MANAJEMEN MEMORI VIRTUAL

Pertemuan 9

Sistem Operasi Kom A

T.A 2021-2022

TUJUAN

1. Mahasiswa mampu memahami konsep dasar memori virtual
2. Mahasiswa mampu memahami segmentation dan paging
3. Mahasiswa mampu memahami page replacement pada memori virtual

PENDAHULUAN

- Memori Virtual adalah skema alokasi penyimpanan di mana memori sekunder dapat dialokasi seolah merupakan bagian dari memori utama.
- Alamat digunakan untuk merujuk ke memori dibedakan dari alamat yang digunakan memori sistem untuk mengidentifikasi penyimpanan fisik, dan alamat yang dihasilkan program yang diterjemahkan secara otomatis ke alamat mesin/fisik yang sesuai.
- Ukuran penyimpanan virtual dibatasi oleh skema pengalamatan sistem komputer dan jumlah memori sekunder yang tersedia tidak sama dengan jumlah aktual lokasi penyimpanan utama

KARAKTERISTIK MEMORI VIRTUAL

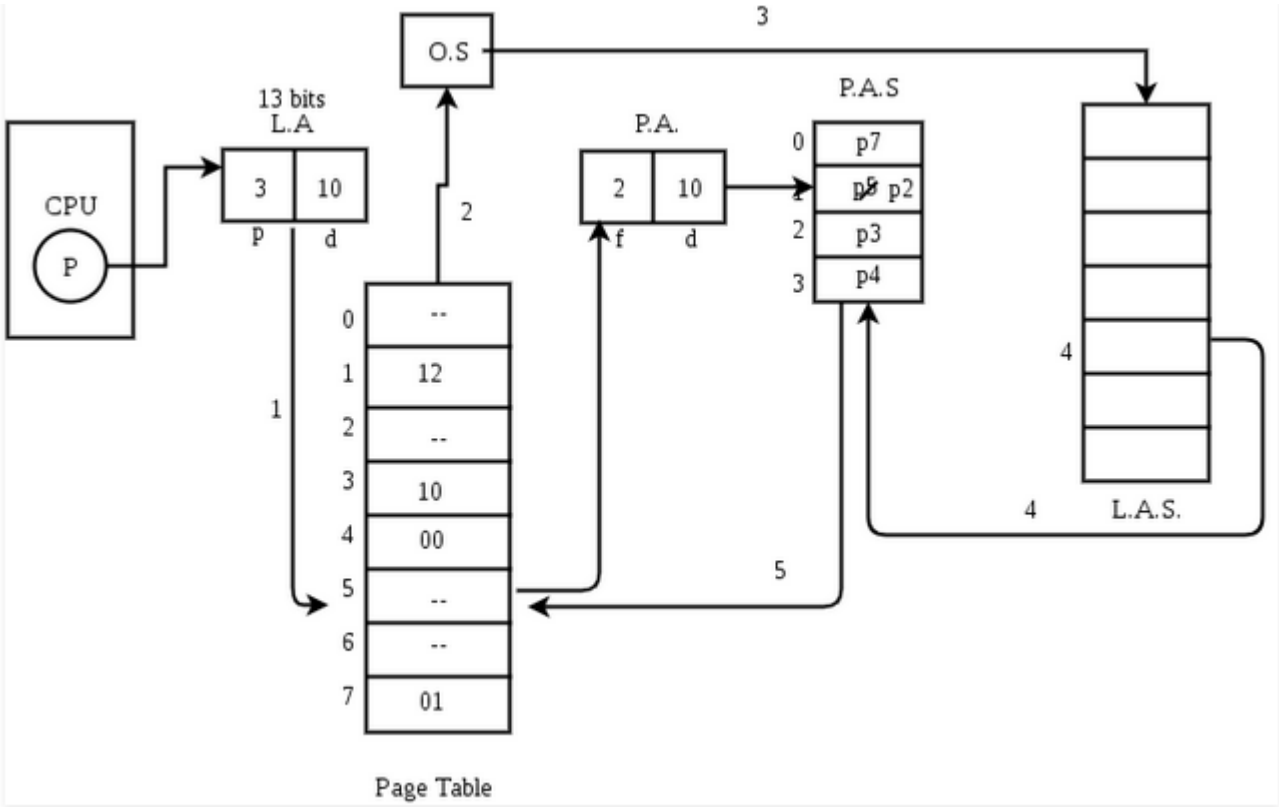
- Semua referensi memori dalam suatu proses adalah alamat logis yang secara dinamis diterjemahkan ke alamat fisik pada saat dijalankan.
- Ini berarti bahwa suatu proses dapat ditukar masuk dan keluar dari memori utama sedemikian sehingga ia menempati tempat yang berbeda di memori utama pada waktu yang berbeda selama pelaksanaan.
- Suatu proses dapat dipecah menjadi beberapa bagian dan bagian-bagian ini tidak perlu secara terus-menerus terletak di memori utama selama eksekusi.
- Kombinasi terjemahan alamat run-time dinamis dan penggunaan halaman atau tabel segmen memungkinkan hal ini terjadi.

LANJUTAN

- Jika karakteristik ini ada, maka tidak perlu semua halaman atau segmen berada di memori utama selama eksekusi.
- Hal ini menunjukkan bahwa halaman yang diperlukan perlu dimuat ke dalam memori pada saat yang diperlukan.
- Memori virtual diimplementasikan menggunakan *Demand Paging* atau *Demand Segmentation*.

DEMAND PAGING

Proses memuat halaman ke dalam memori sesuai permintaan (setiap kali terjadi kesalahan halaman) dikenal sebagai *demand paging*.



LANJUTAN

- *Demand paging* pada prinsipnya hampir sama dengan permintaan halaman (*paging*) hanya saja halaman (*page*) tidak akan dibawa ke ke dalam memori fisik sampai ia benar-benar diperlukan.
- Untuk itu diperlukan bantuan perangkat keras untuk mengetahui lokasi dari halaman saat ia diperlukan.
- Ada tiga kasus yang mungkin dapat terjadi pada saat pengecekan pada halaman yang dibutuhkan, yaitu: halaman ada dan sudah berada di memori – statusnya valid ("1"); halaman ada tetapi masih berada di disk atau belum berada di memori (harus menunggu sampai dimasukkan) – statusnya tidak valid ("0"); halaman tidak ada, baik di memori maupun di *disk* (*invalid reference*).
- Saat terjadi kasus kedua dan ketiga, maka proses dinyatakan mengalami kesalahan halaman.

LANJUTAN

- Permintaan pemberian page menggunakan **swapping**.
- Program swapper yg digunakan:
  - **Lazy swapper** – tidak pernah swap page kedalam memory sampai page benar-benar diperlukan
  - Istilah Swapper berarti memanipulasi seluruh proses, sehingga swapper yang khusus berhubungan dengan pages bernama **pager**

VALID-INVALID BIT

Bagaimana cara menandai sebuah page sudah ada di physical memory?

Gunakan (v)alid-(i)nvalid bit pada page table

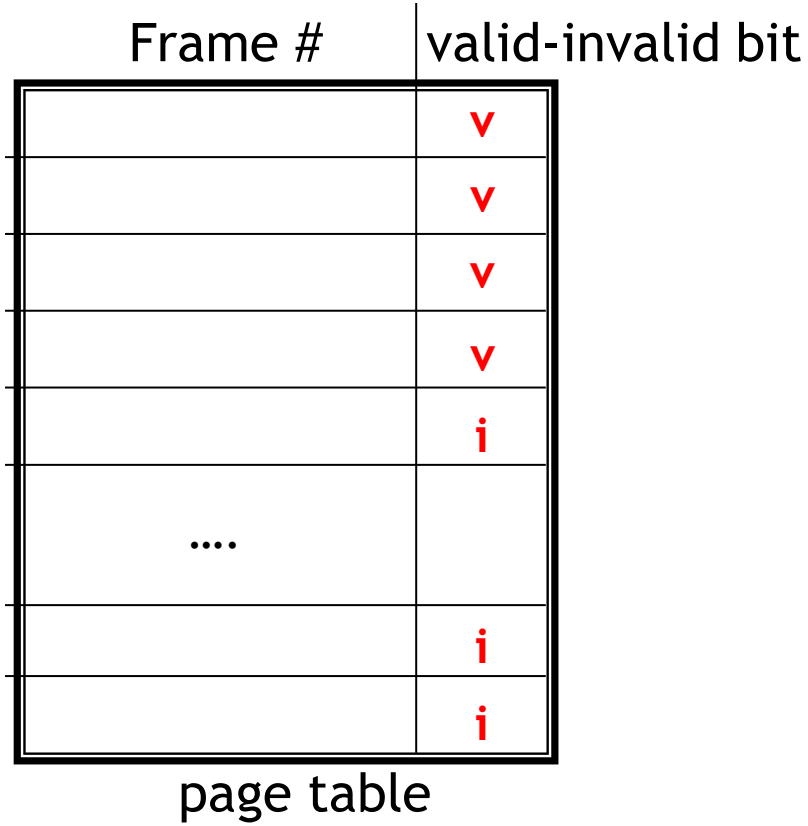
(v)alid : page ada di memory

(i)nvalid : page tidak ada di memory.

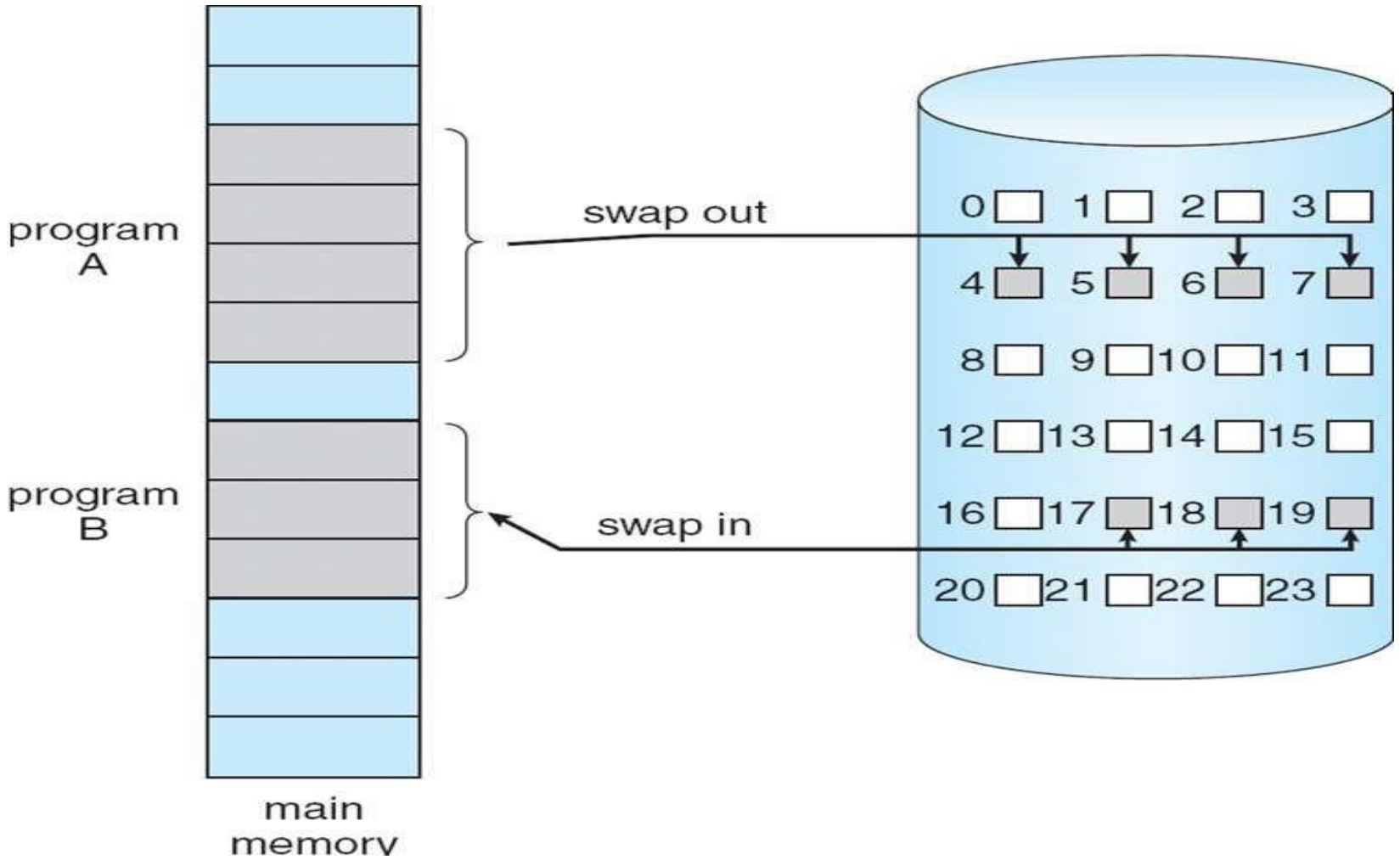
Ada 2 kemungkinan :

- page belum di-load ke memory
- page tersebut tidak berhak diakses proses tsb.

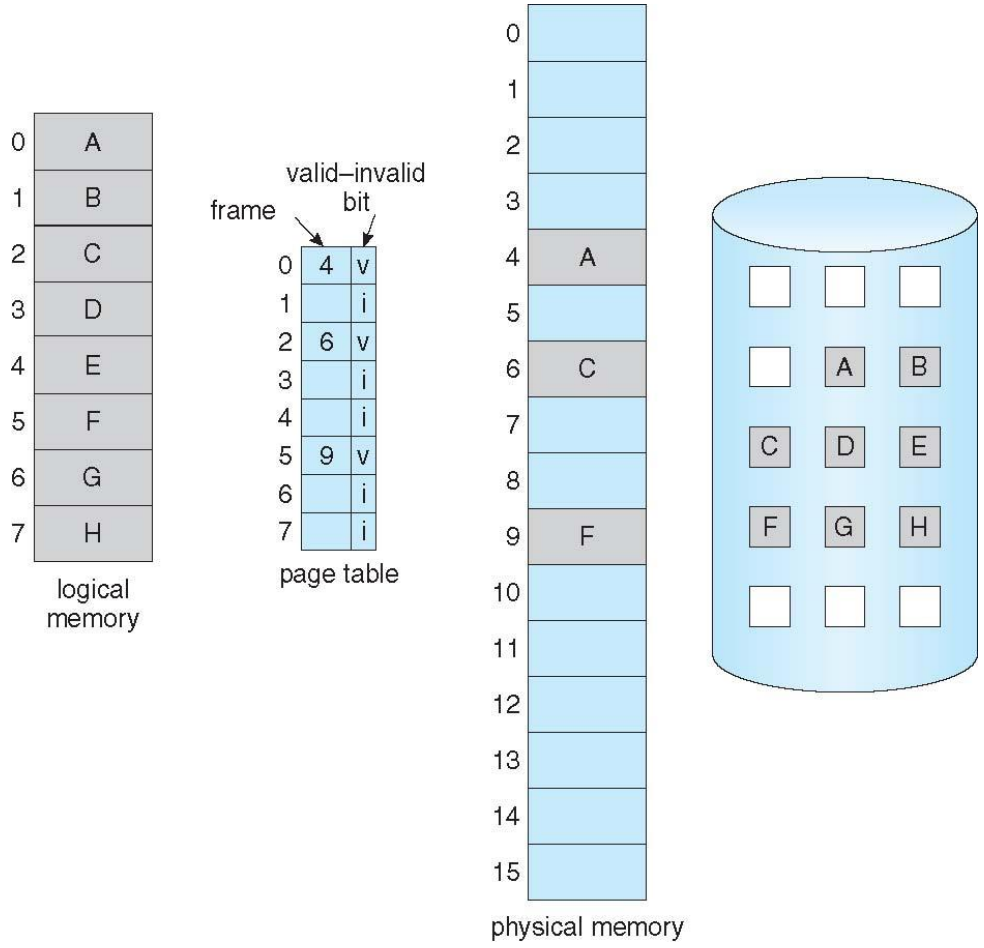
Akses pada page dengan invalid bit disebut dengan page fault.



TRANSFER OF A PAGED MEMORY TO CONTIGUOUS DISK SPACE

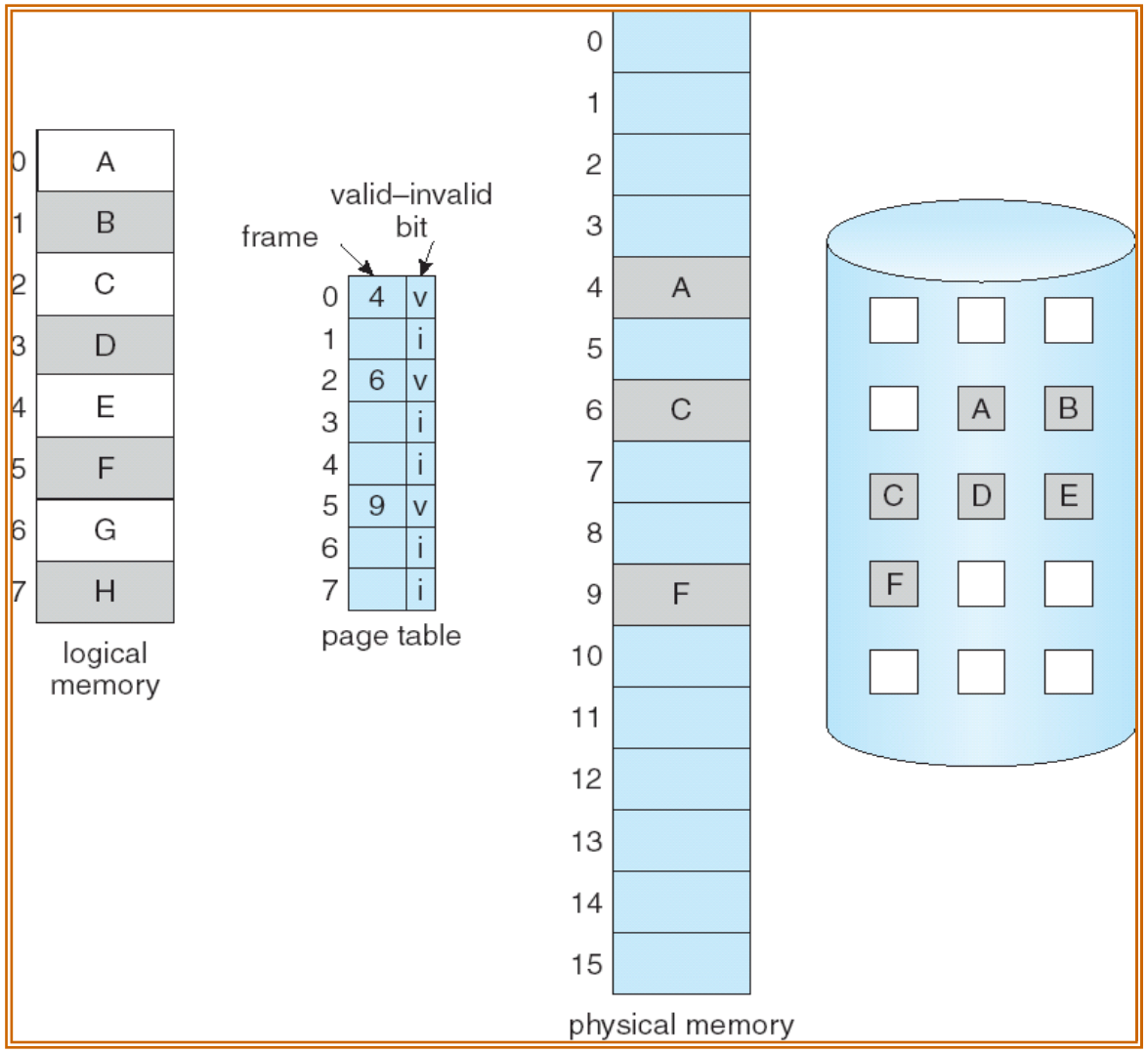


CONTOH



1. Pada awal eksekusi, hanya load Page 0, 2, 5 ke main memory
   - Bit valid untuk Page 0,2,5
2. Saat terjadi eksekusi, CPU harus mengakses data di Page 2
   - Cek Page Table, Page 2 Valid
   - Akses langsung physical memory
3. Kemudian CPU harus mengakses data di Page 1
   - Cek Page Table, Page 1 Invalid -> Page Fault
   - Jalankan prosedur penanganan Page Fault

PAGE TABLE WHEN SOME PAGES ARE NOT IN MAIN MEMORY, BUT IN HDD

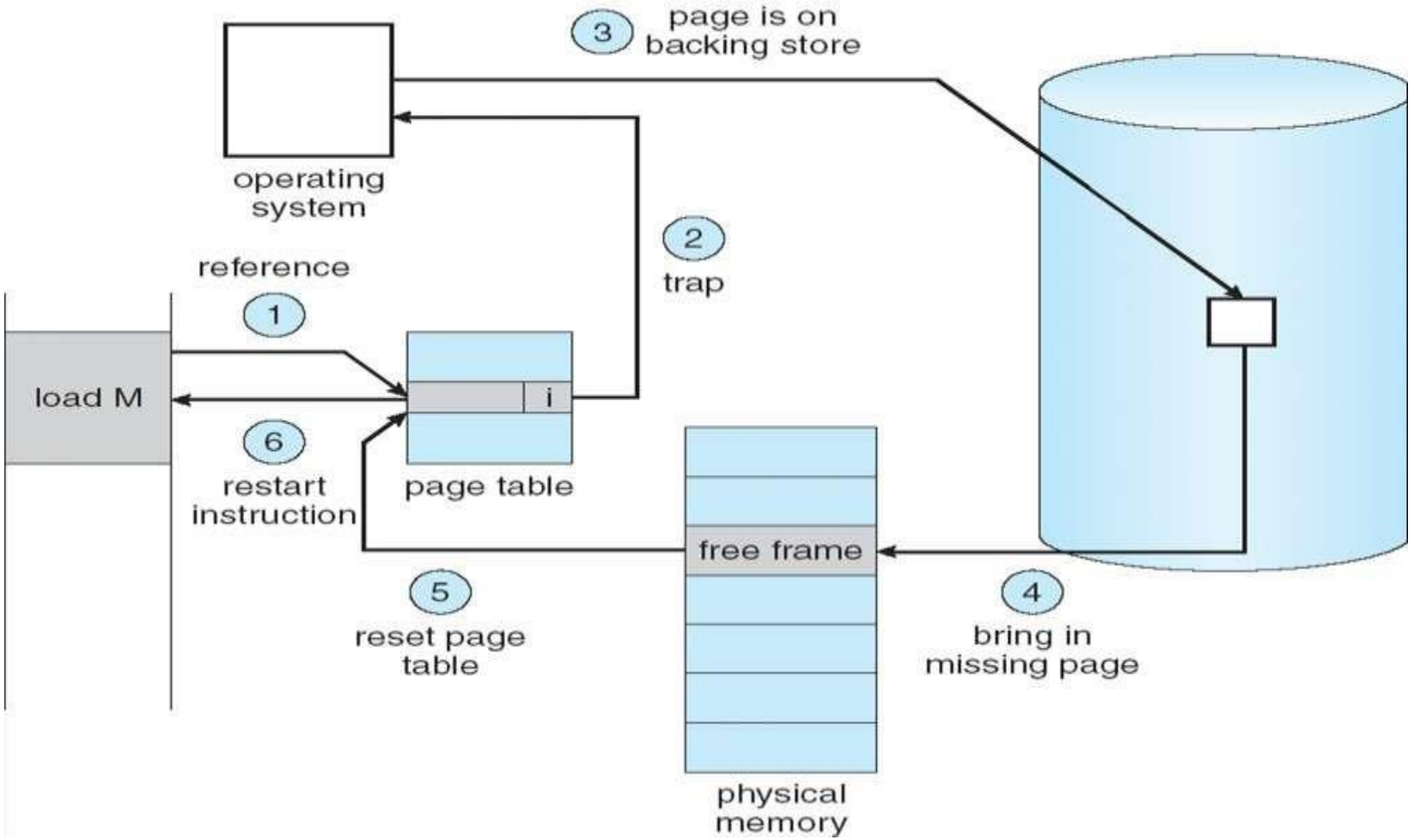


1. Ketika terjadi page fault, maka eksekusi dihentikan sementara karena OS trap.
2. Cek 2 kemungkinan penyebab page fault:
   - Page belum di-load ke memory
   - Page tersebut tidak berhak diakses
3. Jika invalid karena belum di-load, ambil page dari backing store
4. Pindahkan page tadi ke frame bebas
5. Reset page table
6. Restart instruksi

PAGE FAULT

- Jika ada referensi ke sebuah page, ternyata pagenya tidak ada (invalid), maka akan ditrap oleh OS, dan menghasilkan: **page fault**
- Untuk menangani page fault menggunakan prosedur berikut:
  - Memeriksa tabel internal (biasanya ada dlm PCB) unt menentukan valid atau invalid
  - Jika invalid, proses di suspend, jika valid tapi proses belum dibawa ke page, maka bawa page ke memory.
  - Cari sebuah frame bebas (free frame).
  - Jadwalkan operasi sebuah disk untuk membaca page tersebut ke frame yang baru dialokasikan.
  - Saat pembacaan selesai, ubah validation bit menjadi “1” yang berarti page telah ada di memory.
  - Ulangi lagi instruksi program yg ditrap td dari awal sehingga bisa berjalan dgn baik.

STEPS IN HANDLING A PAGE FAULT



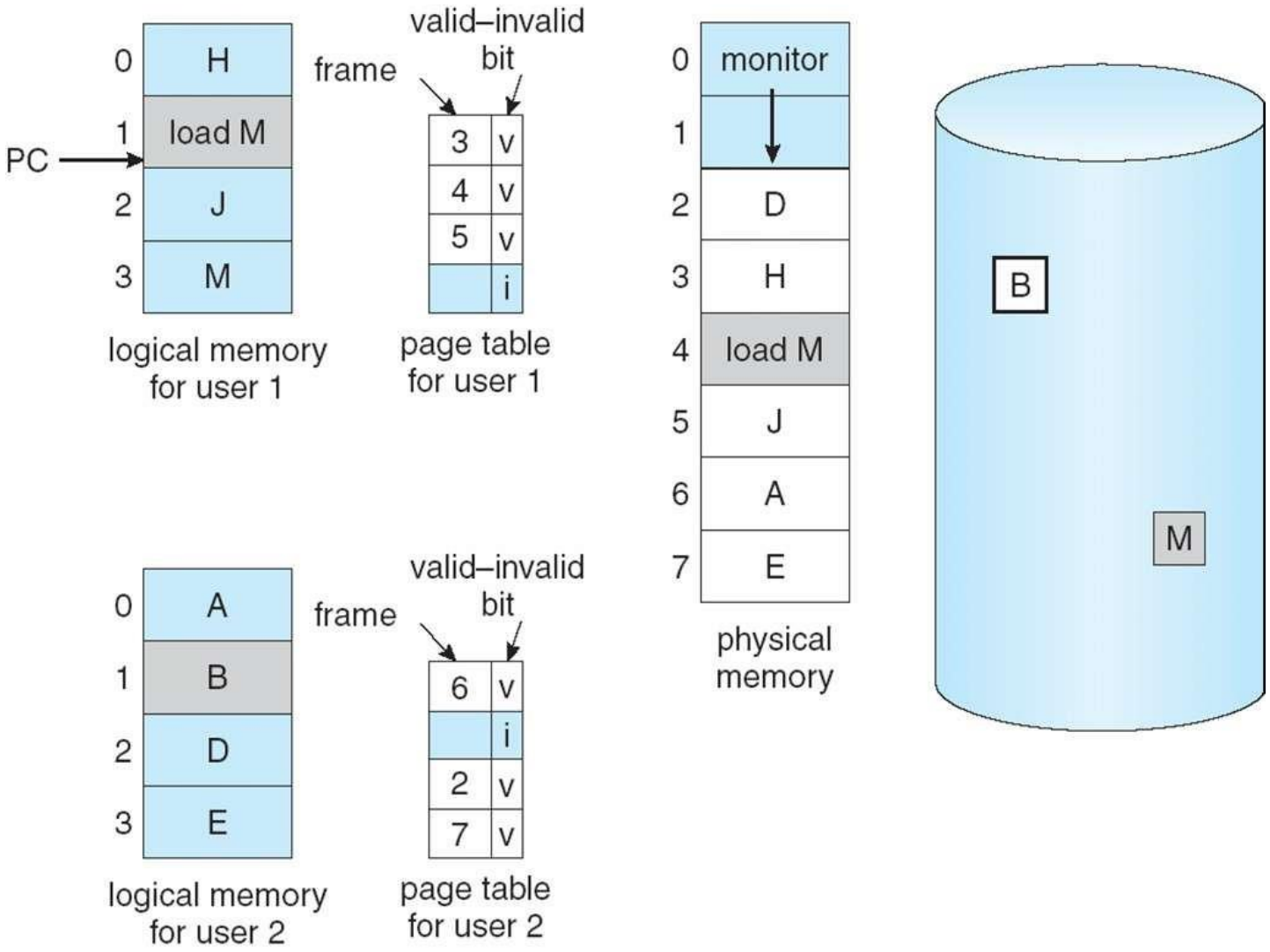
…CONT'D

- Jika CPU mencoba merujuk page yang saat ini tidak tersedia di memori utama, maka akan menghasilkan gangguan akses memori
- OS menempatkan proses yang terputus dalam status pemblokiran. Untuk pelaksanaan, OS harus melanjutkannya dengan membawa page yang diperlukan ke dalam memori.
- OS akan mencari page yang diperlukan di ruang alamat logis.
- Page yang diperlukan akan dibawa dari ruang alamat logis ke ruang alamat fisik. Algoritme page replacement digunakan untuk pengambilan keputusan penggantian page dalam ruang alamat fisik.
- Page table akan diperbarui untuk melanjutkan eksekusi program dan itu akan menempatkan proses kembali ke status siap kemudian sinyal dikirim ke CPU

WHAT HAPPENS IF THERE IS NO FREE FRAME?

- Solution: Page Replacement.
- Pendekatan :
  - Jika tidak ada frame yang kosong, cari frame yang tidak sedang digunakan, lalu kosongkan dengan cara menuliskan isinya ke dalam swap space, dan mengubah semua tabel sebagai indikasi bahwa page tersebut tidak akan berada di memori lagi.
  - Menggunakan algoritma

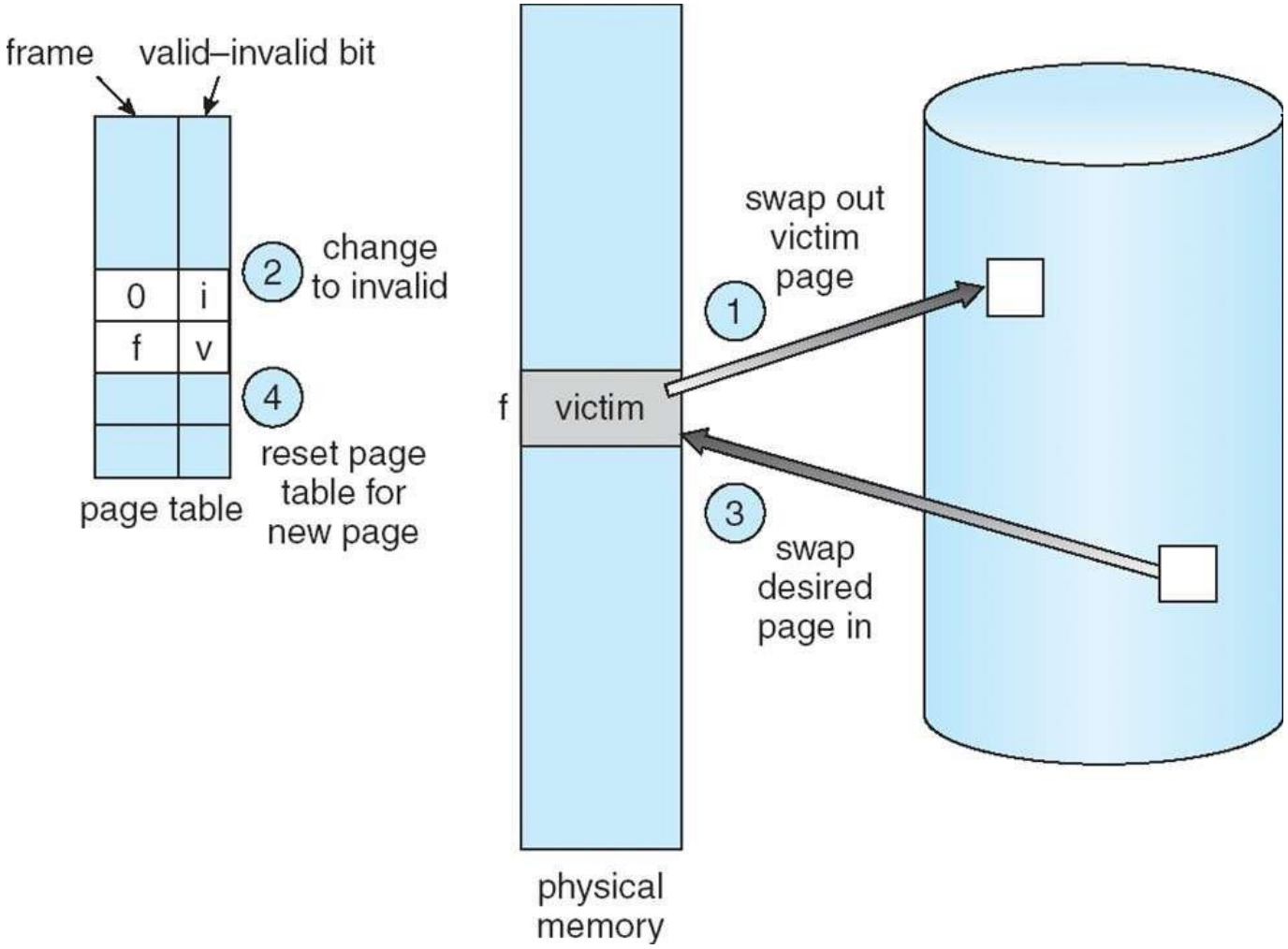
NEED FOR PAGE REPLACEMENT



YANG DILAKUKAN SAAT PAGE REPLACEMENT

- Mencari lokasi page yang diinginkan pada disk.
- Mencari frame yang kosong :
  - Jika ada, maka gunakan frame tersebut.
  - Jika tidak ada, maka kita bisa mengosongkan frame yang tidak sedang dipakai.
    - Gunakan **algoritma page-replacement** untuk menentukan frame yang akan dikosongkan.
  - Tulis page yang telah dipilih ke disk, ubah page-table dan frame-table.
  - Membaca page yang diinginkan ke dalam frame kosong yang baru.
  - Ulangi user process dari awal.

PAGE REPLACEMENT



ALGORITMA UNTUK PAGE AND FRAME REPLACEMENT

Ada dua jenis algoritma yang dibutuhkan pada metode page replacement :

- **Page-replacement algorithm**
- **Frame-allocation algorithm**

Page replacement algorithm bertujuan agar

- page fault serendah mungkin

Frame allocation algorithm menentukan

- Berapa banyak frame yang dialokasikan untuk satu proses?
- Frame mana yang harus di-replace?

ALGORITMA PAGE REPLACEMENT

- Bertujuan untuk mendapatkan **page fault terendah.**
- Ada beberapa Algoritma Page Replacement:
  - Algoritma FIFO
  - Algoritma Optimal
  - Algoritma LRU
  - Algoritma Perkiraan LRU

ALGORTIMA FIFO

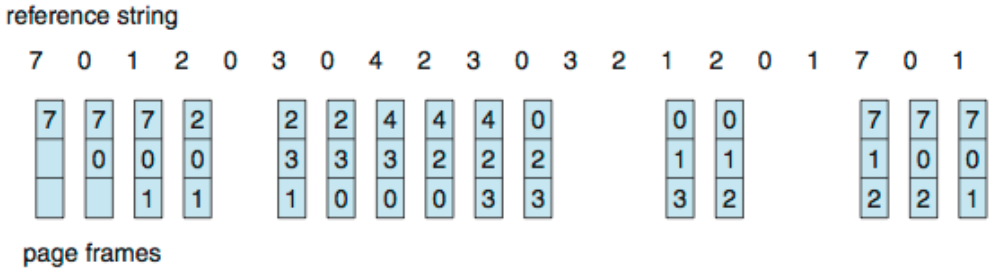
- Page yang diganti adalah page yang **paling lama** berada di memori.
- Mudah diimplementasikan.
- Mudah dimengerti.
- Namun bisa mengalami Anomali Belady.
  - Page fault rate meningkat seiring dengan meningkatnya jumlah frame.
  - Hanya terjadi pada beberapa Algoritma Page Replacement.

FIRST-IN-FIRST-OUT (FIFO) ALGORITHM

Page yang dialokasikan lebih awal akan di-replace lebih awal

Contoh :

- 3 Frame
- Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1



- Terjadi 15 page fault : 7,0,1,2,3,0,4,2,3,0,1,2,7,0,1

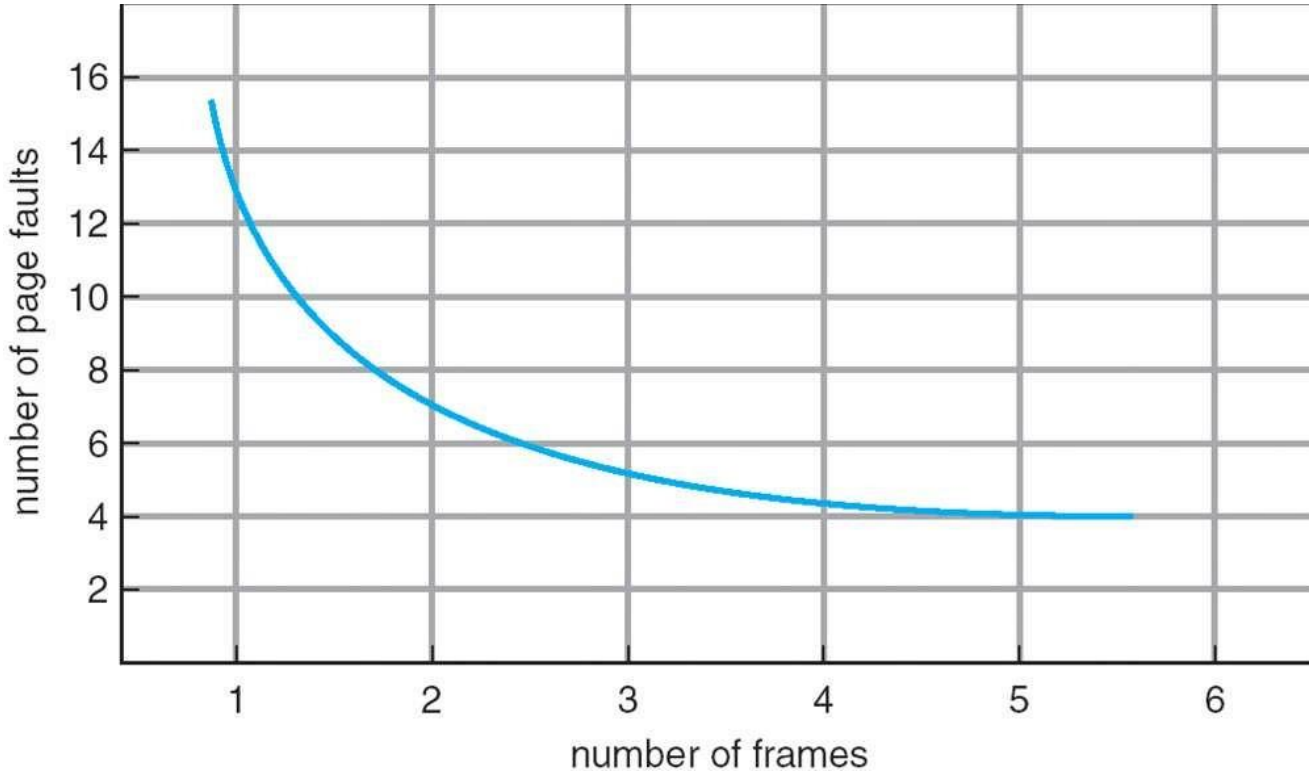
Bagaimana jika jumlah frame ditambah?

- Bisa terjadi lebih banyak page fault (Belady's anomaly)

Bagaimana implementasinya?

- Perlakukan physical memory sebagai FIFO queue

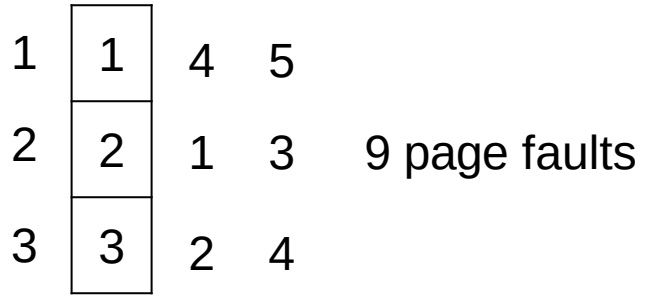
GRAPH OF PAGE FAULTS VERSUS THE NUMBER OF FRAMES



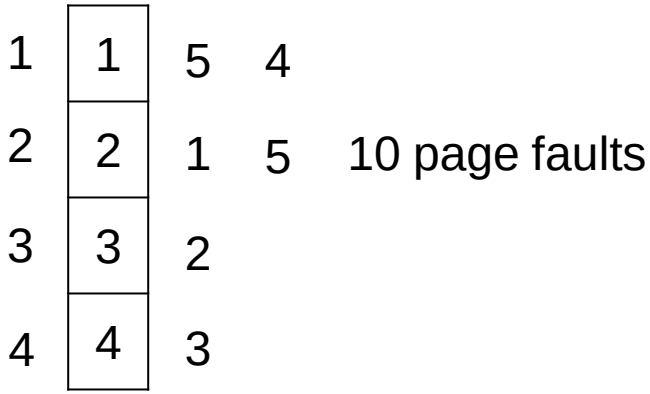
Anomaly Belady: kecepatan page fault akan bertambah jika framenya bertambah

FIRST-IN-FIRST-OUT (FIFO) ALGORITHM

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)



- 4 frames



OPTIMAL ALGORITHM

Replace page yang tidak akan digunakan untuk periode waktu yang lama

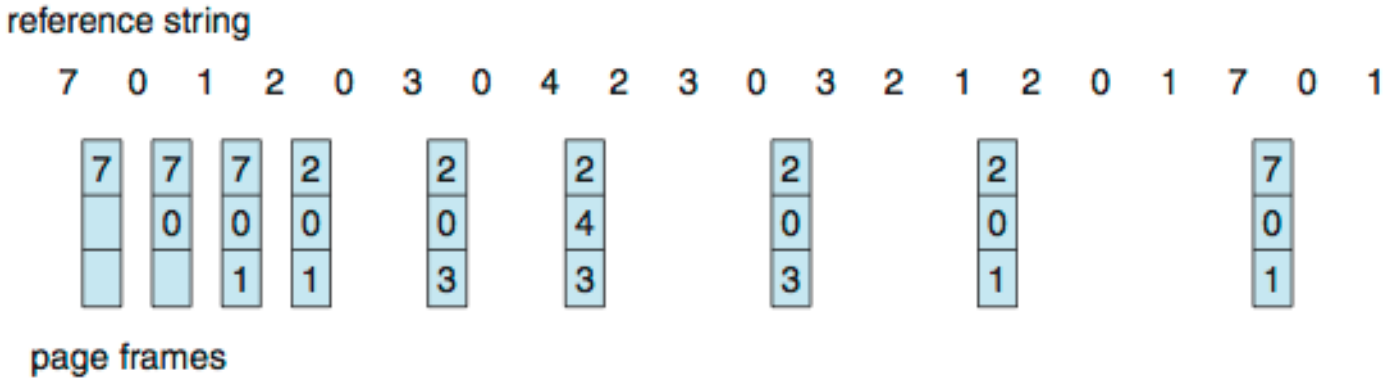
Ideal, tapi tidak realistis diimplementasikan

- Bagaimana kita bisa tahu page yang tidak akan digunakan untuk periode waktu yang lama?

Hanya digunakan untuk mengukur performa algoritma lain

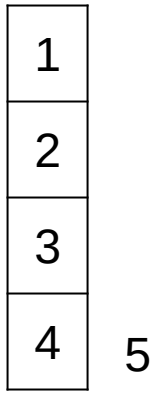
- Bandingkan kondisi ideal dengan kondisi realistis pada algoritma lain

Contoh :



- 4 frames example

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



4

5

6 page faults

- How do you know this?
- Used for measuring how well your algorithm performs

ALGORITMA LEAST RECENTLY USED (LRU)

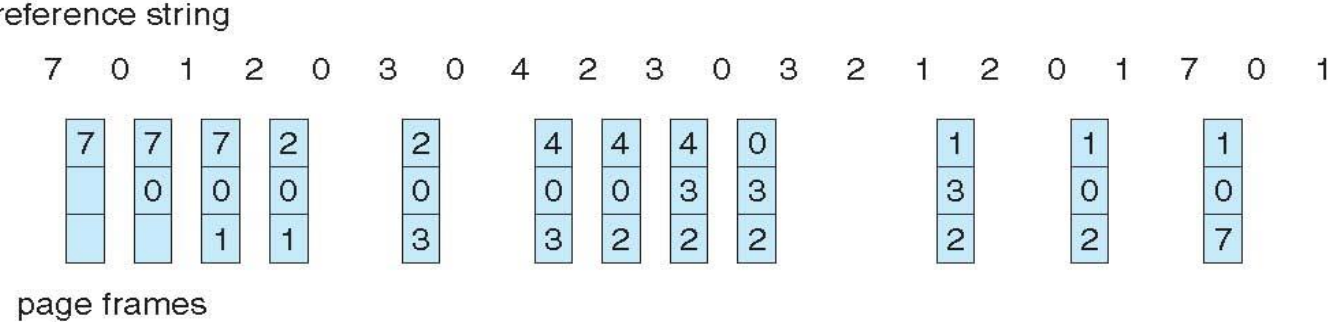
- Page yang diganti adalah page yang **bukan baru saja digunakan** (paling lama tidak digunakan)
- Merupakan perpaduan antara Algoritma FIFO dan Algoritma Optimal.
- Tidak akan mengalami Anomali Belady.
- Dapat diimplementasikan dengan 2 cara, yaitu :
  - Counter
    - Menggunakan clock yang nilainya akan ditambah 1 tiap kali melakukan reference ke suatu page.
    - Harus melakukan pencarian.
  - Stack
    - Tiap mereference ke suatu page, page tersebut dipindah dan diletakkan pada bagian paling atas stack.
    - Page yang diganti adalah page yang berada di stack paling bawah.
    - Tidak perlu melakukan pencarian.
    - Lebih mahal.

LEAST RECENTLY USED

Mengganti page yang sudah tidak digunakan untuk periode waktu yang terlama.

- Realistis karena m

Contoh :

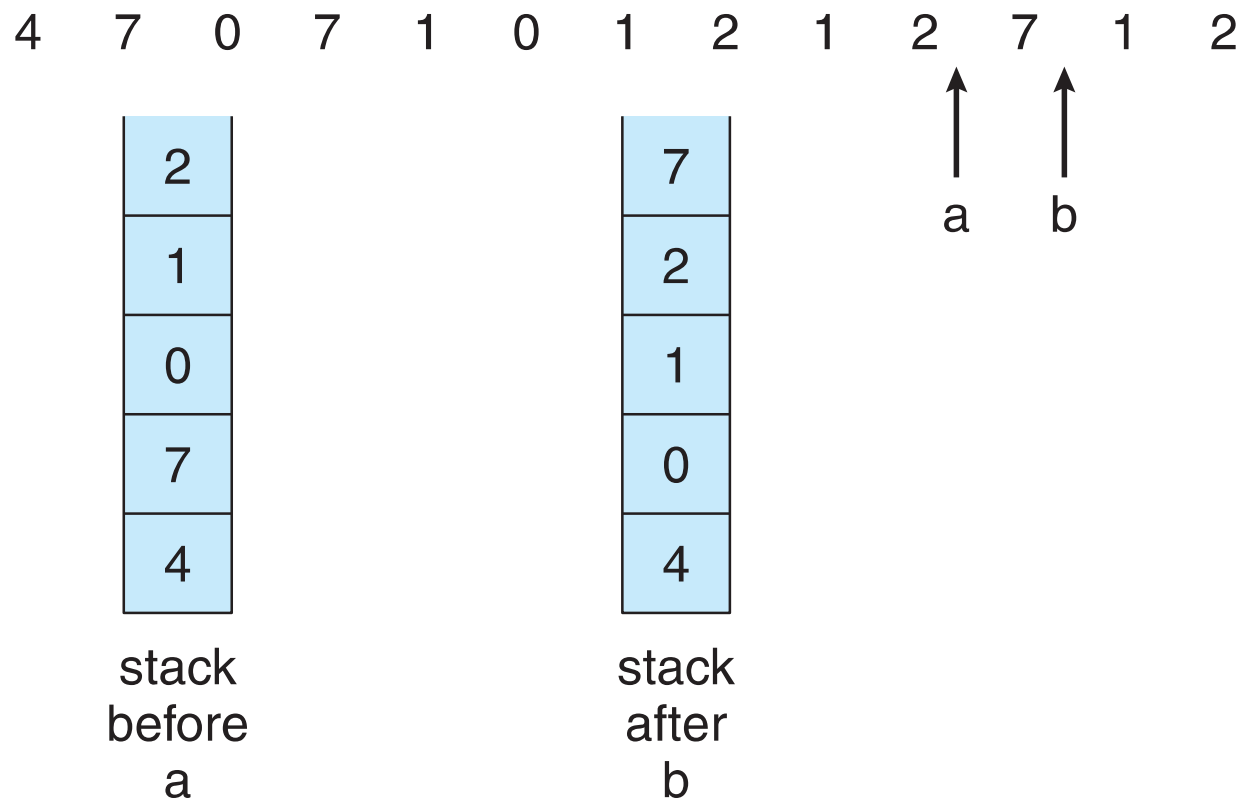


- Terdapat 12 page fault
- Lebih baik dari FIFO, tapi belum Optimal

Terdapat dua cara implementasi LRU

- LRU dengan counter
- LRU dengan stack

CONTOH LRU DENGAN STACK



ALOKASI FRAME

- Alokasi frame berhubungan dengan mekanisme alokasi sejumlah memori bebas untuk proses-proses.
- Fixed Allocation
  - Proses dengan prioritas tinggi ataupun rendah diperlakukan sama.
    - Equal Allocation: semua sama rata
    - Proportional Allocation: sesuai kebutuhan
- Alokasi prioritas
  - Perbandingan frame-nya tidak tergantung pada ukuran relatif dari proses tetapi tergantung pada prioritas proses.

JENIS PAGE REPLACEMENT

- **Global replacement** mengijinkan suatu proses untuk menyeleksi suatu frame yang akan dipindah dari sejumlah frame, meskipun frame tersebut sedang dialokasikan ke proses yang lain.
- Pada **local replacement**, jumlah frame yang dialokasikan untuk proses tidak berubah.
  - Setiap proses dapat memilih dari frame-frame yang dialokasikan untuknya.

THRASHING

- Kegiatan paging yg sangat tinggi
- **Thrashing** ≡ a process is busy swapping pages in and out
- If a process does not have “enough” pages, the page-fault rate is **very high**.
- This leads to:
  - low CPU utilization
  - operating system thinks that it needs to increase the degree of multiprogramming
  - another process added to the system
- Proses menghabiskan waktu lebih banyak untuk paging daripada eksekusi.

TUGAS

Diketahui string acuan dari page :

1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

Berapa banyak page fault yang terjadi untuk algoritma page replacement berikut dengan satu, dua, tiga, empat, lima, enam atau tujuh frame ? Ingat bahwa semua frame diinisialisasi kosong, sehingga setiap page unik pertama akan bernilai masing-masing satu fault

a. LRU
b. FIFO
c. Optimal