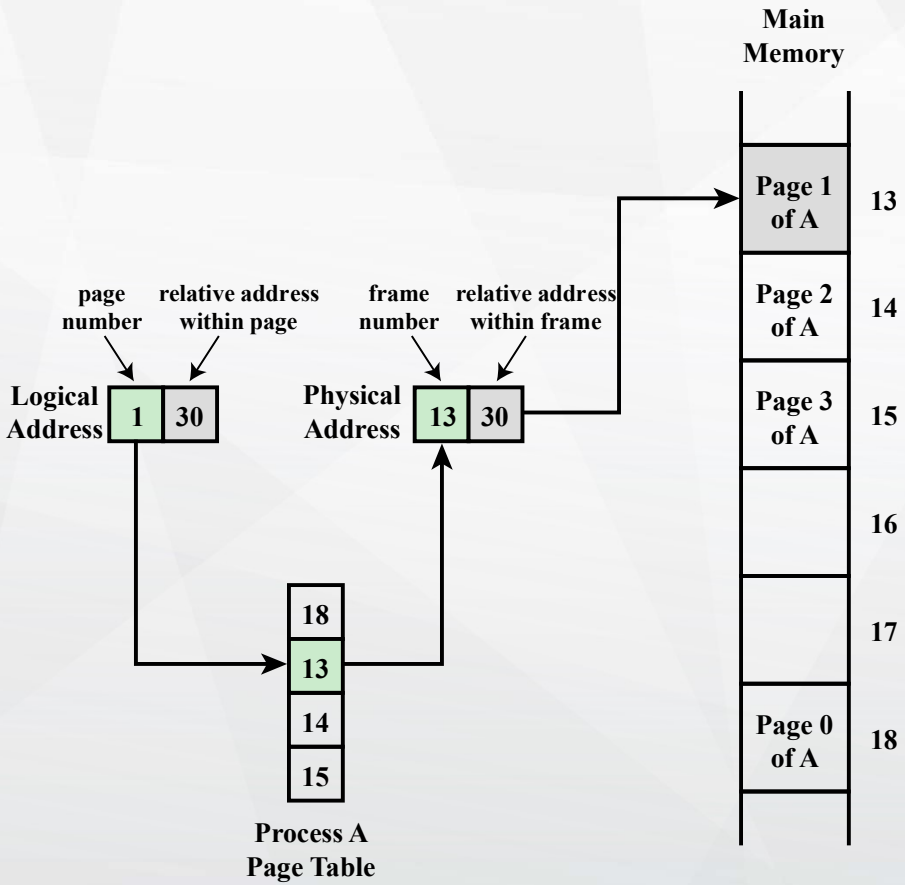



**Figure 8.16 Logical and Physical Addresses**

# Memori Virtual

## Demand Paging

**Setiap halaman proses dibawa masuk hanya saat dibutuhkan**

**Prinsip lokalitas**

Saat bekerja dengan proses yang besar, eksekusi mungkin terbatas pada bagian kecil dari program (subrutin)

Lebih baik menggunakan memori untuk memuat hanya dalam beberapa halaman

Jika program mereferensikan data atau cabang ke instruksi pada halaman yang tidak ada dalam memori utama, a *kesalahan halaman* dipicu yang memberi tahu OS untuk menampilkan halaman yang diinginkan

**Keuntungan:**

Lebih banyak proses dapat dipertahankan dalam memori

Waktu disimpan karena halaman yang tidak digunakan tidak ditukar masuk dan keluar dari memori

**Kekurangan:**

Ketika satu halaman dibawa masuk, halaman lain harus dibuang (*penggantian halaman)*

Jika sebuah halaman dibuang tepat sebelum akan digunakan OS harus membuka halaman itu lagi

*Labrakan*

- Ketika prosesor menghabiskan sebagian besar waktunya untuk bertukar halaman daripada menjalankan instruksi

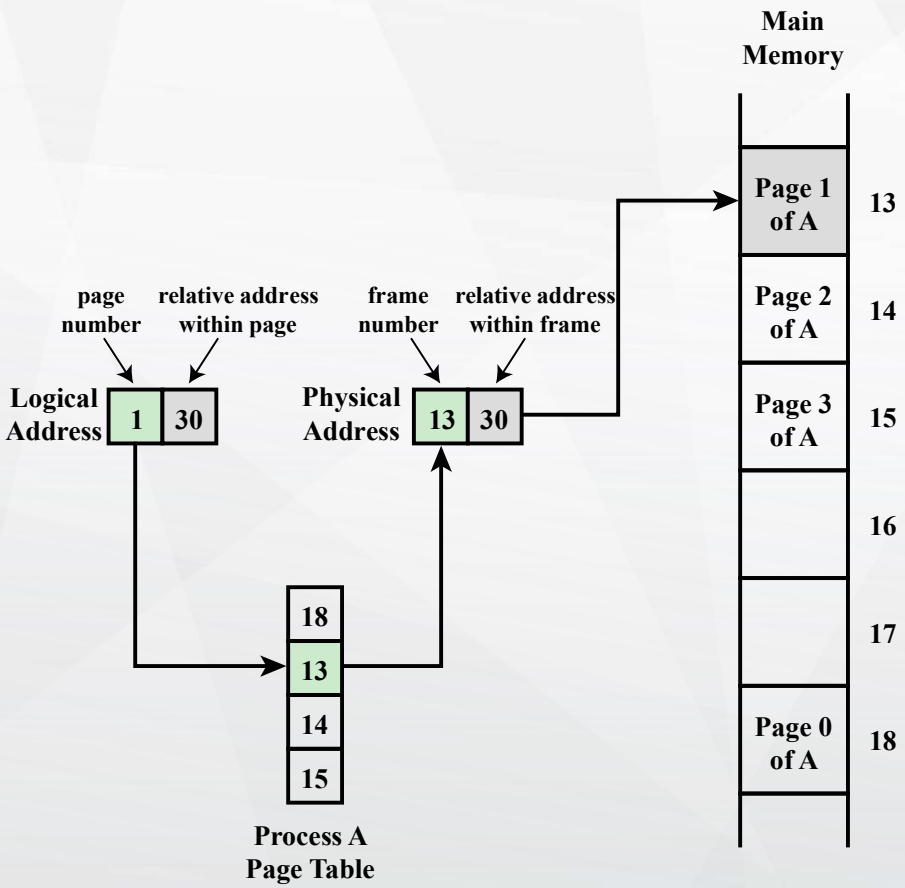



**Figure 8.16 Logical and Physical Addresses**

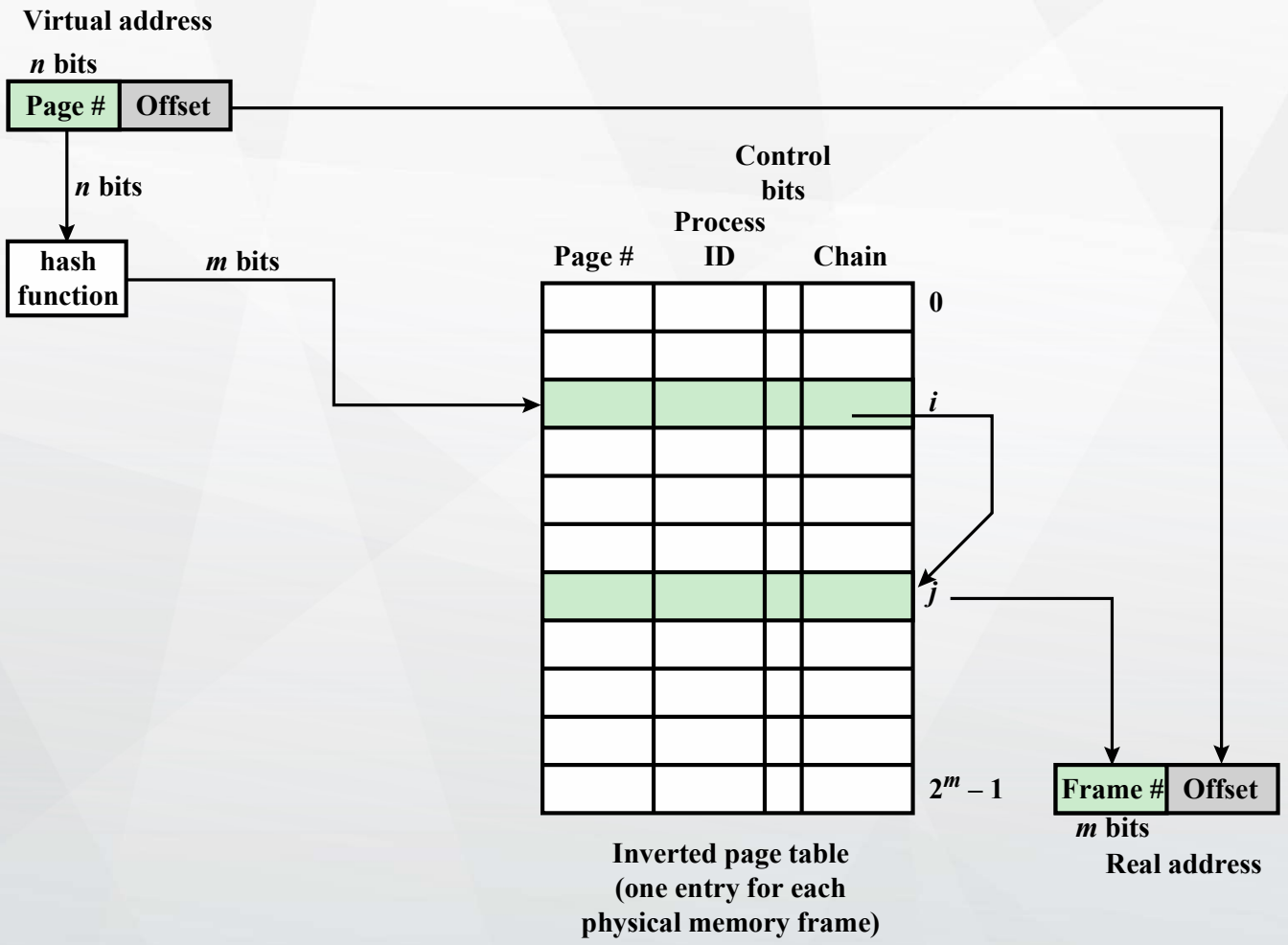



**Figure 8.17 Inverted Page Table Structure**

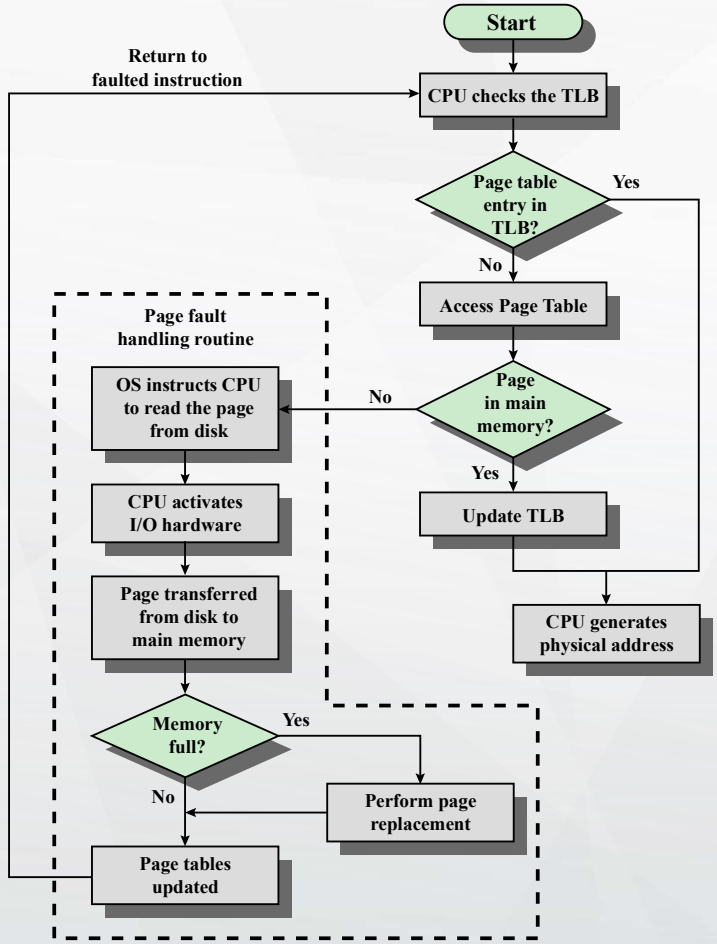



**Figure 8.18 Operation of Paging and Translation Lookaside Buffer (TLB) [FURH87]**

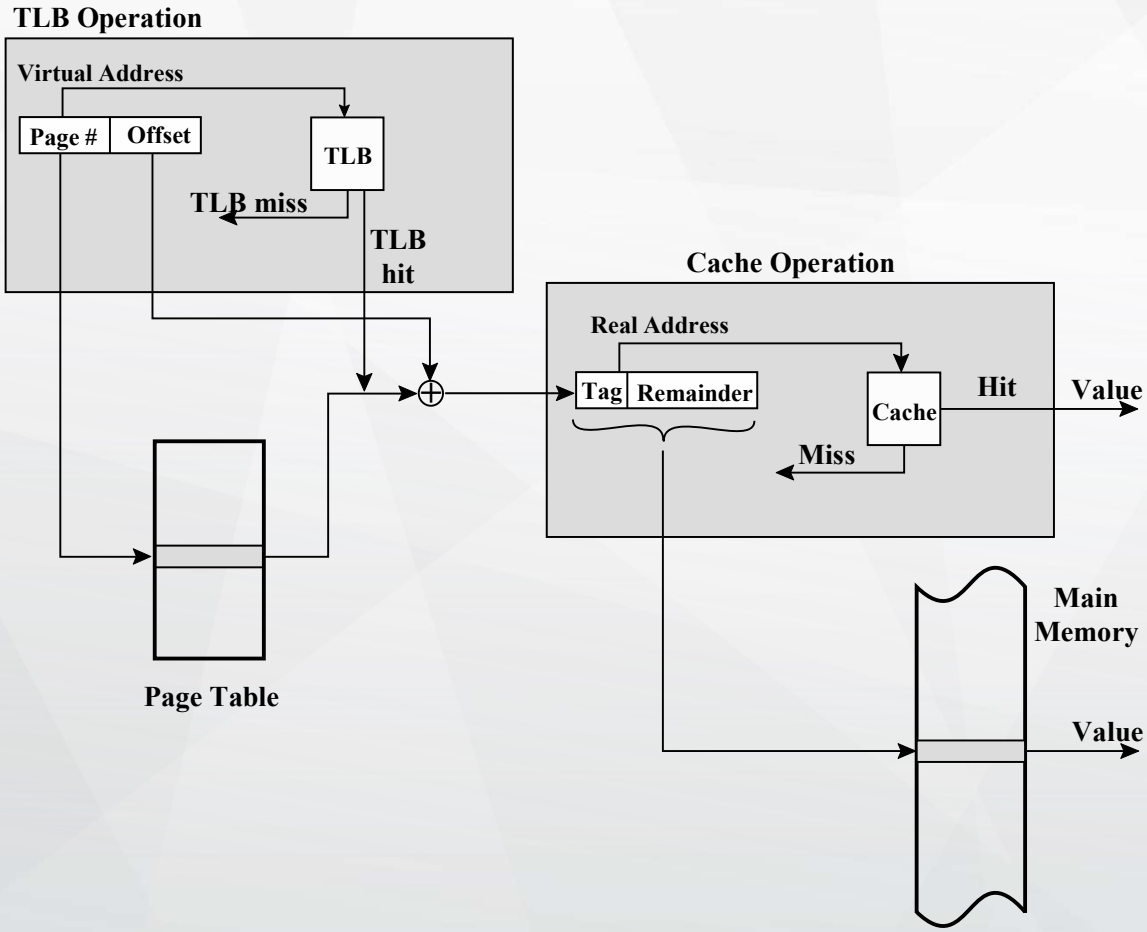



**Figure 8.19 Translation Lookaside Buffer and Cache Operation**

# Segmentasi

**Biasanya terlihat oleh programmer**

**Diberikan sebagai kemudahan untuk mengatur program dan data dan sebagai sarana untuk menghubungkan atribut hak istimewa dan perlindungan dengan instruksi dan data**

**Memungkinkan programmer untuk melihat memori yang terdiri dari beberapa ruang alamat atau segmen**

**Keuntungan:**

Menyederhanakan penanganan struktur data yang berkembang

Mengizinkan program diubah dan dikompilasi ulang secara independen tanpa mengharuskan seluruh rangkaian program ditautkan kembali dan dimuat ulang

Cocok untuk berbagi antar proses

Cocok untuk perlindungan

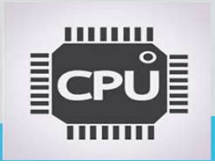
- **Termasuk perangkat keras untuk segmentasi dan paging**
- **Memori tidak tersegmentasi**
  - Alamat virtual sama dengan alamat fisik
  - Berguna dalam aplikasi pengontrol kinerja tinggi dengan kompleksitas rendah
- **Memori halaman tidak tersegmentasi**
  - Memori dipandang sebagai ruang alamat linier halaman
  - Perlindungan dan manajemen memori dilakukan melalui paging
  - Disukai oleh beberapa sistem operasi
- **Memori tidak terpage tersegmentasi**
  - Memori dipandang sebagai kumpulan ruang alamat logis
  - Memberikan perlindungan hingga ke level satu byte
  - Menjamin bahwa tabel terjemahan yang diperlukan ada di chip saat segmen ada di memori
  - Menghasilkan waktu akses yang dapat diprediksi
- **Memori halaman tersegmentasi**
  - Segmentasi digunakan untuk menentukan partisi memori logis yang tunduk pada kontrol akses, dan paging digunakan untuk mengelola alokasi memori di dalam partisi.
  - Sistem operasi seperti UNIX System V mendukung pandangan ini

Intel x86

Penyimpanan
Pengelolaan

# Segmentasi

- **Setiap alamat virtual terdiri dari referensi segmen 16-bit dan offset 32-bit**
  - Dua bit referensi segmen berhubungan dengan mekanisme proteksi
  - 14 bit menentukan segmen
- **Memori virtual tidak tersegmentasi adalah $2^{32}$ = 4Gbytes**
- **Memori virtual tersegmentasi adalah $2^{46}$= 64 terabyte (Tbytes)**
- **Ruang alamat fisik menggunakan alamat 32-bit dengan maksimum 4 Gbytes**
- **Ruang alamat virtual dibagi menjadi dua bagian**
  - Setengahnya global, dipakai bersama oleh semua prosesor
  - Sisanya bersifat lokal dan berbeda untuk setiap proses



# Perlindungan Segmen

- Terkait dengan setiap segmen ada dua bentuk perlindungan:
  - Tingkat hak istimewa
  - Atribut akses
- Ada empat tingkat hak istimewa
  - Paling terlindungi (level 0)
  - Paling tidak terlindungi (level 3)
- Tingkat hak istimewa yang terkait dengan segmen data adalah "klasifikasinya"
- Tingkat hak istimewa yang terkait dengan segmen program adalah "izin" nya
- Program pelaksana hanya dapat mengakses segmen data yang tingkat izinnya lebih rendah dari atau sama dengan tingkat hak istimewa segmen data
- Mekanisme hak istimewa juga membatasi penggunaan instruksi tertentu

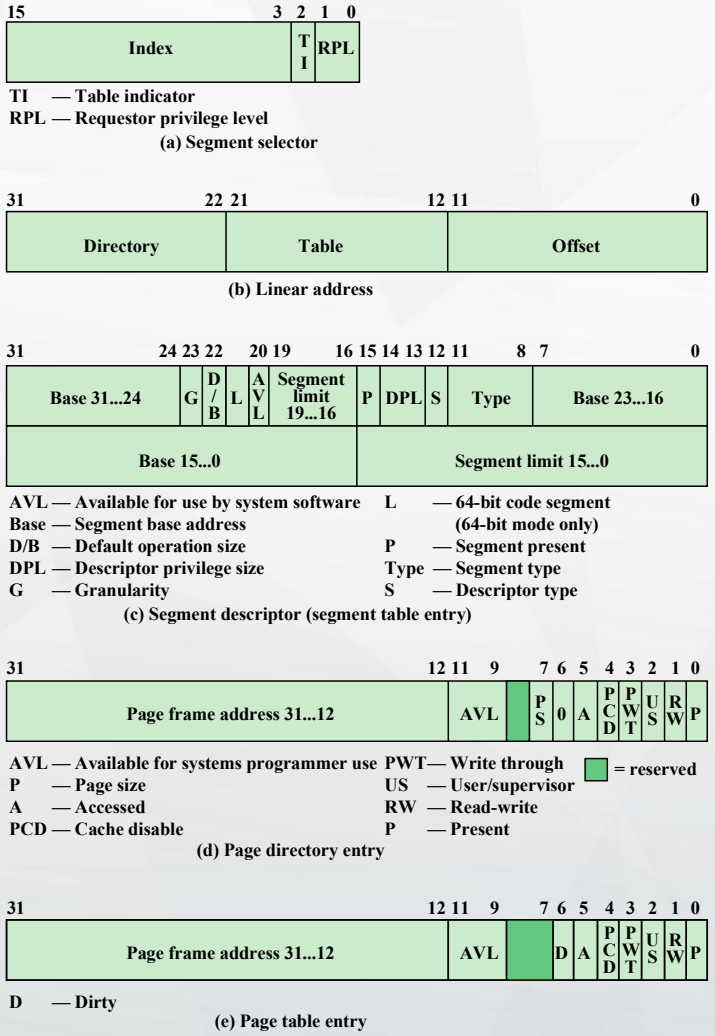



**Figure 8.20 Intel x86 Memory Management Formats**

**Tabel 8.5**
**x86 Parameter Manajemen Memori** (halaman 1 dari 2)

| Segment Descriptor (Segment Table Entry) |
|---|
| **Base**<br>Defines the starting address of the segment within the 4-GByte linear address space. |
| **D/B bit**<br>In a code segment, this is the D bit and indicates whether operands and addressing modes are 16 or 32 bits. |
| **Descriptor Privilege Level (DPL)**<br>Specifies the privilege level of the segment referred to by this segment descriptor. |
| **Granularity bit (G)**<br>Indicates whether the Limit field is to be interpreted in units by one byte or 4 KBytes. |
| **Limit**<br>Defines the size of the segment. The processor interprets the limit field in one of two ways, depending on the granularity bit: in units of one byte, up to a segment size limit of 1 MByte, or in units of 4 KBytes, up to a segment size limit of 4 GBytes. |
| **S bit**<br>Determines whether a given segment is a system segment or a code or data segment. |
| **Segment Present bit (P)**<br>Used for nonpaged systems. It indicates whether the segment is present in main memory. For paged systems, this bit is always set to 1. |
| **Type**<br>Distinguishes between various kinds of segments and indicates the access attributes. |

**Tabel 8.5**
**x86 Parameter Manajemen Memori** (halaman 2 dari 2)

| Page Directory Entry and Page Table Entry |
|---|
| **Accessed bit (A)**<br>This bit is set to 1 by the processor in both levels of page tables when a read or write operation to the corresponding page occurs. |
| **Dirty bit (D)**<br>This bit is set to 1 by the processor when a write operation to the corresponding page occurs. |
| **Page Frame Address**<br>Provides the physical address of the page in memory if the present bit is set. Since page frames are aligned on 4K boundaries, the bottom 12 bits are 0, and only the top 20 bits are included in the entry. In a page directory, the address is that of a page table. |
| **Page Cache Disable bit (PCD)**<br>Indicates whether data from page may be cached. |
| **Page Size bit (PS)**<br>Indicates whether page size is 4 KByte or 4 MByte. |
| **Page Write Through bit (PWT)**<br>Indicates whether write-through or write-back caching policy will be used for data in the corresponding page. |
| **Present bit (P)**<br>Indicates whether the page table or page is in main memory. |
| **Read/Write bit (RW)**<br>For user-level pages, indicates whether the page is read-only access or read/write access for user-level programs. |
| **User/Supervisor bit (US)**<br>Indicates whether the page is available only to the operating system (supervisor level) or is available to both operating system and applications (user level). |

# Paging

**Segmentasi mungkin dinonaktifkan**

Dalam hal ini digunakan ruang alamat linier

**Pencarian tabel halaman dua tingkat**

Pertama, direktori halaman

- 1024 entri maks
- Membagi memori linier 4 Gbyte menjadi 1024 grup halaman dengan 4 Mbyte
- Setiap tabel halaman memiliki 1024 entri yang sesuai dengan 4 halaman Kbyte
- Dapat menggunakan satu direktori halaman untuk semua proses, satu per proses atau campuran
- Direktori halaman untuk proses saat ini selalu dalam memori

Gunakan TLB yang menampung 32 entri tabel halaman

Tersedia dua ukuran halaman, 4k atau 4M

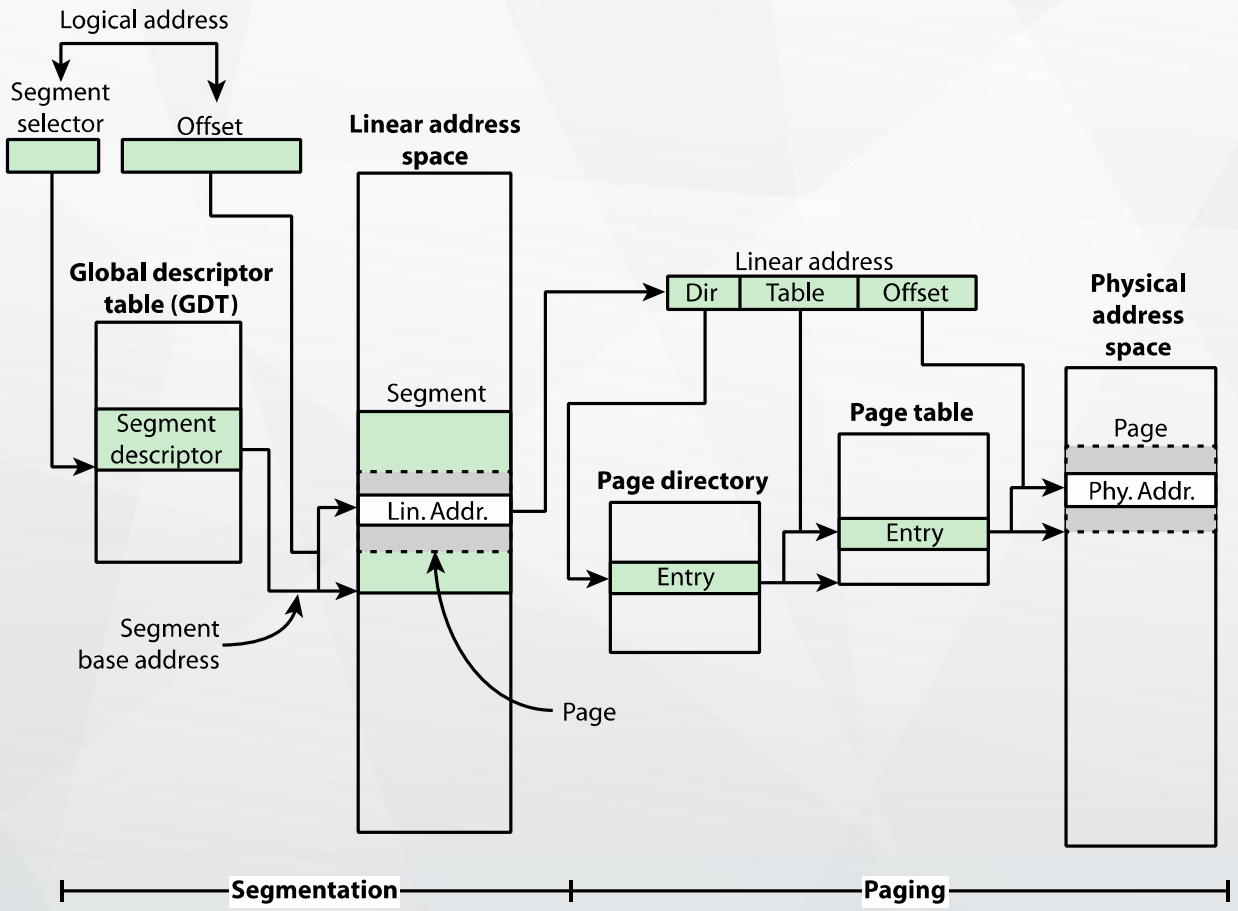



**Figure 8.21 Intel x86 Memory Address Translation Mechanisms**

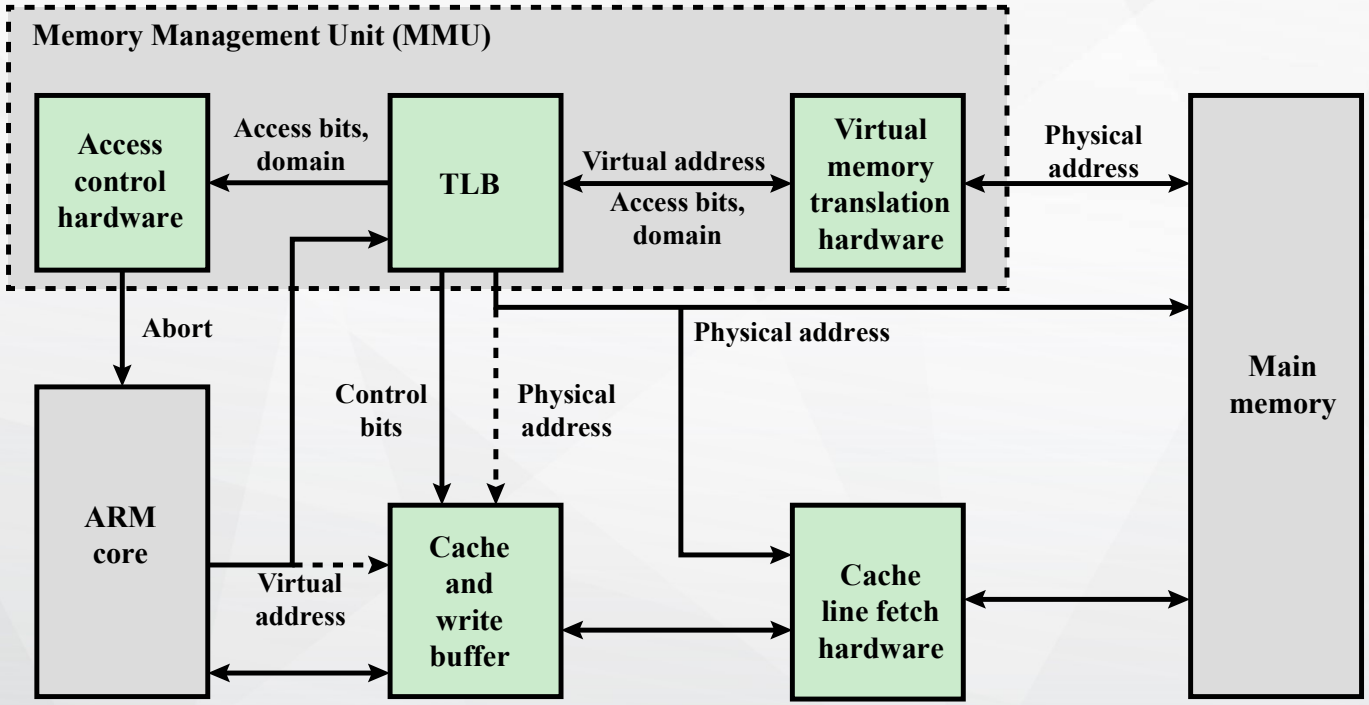



Figure 8.22 ARM Memory System Overview

# Terjemahan Alamat Memori Virtual

**ARM mendukung akses memori berdasarkan bagian atau halaman**

**Supersection (opsional)**

Terdiri dari blok memori utama 16 MB

**Bagian**

Terdiri dari blok memori utama sebesar 1 MB

**Halaman besar**

Terdiri dari 64-kB blok memori utama

**Halaman kecil**

Terdiri dari blok memori utama 4 kB

**Bagian dan supersection didukung untuk memungkinkan pemetaan wilayah memori yang besar sementara hanya menggunakan satu entri di TLB**

**Tabel terjemahan yang disimpan di memori utama memiliki dua tingkatan:**

Tabel tingkat pertama

- Memiliki terjemahan bagian dan superseksi, dan penunjuk ke tabel tingkat kedua

Tabel tingkat kedua

- Tahan terjemahan halaman besar dan kecil

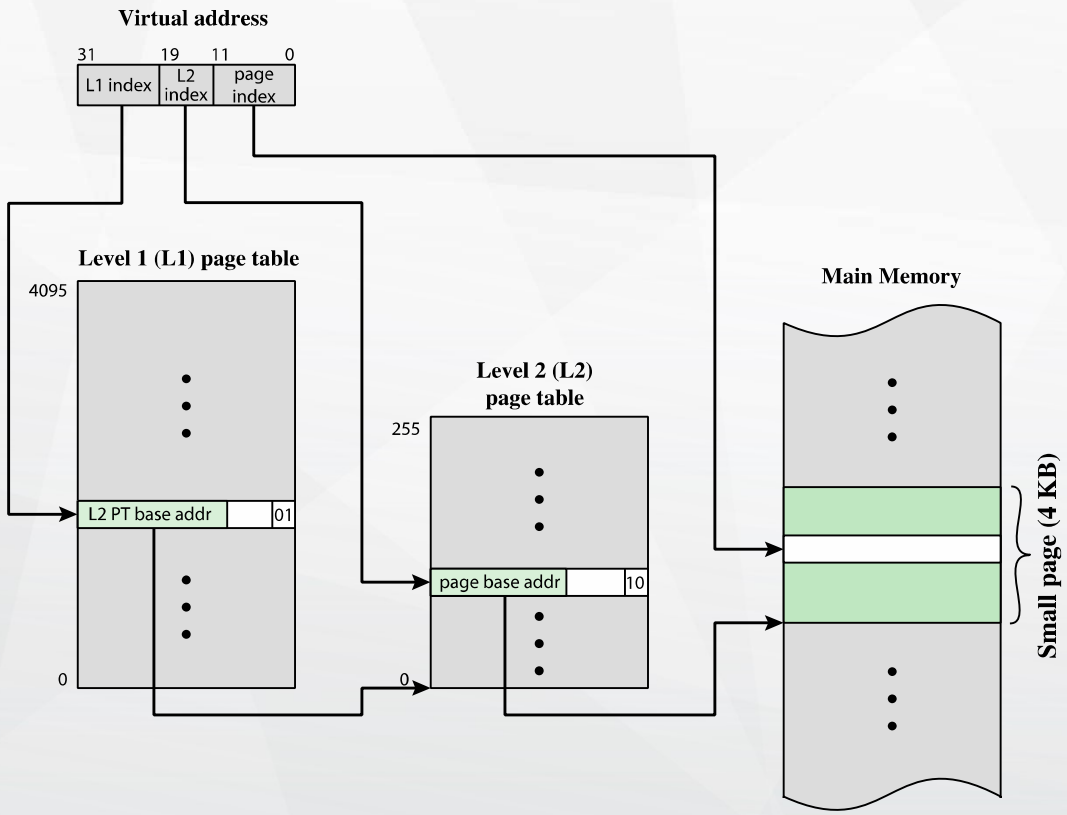
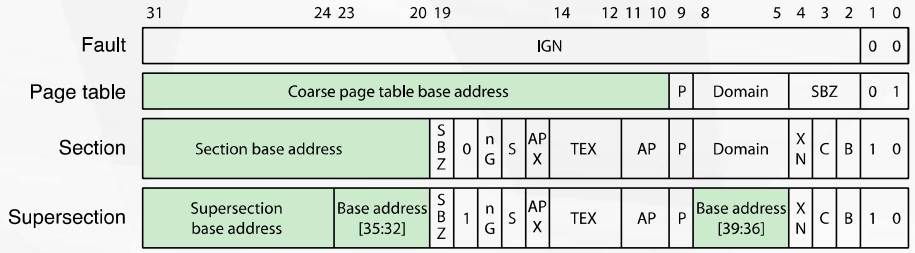
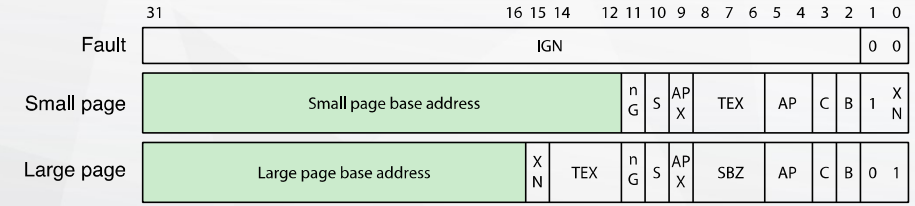


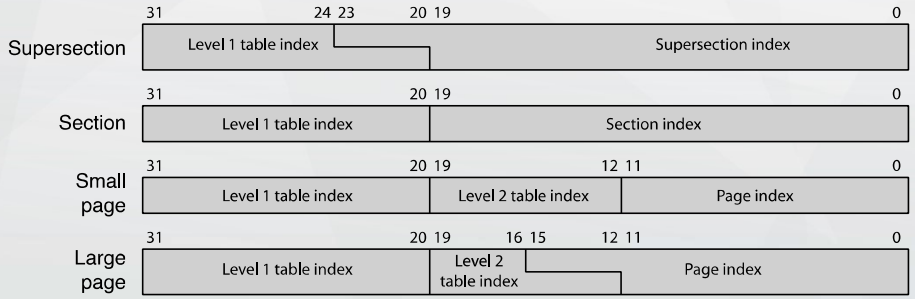

**Figure 8.23 ARM Virtual Memory Address Translation for Small Pages**




**(a) Alternative first-level descriptor formats**




**(b) Alternative second-level descriptor formats**




**(c) Virtual memory address formats**

**Figure 8.24 ARM Memory Management Formats**

## Table 8.6 ARM Memory-Management Parameters

**Access Permission (AP), Access Permission Extension (APX)**
These bits control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, a Permission Fault is raised.

**Bufferable (B) bit**
Determines, with the TEX bits, how the write buffer is used for cacheable memory.

**Cacheable (C) bit**
Determines whether this memory region can be mapped through the cache.

**Domain**
Collection of memory regions. Access control can be applied on the basis of domain.

**not Global (nG)**
Determines whether the translation should be marked as global (0), or process specific (1).

**Shared (S)**
Determines whether the translation is for not-shared (0), or shared (1) memory.

**SBZ**
Should be zero.

**Type Extension (TEX)**
These bits, together with the B and C bits, control accesses to the caches, how the write buffer is used, and if the memory region is shareable and therefore must be kept coherent.

**Execute Never (XN)**
Determines whether the region is executable (0) or not executable (1).

# Kontrol akses

**Bit kontrol akses AP di setiap akses kontrol entri tabel ke wilayah memori dengan proses tertentu**

**Suatu wilayah memori dapat ditetapkan sebagai:**

- Tidak ada akses
- Baca saja
- Baca tulis

Wilayah ini hanya dapat diakses dengan hak istimewa, dipesan untuk digunakan oleh OS dan bukan oleh aplikasi

ARM menggunakan konsep domain:

- Kumpulan bagian dan / atau halaman yang memiliki izin akses khusus
- Arsitektur ARM mendukung 16 domain
- Mengizinkan banyak proses untuk menggunakan tabel terjemahan yang sama sambil mempertahankan beberapa perlindungan satu sama lain

Dua jenis akses domain didukung:

Klien

- Pengguna domain yang harus memperhatikan izin akses individu bagian dan/ atau halaman yang membentuk domain itu

Manajer

- Kontrol perilaku domain dan lewati izin akses untuk entri tabel di domain itu

# Ringkasan

## Bab 8

## Sistem operasi Dukung

**Gambaran umum sistem operasi**
- Tujuan dan fungsi sistem operasi
- Jenis sistem operasi

**Penjadwalan**
- Penjadwalan jangka panjang
- Penjadwalan jangka menengah
- Penjadwalan jangka pendek

**Manajemen memori Intel x86**
- Ruang alamat
- Segmentasi
- paging

**Manajemen memori**
- Swapping
- Mempartisi
- Paging
- Memori virtual
- Terjemahan penyangga tepi pantai
- Segmentasi

Manajemen memori ARM
- Organisasi sistem memori
- Terjemahan alamat memori virtual
- Format manajemen memori
- Kontrol akses