MANAJEMEN MEMORI

1. Konsep dasar memori
   - Konsep Binding
   - Dynamic Loading
   - Dynamic Linking
   - Overlay
2. Ruang Alamat Logika dan Fisik
3. Swapping
4. Pengalokasian Berurutan (Contiguous Allocation)
5. Pengalokasian Tidak Berurutan (Non Contiguous Allocation)

KONSEP DASAR

- Memori sebagai tempat penyimpanan instruksi/ data dari program
- Memori adalah pusat kegiatan pada sebuah komputer, karena setiap proses yang akan dijalankan, harus melalui memori terlebih dahulu
- Untuk dapat dieksekusi, program harus dibawa ke memori dan menjadi suatu proses

KONSEP DASAR

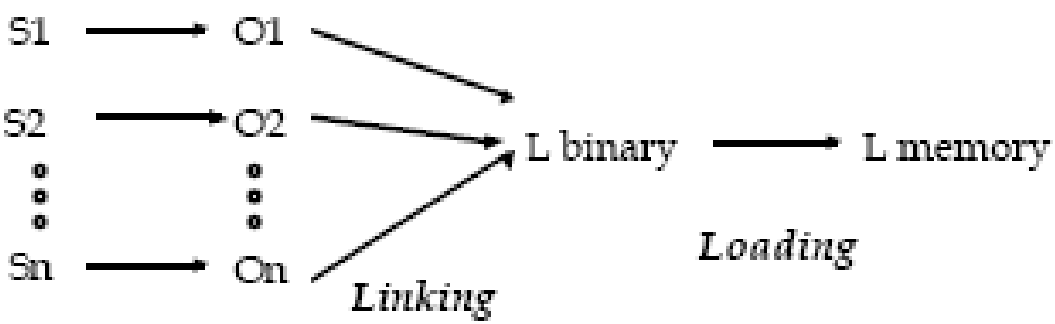
- Manajemen memori :
  - Melacak pemakaian memori (siapa dan berapa besar)
  - Memilih program mana yang akan diload ke memori
  - Alokasi dan dealokasi memori fisik untuk program/ proses-proses dalam menggunakan address space
- Tugas Sistem Operasi :
  - Mengatur peletakan banyak proses pada suatu memori, memori harus dapat digunakan dengan baik → dapat memuat banyak proses dalam suatu waktu

KONSEP BINDING

- Sebelum eksekusi, program berada di dalam disk, dan saat dieksekusi program tersebut perlu berada pada suatu lokasi dalam memori fisik
- Address Binding adalah cara instruksi dan data (yang berada di disk sebagai file executable) dipetakan ke alamat memori
- Alamat (address) pada source program umumnya merupakan alamat simbolik. Sebuah compiler biasanya membutuhkan "mengikat" *(bind)* alamat simbolik ke alamat relokasi

KONSEP BINDING

- Address Binding dapat berlangsung dalam 3 tahap yang berbeda, yaitu : kompilasi, load atau eksekusi dari suatu program



- Bagaimana Sistem Operasi menempatkan program di memori :
  - Kompilasi dan Linking menerjemahkan semua simbol data berdasarkan alamat acuan absolut
  - Proses relokasi (proses mapping program → lokasi memori)
    - Jika program berada di memori, maka semua alamat lojik dalam program harus dikonversi ke alamat fisik.
    - Statis : relokasi alamat dilakukan sebelum program dijalankan
    - Dinamis : relokasi alamat dilakukan pada saat referensi setiap instruksi atau data

DYNAMIC LOADING

- Dengan dynamic loading, suatu routine tidak diload sampai dipanggil. Semua routine disimpan pada disk sebagai format *relocatable load*
- Mekanisme dasar :
  - Program utama diload dahulu dan dieksekusi
  - Bila suatu routine perlu memanggil routine yang lain, routine yang dipanggil lebih dahulu diperiksa apakah routine yang dipanggil sudah diload. Jika tidak, *relocatable linking loader* dipanggil untuk meload routine yang diminta ke memori dan mengupdate tabel alamat dari program yang mencerminkan perubahan ini.

DYNAMIC LINKING

- **Linking ditunda hingga waktu eksekusi**
- **Program-program user tidak perlu menduplikasi system library**
  - **System library dipakai bersama**
  - **Mengurangi pemakaian space : satu rutin library di memori digunakan secara bersama oleh sekumpulan proses**
  - **Contoh : DLL Win32**
- **Mekanisme menggunakan stub (potongan kecil yang mengindikasikan bagaimana meload library jika routine tidak tersedia saat itu)**
  - Saat stub dieksekusi, ia akan memeriksa apakah rutin ybs sudah berada di dalam memori(diakses oleh proses lain yang run), kalau belum ada maka rutin tersebut diload
  - Stub menempatkan dirinya pada alamat rutin dan mengeksekusi rutin tersebut

DYNAMIC LINKING

- Dynamic Linking membutuhkan beberapa dukungan dari OS, misal :
  - Bila proses-proses di memori utama saling diproteksi, maka SO melakukan pengecekan apakah rutin yang diminta berada diluar alamat.
  - Beberapa proses diijinkan untuk mengakses memori pada alamat yang sama
- File dynamic linking berekstensi .dll, .sys, .drv

OVERLAY

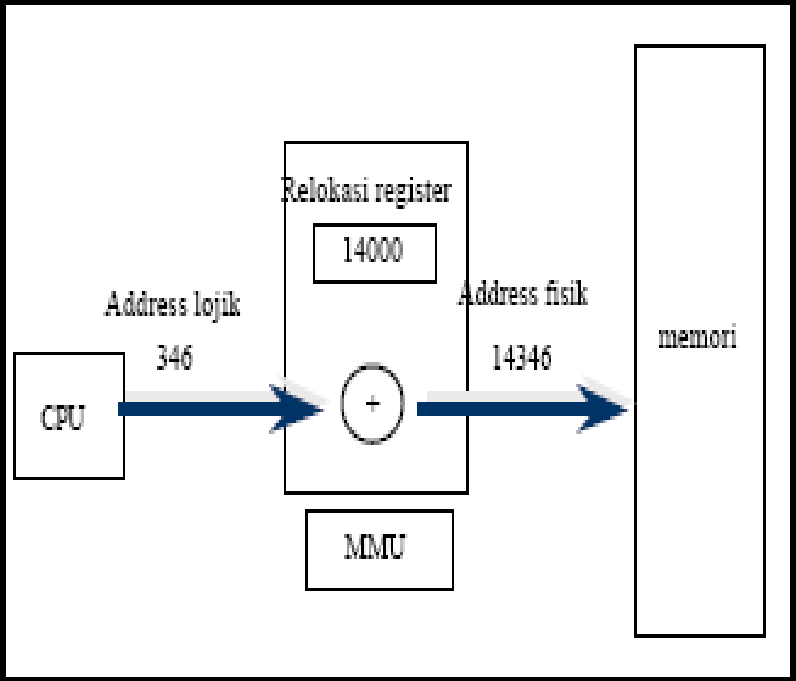
- Hanya instruksi dan data yang diperlukan pada suatu waktu yang disimpan di memori
- Overlay diperlukan jika ukuran proses lebih besar dari memori yang dialokasikan untuknya
- Overlay tidak membutuhkan dukungan khusus dari SO
  - User dapat mengimplementasikan secara lengkap menggunakan struktur file sederhana
  - OS memberitahu hanya jika terdapat I/O yang melebihi biasanya

Ruang Alamat Logika dan Fisik

- Alamat Logika adalah alamat yang di*generate* oleh CPU, disebut juga **Alamat Virtual**
- Alamat Fisik adalah alamat yang terdapat di memori
- Perlu ada penerjemah (translasi) dari alamat logika ke alamat fisik
- MMU (Memory Management Unit) adalah perangkat keras yang memetakan alamat logika ke alamat fisik

Ruang Alamat Logika dan Fisik

- Dalam Skema MMU :
  - Menyediakan perangkat register yang dapat diset oleh CPU: setiap proses mempunyai data set register tersebut (disimpan di PCB)
  - Harga dalam register base/relokasi ditambahkan ke setiap alamat proses user pada saat run dimemori
  - Program-program user hanya berurusan dengan alamat logika saja



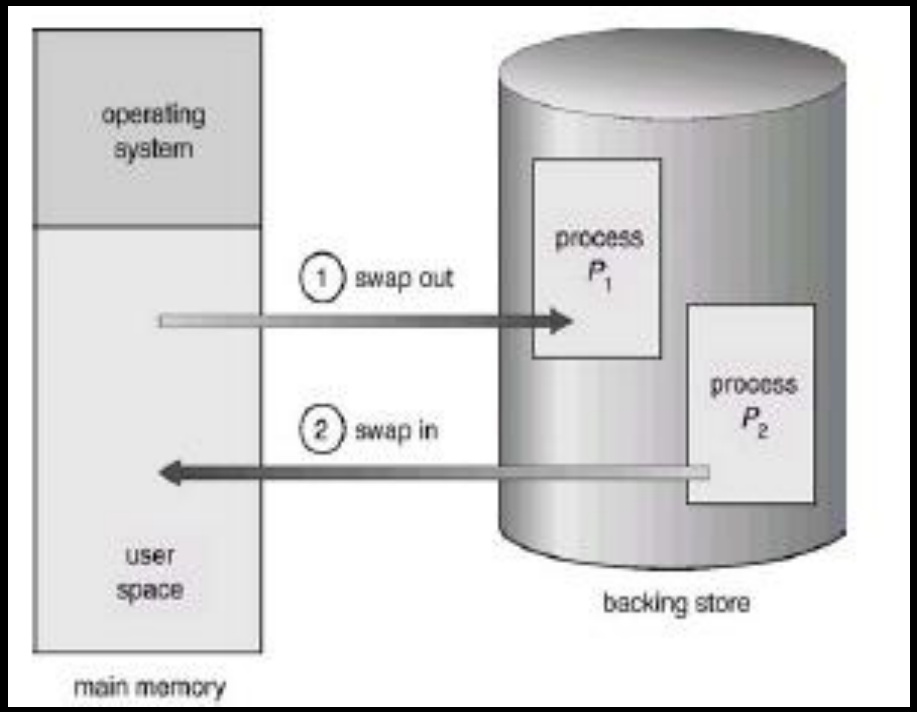
SWAPPING

- Meskipun proses harus berada di memori untuk dieksekusi, tapi proses dapat di *swap* (tukar) sementara keluar memori ke *backing store* dan kemudian membawanya kembali ke memori untuk eksekusi lanjutan

- Penukaran dapat terjadi pada lingkungan dengan multiprogramming dengan penjadwalan CPU Round Robin atau Priority.
  - Bila waktu kuantum habis atau proses yang datang mempunyai prioritas lebih tinggi, maka *memory manager* akan mulai *swap out* proses yang telah selesai atau proses yang yang prioritasnya lebih rendah dan *swap in* proses lainnya ke memori

SWAPPING

- **Skema Swapping**



SWAPPING

- Pada umumnya sebuah proses yang di *swap out* akan menukar kembali ke ruang memori yang sama dengan sebelumnya
- Penukaran membutuhan sebuah *backing storage*
- Bila *CPU Scheduler* memutuskan untuk mengeksekusi proses, OS memanggil dispatcher
  - Dispatcher memeriksa untuk melihat apakah proses selanjutnya pada *ready queue* ada di memori
  - Jika tidak dan tidak terdapat cukup memori bebas, maka *dispatcher swap out* sebuah proses yang ada di memori dan *swap in* proses tersebut

Memori dan Proses

- Monoprogramming →hanya ada satu proses di memori dan sistem operasi

- Multiprogramming → lebih dari satu proses siap di memori
  - Alokasi memori dengan partisi tetap untuk setiap proses
  - Alokasi memori dengan partisi beragam sesuai besarnya proses
  - Alokasi memori dibantu dengan disk (swap area), proses dapat berpindah dari memori ke disk
  - Virtual memori

PENGALOKASIAN MEMORI

- Salah satu tanggung jawab Sistem Operasi adalah mengontrol akses ke sumber daya sistem. Salah satunya adalah memori

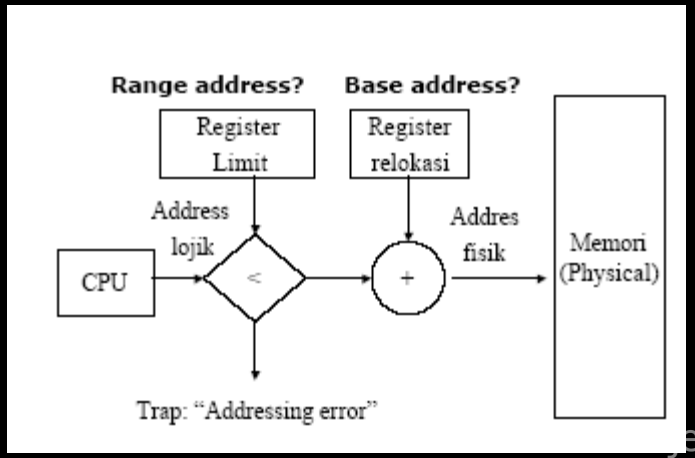
- Pengalokasian memori dibagi 2 tipe, yaitu :
  - Pengalokasian berurutan (Contiguous Allocation)
  - Pengalokasian tidak berurutan (Non Contiguous Allocation)

CONTIGUOUS ALLOCATION

- Pada Multiprogramming memori utama harus mengalokasikan tempat untuk sistem operasi dan beberapa user proses
- Memori harus mengakomodasi baik OS dan proses user
- Memori dibagi menjadi 2 partisi :
  - Untuk OS yang resident
  - Untuk Proses User
- Ada 2 tipe Contiguos Allocation :
  - Single Partition (Partisi Tunggal)
  - Multiple Partition (Partisi Banyak)

CONTIGUOUS ALLOCATION

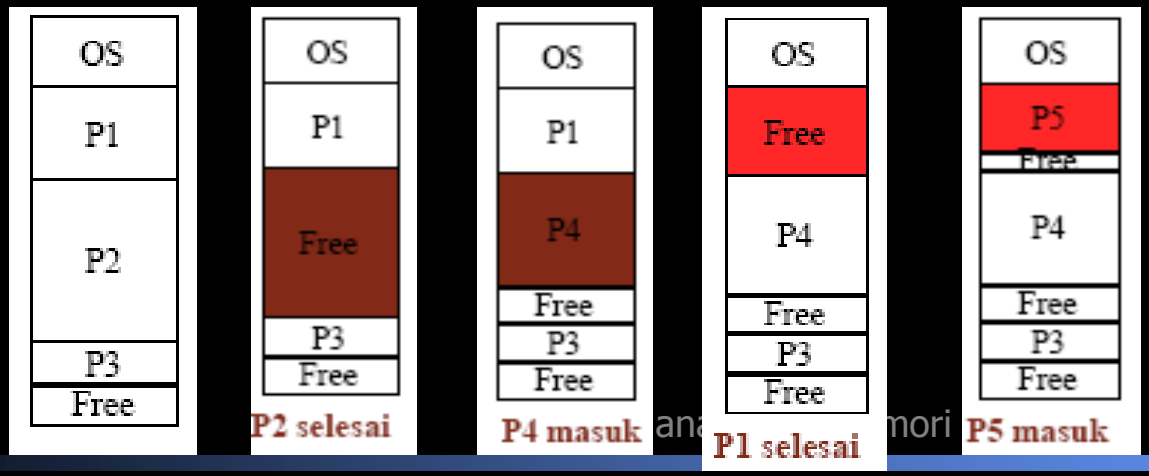
- Single Partition (Partisi Tunggal)
  - Pada skema ini, diasumsikan OS ditempatkan di memori rendah, dan proses user dieksekusi di memori tinggi
  - Proteksi dapat dilakukan dengan dengan menggunakan register relokasi dan register limit
    - Register relokasi → berisi nilai dari alamat fisik terkecil
    - Register Limit → berisi jangkauan alamat logika
    - Alamat logika harus lebih kecil dari register limit



- Misal:
  - Register relokasi (base register) = 100.040
    Register limit (limit register) = 300.000
  - Address lojik = 10.200 (dari CPU)
    Address fisik = 110.240 (akses ke physical memory)
  - address lojik = 810.000
    address fisik= error

CONTIGUOUS ALLOCATION

- Multiple Partition (Partisi Banyak)
  - **Ruang kosong → blok memori yang tersedia, ruang kosong dengan berbagai ukuran tersebar pada memori**
  - **Proses akan dialokasikan memori pada ruang kosong yang cukup besar untuk ditempatinya**
  - **OS akan mengelola informasi mengenai :**
    - **Partisi yang dialokasikan**
    - **Partisi bebas (ruang kosong)**
  - **Contoh multiple allocation**



CONTIGUOUS ALLOCATION

- Multiple Partition (Partisi Banyak)
  - **Ada 2 skema dalam Multiple Partition Allocation:**
    - Partisi Fixed Size (MFT)
      - **Memori dibagi menjadi beberapa blok dengan ukuran tertentu yang seragam**
      - **Setiap partisi berisi tepat 1 proses**
      - **Digunakan oleh IBM OS/360 yang disebut Multiprogramming with a Fixed number of Task (MFT)**
      - **Masalah yang muncul pada MFT :**
        - **Sifat Program dinamis (alokasi dan dealokasi)**
        - **Memori yang teralokasi mungkin lebih besar dari memori yang diminta, sehingga mengakibatkan fragmentasi internal**

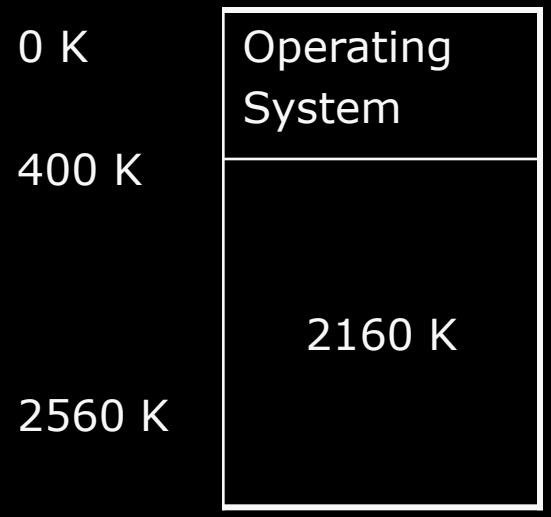
MULTIPLE PARTITION

- **Partisi variable Size (MVT)**
  - Pembagian memori sesuai dengan request dari proses-proses yang ada
  - Peranan memori manajemen semakin penting : list dari partisi yang digunakan, free dll
  - Masalah pada MVT :
    - » Terjadi fragmentasi external
      - Ruang memori free tapi tidak contiguous
      - Hole-hole ada diantara proses
      - Tidak dapat digunakan karena proses terlalu besar untuk menggunakannya

MULTIPLE PARTITION

- Pada MVT OS akan menyimpan tabel yang berisi bagian memori yang tersedia dan yang digunakan:
  - Mula-mula,semua memori tersedia untuk proses user sebagai satu blok besar (*large hole*)
  - Bila proses datang dan memerlukan memori, dicari *hole* yang cukup untuk proses tersebut
  - Bila ditemukan, memory manager akan mengalokasikan sejumlah memori yang dibutuhkan dan menyimpan sisanya untuk permintaan berikutnya
- Contoh :
  - Diasumsikan tersedia memori 2560 Kb dan untuk OS 400 Kb. Sisa 2160 Kb digunakan untuk user proses
  - Diasumsikan terdapat 5 job (P1 s/d P5) terdapat pada *input queue.*
  - Diasumsikan penjadwalan FCFS digunakan untuk load job ke memori. Penjadwalan CPU secara Round Robin (quantum time =1) untuk penjadwalan job yang sudah ada di memori

MULTIPLE PARTITION



JOB QUEUE		
Proses	Memori	Time
P1	600 K	10
P2	1000 K	5
P3	300 K	20
P4	700 K	8
P5	500 K	15

MULTIPLE PARTITION

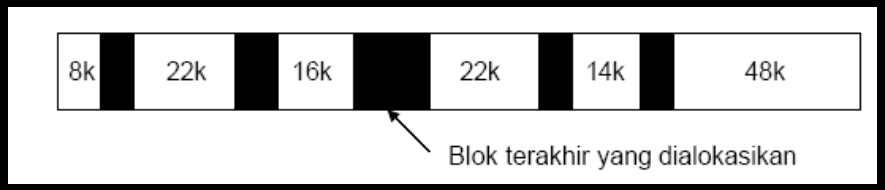
Menggunakan MVT, terdapat beberapa kali hole untuk ukuran berbeda

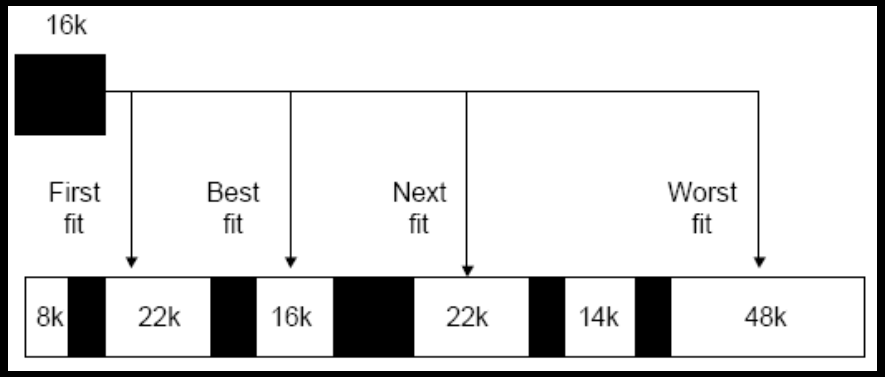
- Bila proses datang dan memerlukan memori, dicari dari hole yang cukup untuk proses

- *Dynamic Storage Allocation* dapat dilibatkan untuk memenuhi permintaan ukuran n dari hole bebas :
  - *First Fit* → alokasi hole yang pertama yang memenuhi permintaan
  - *Best Fit* → alokasi hole terkecil yang memenuhi permintaan
    Dalam stratagi ini memerlukan pencarian keseluruhan hole, kecuali bila ukuran sudah terurut
  - *Worst Fit* → alokasi hole terbesar. Strategi ini memerlukan pencarian keseluruhan hole, kecuali bila ukuran sudah terurut

- Diantara algoritma diatas, *first fit* dan *best fit* lebih baik dibandingkan *worst fit* dalam hal menurunkan waktu dan utilitas penyimpanan. Dan *First Fit* lebih cepat

MULTIPLE PARTITION





- Diantara algoritma diatas, *first fit* dan *best fit* lebih baik dibandingkan *worst fit* dalam hal menurunkan waktu dan utilitas penyimpanan. Dan *First Fit* lebih cepat

CONTIGUOUS ALLOCATION
SISTEM BUDDY

- Sistem Buddy merupakan cara mengelola memori utama dengan memanfaatkan kelebihan penggunaan bilangan biner (2^k; k = 0,1,2 …)
- Contoh :
  - Suatu memori utama pada awalnya memiliki satu lubang besar berukuran 1 Mbyte.Jika suatu proses A berukuran 90 Kbyte memasuki memori, maka permintaan tersebut dialokasikan ke lokasi terdekat yaitu 128 kbyte, karena tidak ada, maka blok 1 Mbyte dipecah menjadi 2, masing-masing berukuran 512 kbyte

FRAGMENTASI EXTERNAL & INTERNAL

- Fragmentasi external terjadi pada situasi dimana terdapat cukup ruang memori total untuk memenuhi permintaan, tetapi tidak dapat langsung dialokasikan karena tidak berurutan
- Fragmentasi Internal terjadi pada situasi dimana memori yang dialokasikan lebih besar daripada memori yang diminta, sehingga terdapat sebagian memori untuk satu partisi tertentu yang tidak digunakan

FRAGMENTASI EXTERNAL & INTERNAL

- Lubang-lubang kecil diantara blok-blok memori yang digunakan dapat diatasi dengan *memori compaction*
- *Memori Compaction* adalah operasi menggabungkan semua lubang kecil menjadi satu lubang besar dengan memindahkan semua proses agar proses saling berdekatan

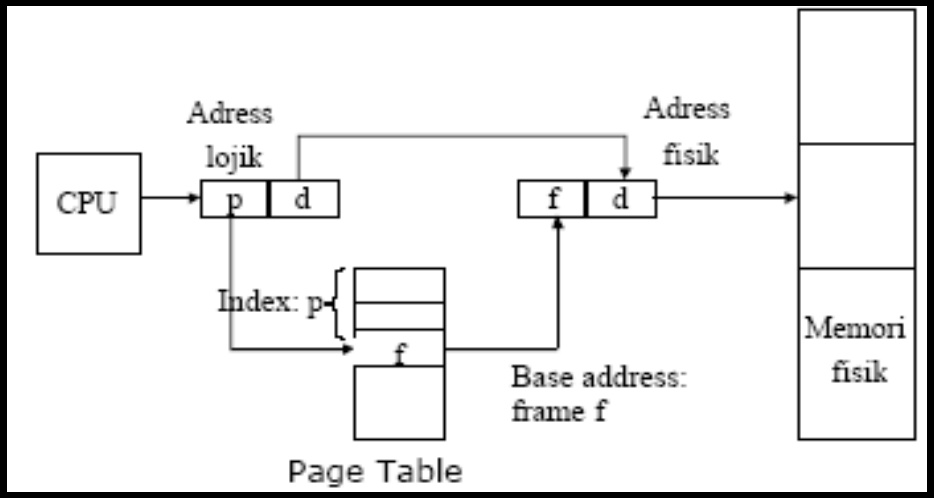
NON CONTIGUOS ALLOCATION

- **Paging**
  - Paging adalah solusi untuk permasalahan fragmentasi external
  - Memori fisik dibagi ke dalam blok-blok ukuran tetap yang disebut *"frame"*
  - Memori logika dibagi ke dalam blok-blok dengan ukuran yang sama yang disebut *"page"*
  - Untuk menjalankan program berukuran n page, harus dicari frame kosong sebanyak n untuk meload program
  - Page table digunakan untuk translasikan alamat lojik ke alamat fisik

NON CONTIGUOUS ALLOCATION
PAGING

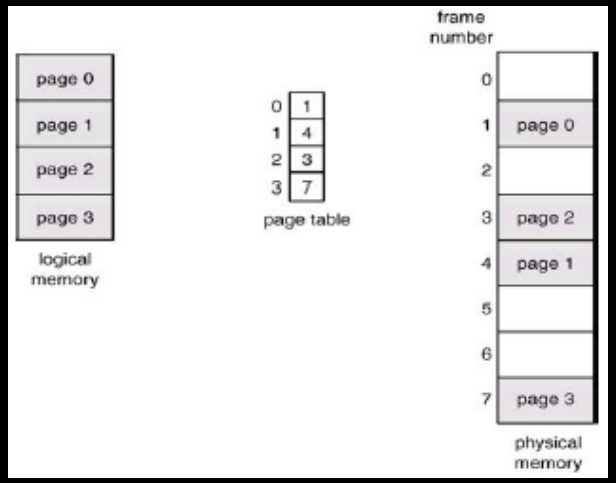
- **Alamat yang dibangkitkan CPU dibagi menjadi :**
  - Page number (p) → digunakan sebagai index ke page table. Page table berisi alamat basis dari setiap page pada memori fisik
  - Page Offset (d) → dikombinasikan dengan alamat basis untuk mendefinisikan alamat memori fisik yang dikirim ke unit memori

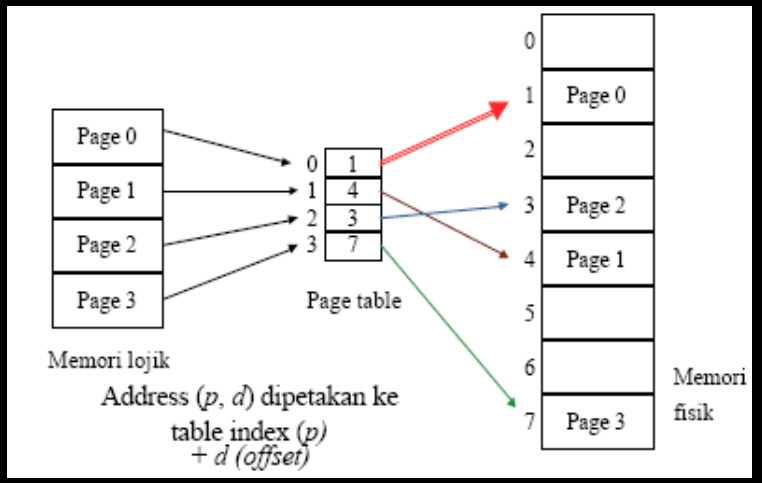
- **Skema translasi alamat**



NON CONTIGUOUS ALLOCATION
PAGING

- **Model Paging**





- **Ukuran page atau frame ditentukan oleh hardware**
  - **Ukuran page merupakan bilangan 2 pangkat k mulai 512 sampai 8192 tergantung arsitektur komputer**

NON CONTIGUOUS ALLOCATION
SEGMENTASI

- **Segmentasi**
  - Segmentasi adalah skema pengaturan memori yang mendukung user untuk melihat memori tersebut
  - Tiap-tiap segmen memiliki nama dan panjang.
  - Pandangan user mengenai memori:

NON CONTIGUOUS ALLOCATION
SEGMENTASI

- **Segmentasi**
  - Dukungan Hardware :
    - Pemetaan ke alamat fisik dilakukan dengan menggunakan tabel segmen, masing-masing berisi base dan limit