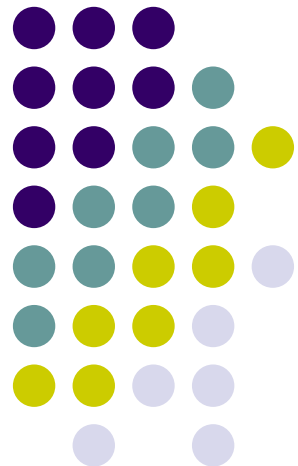
9

Memori



Memory Management

- Latar Belakang
- Swapping
- Contiguous Allocation
- Paging
- Segmentation
- Segmentation dengan Paging

Latar Belakang

- Untuk dieksekusi program harus berada dalam memori
  - Eksekusi: proses
  - Alokasi resources memori: ruang (tempat storage) untuk menyimpan data, instruksi, stack dll.
- Problem: Memori secara fisik (besarnya storage) sangat terbatas ukurannya,
  - Manajemen storage: alokasi dan dealokasi untuk proses-proses
  - Utilisasi: optimal dan efisien

alamat Binding

- Sebalum eksekusi program berada di dalam disk, dan saat dieksekusi ia perlu berada pada suatu lokasi dalam memori fisik
- alamat binding adalah menempatkan alamat relatif ke dalam adress fisik memori yang dapat berlangsung dalam di salah satu tahapan: kompilasi, load, atau eksekusi

Tahapan Running Program

- Tahapan kompilasi: source program (source code) dikompilasi menjadi object module (object code)
- Tahapan link & load: object module di-link dengan object module lain menjadi load module (execution code) kemudian di-load ke memori untuk dieskekusi
- Tahapan eksekusi: mungkin juga dilakukan dynamic linking dengan resident library

Alamat Binding: Saat Kompilasi

- Jika lokasi dari proses sudah diketahui sebelumnya maka saat kompilasi alamat-alamat instruksi dan data ditentukan dengan alamat fisik
- jika terjadi perubahan pada lokasi tersebut maka harus direkompilasi

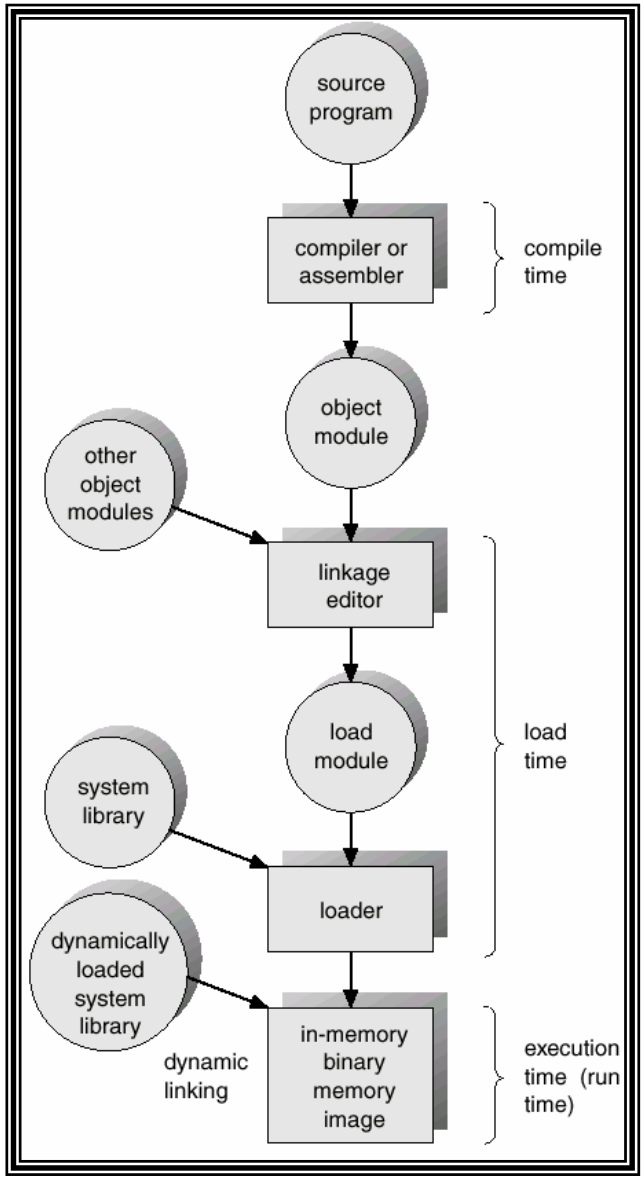
Alamat Binding: Saat Load

- Code hasil kompilasi masih menunjuk alamat-alamat secara relatif, saat di-load alamat-alamat disubstitusi dengan alamat fisik berdasar relokasi proses yang diterima
- Jika terjadi perubahan relokasi maka code di-load ulang

Alamat Binding: Saat Eksekusi

- Binding bisa dilakukan ulang selama proses
  - hal ini untuk memungkinkan pemindahan proses dari satu lokasi ke lokasi lain selama run
- Perlu adanya dukungan hardware untuk pemetaan adress
  - contoh: base register dan limit register

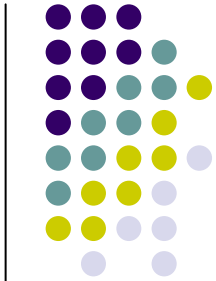
Tahapan Pemrosesan User rogram



Ruang Alamat Logik vs. Fisik

- Konsep ruang alamat logik terhadap ruang alamat fisik adalah hal pokok dalam manajemen memori
- alamat logik: alamat yang di-generate oleh CPU (disebut juga virtual alamat)
  - Berdasarkan eksekusi program
  - Note: Besarnya alamat program dapat lebih besar dari kapasitas memori fisik.
- alamat fisik: alamat yang dikenal oleh unit memory
  - alamat sebenarnya yang digunakan untuk mengakses memori.
- Perlu ada penerjemahan (translasi) dari alamat logik ke alamat fisik.

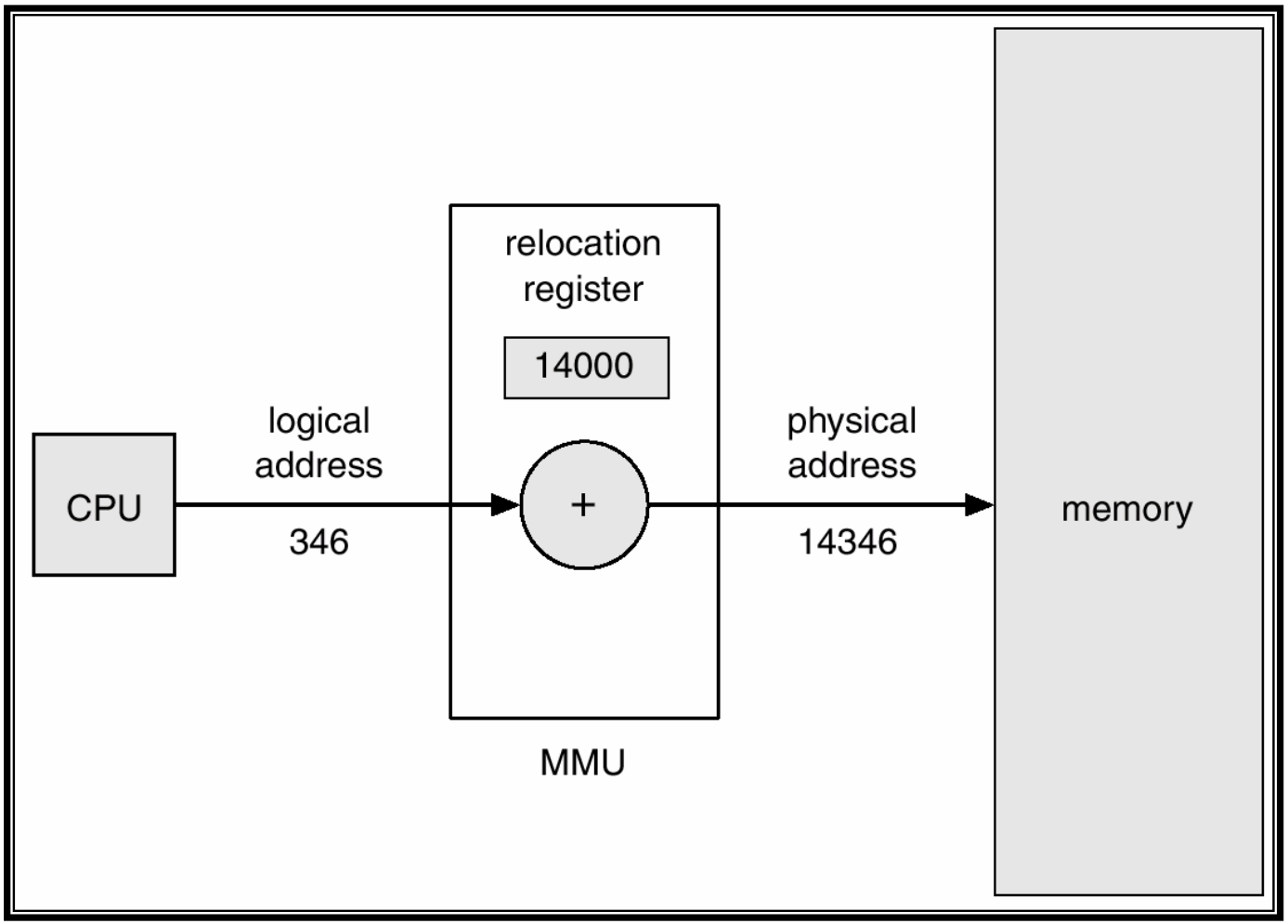
Memory-Management Unit (MMU)



- Perangkat Hardware yang memetakan alamat logik (virtual) ke alamat fisik.

- Dalam skema MMU
  - Menyediakan perangkat register yang dapat di set oleh setiap CPU: setiap proses mempunyai data set register tsb (disimpan di PCB).
    - Base register dan limit register.
  - Harga dalam register base/relokasi ditambahkan ke setiap address proses user pada saat run di memori
  - Program user hanya berurusan dengan addressaddress logik saja

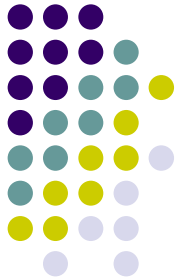
Relokasi Dinamik menggunakan Register Relokasi



Dynamic Loading

- Rutin tidak akan di load jika tidak dipanggil (execute).
- Pro's: utilisasi memory-space, rutin yang tidak dieksekusi tidak akan dipanggil (program behaviour: 70-80% dari code).
  - Handling execption, error, atau pilihan yang jarang digunakan.
- Tidak perlu dukungan khusus dari OS:
  - **Overlay**: memori terbatas dan program lebih besar dari memori.
  - Disusun berdasarkan hirarkis dalam bentuk tree: root – branch dan leaves (misalkan root harus ada di memory, sedangkan yang lain dapat di load bergantian).
  - Tidak dilakukan otomatis tapi harus dirancang oleh programmer (user).

Dynamic Linking



- Linking ditunda sampai saat eksekusi
  - code menjadi berukuran kecil.
- Program-program user tidak perlu menduplikasi system library
  - system library dipakai bersama
  - Mengurangi pemakaian space: satu rutin library di memory digunakan secara bersama oleh sekumpulan proses.
  - Contoh: DLL (dynamic linking library) Win32
- Mekanisme menggunakan skema Stub
  - stub: suatu potongan kecil code menggantikan referensi rutin (dan cara meload rutin tsb)

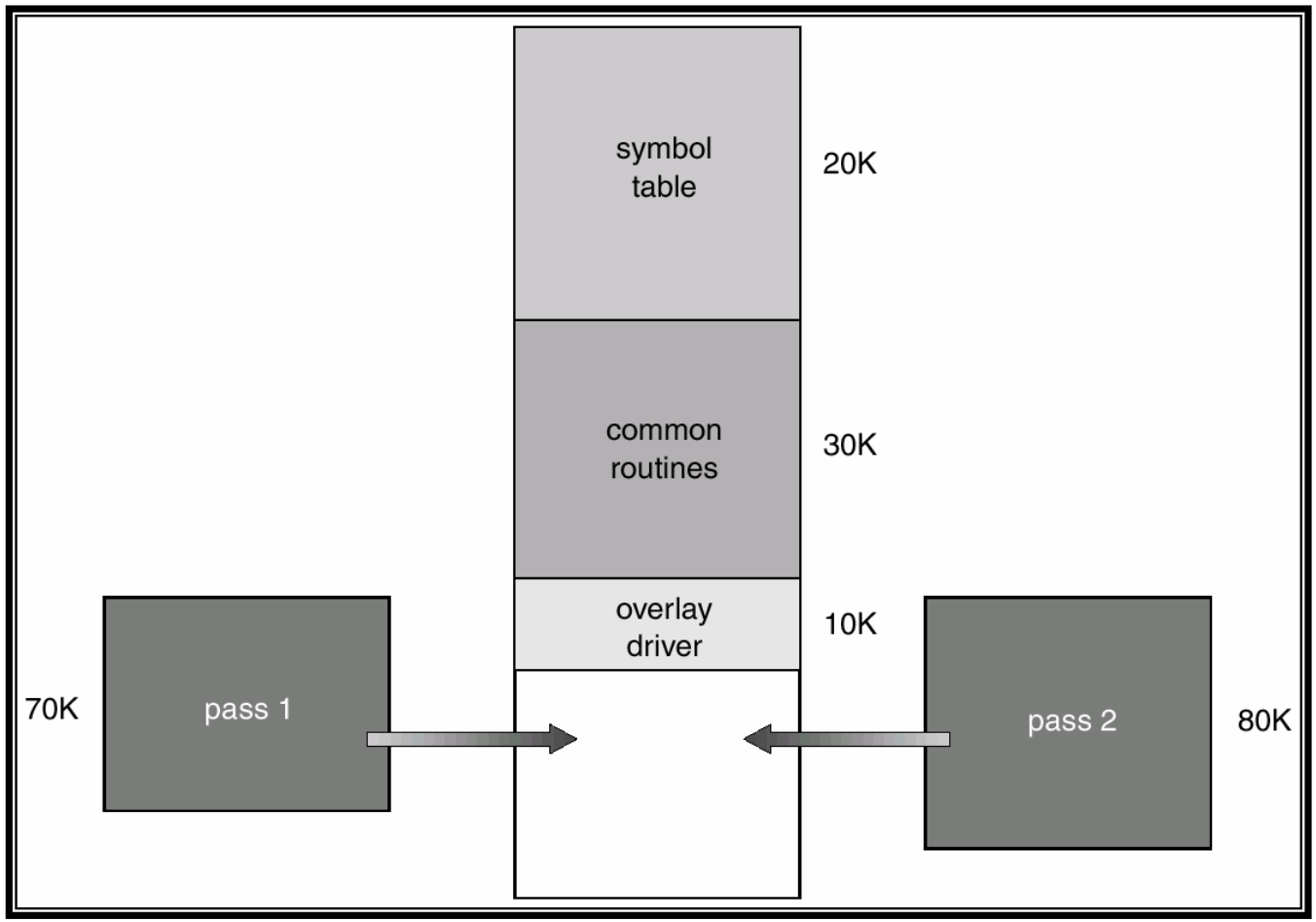
Overlay

- Overlay membagi program yang besar menjadi bagian-bagian yang lebih kecil dan daat dimuat dalam memori utama.

- Dibutuhkan ketika proses yang ada lebih besar dibandingkan memori yang tersedia

- Diimplementasikan oleh user, tidak ada dukungan khusus dari sistem operasi, disain program pada struktrur overlay cukup kompleks.

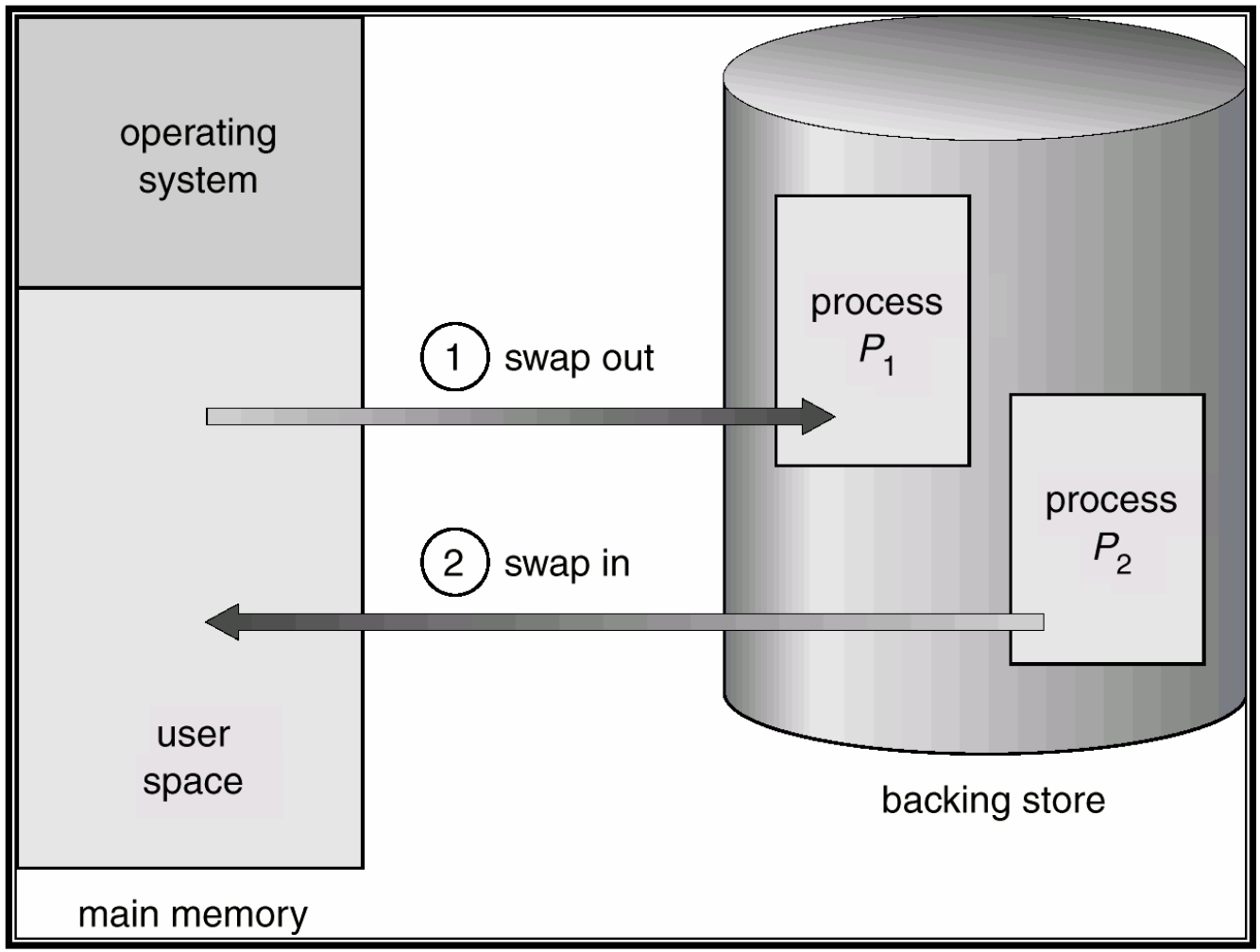
Overlay pada Two-Pass Assembler



Swapping

- Suatu proses dapat di-swap secara temporary keluar dari memori dan dimasukkan ke *backing* store, dan dapat dimasukkan kembali ke dalam memori pada eksekusi selanjutnya.
- *Backing store* –disk cepat yang cukup besar untuk mengakomodasi copy semua memori image pada semua user; menyediakan akses langsung ke memori image.
- *Roll out, roll in* – varian swapping yang digunakan dalam penjadualan prioritas; proses dengan prioritas rendah di-swap out, sehingga proses dengan prioritas tinggi dapat di-load dan dieksekusi.
- Bagian terbesar dari swap time adalah transfer time, total transfer time secara proporsional dihitung dari jumlah memori yang di swap.
- Modifikasi swapping dapat ditemukan pada sistem UNIX, Linux dan Windows.

Skema Swapping

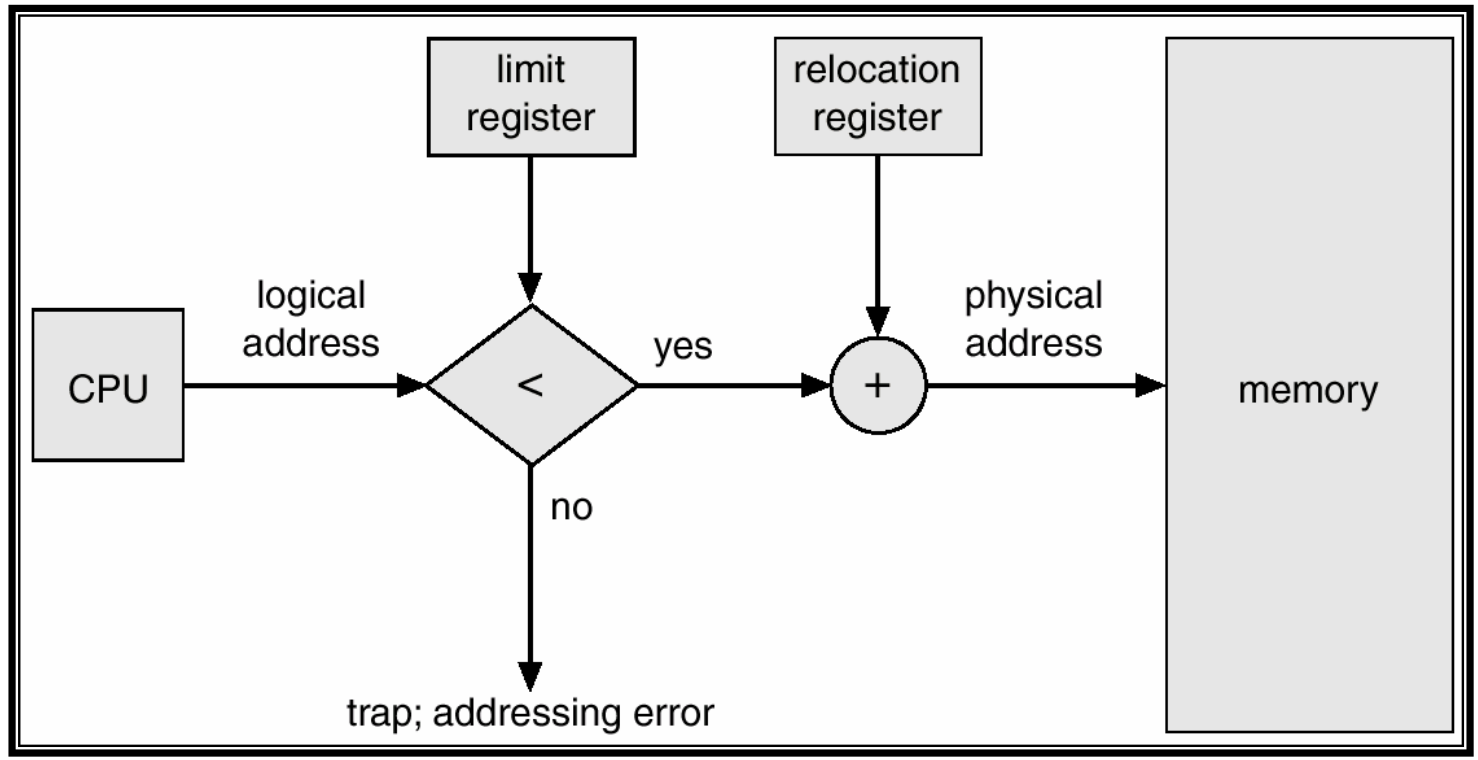


Contiguous Allocation

- Memori utama biasanya terbagi dalam dua bagian:
  - Resident operating system, biasanya tersimpan di alamat memori rendah termasuk interrupt vector .
  - User proces menggunakan memori beralamat tinggi/besar.

- Single-partition allocation
  - Relokasi register digunakan untuk memproteksi masing-masing user proses dan perubahan kode sistem operasi dan data.
  - Relokasi register terdiri dari alamat fisik bernilai rendah; limit register terdiri dari rentang/range alamat logik, setiap alamat logik harus lebih kecil dari limit register.

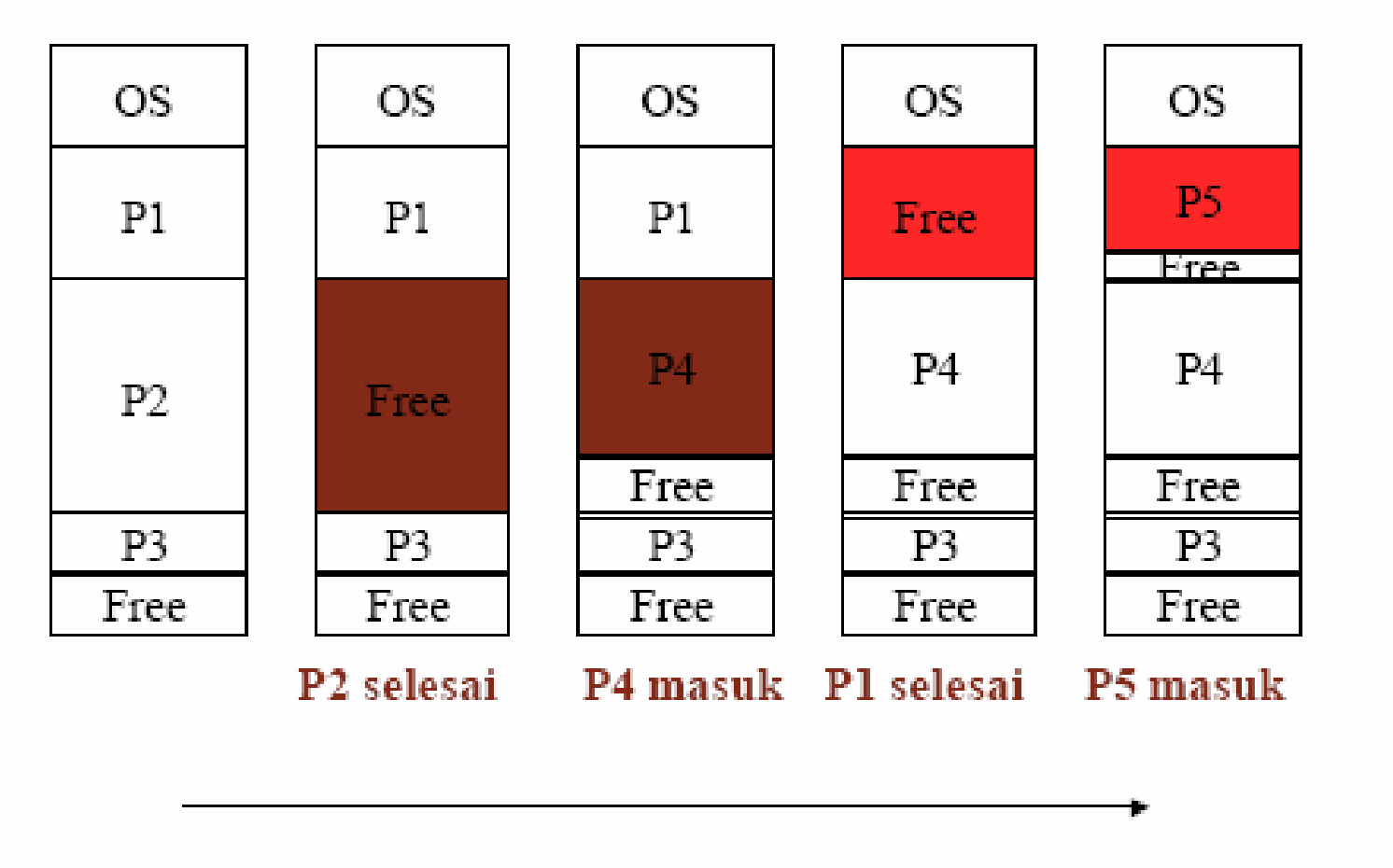
Dukungan Hardware untuk Relokasi dan Limit Register



Multiple-Partition Allocation

- Partisi Fixed-Sized (MFT)
  - Memori dibagi menjadi beberapa blok dengan ukuran tertentu yang seragam
  - Jumlah proses yang bisa running max hanya sejumlah blok yang disediakan(misal IBM OS/360)
- Partisi Variabel-Size (MVT)
  - Pembagian memori sesuai dengan request dari proses-proses yang ada.
    - Lebih rumit karena ukuran alokasi (partisi) memori dapat bervariasi
  - Peranan memori manajemen semakin penting: list dari partisi yang digunakan, free dll.

Contoh: Multiple Allocation



Masalah pada Dynamic Storage-Allocation

Bagaimana agar proses berukuran n dapat menempati hole yang bebas

- **First-fit**: Mengalokasikan proses pada hole pertama yang ditemui yang besarnya mencukupi
- **Best-fit**: Mengalokasikan proses pada hole dengan besar paling cocok (fragmentasinya kecil).
- **Worst-fit**: Mengalokasikan proses pada hole dengan fragmentasi terbesar.

First-fit dan best-fit lebih baik dibandingkan worst-fit dalam hal kecepatan dan pemanfaatan storage.

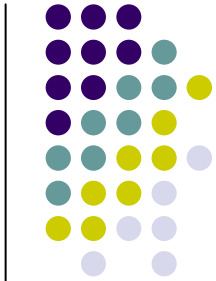
Fragmentasi (issue)

- **External** (masalah variable sized partition):
  - Ruang memori free, namun tidak contiguous.
  - Hole-hole ada di antara proses-proses berturutan.
  - Tidak dapat digunakan karena proses terlalu besar untuk menggunakannya.
- Internal (masalah fixed size):
  - Sifat program dinamis (alokasi dan dealokasi).
  - Memori yang teralokasi mungkin lebih besar dari memori yang diminta (wasted).

Paging

- Membagi memori fisik ke dalam blok (**page**, frame) dengan ukuran tertentu (fixed) yang seragam.
  - Memudahkan manajemen free memory (hole) yang dapat bervariasi.
  - Tidak perlu menggabungkan hole menjadi blok yang besar seperti pada variable partition (compaction).
  - OS lebih sederhana dalam mengontrol (proteksi dan kebijakan) pemakaian memori untuk satu proses.
- Standard ukuran blok memori fisik yang dialokasikan (de-alokasi) untuk setiap proses.
  - Ukuranya (tergantung OS): 512 byte s/d 16 KB.

Page Allocation



- Alokasi:
  - Terdapat “free list” yang menyimpan informasi “frame” di memori fisik yang tidak digunakan
  - Tergantung besarnya proses => memerlukan n pages
  - Alokasi frame diberikan sesuai dengan permintaan (demand, expand).
- Implikasi:
  - User’s (program) view (logical address): memori dialokasikan secara sinambung (contiguous)
  - Fakta (physical address): memori fisik tersebar (noncontiguous) sesuai dengan frame yang dialokasikan.

Skema Paging

- Bagaimana menjembatani antara “user’s view” dan alokasi memori sebenarnya?
  - Penerjemahan (translasi) alamat logical ke alamat fisik => tugas dari OS (user/program “transparant”).
  - Perlu dukungan hardware (CPU) => address translation.

- Setiap proses mempunyai informasi “pages” yang dialokasikan oleh OS
  - Mapping setiap alamat logical ke alamat fisik
  - **Issue:** mekanisme mudah, cepat dan efisien.
  - **Page table**: berisi “base address” (alamat fisik) dari frame yang telah dialokasikan ke proses tsb.

Page table

- Setiap OS mempunyai cara menyimpan page table untuks setiap proses
- Page table bagian dari setiap proses.
  - Page table berada di memori, saat proses tersebut dieksekusi.
  - Informasi page table disimpan oleh PCB: pointer ke page table dari proses tersebut.
  - Setiap kali terjadi context switch => informasi page table untuk proses yang baru harus di restore (misalkan referensi/pointer lokasi page table tsb. di memori).

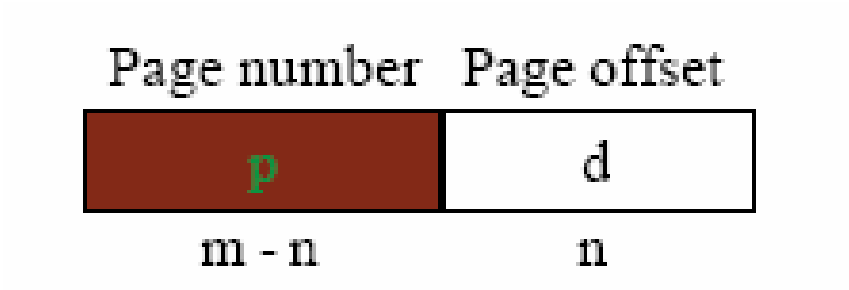
Page table (h/w support)

- Menggunakan “fast register”
  - Contoh: DEC PDP11 : 16 bit address (logical 2**16**): 64K, page size 8K (2**13**).
  - Memerlukan page table dengan: 8 entry (dapat diterapkan pada hardware register, hanya 3 bit)

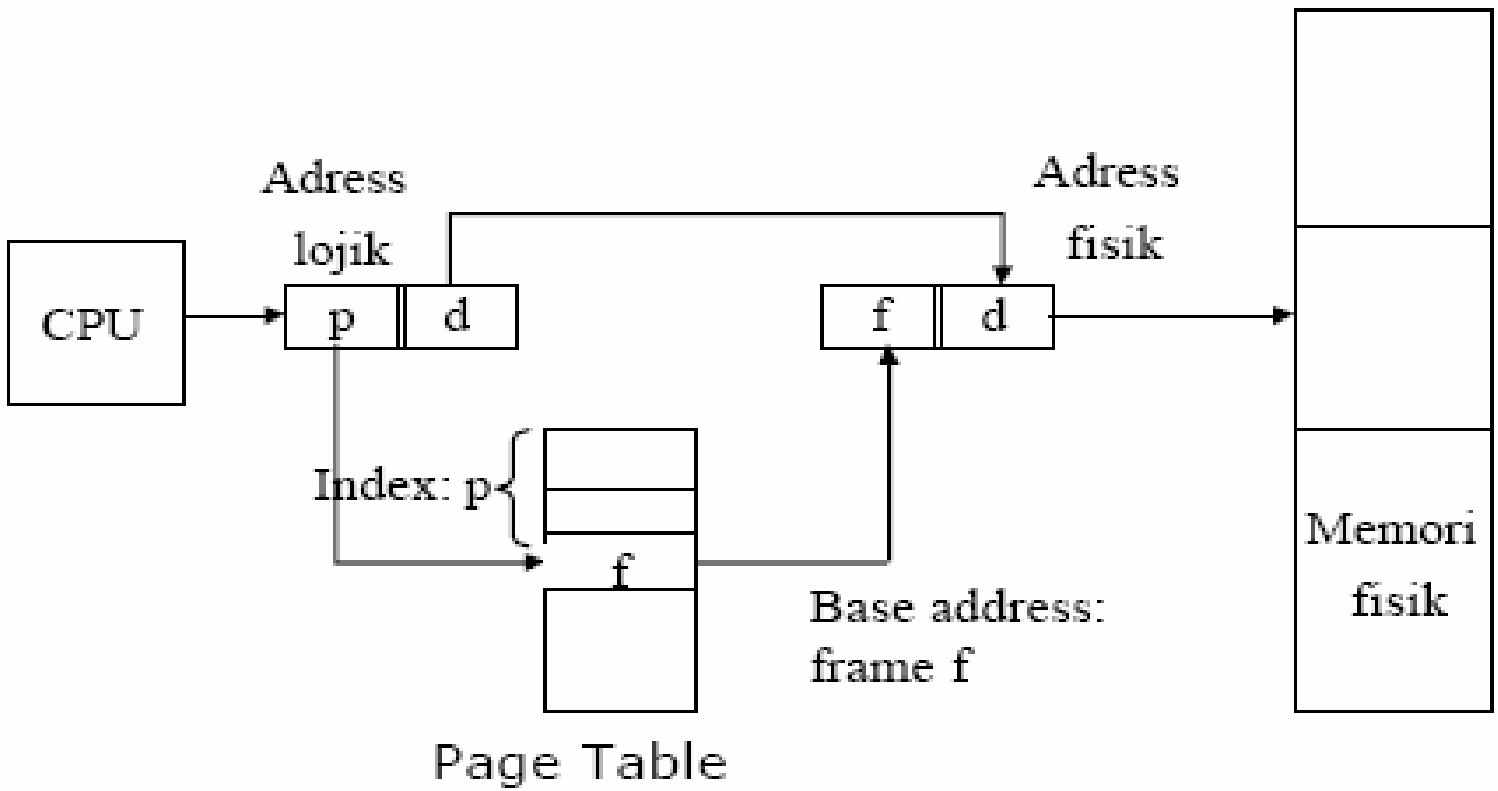
- Untuk komputer modern sulit menggunakan fast register
  - Pentium : 32 bit address logical (total: 4 GB), page size (8K), maka mempunyai potensi entry: 524.288 entry.
  - Page table disimpan pada memori (bagian program) dengan menggunakan page table base register
  - **Page-table base register (PTBR)** : pointer ke page-table di memori.
  - **Page-table length register (PTLR) : besarnya** ukuran page table (karena tidak semua proses memerlukan ukuran page tabel max.)

Paging: translation

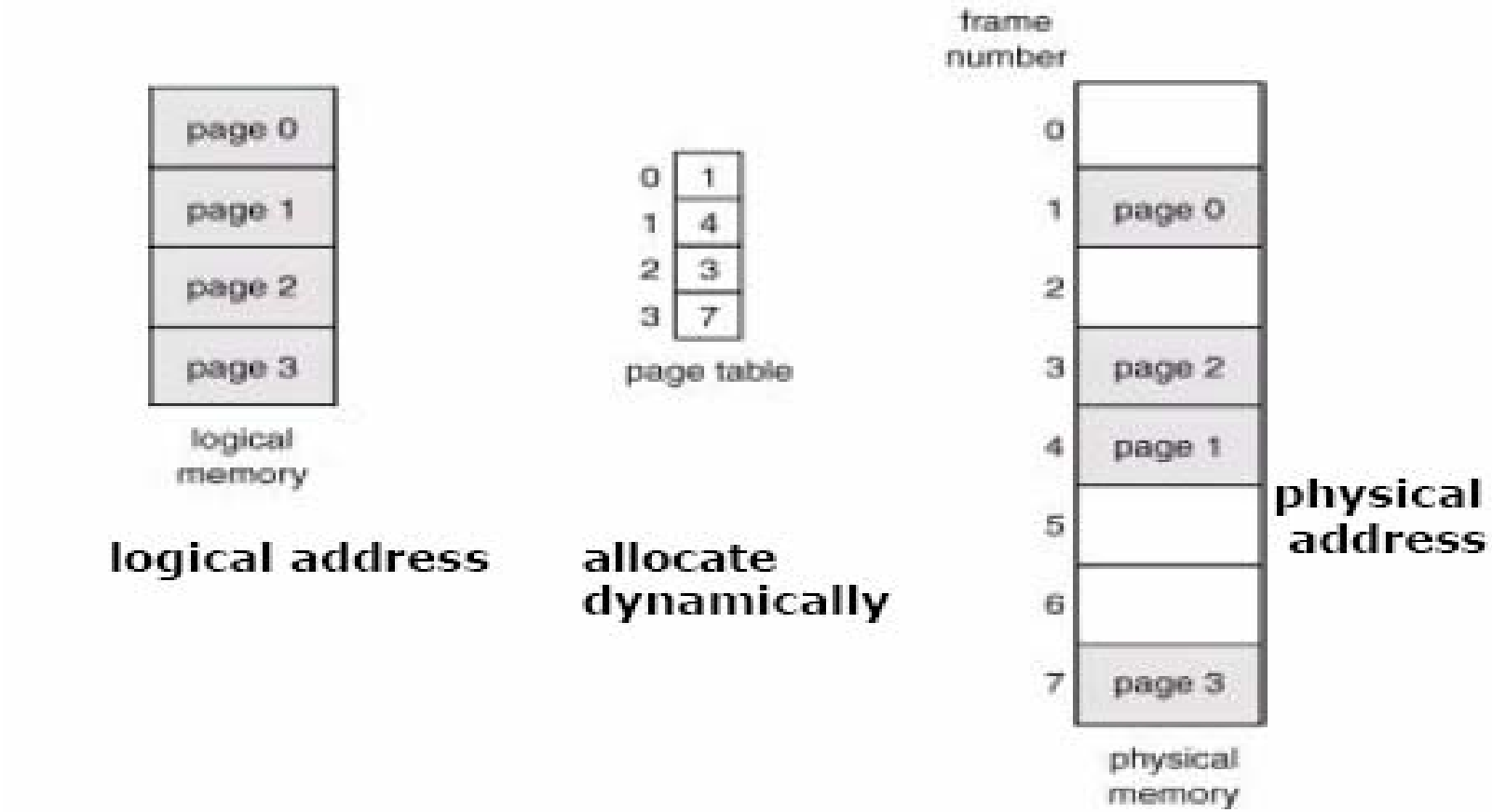
- Address logik dari CPU dianggap terdiri atas dua bagian:
  - Page number (p): merupakan indeks dalam tabel yang berisi **base address** dari tiap page dalam memori fisik
  - Page offset (d): menunjukkan lokasi address memori berdasarkan “base address” pada page tersebut.



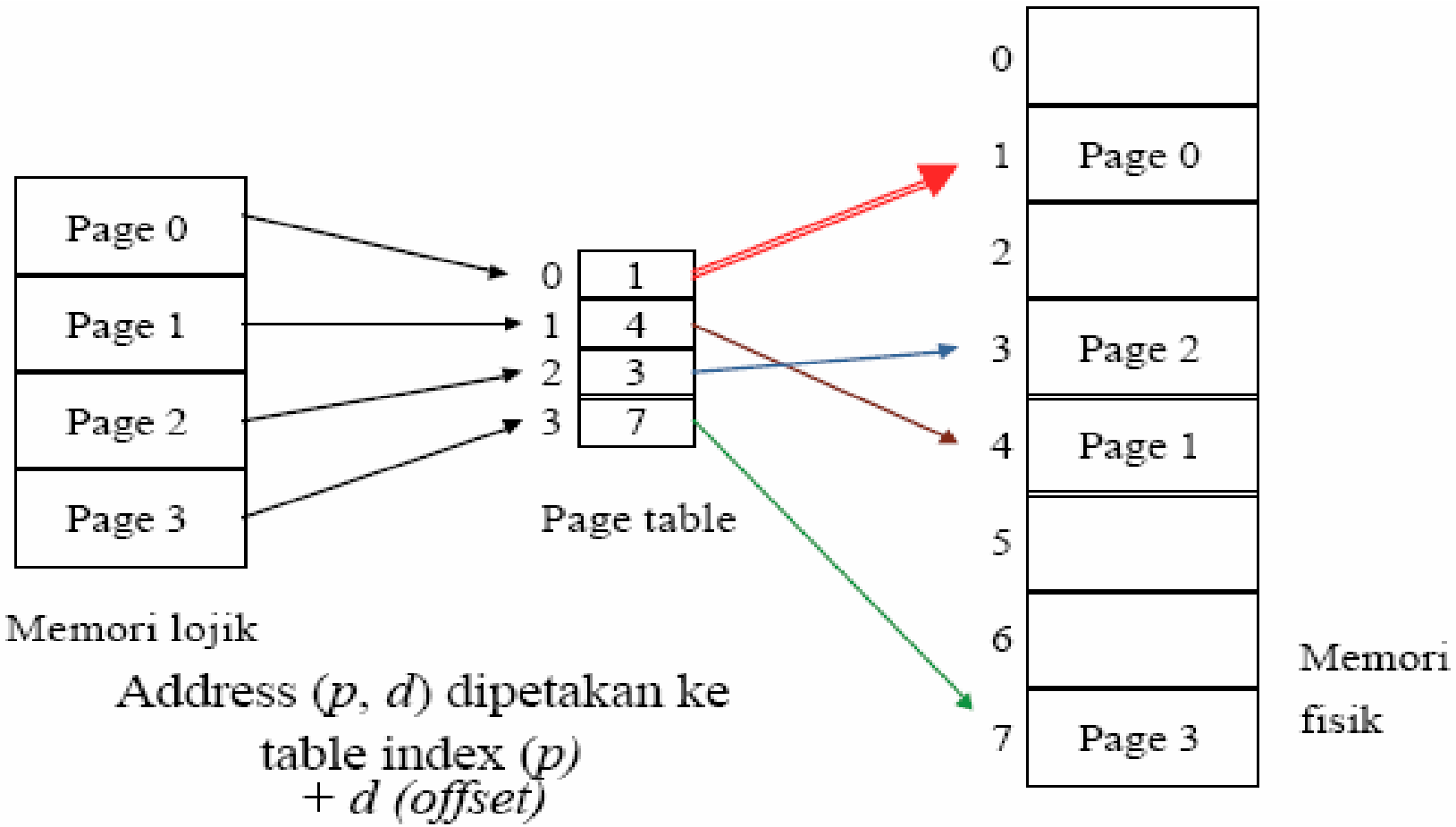
Address: hardware support



Contoh Paging



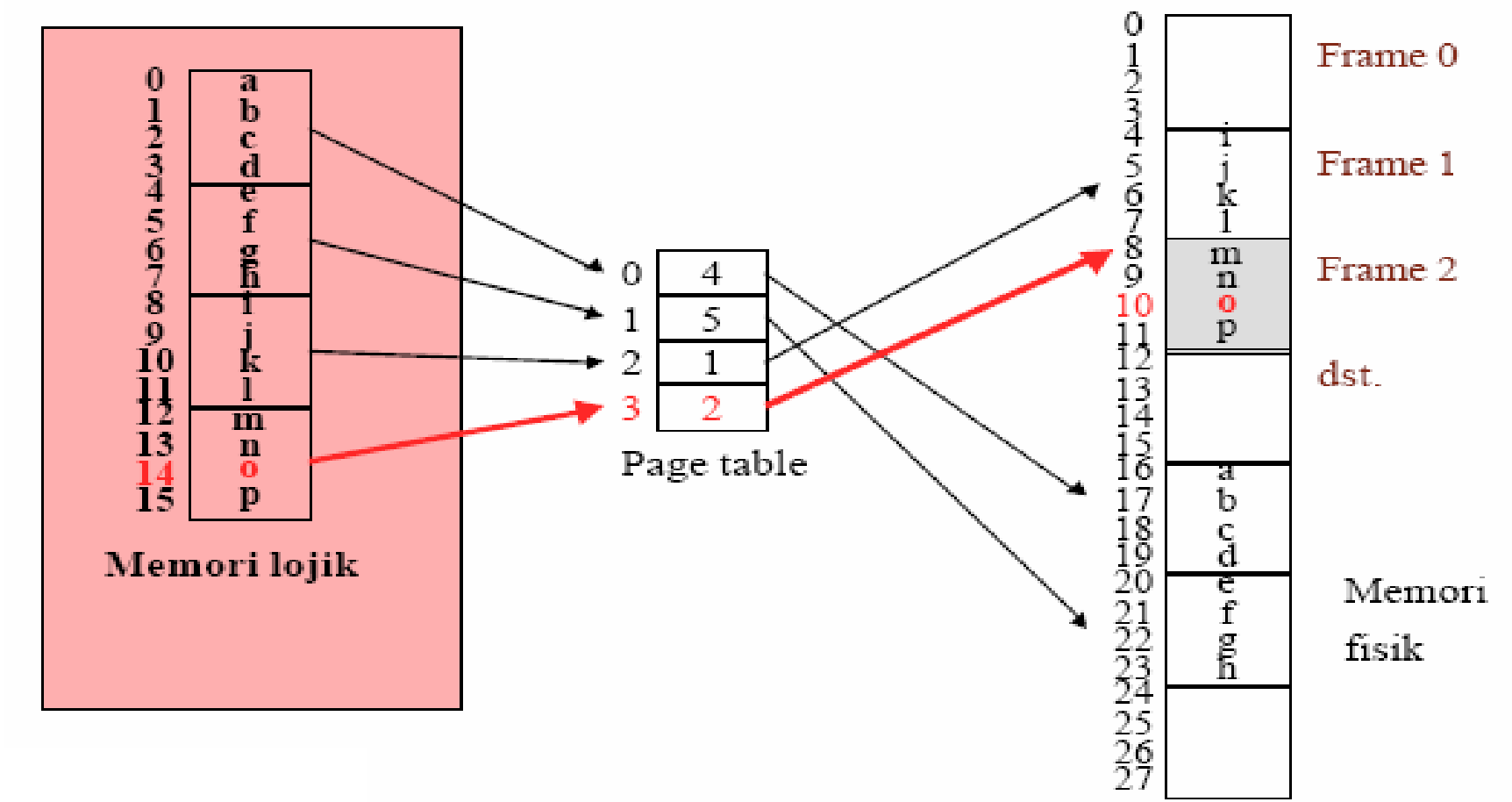
Model Paging



Contoh :

- Misalkan LA: 4 bits (max. logical addres: 16 lokasi)
  - Page size => 4 bytes (ditentukan oleh designerOS).
    - 2 bits: menunjuk ke alamat dari masing-masing byte dalam setiap page tersebut.
  - Page table: tersisa 2 bits
    - Max. 4 entry
    - Jadi setiap proses max. akan menggunakan 4 pages => mencakup seluruh alamat logical.

Contoh (2)


0 a
1 b
2 c
3 d
4 e
5 f
6 g
7 h
8 i
9 j
10 k
11 l
12 m
13 n
14 o
15 p
Memori lojik
0 4
1 5
2 1
3 2
Page table
0
1
2
3
4 i
5 j
6 k
7 l
8 m
9 n
10 o
11 p
12
13
14
15
16 a
17 b
18 c
19 d
20 e
21 f
22 g
23 h
24
25
26
27
Frame 0
Frame 1
Frame 2
dst.
Memori
fisik

Contoh (3) :

- Logical address: 11 10 (program view: 14 desimal => “o”)
- Page translation (physical memory allocation):
  - Bagian: p (index page) => base address dari frame.

    Binary 11 => 3 (index = 3 dari page table)

    => berisi base address untuk frame 2 di memori.
  - Bagian offset: d (displacement)

    Binary 10 => 2
  - Alamat fisik:

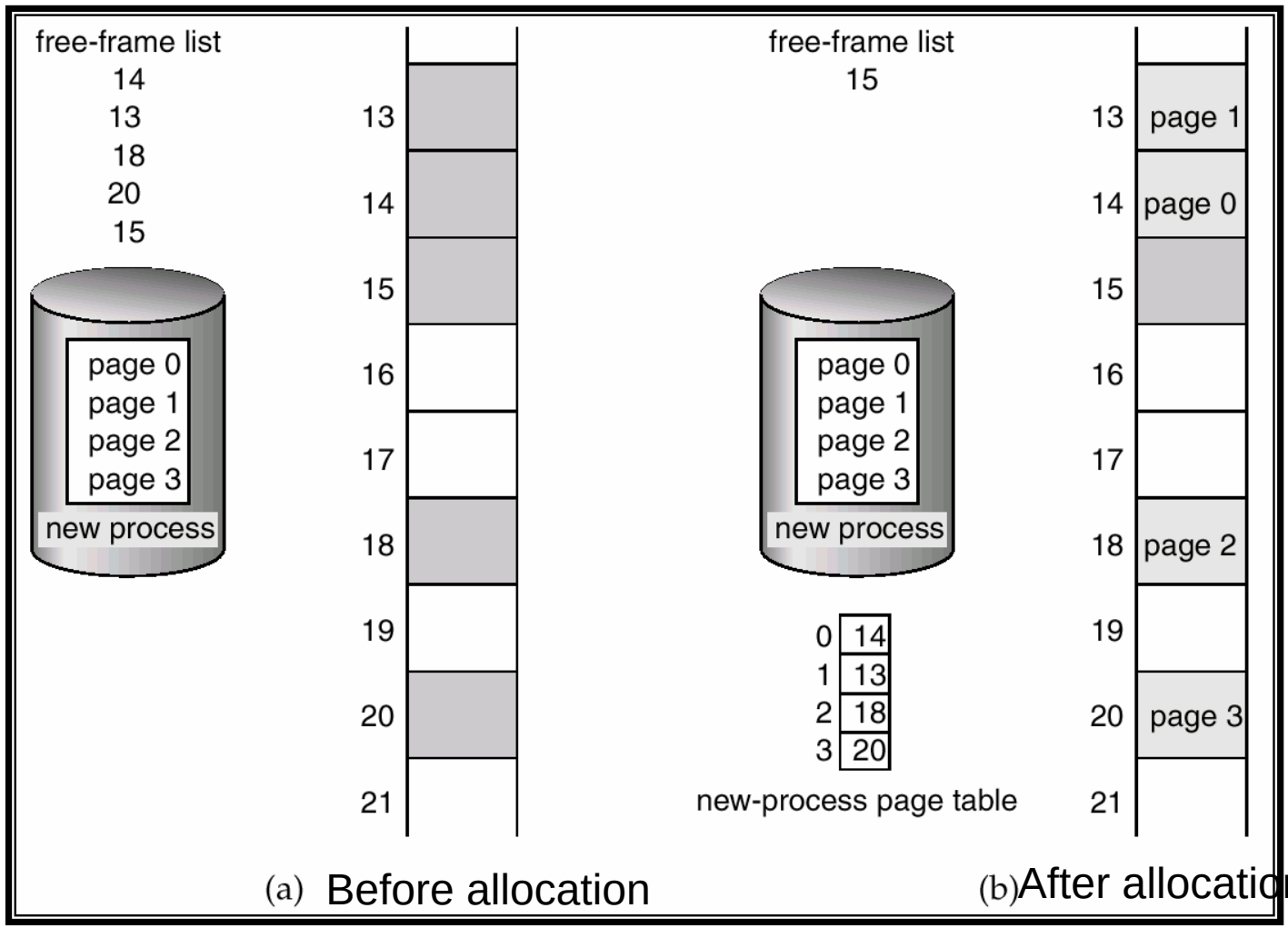
    base address frame 2 : 2 * 4 => 8;

    => 8 + 2 = 10 (berisi “o”).

Frame table

- OS harus mempunyai informasi “frame” dari memori fisik:
  - Berapa banyak frame yang bebas?
  - Mana saja frame yang bebas (identifikasi) => frame table (list)
  - Informasi hubungan antara satu frame dengan page mana dari proses yang aktif
  - List ini akan terus di-update, misalkan jika proses terminate maka semua frame yang dialokasikan akan di kembalikan ke free list.

Frame Bebas


free-frame list
14
13
18
20
15
page 0
page 1
page 2
page 3
new process
13
14
15
16
17
18
19
20
21
(a) Before allocation
free-frame list
15
page 0
page 1
page 2
page 3
new process
0 14
1 13
2 18
3 20
new-process page table
13 page 1
14 page 0
15
16
17
18 page 2
19
20 page 3
21
(b) After allocation

Page size

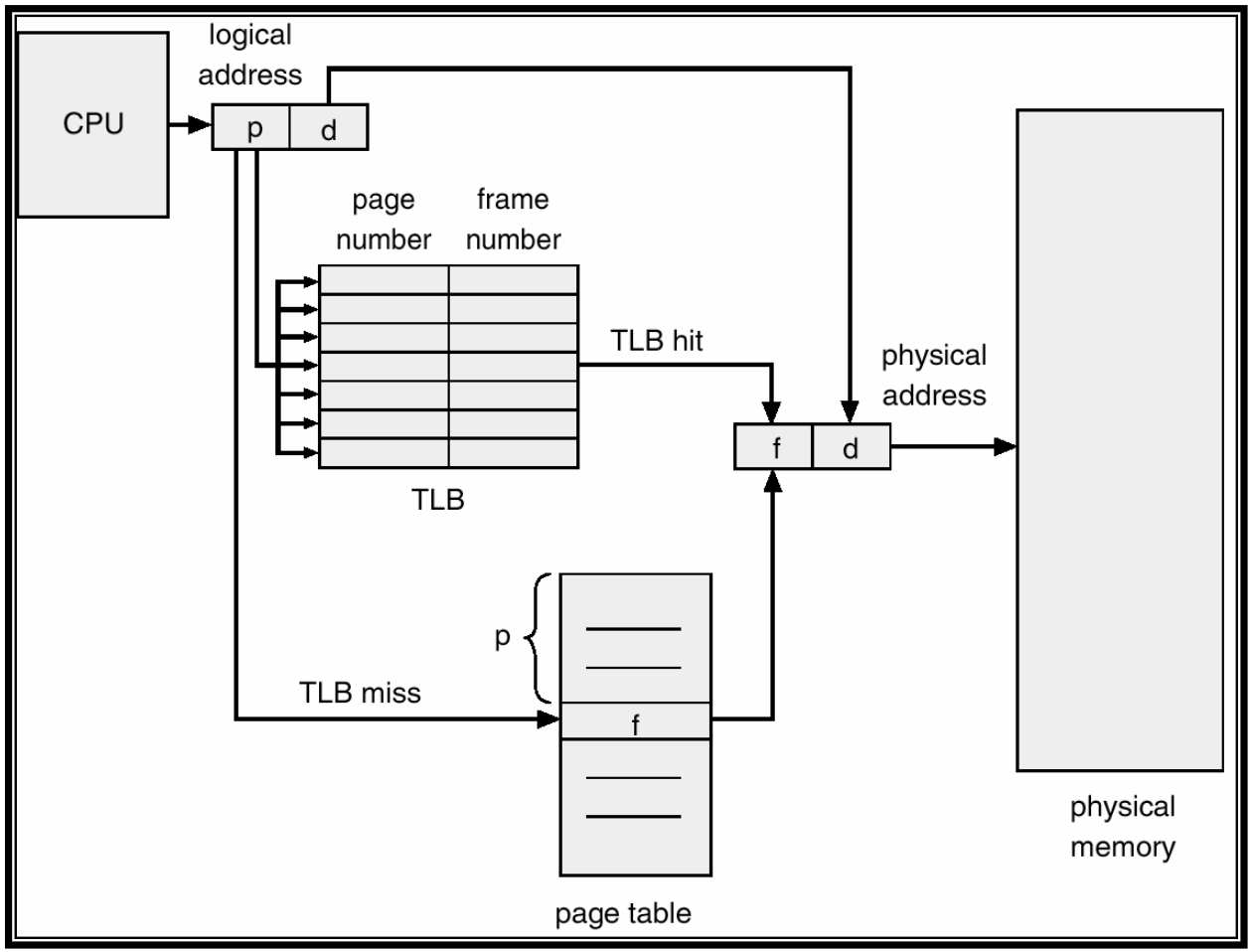
- **Fragmentasi internal pada page terakhir**
  - Tidak ada fragmentasi eksternal
  - Fragmentasi internal bisa terjadi
  - Worst-case:
    - Untuk proses yang memerlukan n page + 1 byte
    - bila ukuran page = 4096 byte maka akan terbuang 4095 byte / process

  - **Besarnya ukuran pages**
    - Independent dari program/proses (system wide)
    - Intuitif: small pages preferable
      - Apakah keuntungan ukuran pages kecil?
    - Page table entry dapat dikurangi dengan memperbesar ukuran pages
      - Apakah keuntungan ukuran pages besar?
    - Umumnya page disesuaikan dengan kapasitas memori (tipikal) pada sistim (range: 2 – 8 Kbytes)

Implementasi Page Table

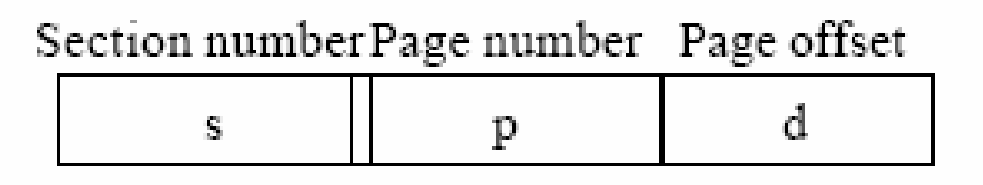
- Page table disimpan di main memory.
- *Page-table base register (*PTBR) menunjuk ke page table.
- *Page-table length register* (PRLR) mengindikasikan ukuran page table.
- Pada skema ini, setiap akses data/instruksi membutuhkan dua memori akses. Satu untuk page table dan satu untuk data/instruksi.
- Masalah yang ada pada dua akses memori dapat diselesaikan dengan menggunakan cache memori berkecepatan tinggi yang disebut *associative memory* or *translation look-aside buffers (TLBs)*

Paging Hardware dengan TLB

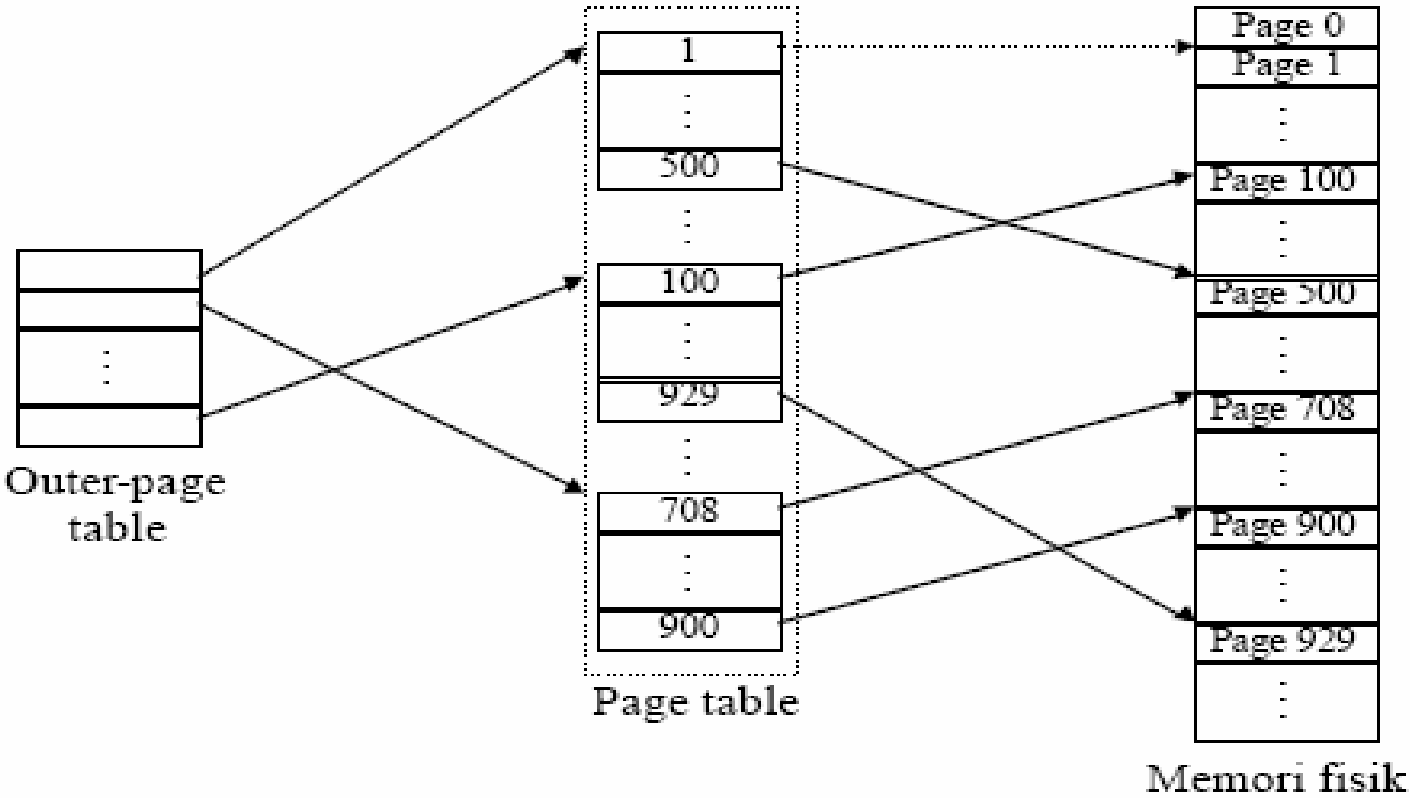


Multilevel Paging

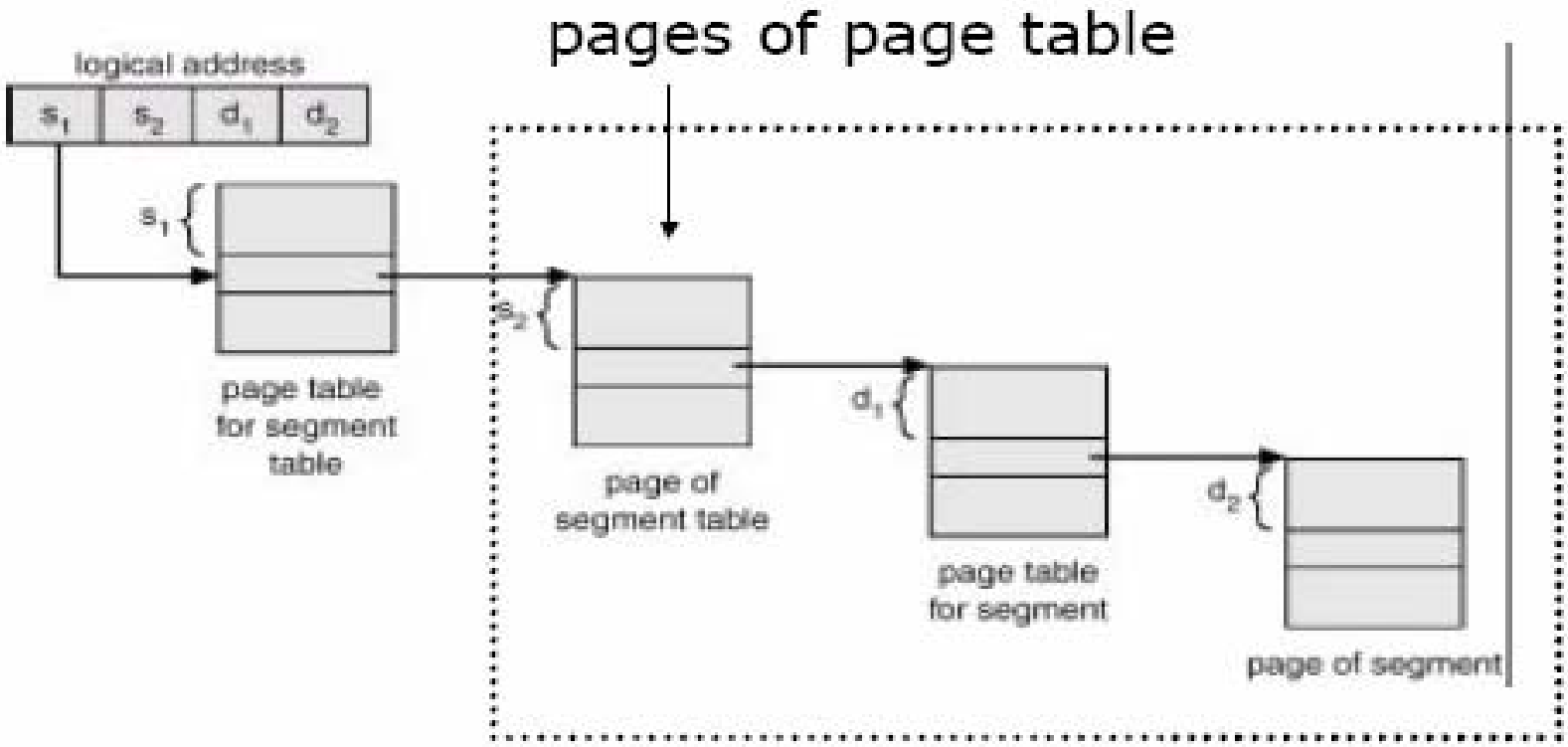
- Address logical besar => page table menjadi besar.
  - Misalkan: LA => 32 bits, dan ukuran page frame: 12 bits, maka page table: 20 bits (2^20 => 1 MB).
  - Page table dapat dipisah dalam bentuk pages juga, sehingga tidak semua page table harus berada di memori.
- Address lojik terdiri atas: section number *s*, page number *p*, offset *d*
  - *s* indeks ke dalam outer page table dan *p* displacement dalam page table



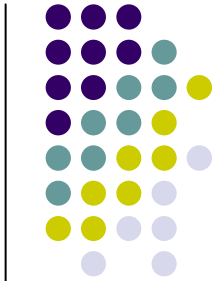
Two level page table



Translation: multilevel



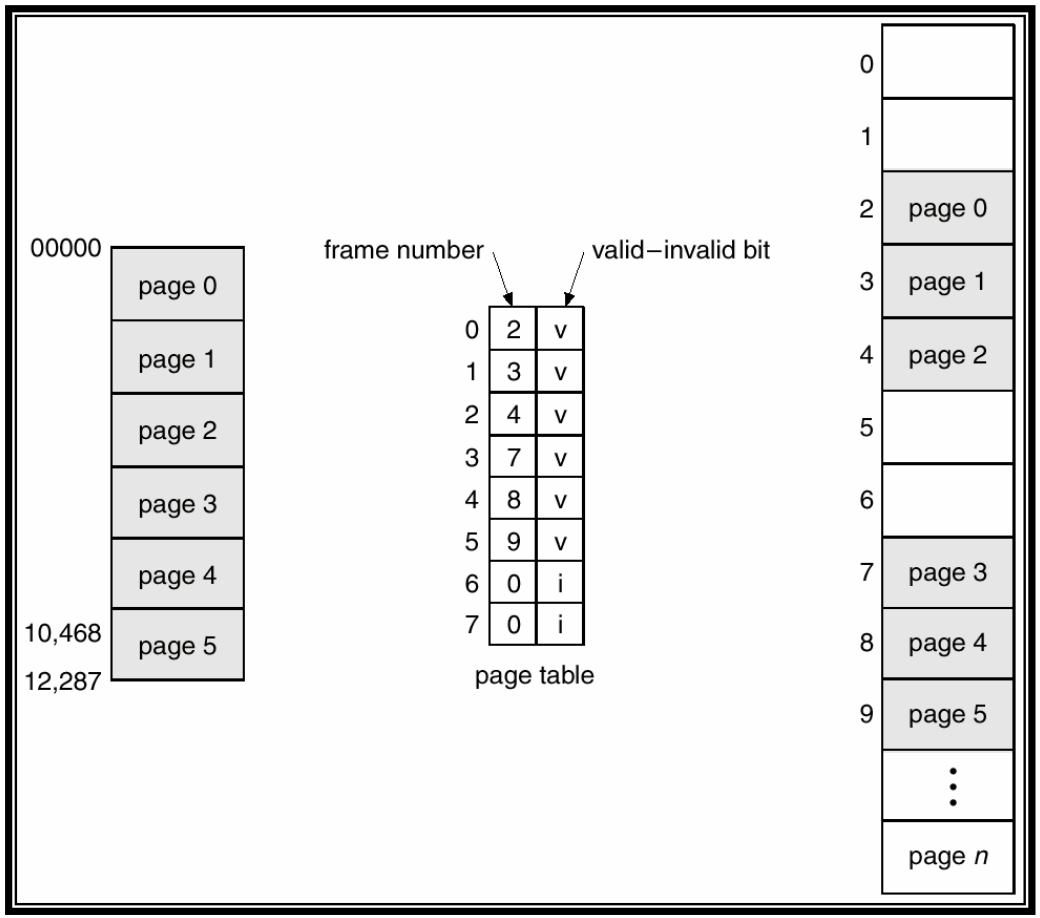
Proteksi Memory



- Proteksi memori diimplementasikan dengan asosiasi proteksi bit pada setiap frame

- *Valid-invalid* bit ditambahkan/dimasukkan pada page table :
  - Bit akan diset valid jika page yang bersangkutan ada pada area ruang alamat logika
  - Bit akan diset "invalid" jika page yang bersangkutan berada di luar area ruang alamat logika.

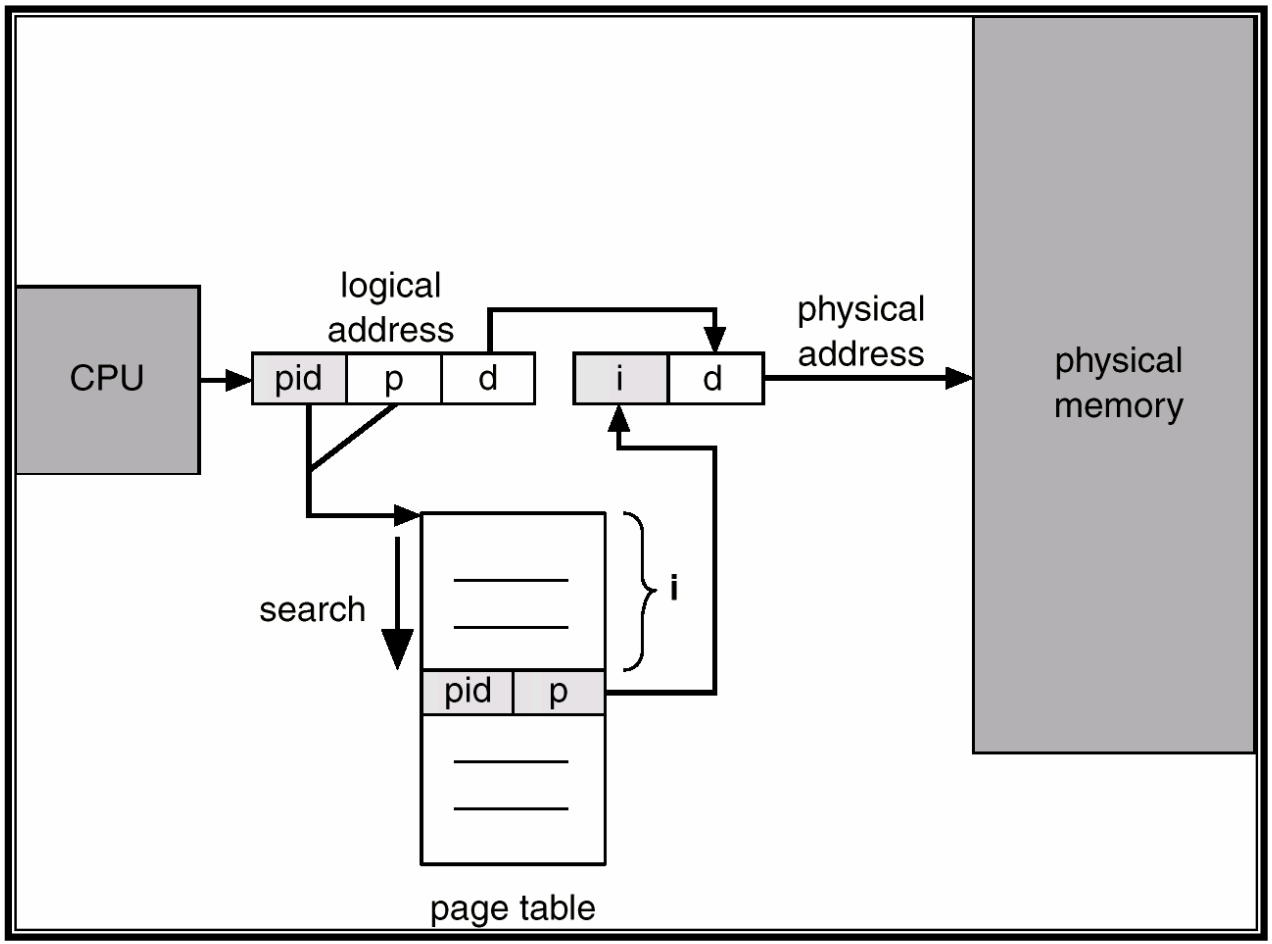
Valid (v) or Invalid (i) Bit pada Page Table



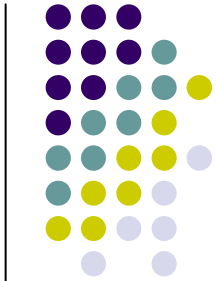
Inverted Page Table

- Satu masukan untuk setiap real page dari memori
- Masukan dari alamat virtual disimpan pada lokasi real memori, dengan informasi proses pada page
- Penurunan memori dibutuhkan untuk menyimpan setiap page table, tetapi setiap kenaikan waktu dibutuhkan untuk mencari tabel saat pager refference dilakukan

Arsitektur Inverted Page Table



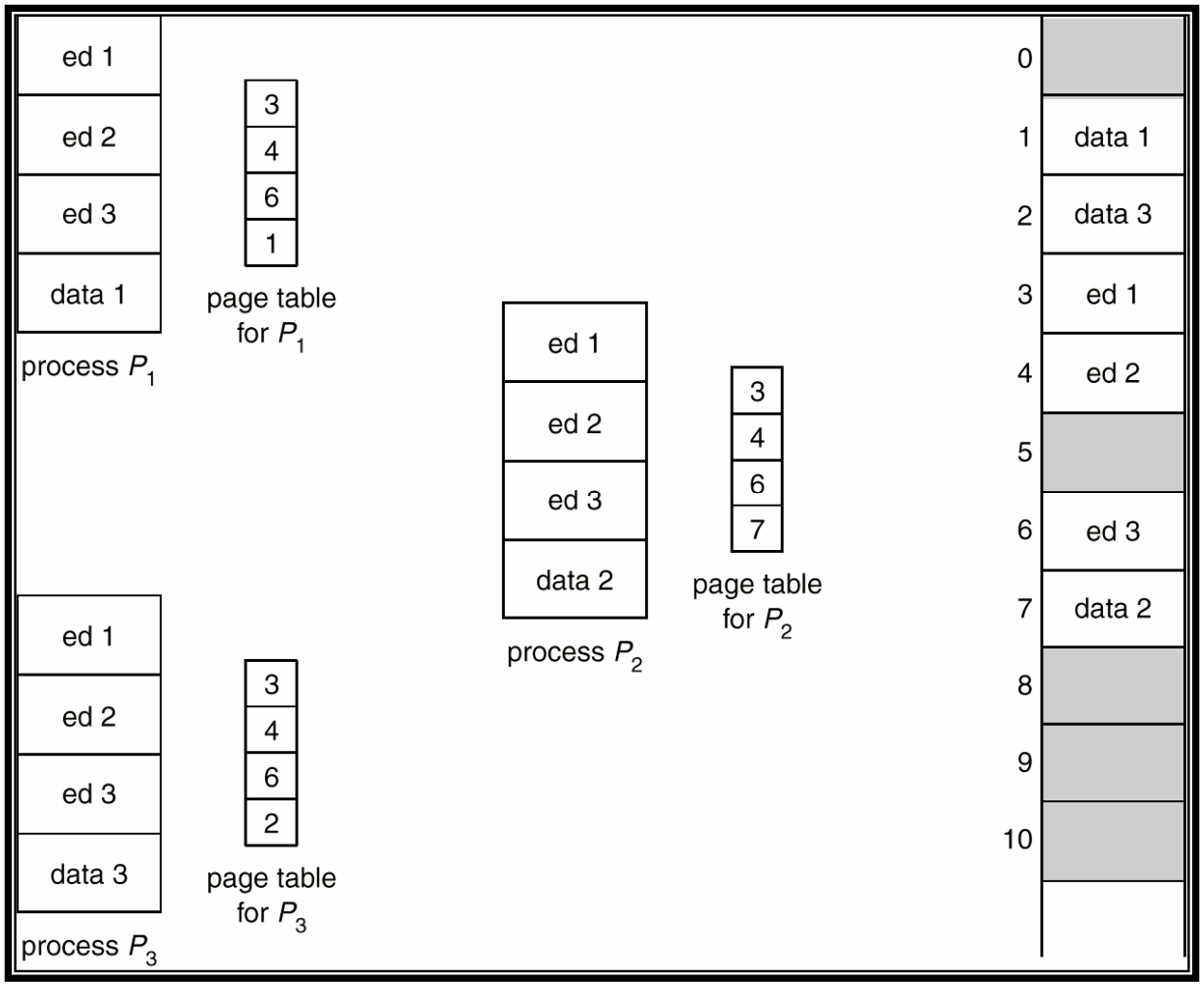
Shared Pages



- Shared code
  - Satu copy kode read-only (reentrant) dibagi diantara proses (contoh text editor, compiler, window system).
  - Shared code harus dimunculkan pada lokasi yang sama pada alamat logik semua proses.

- Private code dan data
  - Setiap proses menyimpan sebagian copy kode dan data.
  - Page untuk kode private dan data dapat ditampilkan dimana saja pada ruang alamat logik.

Contoh Shared Pages

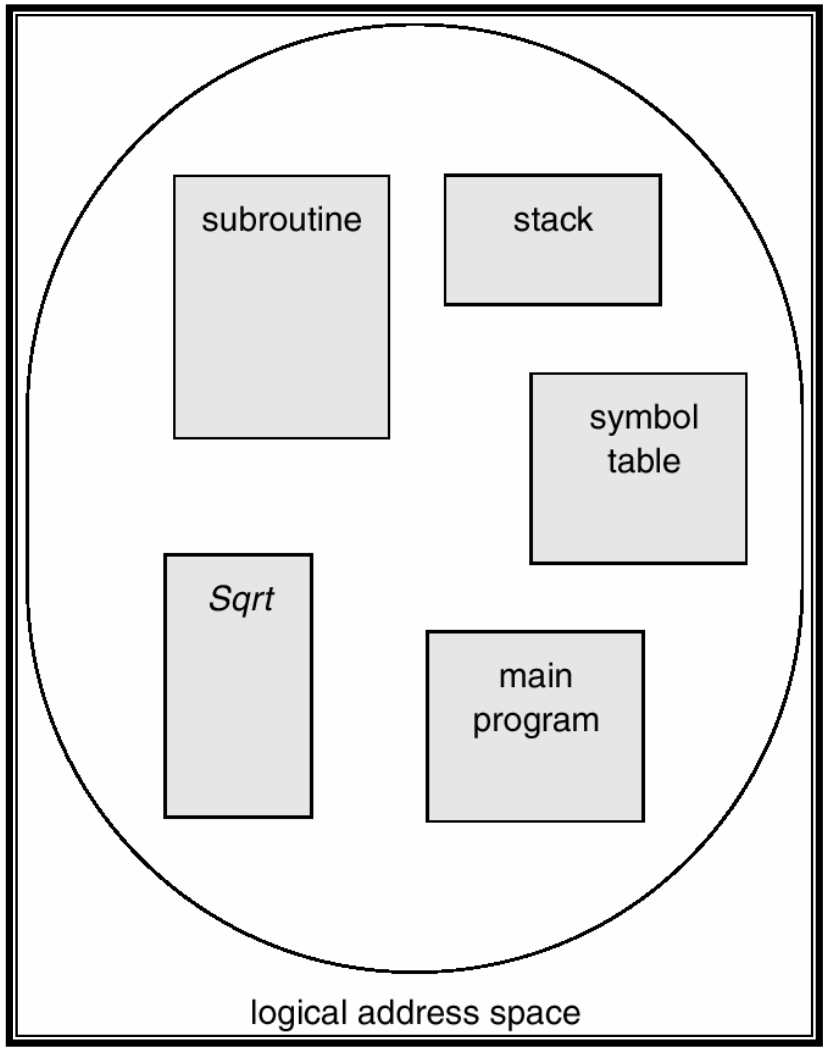


Segmentasi

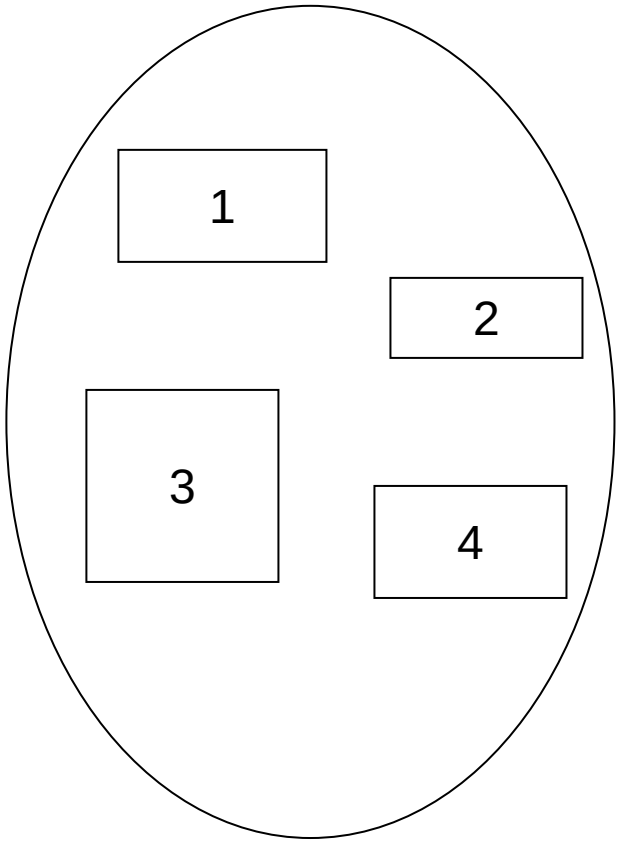
- Skema pengaturan memori yang mendukung user untuk melihat memori tersebut..
- Sebuah program merupakan kumpulan dari segment. Sebuah segement berisi unit logik seperti:

  main program,
  procedure,
  function,
  method,
  object,
  local variables, global variables,
  common block,
  stack,
  symbol table, arrays

User View Program



Pandangan Logik Segmentasi



user space

physical memory space

Arsitektur Segmentasi

- Alamat logik terdiri dari dua tuple:

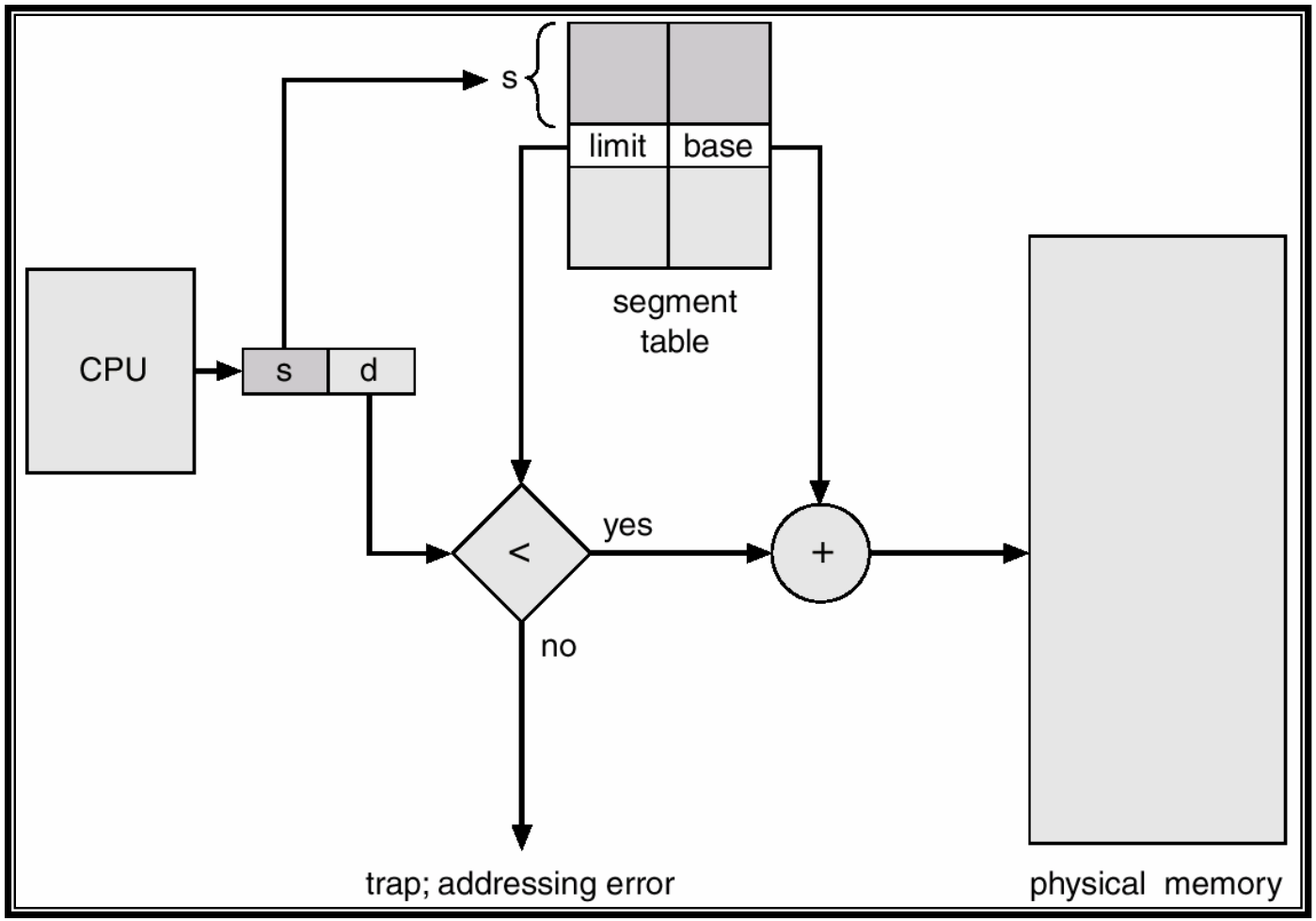
  <segment-number, offset>,

  - Harus diset oleh programmer atau compiler untuk menyatakan berapa besar segment tersebut
  - Implikasi: segment bervariasi besarnya (bandingkan dengan page table: fixed dan single/flat address space => hardware yang menentukan berapa size)
- *Segment table* – mapping dari LA ke PA
  - base table – berisi lokasi awal dari physical address dimana segment berada di memori.
  - limit table – berisi panjang (besar) dari segmen tersebut.

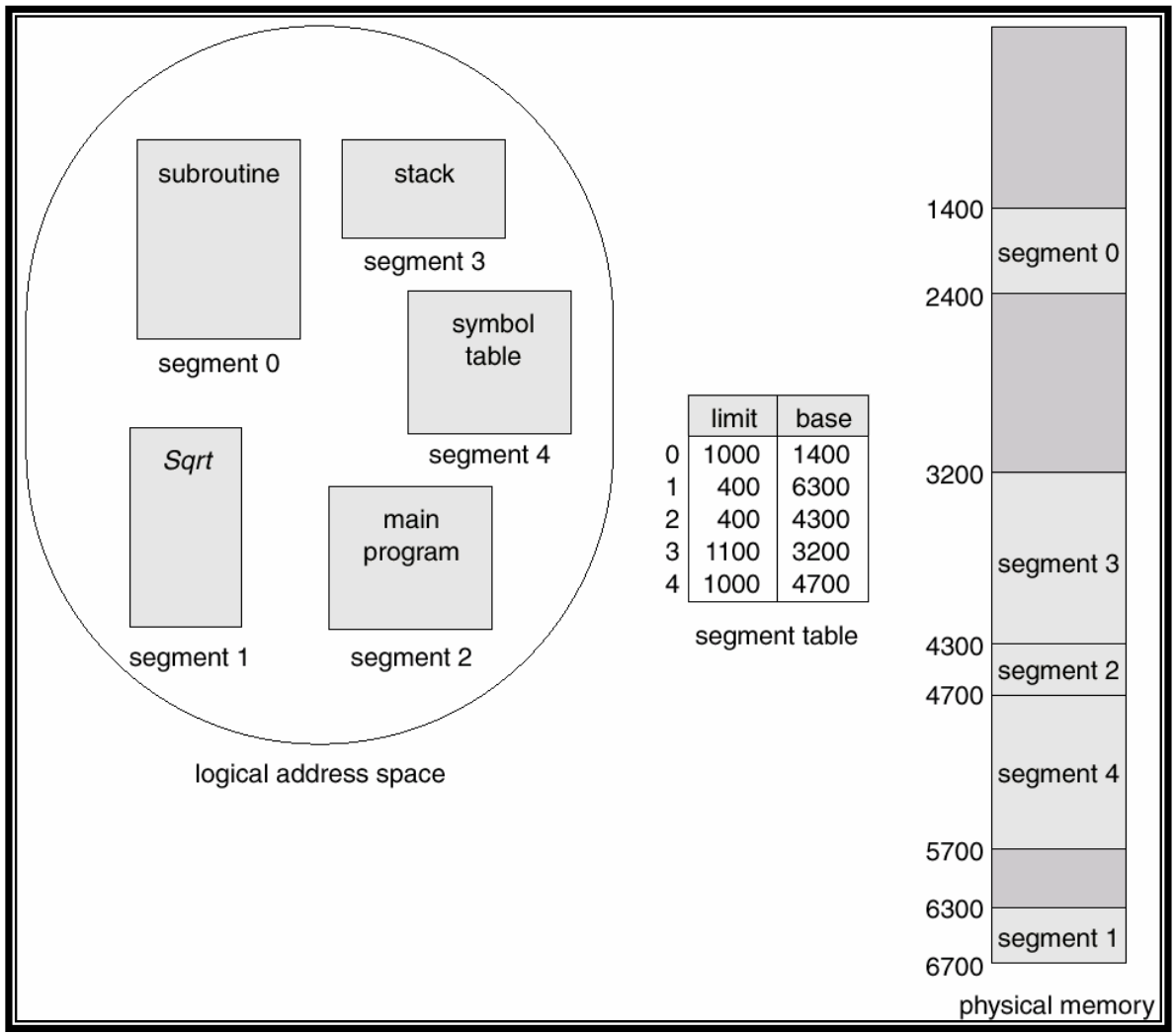
Arsitektur Segmentasi (Cont.)

- Relokasi.
  - Dynamic
  - Melalui segment table
- Sharing.
  - Shared segments
  - Nomor segment yang sama
- Alokasi.
  - first fit/best fit
  - external fragmentation

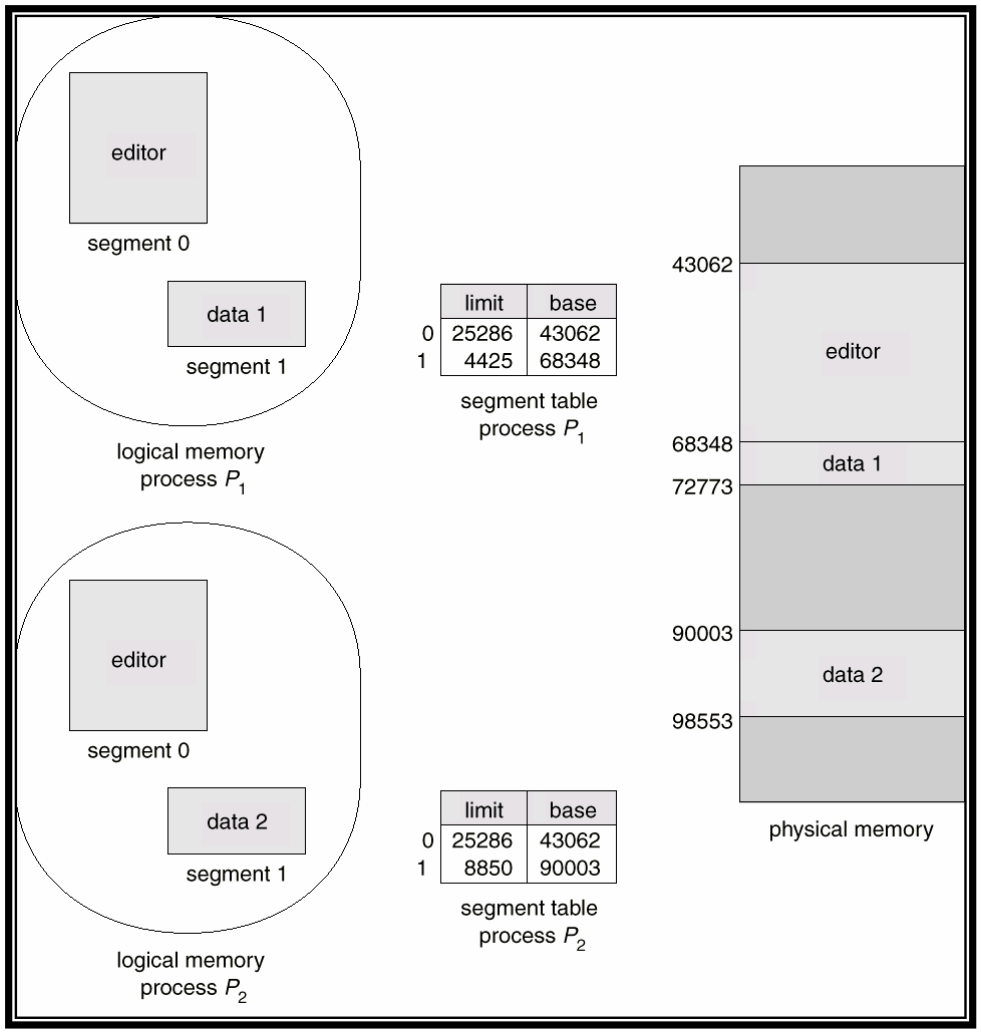
Segmentasi Hardware



Contoh Segmentasi



Sharing of Segments



Segmentasi Paging

- Intel 386
  - Logical address (32 bits) dibagi atas 2:
    - Selector: Segment: S (13 bits), Descriptor Table (1 bit: Local or Global); Protection (2 bits)
    - Offset: 16 bits
  - Melalui Descriptor table
    - Selector menentukan entry pada table, melihat protection, dan menguji limit (tabel berisi informasi limit)
    - Menghasilkan linear address: Base address segment + offset
  - Logical Linear address: paging (besar page: 4 K), 2 level (10 bits untuk direktori dan 10 bits untuk page number), offset: 12 bits.

Alamat Translasi Intel 30386