MODUL SISTEM OPERASI

(CCI210)

MODUL SESI 7

MEMORY MANAGEMENT

DISUSUN OLEH

ADI WIDIANTONO, SKOM, MKOM.

UNIVERSITAS ESA UNGGUL

2020

VIRTUAL MEMORY

A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu :

1. Memahami konsep dasar dari vitual memory.
2. Memahami teknik dari virtual memory.
3. Mamahami proses kerja Sistem Operasi dalam menjalankan virtual memory.

B. Uraian dan Contoh

1. Virtual Memori

Komputer dapat menangani lebih banyak memori daripada jumlah yang terpasang secara fisik pada sistem. Memori ekstra ini sebenarnya disebut virtual memori dan ini adalah bagian dari hard disk yang diatur untuk meniru RAM komputer.

1.1. Definisi

Virtual memory adalah : Suatu teknik pemetaan memori yang melibatkan memori sekunder, umumnya disk. Secara sistem logika, ukuran memori lebih besar daripada ukuran memori utama secara fisik. Virtual memori melibatkan mekanisme swapping.

Virtual Memori memiliki dua tujuan. Pertama, ini memungkinkan untuk memperluas penggunaan memori fisik dengan menggunakan disk. Kedua, ini memungkinkan memiliki perlindungan memori, karena setiap alamat virtual diterjemahkan ke alamat fisik.

1.2. Konsep Kerja

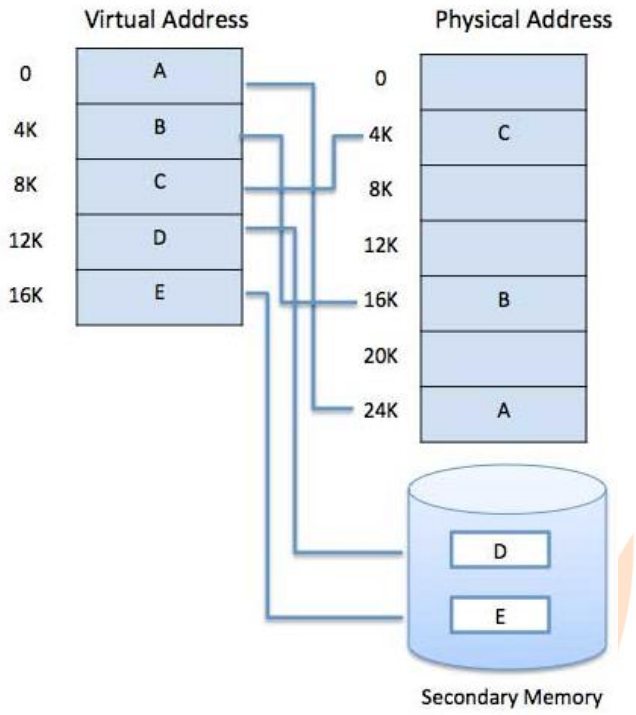
Secara konsep, kerja dari virtual memori adalah sbb:

- Image proses akan diinisialisasi di area swap space, yaitu suatu lokasi di media penyimpan sebagai ekstensi memori utama.
- Swap space dapat berupa suatu berkas atau partisi khusus, dan unit terkecilnya disebut page.
- Pengalihan page dari swap space ke memori utama menggunakan teknik lazy swapper, yaitu hanya page proses yang dibutuhkan yang akan dialihkan ke memori utama.

Konsep dasar dari Virtual Memory adalah

- Pemisahan antara “user logical memory” (virtual) dengan “physical memory”.
- Logical address space (program) dapat lebih besar dari alokasi memori fisik yang diberikan.
- Hanya sebagian kecil dari program yang harus berada di memori untuk eksekusi.
- Terdapat bagian dari disk menyimpan sisa page (program) yang sedang dijalankan di memori.

Mikroprosesor modern yang ditujukan untuk penggunaan umum, unit manajemen memori, atau MMU, dibangun ke dalam perangkat keras. Tugas MMU adalah menerjemahkan alamat virtual menjadi alamat fisik. Contoh dasar diberikan di gambar bawah ini –



Gambar 1.1. Vitual Memori

Virtual Memory dapat diimplementasikan melalui teknik:

- ❖ Demand Paging
- ❖ Demand Segmentation

1.3. Keuntungan

Keuntungan dalam menggunakan virtual memori:

- Lebih sedikit memori yang diperlukan per proses.
- System response menjadi lebih cepat, karena tidak semua bagian image proses perlu dialokasikan ke memori utama → proses dapat dieksekusi lebih cepat.
- Lebih banyak proses yang dapat dijalankan → multiprogramming

1.4. Terminology

Berikut ini terminology yang digunakan dalam Virtual Memori

Virtual memory	A storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program-generated addresses are translated automatically to the corresponding machine addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations.
Virtual address	The address assigned to a location in virtual memory to allow that location to be accessed as though it were part of main memory.
Virtual address space	The virtual storage assigned to a process.
Address space	The range of memory addresses available to a process.
Real address	The address of a storage location in main memory.

1.5. KARAKTERISTIK PAGING & SEGMENTATION

Vitual memori diimplementasikan dengan teknik paging dan segmentation, berikut ini karakteristik dari teknik tersebut:

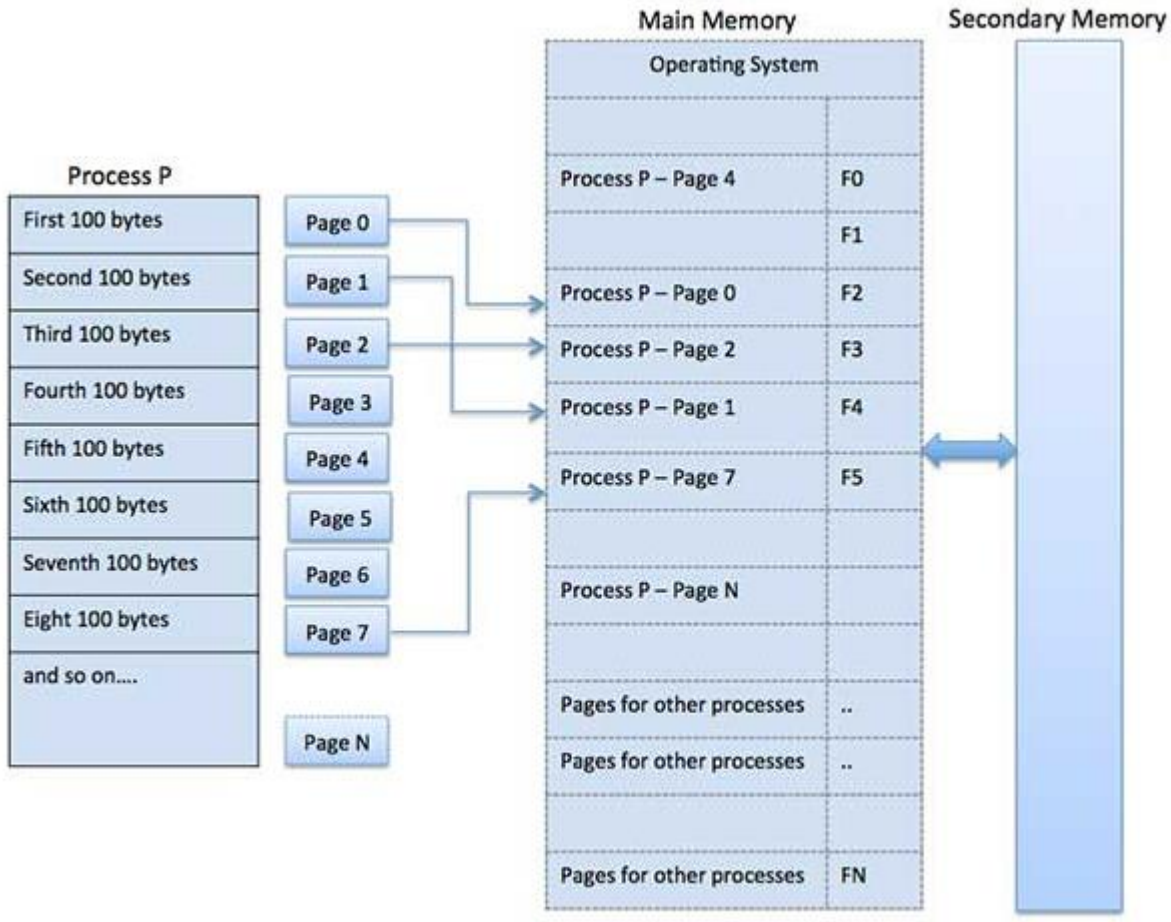
Simple Paging	Virtual Memory Paging	Simple Segmentation	Virtual Memory Segmentation
Main memory partitioned into small fixed-size chunks called frames	Main memory partitioned into small fixed-size chunks called frames	Main memory not partitioned	Main memory not partitioned
Program broken into pages by the compiler or memory management system	Program broken into pages by the compiler or memory management system	Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)	Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)
Internal fragmentation within frames	Internal fragmentation within frames	No internal fragmentation	No internal fragmentation
No external fragmentation	No external fragmentation	External fragmentation	External fragmentation
Operating system must maintain a page table for each process showing which frame each page occupies	Operating system must maintain a page table for each process showing which frame each page occupies	Operating system must maintain a segment table for each process showing the load address and length of each segment	Operating system must maintain a segment table for each process showing the load address and length of each segment
Operating system must maintain a free frame list	Operating system must maintain a free frame list	Operating system must maintain a list of free holes in main memory	Operating system must maintain a list of free holes in main memory
Processor uses page number, offset to calculate absolute address	Processor uses page number, offset to calculate absolute address	Processor uses segment number, offset to calculate absolute address	Processor uses segment number, offset to calculate absolute address
All the pages of a process must be in main memory for process to run, unless overlays are used	Not all pages of a process need be in main memory frames for the process to run. Pages may be read in as needed	All the segments of a process must be in main memory for process to run, unless overlays are used	Not all segments of a process need be in main memory for the process to run. Segments may be read in as needed
	Reading a page into main memory may require writing a page out to disk		Reading a segment into main memory may require writing one or more segments out to disk

2. PAGING

Komputer dapat menangani lebih banyak memori daripada jumlah yang terpasang secara fisik pada sistem. Memori ekstra ini sebenarnya disebut memori virtual dan itu adalah bagian dari harddisk yang diatur untuk meniru RAM komputer. Teknik paging berperan penting dalam mengimplementasikan memori virtual.

Paging adalah teknik manajemen memori di mana ruang alamat proses dipecah menjadi blok-blok dengan ukuran yang sama yang disebut halaman/page (ukuran pangkat 2, antara 512 byte dan 8192 byte). Ukuran proses diukur dalam jumlah halaman/page.

Demikian pula, memori utama dibagi menjadi blok memori (fisik) berukuran tetap kecil yang disebut bingkai/frame dan ukuran bingkai dijaga agar tetap sama dengan ukuran halaman untuk mendapatkan pemanfaatan optimal dari memori utama dan untuk menghindari fragmentasi eksternal.



Gambar 2.1. Page

Address Translation

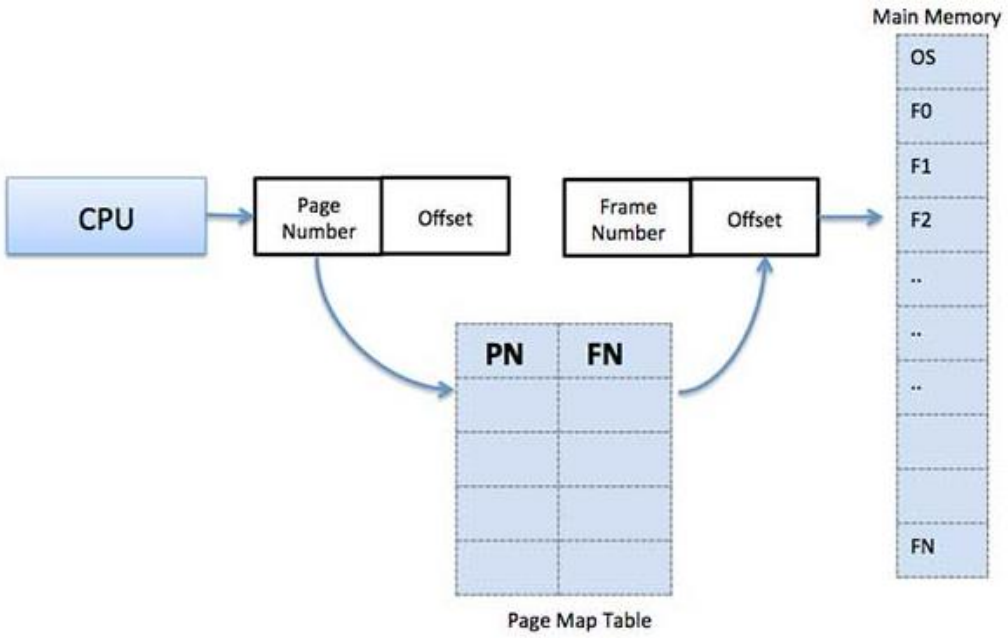
Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

Frame address is called **physical address** and represented by a **frame number** and the **offset**.

Physical Address = Frame number + page offset

A data structure called **page map table** is used to keep track of the relation between a page of a process to a frame in physical memory.



Gambar 2.2. Page Map Table

Ketika sistem mengalokasikan bingkai ke halaman mana pun, itu menerjemahkan alamat logis ini menjadi alamat fisik dan membuat entri ke tabel halaman untuk digunakan selama pelaksanaan program.

Ketika suatu proses akan dieksekusi, halaman terkait dimuat ke dalam frame memori yang tersedia. Misalkan Anda memiliki program sebesar 8Kb tetapi memori Anda hanya dapat menampung 5Kb pada suatu waktu tertentu, maka konsep paging akan muncul. Saat komputer kehabisan RAM, sistem operasi (OS) akan memindahkan halaman memori yang menganggur atau tidak diinginkan ke memori sekunder untuk membebaskan RAM untuk proses lain dan mengembalikannya saat dibutuhkan oleh program. Proses ini berlanjut selama seluruh eksekusi program di mana OS terus

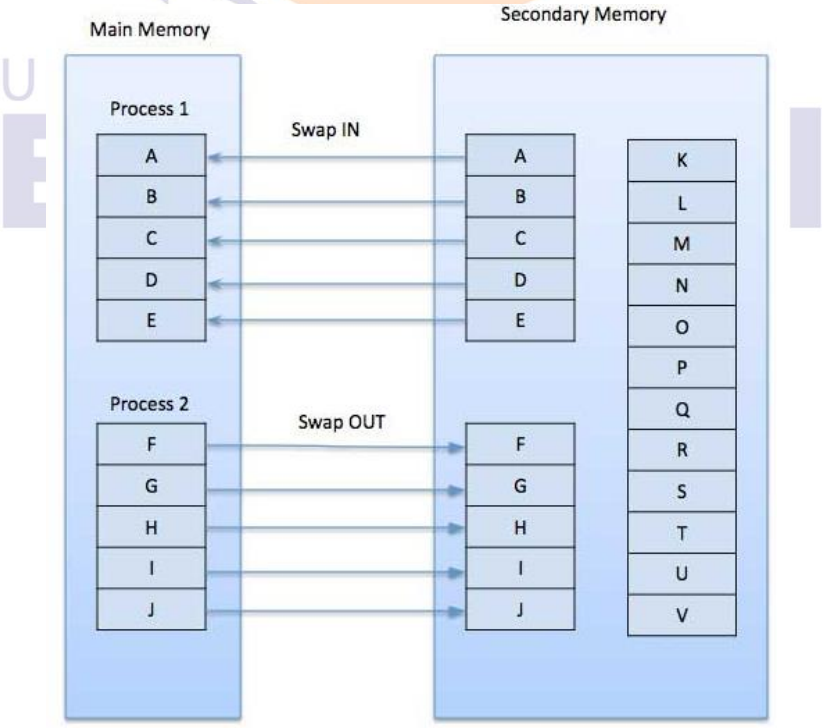
menghapus halaman idle dari memori utama dan menulisnya ke memori sekunder dan mengembalikannya saat diperlukan oleh program.

Keuntungan dan Kerugian Paging

- Paging mengurangi fragmentasi eksternal, tetapi masih mengalami fragmentasi internal.
- Paging mudah diimplementasikan dan diasumsikan sebagai teknik manajemen memori yang efisien.
- Karena ukuran halaman dan bingkai sama, pertukaran menjadi sangat mudah.
- Tabel halaman membutuhkan ruang memori ekstra, jadi mungkin tidak baik untuk sistem yang memiliki RAM kecil.

2.1. DEMAND PAGING

Sistem *demand paging* sangat mirip dengan sistem halaman dengan pertukaran di mana proses berada di memori sekunder dan halaman dimuat hanya atas permintaan, bukan sebelumnya. Ketika pertukaran (switch) konteks terjadi, sistem operasi tidak menyalin halaman program lama ke disk atau halaman program baru mana pun ke dalam memori utama. Sebaliknya, ia hanya mulai menjalankan program baru setelah memuat halaman pertama dan mengambilnya. halaman program sebagaimana mereka direferensikan.



Gambar 2.3. Demand Paging

Saat menjalankan program, jika program mereferensikan halaman yang tidak tersedia di memori utama karena ditukar beberapa saat yang lalu, prosesor memperlakukan referensi memori yang tidak valid ini sebagai kesalahan halaman dan mentransfer kontrol dari program ke sistem operasi ke menuntut halaman itu kembali ke dalam memori.

Berikut adalah keuntungan dari Demand Paging -

- Memori virtual yang besar.
- Penggunaan memori lebih efisien.
- Tidak ada batasan pada tingkat multiprogramming.

Kekurangan

- Jumlah tabel dan jumlah overhead prosesor untuk menangani interupsi halaman lebih besar daripada kasus teknik manajemen halaman sederhana.

2.1.1. PAGE REPLACEMENT

Algoritme penggantian halaman adalah teknik yang digunakan Sistem Operasi untuk memutuskan halaman memori mana yang akan ditukar, menulis ke disk ketika halaman memori perlu dialokasikan. Page replacement diperlukan untuk:

- Mencegah alokasi yang berlebihan dari memori dengan memodifikasi layanan rutin page-fault melalui page
- Menggunakan *modify bit* untuk mengurangi overhead transfer page – hanya modifikasi page yang ditulis di disk.
- Page replacement melengkapi pemisahan antara memori logik dan memori fisik – virtual memori yang besar dapat memenuhi kebutuhan memori fisik yang kecil

DASAR PAGE REPLACEMENT

1. Tentukan lokasi yang diminta page pada disk.
2. Tentukan frame bebas :
    - ❖ Jika tersedia frame bebas, maka dapat digunakan
    - ❖ Jika tidak tersedia frame bebas, gunakan algoritma penggantian untuk memilih kandidat frame.

3. Baca page yang dituju ke dalam frame bebas (yang baru). Update page dan frame table.
4. Restart process.

2.1.2. ALGORITMA PAGE REPLACEMENT

Algoritma page replacement secara sederhana adalah sbb:

- Pilih page fault terendah.
- Evaluasi algoritma dengan menjalankan particular string dari memori acuan (reference string) dan menghitung jumlah page fault dari string.
- Contoh, reference string sebagai berikut :

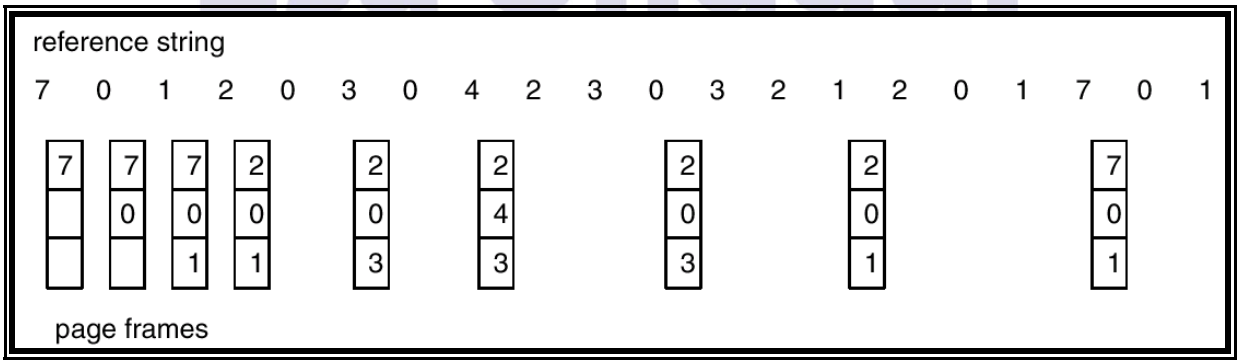
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

Algoritma dasar, terdapat algoritma dasar tertentu yang digunakan untuk pemilihan page yang akan diganti. Algoritma penggantian yang dikenal diantaranya adalah:

- Optimal
- Least Recently Used (LRU)
- First In First Out (FIFO)

ALGORITMA OPTIMAL

- ❖ Melakukan pemlilihan terhadap page yang tenggang waktu ke referensi berikutnya paling lama.
- ❖ Algoritma ini tidak mungkin diterapkan, karena akan memerlukan sistem operasi yang memeiliki pengetahuan tentang event yang akan terjadi di masa dating dengan sempurna.

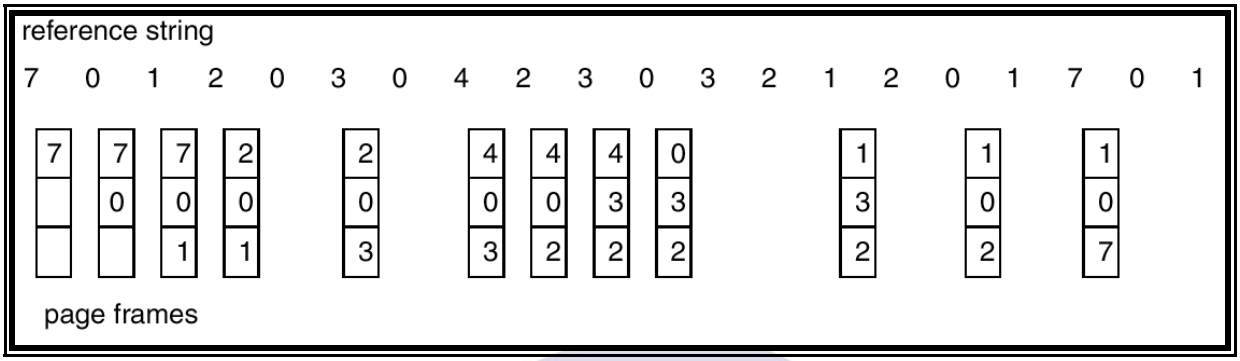


Gambar 2.3. Algoritma Optimal

ALGORITMA LRU

- ❖ Mengganti page yang paling lama tidak digunakan/diakses.

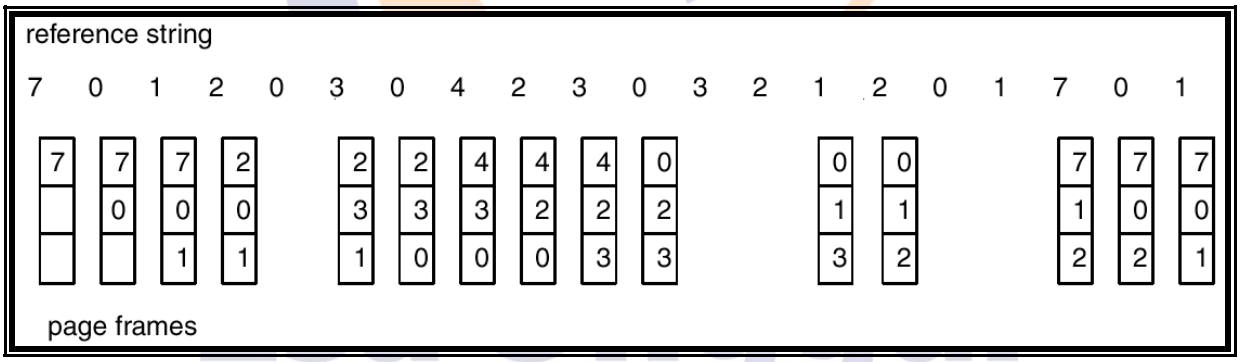
- Asumsi page yang diakses sekarang → kemungkinan besar akan diakses lagi (sulit diprediksi juga).
- Masalah: mendeteksi (memelihara) LRU semua page → bantuan hardware yang cukup rumit.



Gambar 2.4. Algoritma LRU

ALGORITMA FIFO

- Mengganti page yang terlama berada di memori.
- Data struktur FIFO queue yang menyimpan kedatangan pages di memori.
- Masalah: menambah page frame → page fault tidak berkurang.



Gambar 2.5. Algoritma FIFO

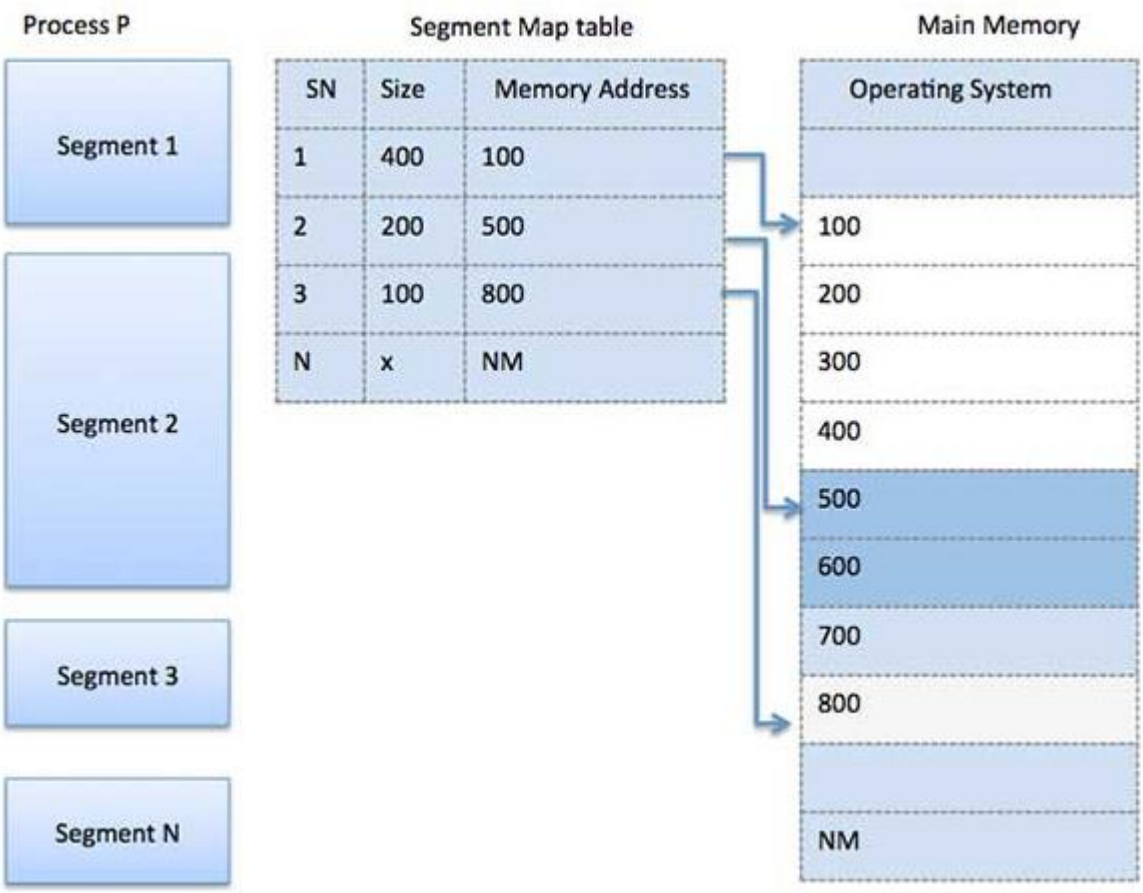
3. Segmentasi (*Segmentation*)

Segmentasi adalah teknik manajemen memori di mana setiap pekerjaan dibagi menjadi beberapa segmen dengan ukuran berbeda, satu untuk setiap modul yang berisi bagian yang menjalankan fungsi terkait. Setiap segmen sebenarnya adalah ruang alamat logis yang berbeda dari program tersebut.

Ketika suatu proses akan dieksekusi, segmentasinya yang sesuai dimuat ke dalam memori yang tidak bersebelahan meskipun setiap segmen dimuat ke dalam blok memori yang tersedia yang berdekatan.

Manajemen memori segmen bekerja sangat mirip dengan paging tetapi di sini segmen memiliki panjang variabel dimana seperti pada halaman halaman memiliki ukuran tetap.

Segmen program berisi fungsi utama program, fungsi utilitas, struktur data, dan sebagainya. Sistem operasi memelihara tabel peta segmen untuk setiap proses dan daftar blok memori bebas bersama dengan nomor segmen, ukurannya dan lokasi memori yang sesuai di memori utama. Untuk setiap segmen, tabel menyimpan alamat awal segmen dan panjang segmen. Referensi ke lokasi memori mencakup nilai yang mengidentifikasi segmen dan offset.



Gambar 3.1. Segmentasi

C. Daftar Pustaka

1. Operating Systems Internals And Design Principles, Seventh Edition, Ch 8, William Stalling, Prentice Hall, 2012