Sistem Operasi

Memory Management

Part 1 of 2

Manajemen Memori

- Memori adalah pusat kegiatan pada sebuah komputer, karena setiap proses yang eksekusi, **harus** berada memori terlebih dahulu.
- Sistem Operasi bertugas untuk mengatur penggunaan memori untuk banyak proses
  - Memori harus digunakan dengan baik, sehingga dapat memuat banyak proses dalam suatu waktu.
- Sebelum masuk ke memori, suatu proses harus menunggu. Hal ini disebut Input Queue (Long term scheduler)

Tujuan Manajemen Memory

- Meningkatkan utilitas CPU
- Data dan instruksi dapat diakses lbh cepat oleh CPU
- Memori kapasitasnya terbatas, jadi harus efisien
- Efisiensi Transfer data
  Memori Utama ←→CPU

Manajemen Memori

- Main memory dan registers satu-satunya storage CPU yang dapat diakses secara **langsung**
- Register mengakses dalam satu CPU clock (atau kurang)
- Main memory lebih lama dari register
- **Cache** berada diantara main memory dan CPU registers

Syarat Pengelolaan Memori

- **Relokasi**: mengkonversi alamat logika program ke alamat fisik memori
- **Protection:** diperlukan untuk menjamin operasi-operasinya sesuai dan tepat
- **Sharing**: memori dipakai bersama-sama
- **Organisasi logika** : OS & hw berhubungan dg user program dalam 1 modul
- **Organisasi fisik** : ada pengaturan yg jelas antara mem utama dan mem sekndr pada long term scheduling

Konsep Dasar

- Address Binding
- Dynamic Loading
- Dynamic Linking
- Overlay
- Proteksi Memory

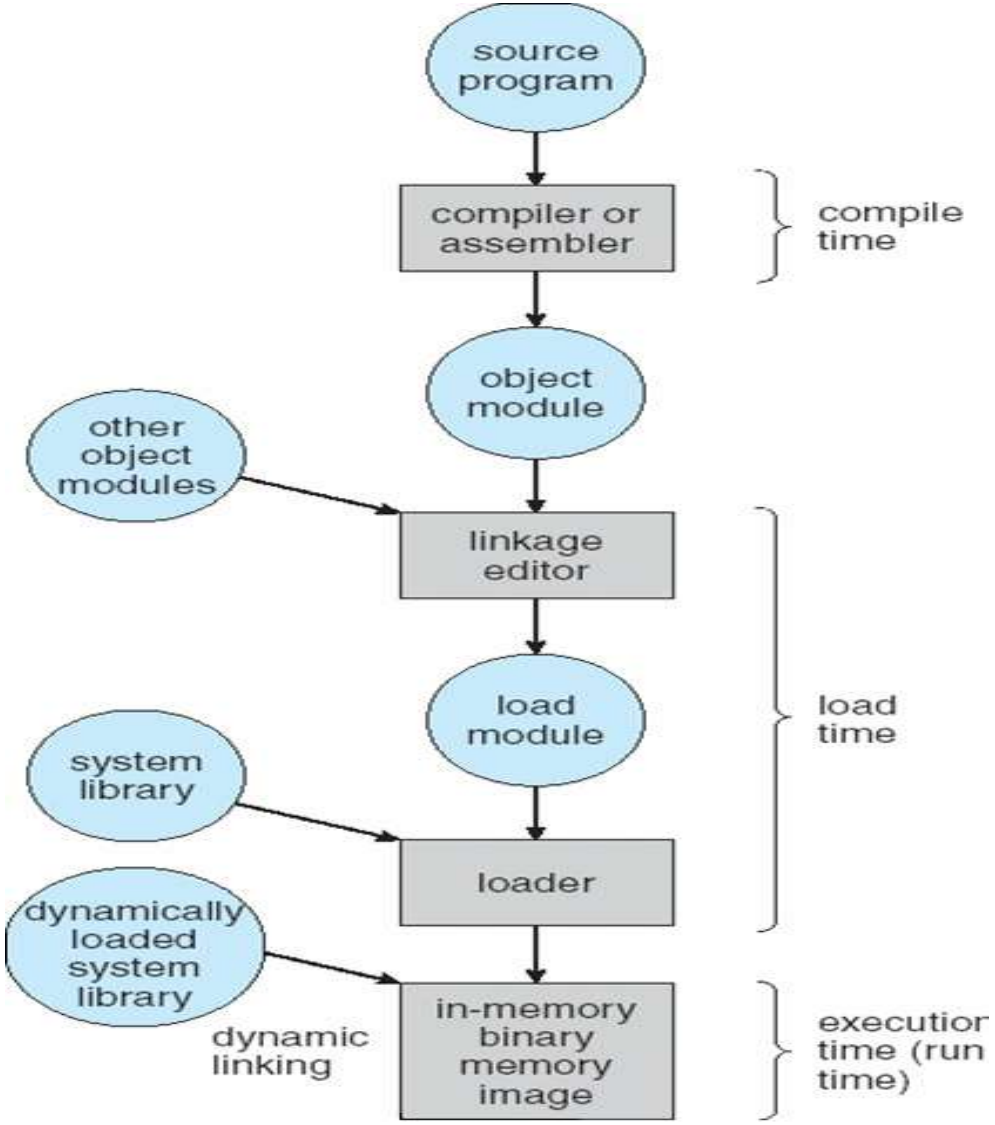
Address Binding

- Pemetaaan alamat suatu data & program ke alamat memory tertentu

Address Binding

- **Compile Time**: saat program di-compile, menggunakan *absolute code*.
  - Contoh: program DOS
- **Load Time:** pada saat program dipanggil / load, menggunakan relocatable code.
  - Variable2 nya berada pada suatu stack yg sudah dipesan sebelumnya dengan pasti
- **Execution Time:** pada saat program dijalankan.
  - Binding akan ditunda sampai run time. Kode dapat dipindah antar segment dan page pada memory.

Multistep Processing of a User Program



Dynamic Loading

- Tidak semua bagian program diload ke memori
- Suatu routine tidak akan diload sampai dibutuhkan
- Tidak perlu campur tangan OS ➔ tergantung desain program aplikasinya
- Sangat berguna jika menangani banyak kode yg jarang diakses
- Ketika terjadi pemanggilan, rutin pemanggil akan memeriksa di memory, apakah rutin yg dibutuhkan itu sudah ada atau belum, jika belum, dipanggil dan dialokasi ke memory

Static Linking

- Menghubungkan seluruh routine yang ada pada program ke dalam suatu ruang alamat di memory
  - Dibuat oleh linker
- Setiap program memiliki salinan dari seluruh rutin dan data yg dibutuhkan.
- Biasanya digabungkan dgn executable file
  - Contoh: EXE Delphi dan C/C++
- Kelebihan: library pasti ada dan versinya pasti benar, mudah pendistribusian file
- Kekurangan: ukuran file besar

Dynamic Linking

- Mirip Dynamic Loading → pada proses lingking
- Shared libabry (misal file .dll, .sys, .drv)

Dynamic Linking

- Menghubungkan semua rutin yang ada scr dinamis.
- Tidak membuang-buang tempat di disk dan memori.
  - Kumpulan data yang ada dapat digunakan bersama-sama.
- Membutuhkan bantuan sistem operasi.
  - Operating system dibutuhkan untuk memeriksa apakah routine itu ada dalam processes' memory address
- Linking dilaksanakan pada execution time
- Sekumpulan kode kecil yg disebut *stub*, digunakan untuk mencari memory-resident library routine yang tepat
  - Stub akan mengganti dirinya sendiri dengan address dari routine, dan kemudian mengeksekusi routine
- Dynamic linking digunakan untuk file libraries
  - System also known as **shared libraries (.dll)**
- Kelebihan: ukuran file kecil, irit, dipakai bersama
- Kekurangan: jika dll hilang, perbedaaan versi

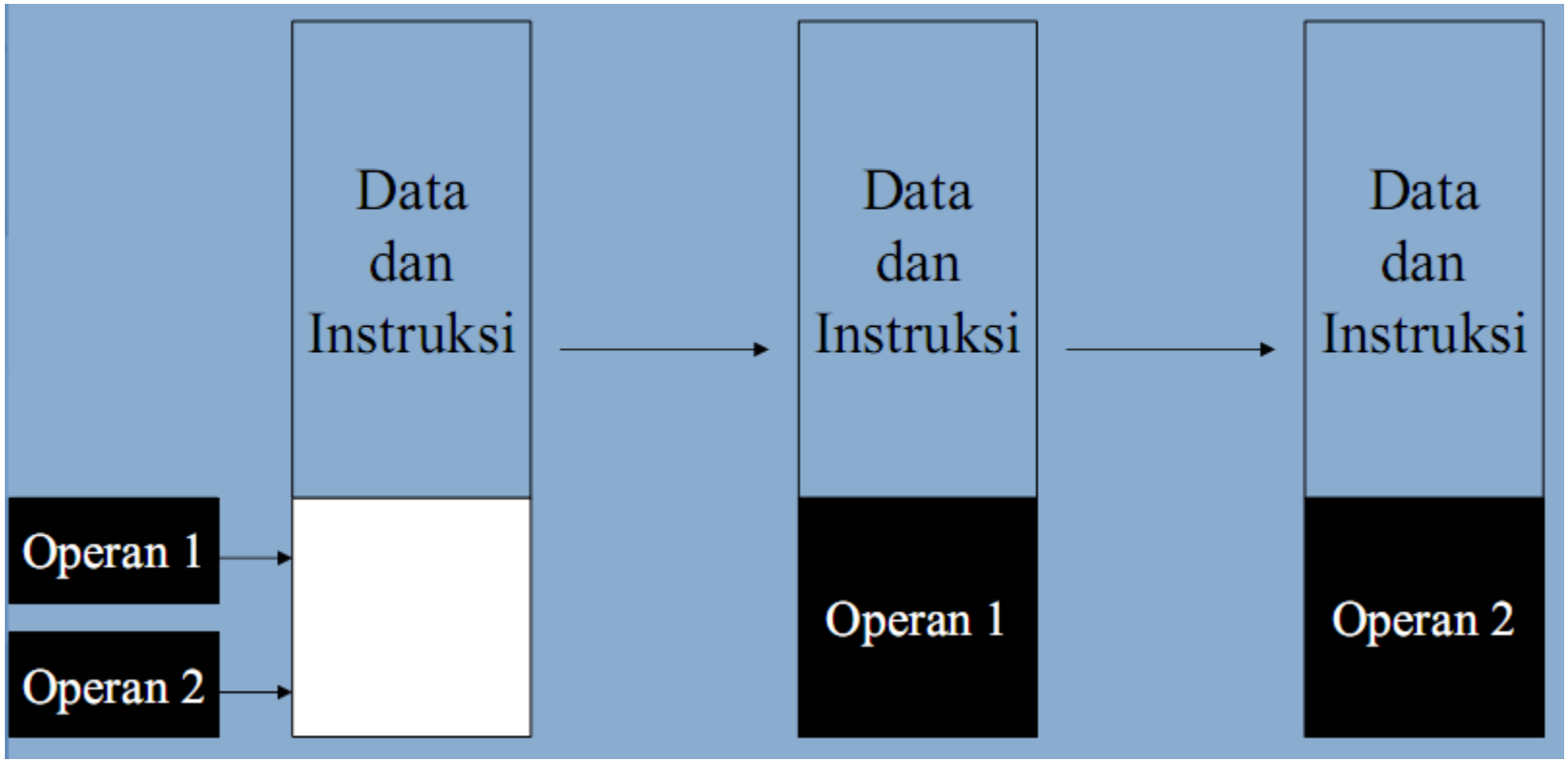
Overlay

- Strategi untuk memaggil suatu program yang membutuhkan memori lebih besar dari yang tersedia.
- Membagi program yang besar menjadi bagian bagian kecil dan dapat dimuat di memori
- Bagian utama selalu berada di memori utama
- Bagian pendukung diletakkan di memori sekunder
- Tidak perlu campur tangan OS ➔ tergantung desain program aplikasinya

Overlays

- Caranya:
  - Data dan instruksi yang diperlukan dimasukkan langsung ke memori utama.
  - Routine-nya dimasukkan ke memori secara bergantian. (dibagi-bagi / dipecah2).
  - Bagian pendukung lain dimasukkan ke memory sekunder
  - Memerlukan algoritma tambahan untuk melakukan overlays.
- Tidak memerlukan bantuan dari sistem operasi.
- Sulit untuk dilakukan.

Contoh overlays



Proteksi Memory

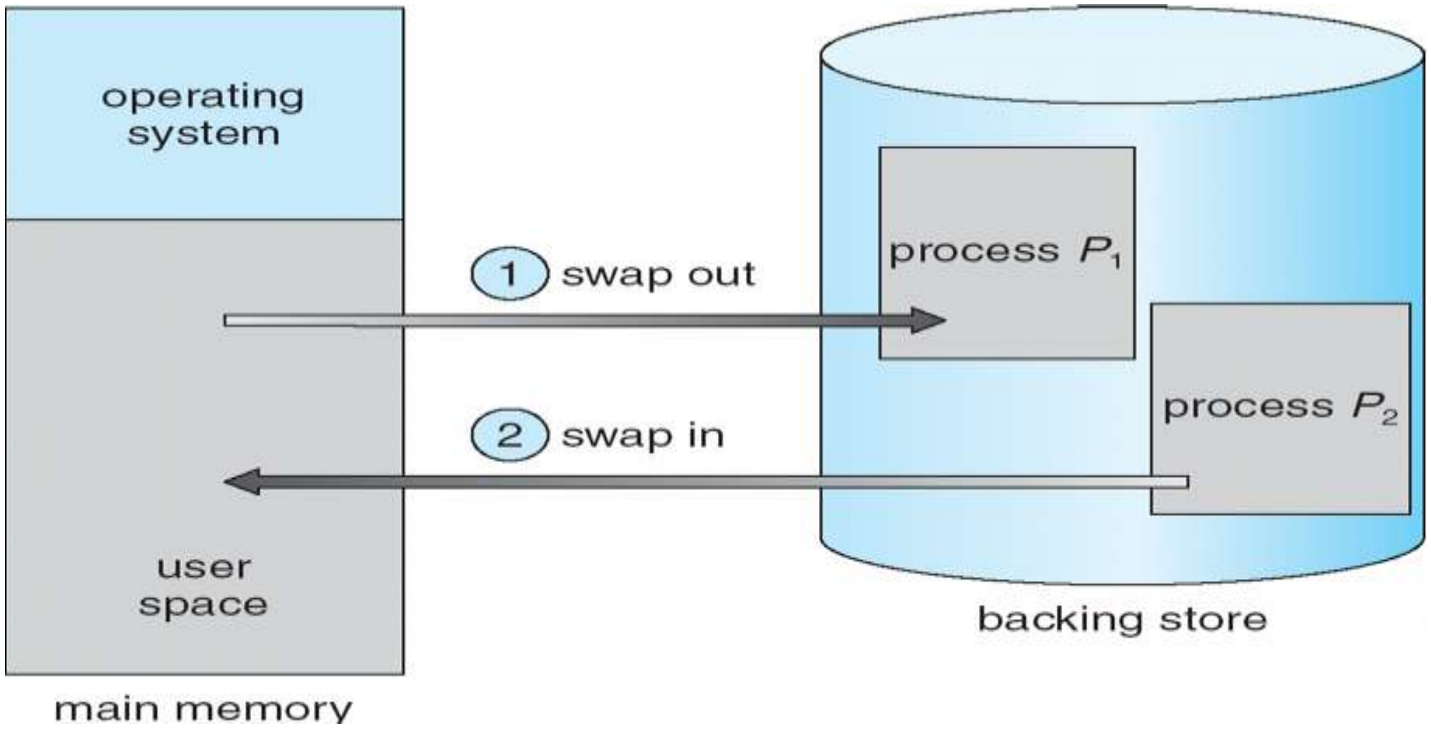
- melindungi OS dari proses yang sedang dijalankan oleh user, atau melindungi suatu proses dari proses lainnya.

Swapping

- Sebuah proses harus berada di dalam memori untuk dapat dijalankan.
- Sebuah proses dapat di-**swap** sementara keluar memori ke sebuah penyimpanan cadangan (backing store) untuk kemudian dikembalikan lagi ke memori.
- **Roll out, roll in** adalah penjadualan swapping berbasis pada prioritas
  - proses berprioritas rendah di-swap keluar memori agar proses berprioritas tinggi dapat masuk dan dijalankan di memori
- **Backing store** – hardisk kecepatan tinggi untuk menyimpan copy seluruh isi memori
  - harus dapat direct access ke memory images

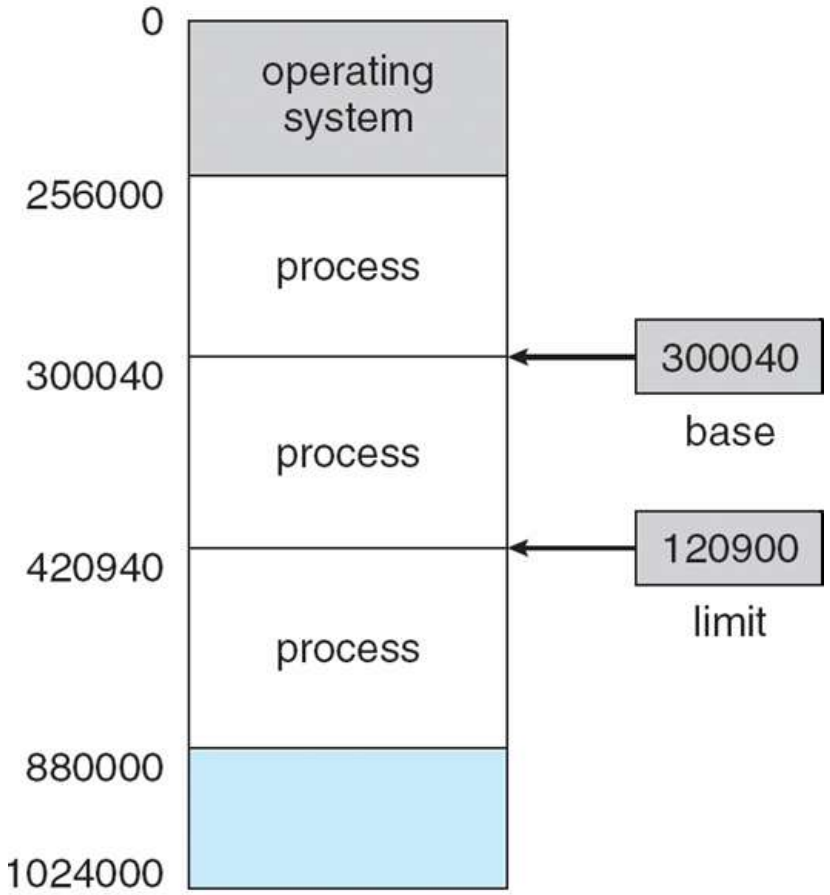
Swapping & Semantic View

- Swapping butuh waktu transfer
  - Misal file 1MB, kecepatan transfer hdd 5MB/s
  - Waktu yg dibutuhkan: 1000 kb / 5000 KBps = 1/5 detik = 200 ms
- Total transfer time berbading lurus dengan jumlah memory yg diswap



Base and Limit Registers untuk proteksi memory

- A pair of **base** and **limit** registers define the logical address space



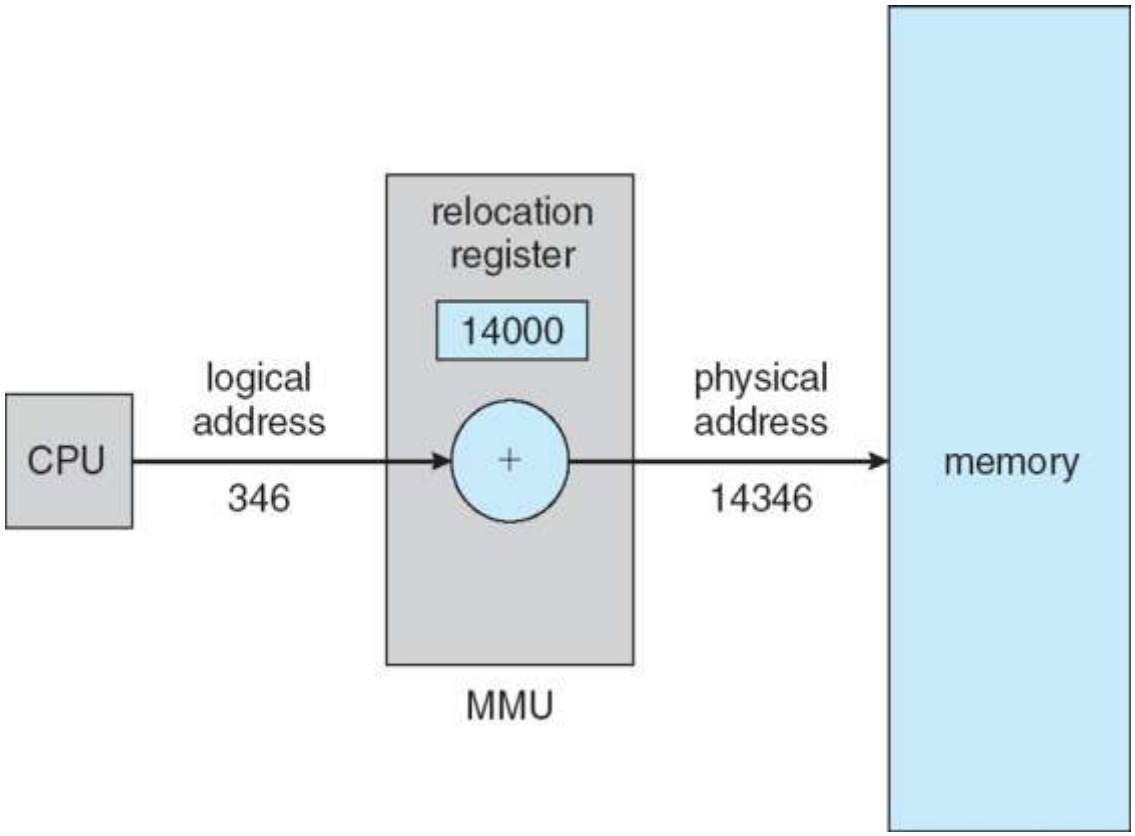
Logical & Phisycal Address

- **Alamat Logika** (**alamat virtual**) : alamat yg dibentuk di CPU
- **Alamat fisik** adalah alamat yang ada di memori fisik.
- Untuk mengubah dari alamat logika ke alamat fisik diperlukan suatu perangkat keras yang bernama MMU (Memory Management Unit).
- Pengubahan dari alamat logika ke alamat fisik adalah pusat kegiatan manajemen memori.
- Logical & physical addresses **sama** pada compile-time & load-time
- Logical (virtual) & physical addresses **beda** pada execution-time

Memory-Management Unit (MMU)

- Hardware yang memetakan **virtual** ke **physical** address
- Pada MMU scheme, nilai dalam *relocation register* **ditambahkan** ke setiap address yg di generated oleh sebuah process pada saat dia dikirim ke memory
- User program deals with *logical* addresses
  - Tidak akan melihat *real* physical addresses

Dynamic relocation using a relocation register



Pencatatan Pemakaian Memory

- Bit Map → Peta Bit
- Linked List
- Buddy

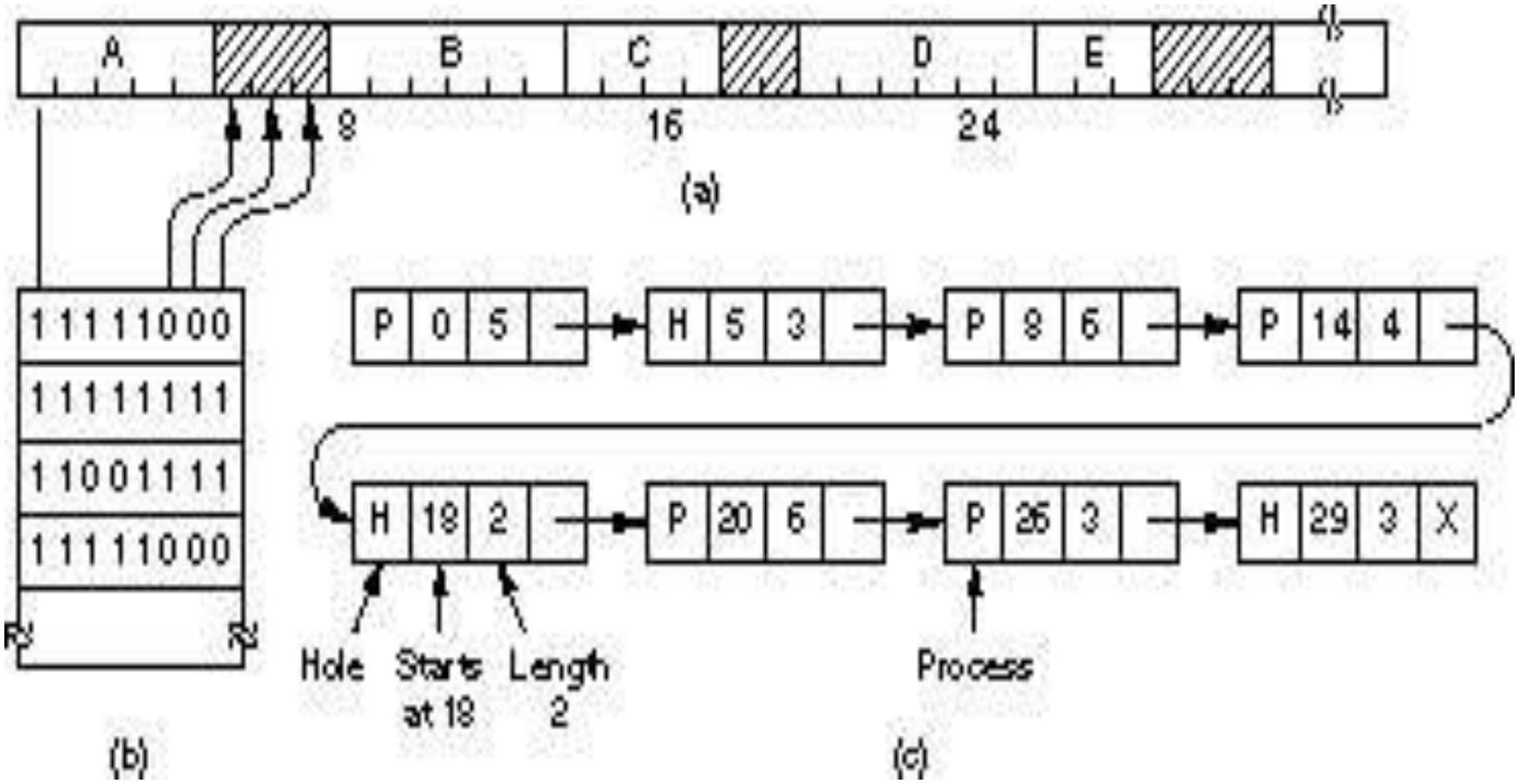
Bit Map

- Memori dibagi menjadi blok/unit
- Tiap unit terdiri beberapa word/kb
- Tiap unit diwakili oleh 1 bit
- Bit = 0 ➔ kosong; Bit = 1 ➔ isi
- Semakin kecil ukuran unit ➔ semakin banyak bit map dibutuhkan

Linked List

- Menggunakan konsep struktur data pointer
- P :Proses ; H: hole

Pencatatan Pemakaian Memory(lanjutan)



Bit = 0 ➔ kosong - arsir, Bit = 1 ➔ isi -putih

Peng-Alokasi-an Memori

- Contiguous :
  - Partisi statis
  - Partisi dinamis
  - Buddy sistem
- Non Contiguous :
  - Paging
  - Segmentasi

Partisi Statis

- Ukuran Partisi Seragam
- Ukuran Partisi Beragam

Ukuran partisi seragam

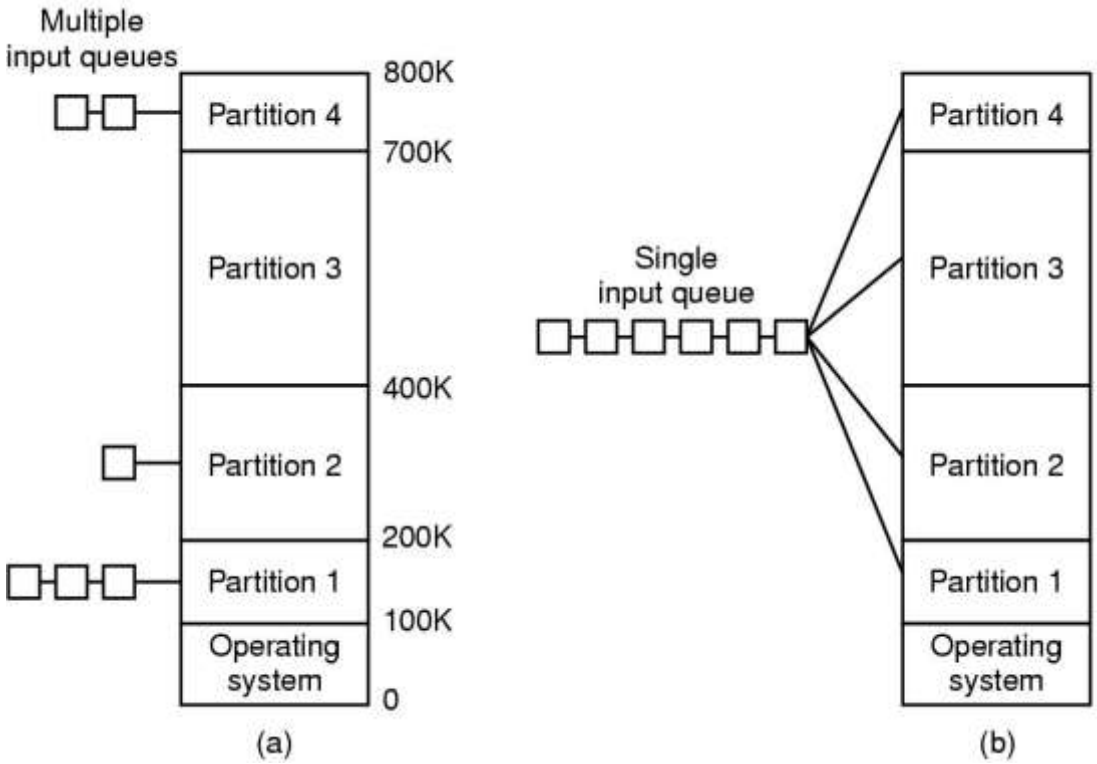
- Program yang ukurannya lebih besar daripada partisi tidak dapat diload ke sistem, berarti tidak pernah bisa dijalankan.
  ➔ Perlu menggunakan teknik overlay
- Program yang ukurannya sangat kecil dibandingkan dengan partisi akan terjadi banyak hole memori
  ➔ nganggur tidak dapat dimanfaatkan.

Partisi tidak seragam

- Untuk menyelesaikan masalah yang muncul pada partisi memori berukuran seragam

Strategi Penempatan Program ke partisi:

- Satu antrian untuk tiap partisi,
- Satu antrian untuk seluruh partisi



Partisi statis

Multiprogramming dengan Partisi Dinamis

- Partisi statis → banyak fragmentasi

Fragmentasi

- Fragmentasi adalah munculnya hole-hole yang **tidak cukup besar** untuk menampung permintaan dari proses.
- Fragmentasi Eksternal: apabila terdapat dalam bentuk banyak hole yang berukuran kecil dan tidak berurutan
- Fragmentasi Internal: apabila terdapat di dalam blok memori yang sudah dialokasikan secara statis

Mengatasi Fragmentasi Eksternal

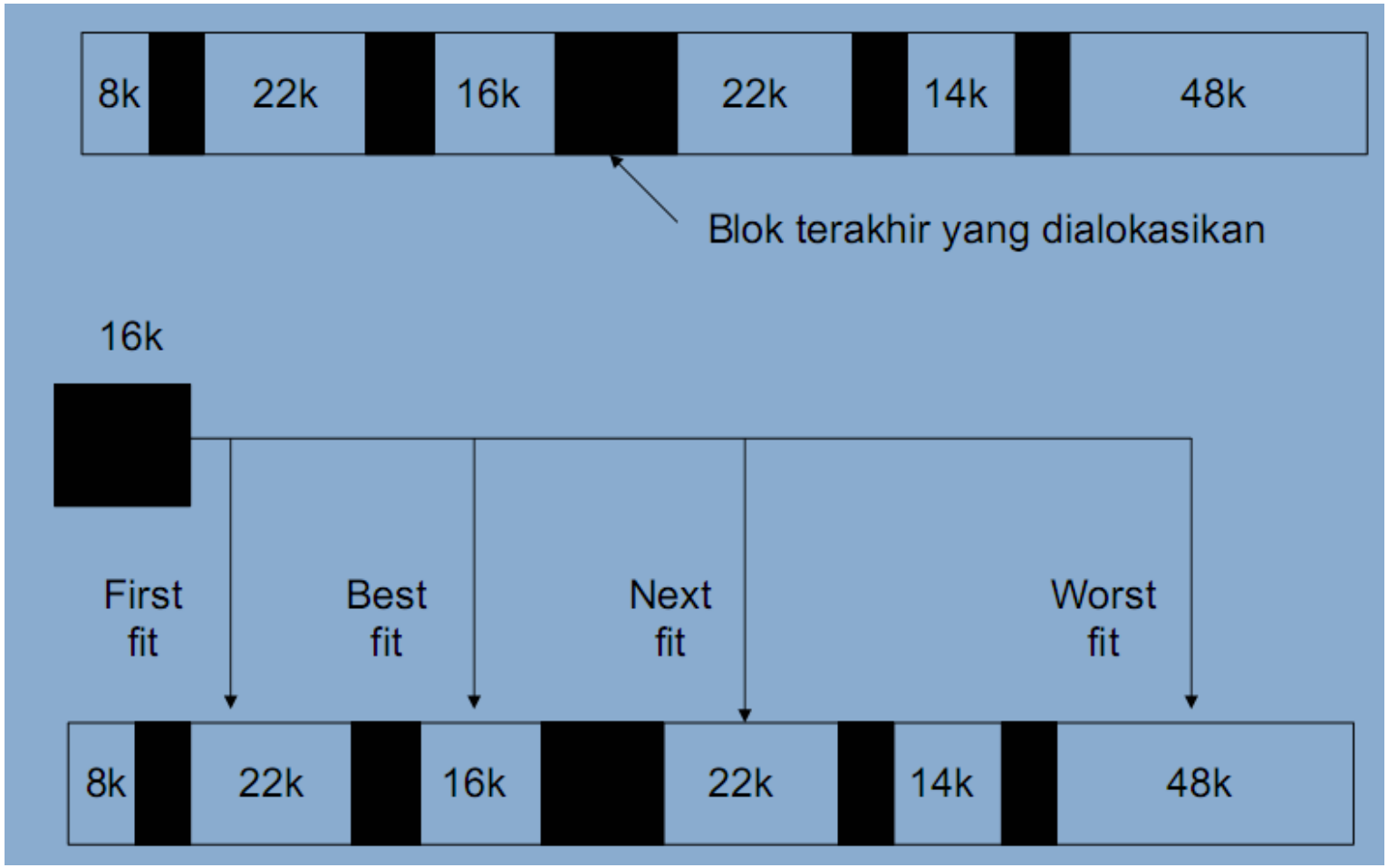
- **compactation**, yaitu mengatur kembali isi memori agar memori yang kosong diletakkan bersama di suatu bagian yang besar.
- Compactation hanya dapat dilakukan apabila relokasi bersifat **dinamis** dan pengalamatan dilakukan pada saat **runtime**.
- Solusi lain untuk fragmentasi eksternal adalah **paging** dan **segmentasi**.
- Partisi fixed berukuran berbeda **lebih baik** dalam meminimalisasi fragmentasi intern daripada partisi fixed berukuran sama.

Compaction

Algoritma Pengalokasian Memory pada partisi dinamis

- First fit : Mengalokasikan hole pertama yang besarnya mencukupi. Pencarian dimulai dari awal.
- Best fit : Mengalokasikan hole terkecil yang besarnya mencukupi (tepat).
- Next fit : Mengalokasikan hole pertama yang besarnya mencukupi.
  - Pencarian dimulai dari akhir pencarian sebelumnya.
- Worst fit : Mengalokasikan hole terbesar yang tersedia.
- First-fit and best-fit better than worst-fit in terms of speed and storage utilization

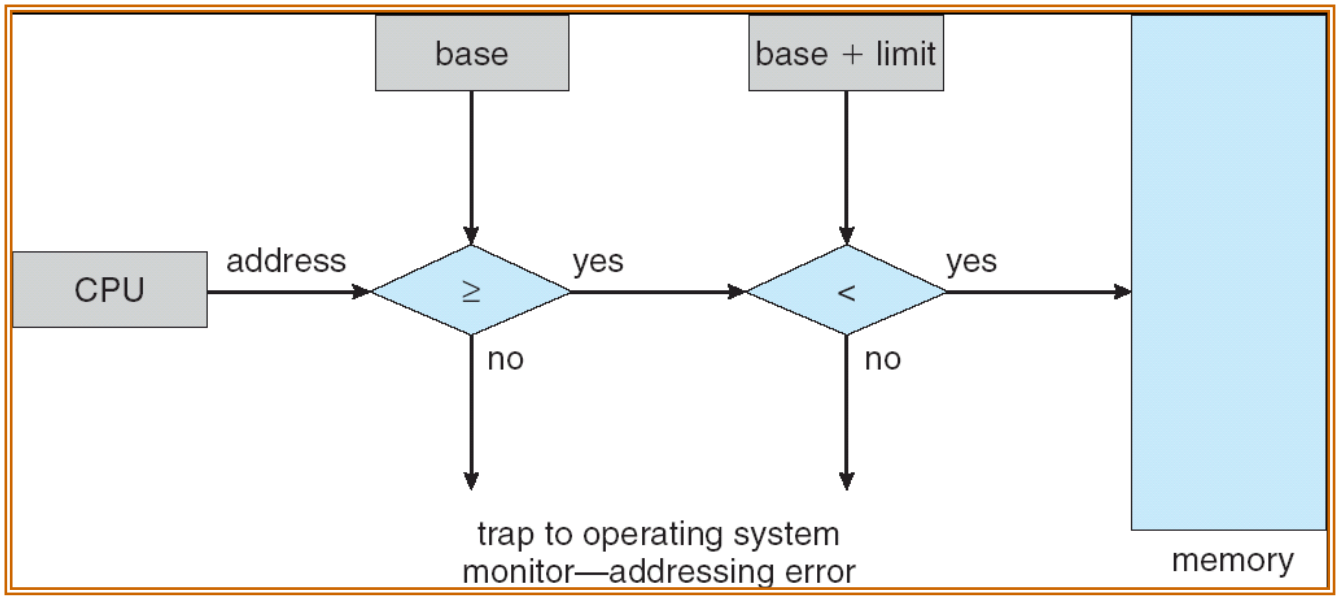
Contoh Pengalokasian Memori



Buddy System

- Pengelolaan memori dengan memanfaatkan bilangan biner (2^k; k=0,1,2..)

HW address protection with base and limit registers



Contiguous Memory Allocation

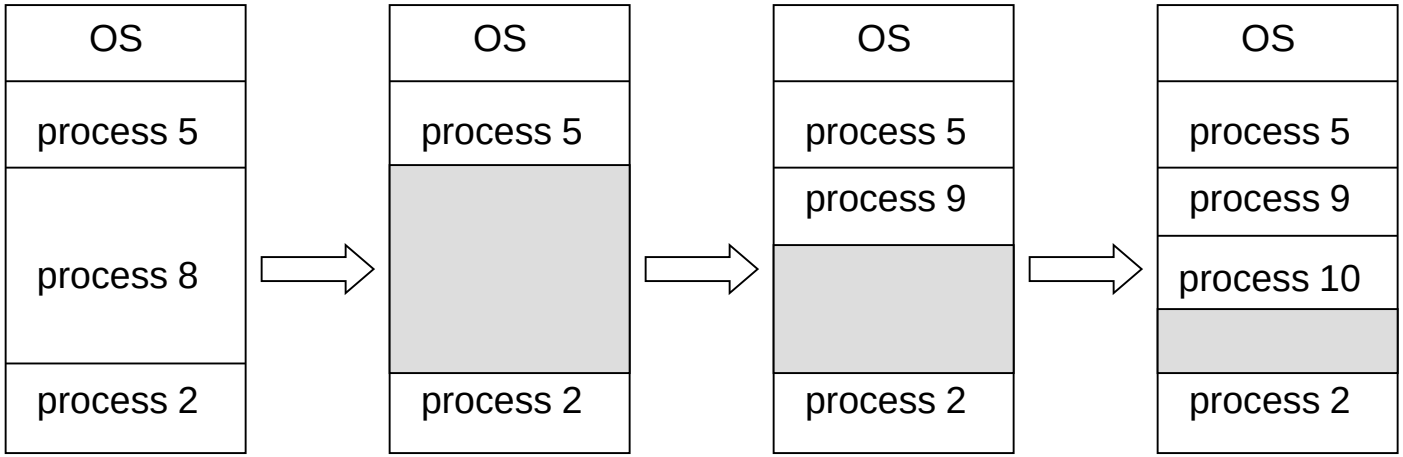
- Main memory dibagi menjadi 2:
  - Resident operating system, menggunakan low memory
  - User processes ada di high memory
- Contiguous Memory Allocation: alamat memori diberikan kepada proses secara berurutan dari kecil ke besar.
- Keuntungan contiguous daripada Non-contiguous:
  - sederhana, cepat, mendukung proteksi memori.
- Kerugian contiguous daripada non-contiguous:
  - jika tidak semua proses dialokasikan di waktu yang sama, akan sangat tidak efektif dan mempercepat habisnya memori.

Contigous Memory Allocation

- Jenis partisi:
  - Partisi tunggal: alamat pertama memory yang dialokasikan untuk suatu proses adalah alamat setelah alamat yang dialokasikan untuk proses sebelumnya.
  - Partisi banyak: adalah dimana Sistem Operasi menyimpan informasi tentang semua bagian memori yang tersedia untuk digunakan (disebut hole).

Contiguous Allocation (Cont.)

- Multiple-partition allocation
  - Hole – block of available memory;
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it
  - Operating system maintains information about:
    a) allocated partitions b) free partitions (hole)



Contiguous Allocation (Cont.)

- Ada 2 cara pengaturan partisi pada sistem partisi banyak: partisi tetap, dan partisi dinamis.
  - Partisi tetap adalah apabila memori dipartisi menjadi blok-blok yang ukurannya ditentukan dari awal.
    - Terbagi lagi atas partisi tetap berukuran sama, dan partisi tetap berukuran berbeda.
  - Partisi dinamis adalah memori dipartisi menjadi bagian-bagian dengan jumlah dan besar yang tidak tentu.

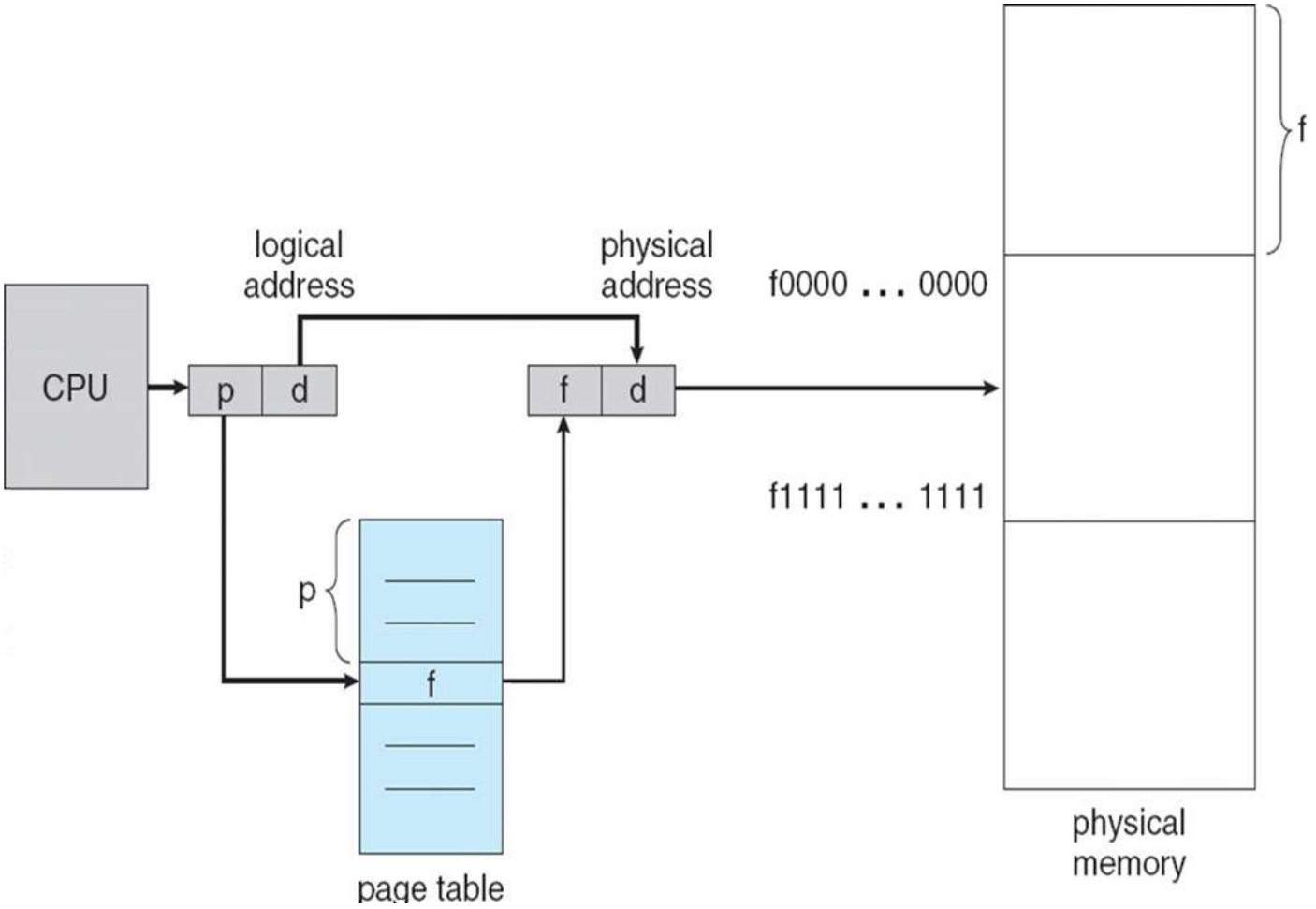
Non-contiguous Allocation

- Paging
- Segmentasi

Paging (Penghalamanan)

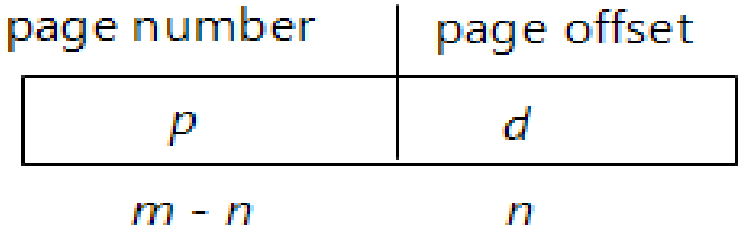
- Suatu metode yang memungkinkan suatu alamat memori fisik yang tersedia dapat tidak berurutan letaknya.
- Memori logic dibagi menjadi blok-blok yang ukurannya tetap yang dinamakan **page** (ukurannya adalah bilangan 2 pangkat k, (2^k) diantara 512 bytes dan 8192 bytes, tergantung arsitektur memory).
- Memori fisik dibagi juga menjadi blok-blok yang ukurannya tetap yang dinamakan **frame**.
- Lalu kita membuat suatu **page table** yang akan menterjemahkan memori virtual menjadi memori fisik.

Paging Concept



Page

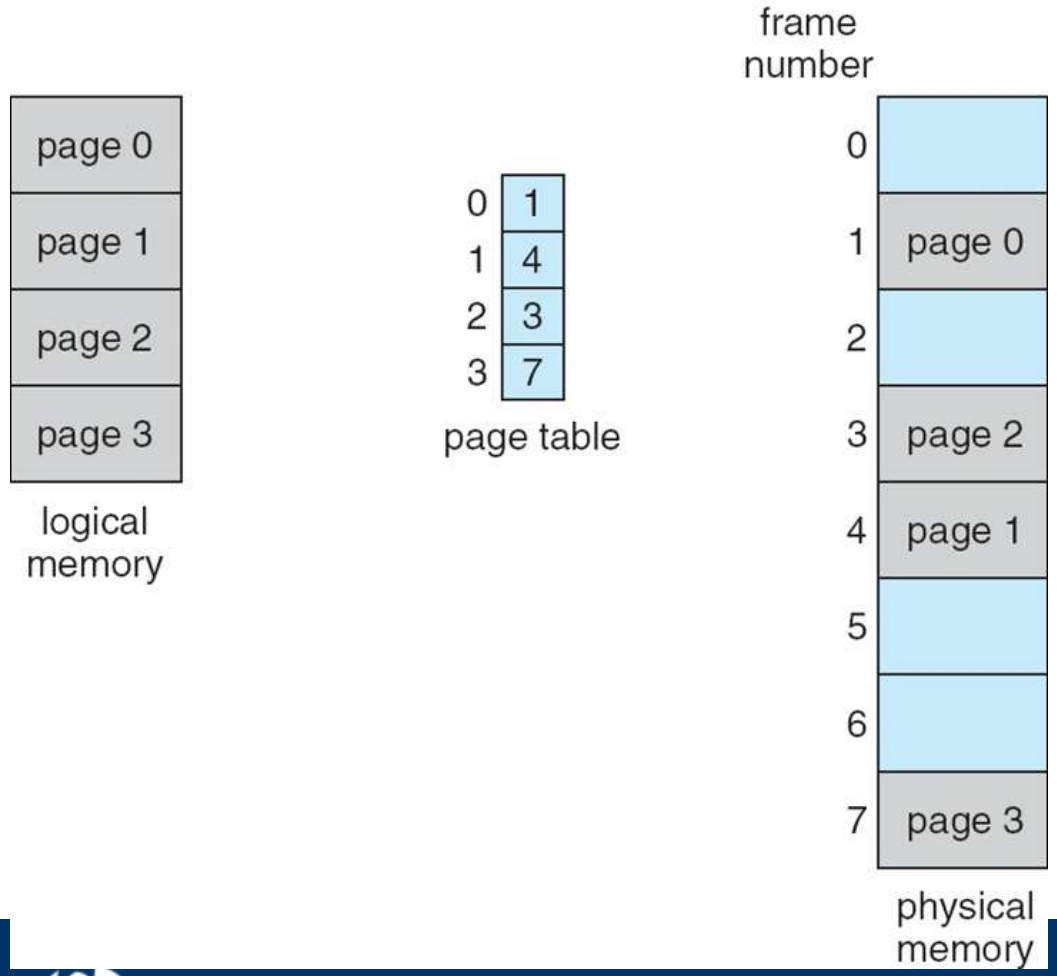
- Alamat yang dihasilkan oleh CPU (memori logic) terdiri 2 bagian yaitu:
- Page Number (p) & Page Offset(d):
  - **Page number** akan menjadi indeks dari page table yang mengandung base address dari setiap alamat di memori fisik.
  - **Page Offset** akan digabung dengan base address untuk mendefinisikan alamat fisik memory yg sebenarnya.



Page Table

- Sebuah rangkaian array dari masukan-masukan (entries) yang mempunyai indeks berupa nomor page (p).
- Untuk proteksi : Setiap masukan terdiri dari **bit valid/invalid** dan nomor page (p).
- Alamat fisik dibentuk dengan menggabungkan nomor frame (f) dengan offset (d).

Paging Model of Logical & Physical Memory



Paging Example :

32-byte memory and 4-byte page table