# Ch07. Memori Utama

IKI 20250 – Sistem Operasi
Fakultas Ilmu Komputer UI

Revisi: 11 April 2012

# A. KONSEP

# TUJUAN PEMBELAJARAN

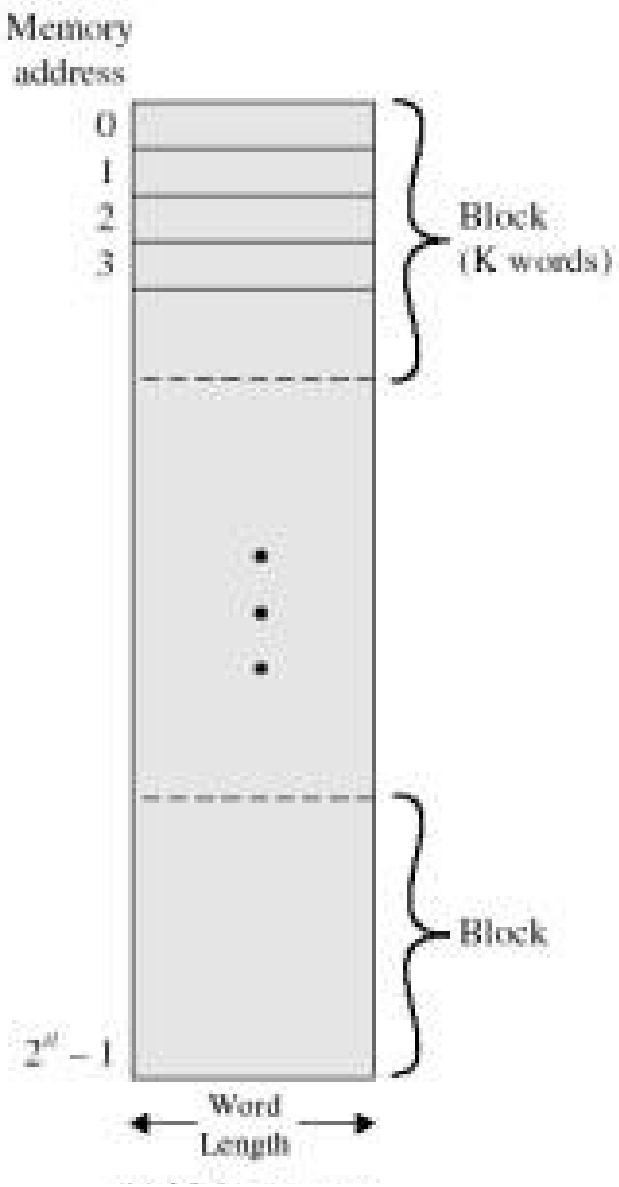
- Memahami konsep dasar memori
- Memahami pengalamatan memori dengan menggunakan teknik paging

# PENDAHULUAN

- CPU membutuhkan memori untuk menempatkan program dan data saat eksekusi.
- Memori  terdiri dari kumpulan array byte/word dalam jumlah besar, masing-masing memiliki alamat yang berbeda
- CPU hanya bisa mengakses langsung memori dan register

# MEMORI

Ilustrasi hardware memori

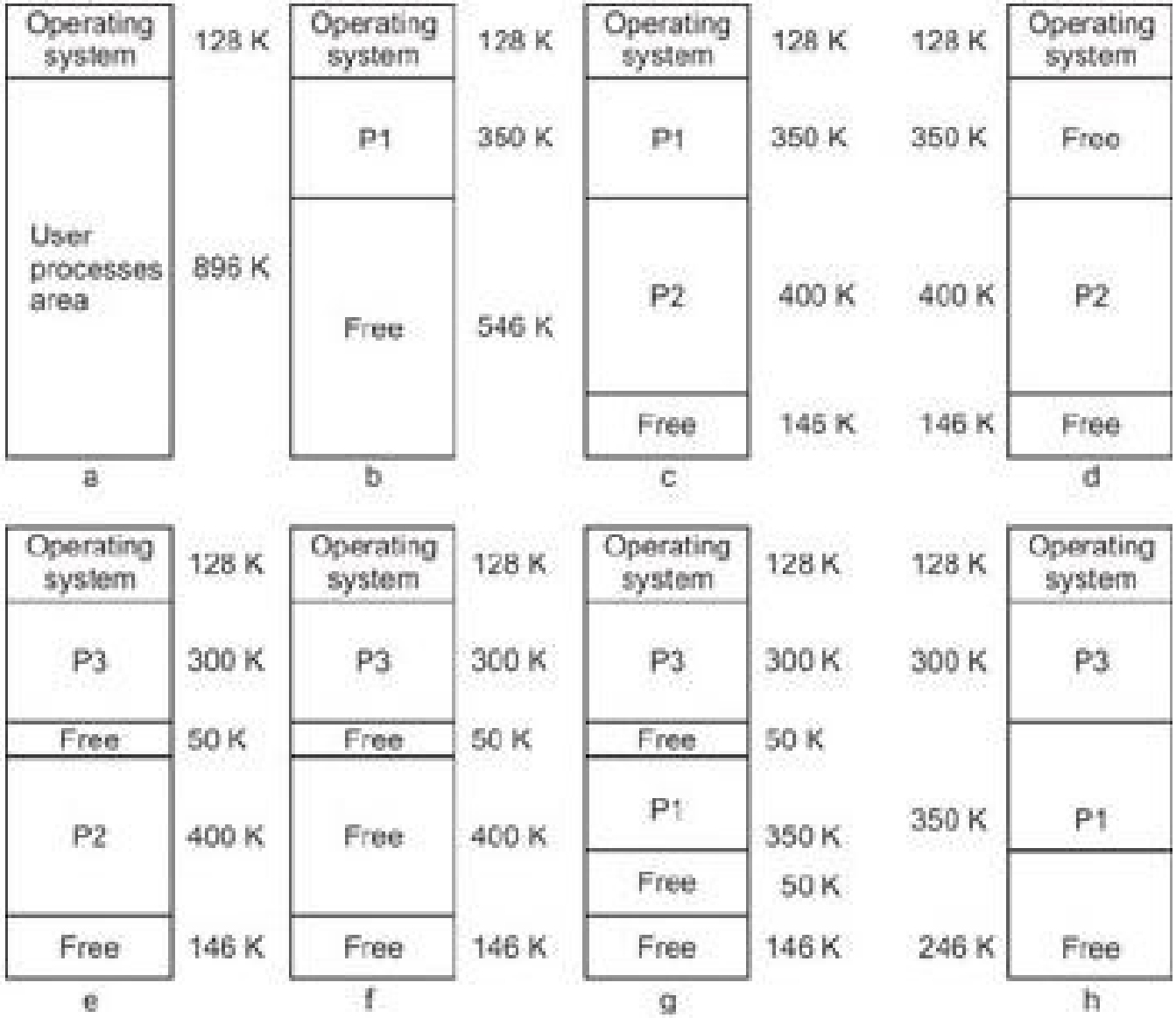



Ilustrasi memori secara logik

# PENDAHULUAN

- Agar instruksi atau data dapat dieksekusi oleh CPU, maka instruksi atau data harus dikirim dahulu ke memory. Dari memori, instruksi atau data dipindahkan ke register CPU
- Metode pengalamatan memori:
  - Berurutan (contiguous)
    - Barisan alamat suatu proses harus ditempatkan secara berurutan dimemori
  - Tidak Berurutan (uncontiguous)
    - Paging
    - Segmentasi

# PENGALAMATAN MEMORI BERURUTAN



# MEMORY LOGIC DAN MEMORY FISIK

- Memory Logik
  - Wilayah memori logik untuk satu proses
  - Menggunakan alamat logik
  - Butuh translator  untuk mendapatkan alamat fisik
- Memori Fisik
  - Bentuknya fisik, dapat dilihat
  - Umumnya menyimpan banyak proses
  - Menggunakan alamat fisik

# PENGALAMATAN MEMORI TIDAK BERURUTAN (CONT.)

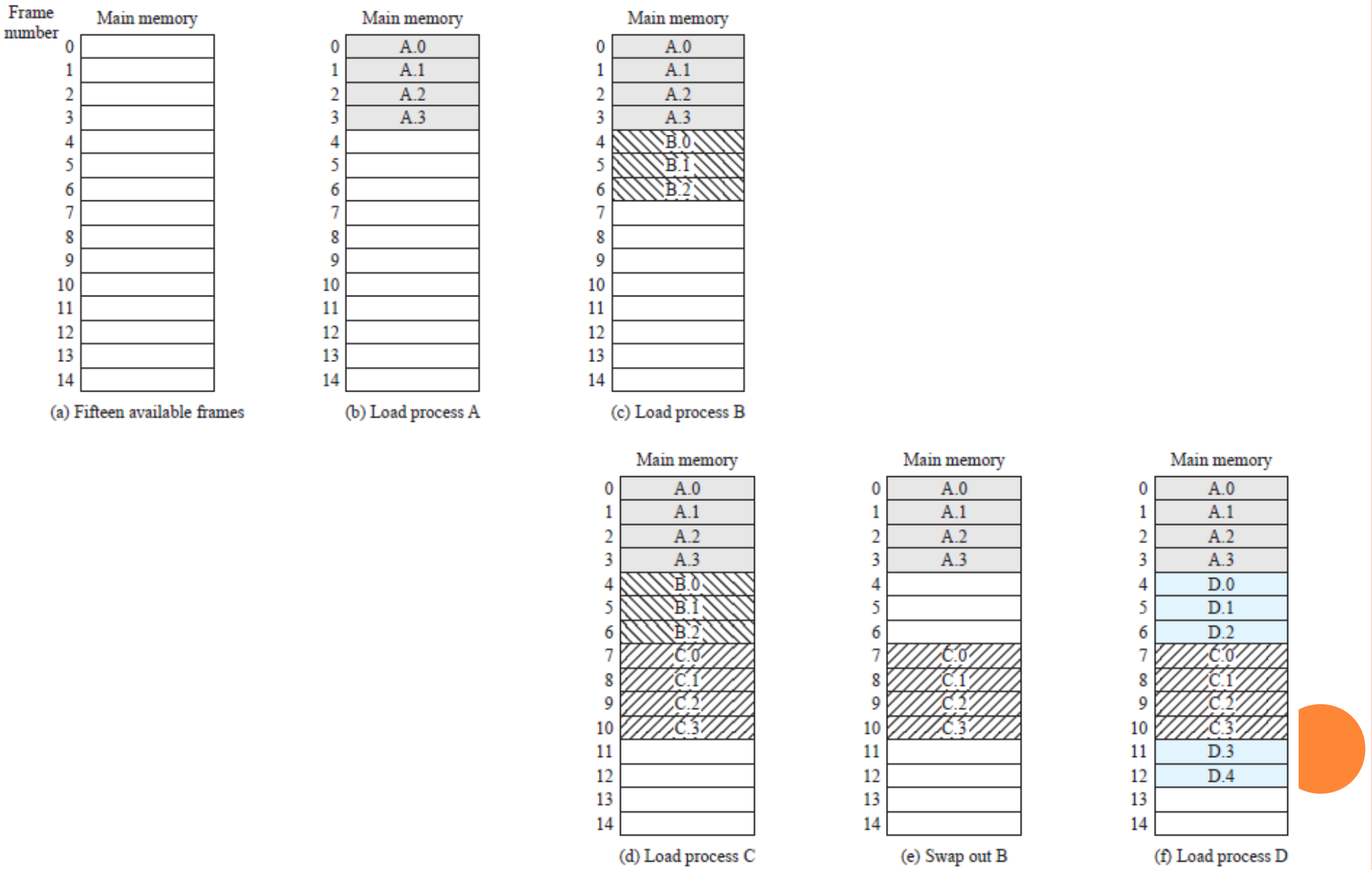
A. Paging:

- Menggunakan memori logik dan memori fisik
- Butuh hardware Memory Management Unit(*mapper)* yang melakukan pemetaan alamat logik ke alamat fisik
- Proses dipecah-pecah secara logik menjadi page-page

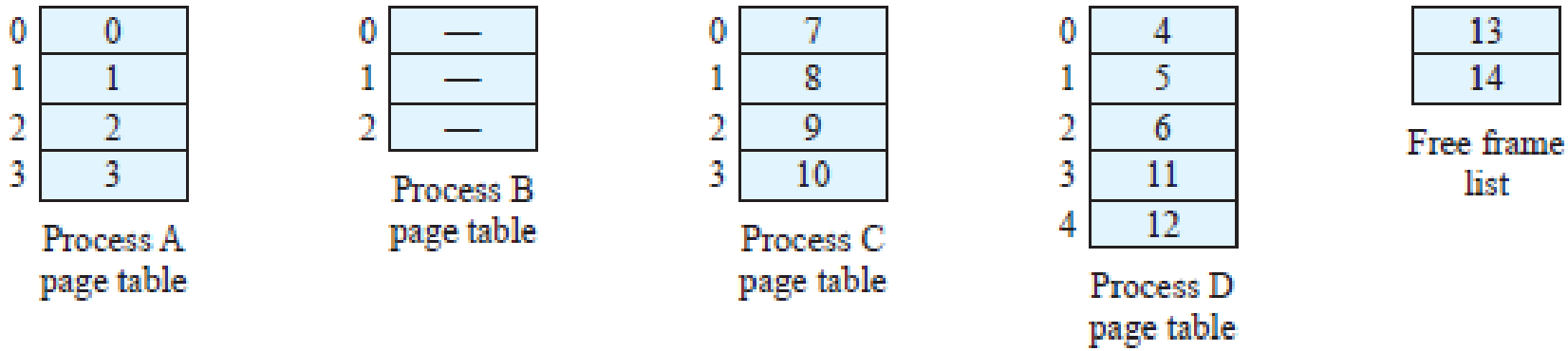
# PAGING (CONT.)

- Memori fisik dibagi menjadi frame-frame
- Page menempati ruang frame secara tidak beraturan
- Ukuran 1 frame = ukuran 1 page
- Ukuran 1 frame dan 1 page tetap

# ILUSTRASI PENEMPATAN PROSES

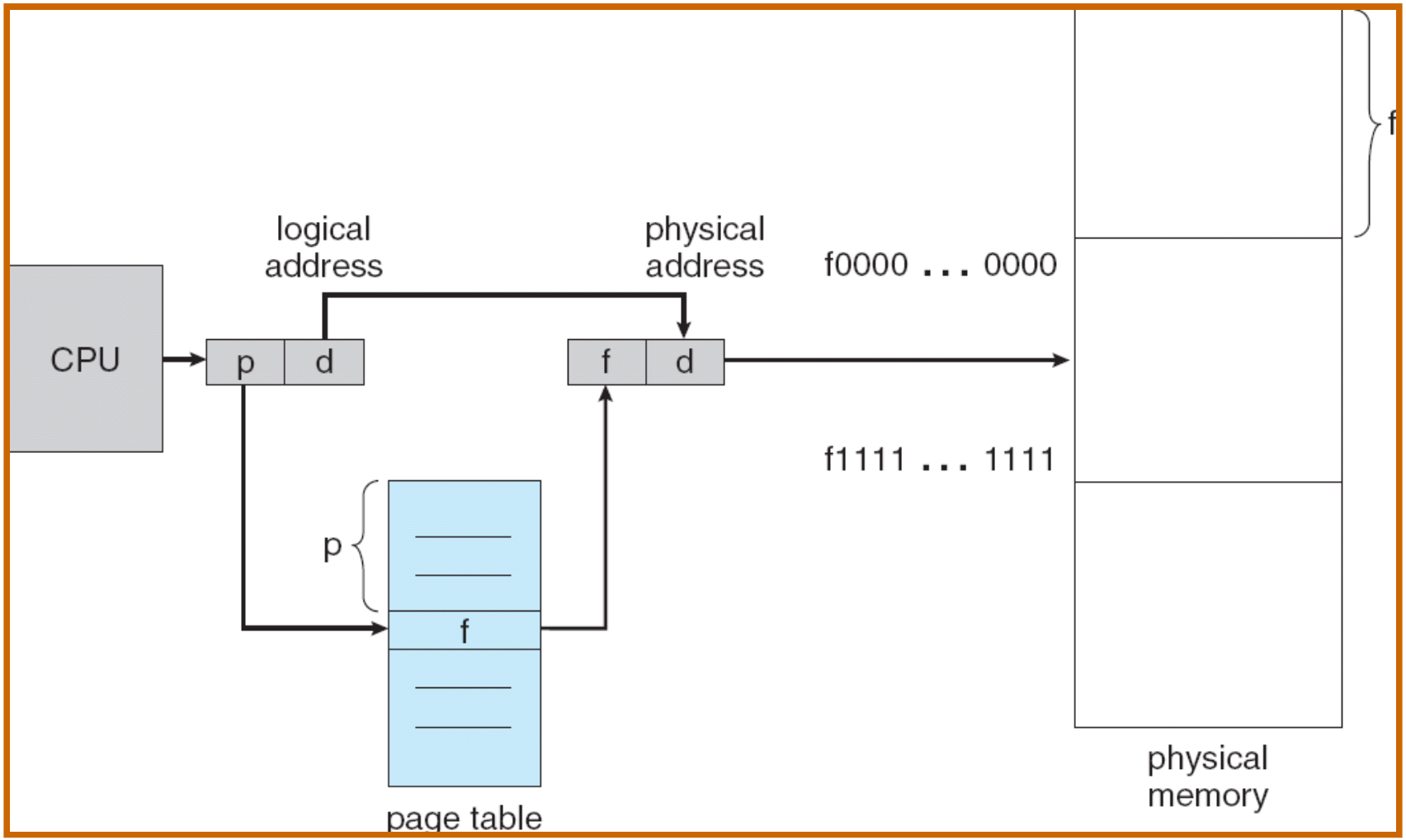


# ILUSTRASI PENEMPATAN PROSES



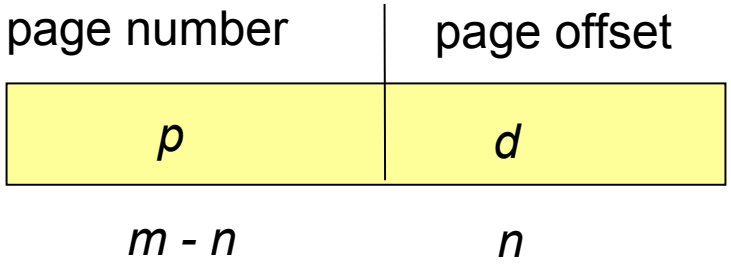

Page table untuk setiap proses

# PAGING


logical
address
physical
address
f0000 . . . 0000
f1111 . . . 1111
CPU
p
d
f
d
p
f
page table
f
physical
memory

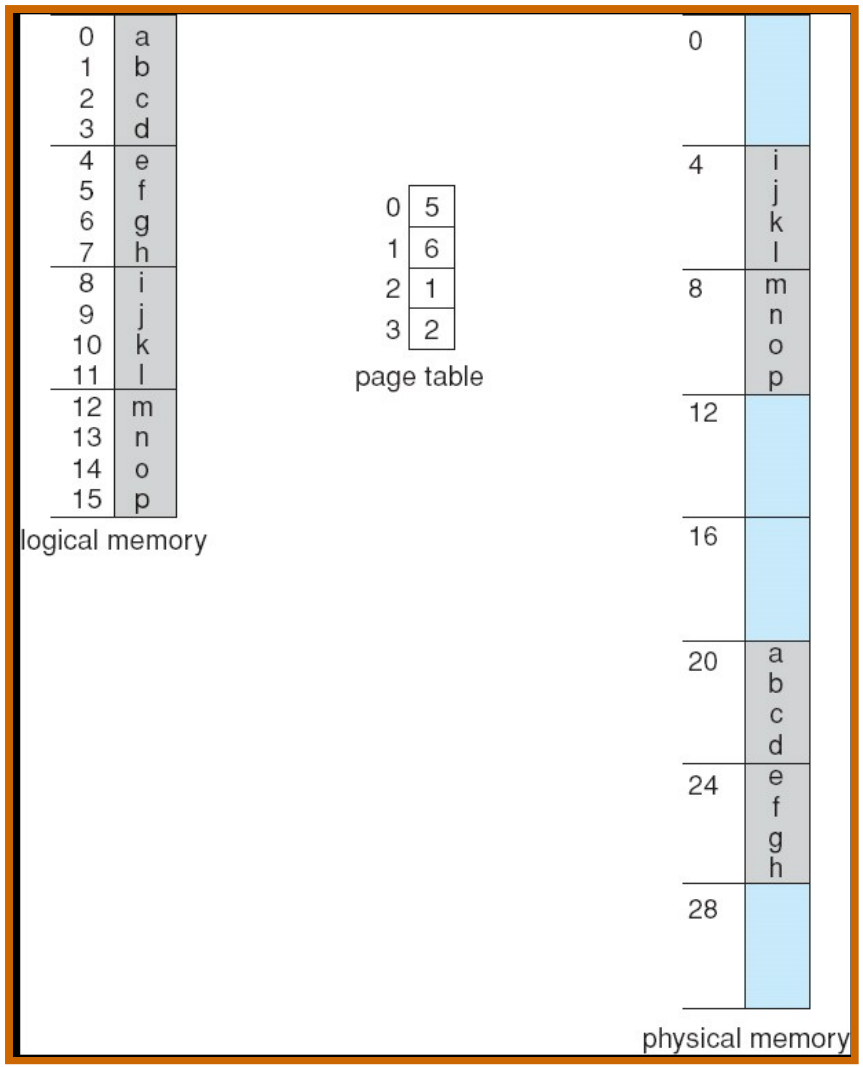
# SKEMA PENERJEMAHAN ALAMAT

- Alamat yang dihasilkan CPU dibagi menjadi dua
  - **Page number (*p*)**
    - Nomor page Proses
  - **Page offset (d)**
    - Nomor offset suatu page
    - Dalam mode binary: bit offset digabung dengan bit nomor page untuk menghasilkan alamat fisik.
    - Dalam mode desimal: (ukuran page x no frame) + nilai offset = alamat fisik




  - Ukuran ruang alamat logik = $2^m$
  - Ukuran page = $2^n$

# CONTOH PAGING




Alamat logik 0 = page 0, offset 0
Alamat fisik = 20 (=((5x4)+0)

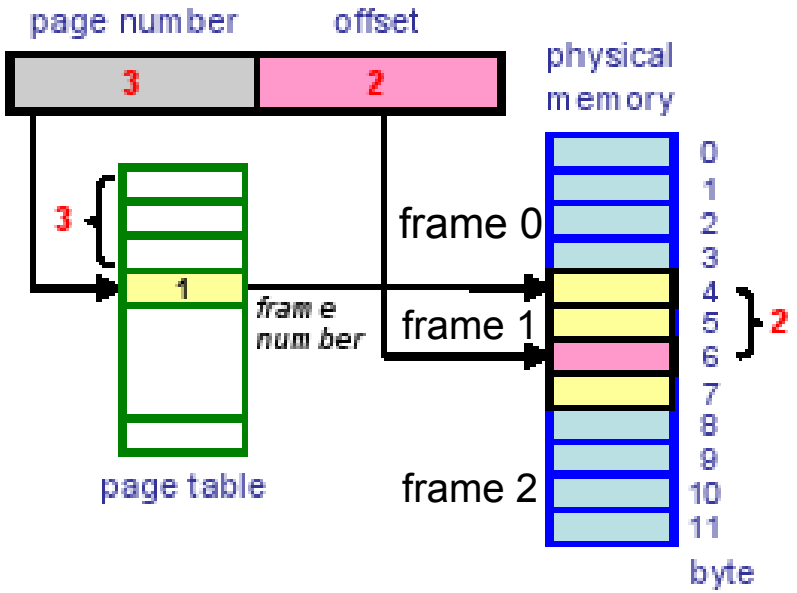
Alamat logik 3 = page 0, offset 3
Alamat fisik = 23 (=((5x4)+3)

Alamat logik 4 = page 1, offset 0
Alamat fisik = 24 (=((6x4)+0)

Alamat logik 13 = page 3, offset 1
Alamat fisik = 20 (=((2x4)+1)

Ukuran memori fisik 32 byte dan ukuran page 4 byte

# CONTOH PAGING DENGAN UKURAN PAGE 4 BYTE




Frame Address:
4 = 1 * 4 (size)

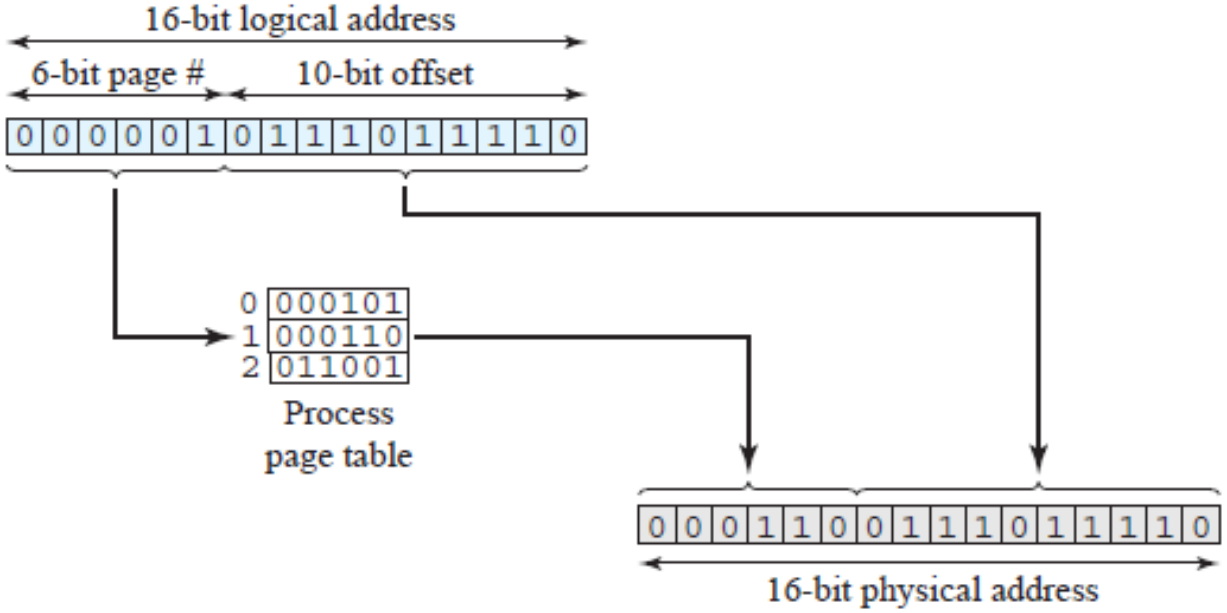
Physical Address:
6 = 1 * 4 + 2 (offset)

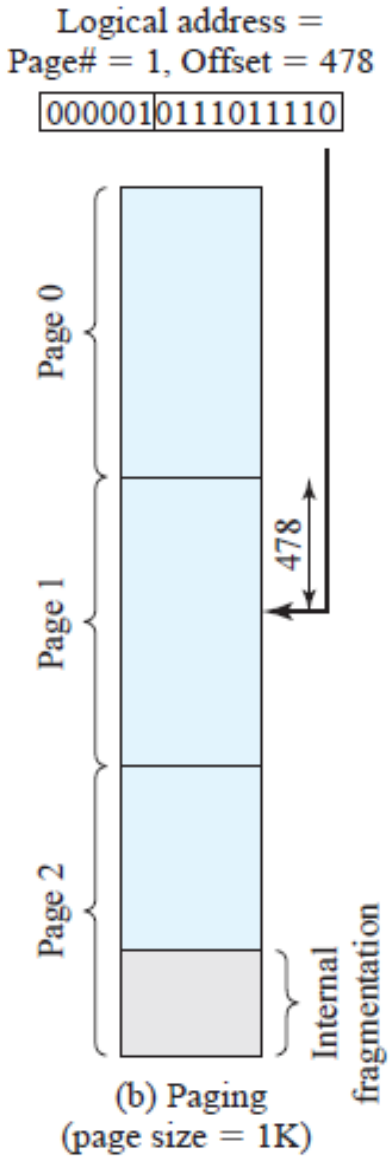
physical_address = frame_number * frame_size + offset

# LATIHAN

Diketahui :
ukuran page = 1 KiB=$2^{10}$
Alamat logika = 1502 = 0000010111011110
ruang alamat logik= 16 bit

Tentukan nomor page dan nomor offset!



# LATIHAN

- Diketahui ukuran memori fisik =1GB, ukuran frame 4KB. Berapa jumlah page, jika besar proses user 25KB?

  A. 249

  B. 251

  C. 250

  D. 6

  E. 7

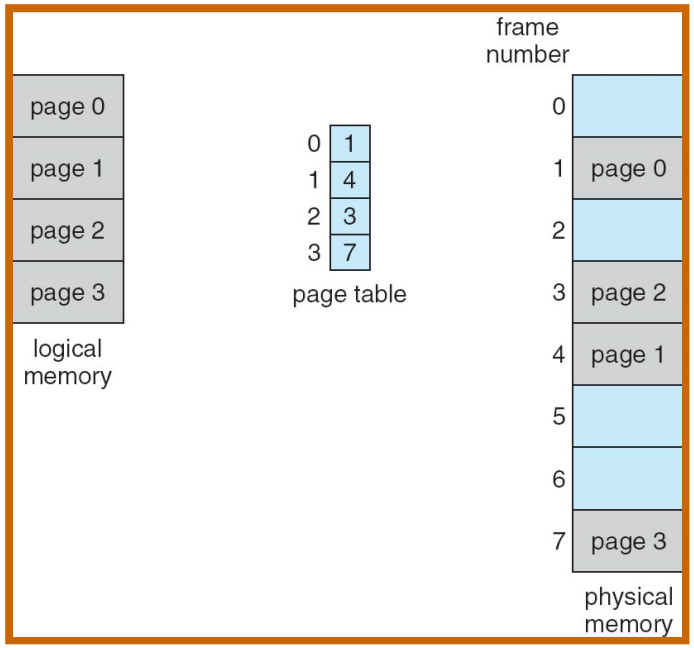
# LATIHAN PAGING

- Memori fisik mempunyai kapasitas 256 MiB ($2^{28}$). Ukuran framenya 64 byte
  - Berapa jumlah framenya?
  - Jika besar program A adalah 10KiB, berapa page yang dibutuhkan?
  - Berdasarkan tabel dibawah ini, berapa alamat fisik dari alamat logika : 30, 234, 377

| Page Number | Frame |
|---|---|
| 0 | 6 |
| 1 | 2 |
| 2 | 4 |
| 3 | 1 |
| 4 | 0 |
| 5 | 3 |
| 6 | 8 |
| 7 | 13 |
| ... | ... |

# UKURAN PAGE

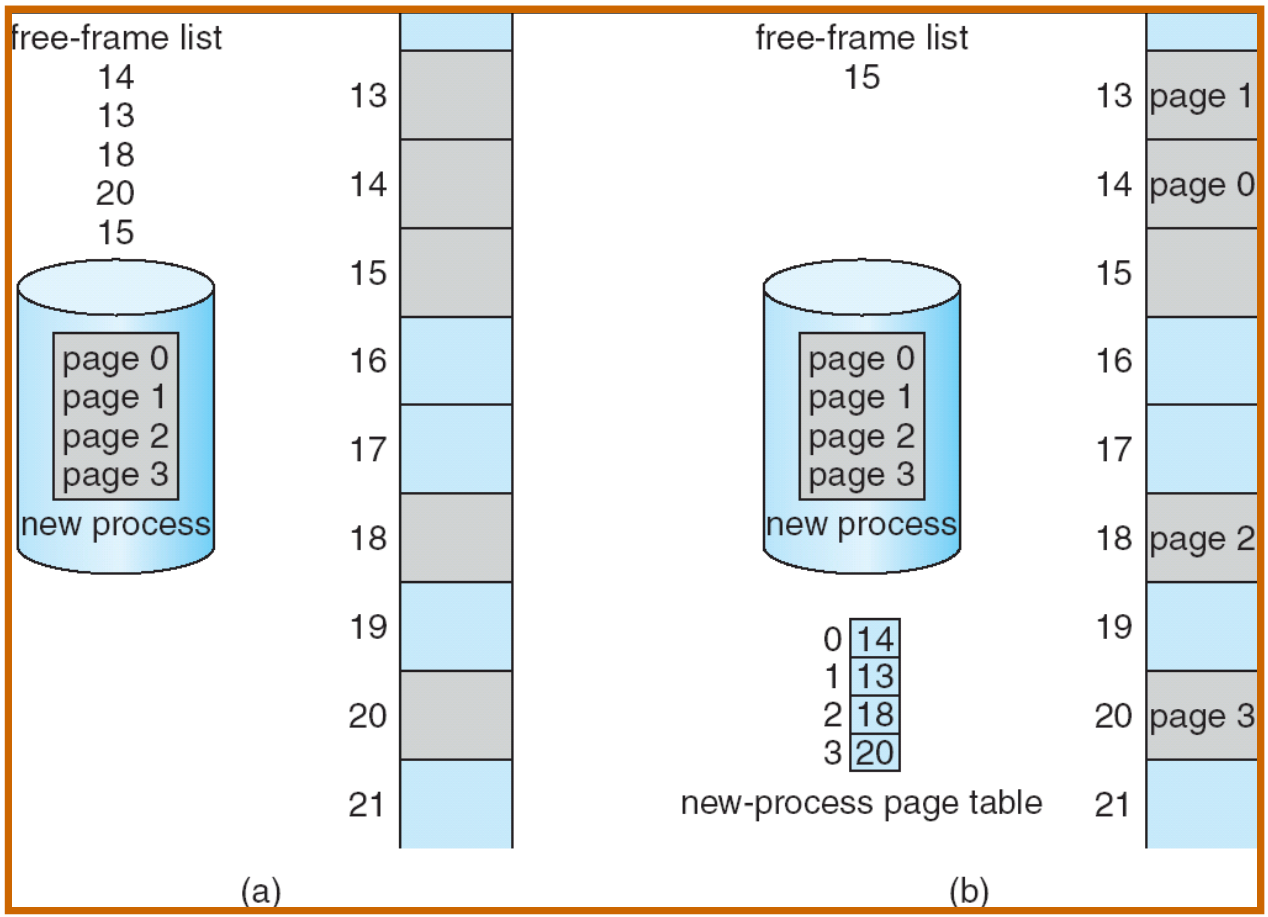
- Ukuran page dan frame didefinisikan oleh hardware.
- Ukuran *page* umumnya $2^n$ dengan variasi besar tiap page antara 512 byte hingga 16MiB, tergantung dari arsitektur mesin.



# FRAGMENTASI INTERNAL

- Kelemahan paging → Fragmentasi Internal
  - Proses A = 11KB
  - Page size= 2KB
  - Page yang dibutuhkan :
    - (Process A / Page size)=(floor) (11/2) = 5
    - 11 – (5*2)= 1 KB
    - Proses A butuh 5 page + 1 KB
    - Jika n pages + 1 byte maka jumlah page = (n + 1) page
    - Prosess A butuh 6 Page

# ALOKASI PAGE




Sebelum alokasi

Setelah alokasi

Keterangan :

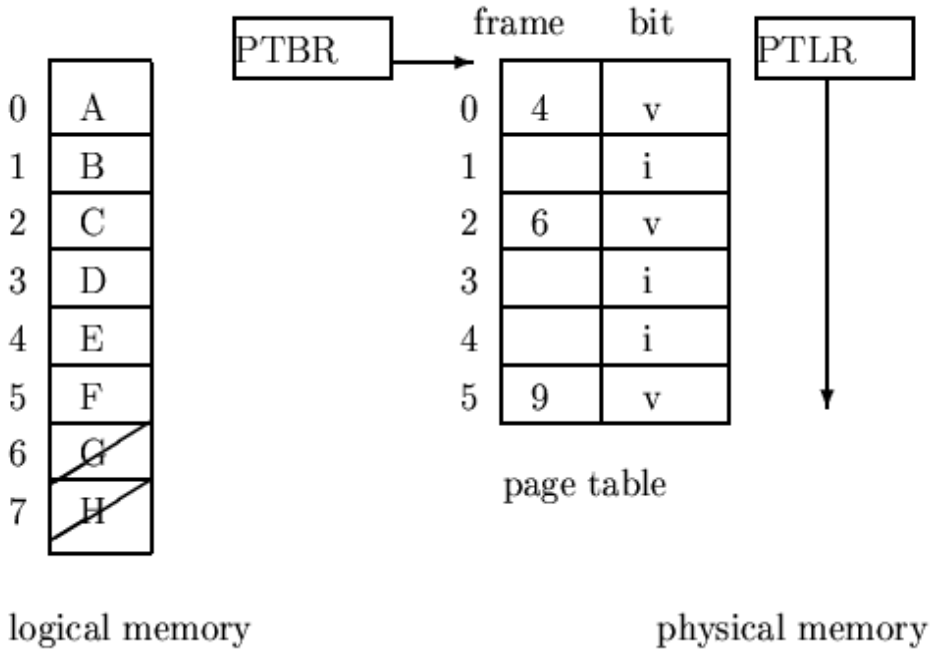
Sebuah proses mempunyai 4 page.

4 page membutuhkan 4 frame

# PAGE TABLE

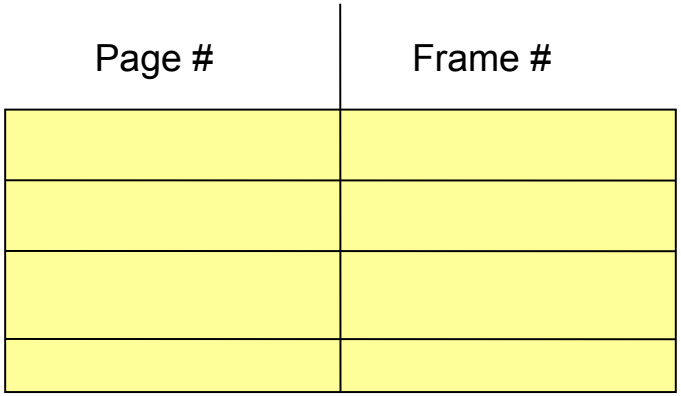
- Page table dapat disimpan dalam register atau memori.
- Jika terlalu besar, page table disimpan di memori.
- *Page-table base register* (PTBR) menunjuk *ke* lokasi memori dari sebuah page table
- **PTLR** (Page-Table Length Register)
  - Register yang menyimpan jumlah entri page table suatu proses
- Jumlah akses memori untuk resolusi alamat fisik?
  - Butuh dua kali akses memori = akses ke page table + akses ke entri frame

# PTBR DAN PTLR



# PAGE TABLE DENGAN TLB

- Minimalisasi akses memory ➔ gunakan TLB (*Translation Look-aside Buffers* )




pemetaan alamat logik (p, d)

Jika nomor page p berada dalam TLB, ambil alamat framenya

Jika tidak ada, maka cari nomor page p pada page table dan ambil alamat framenya

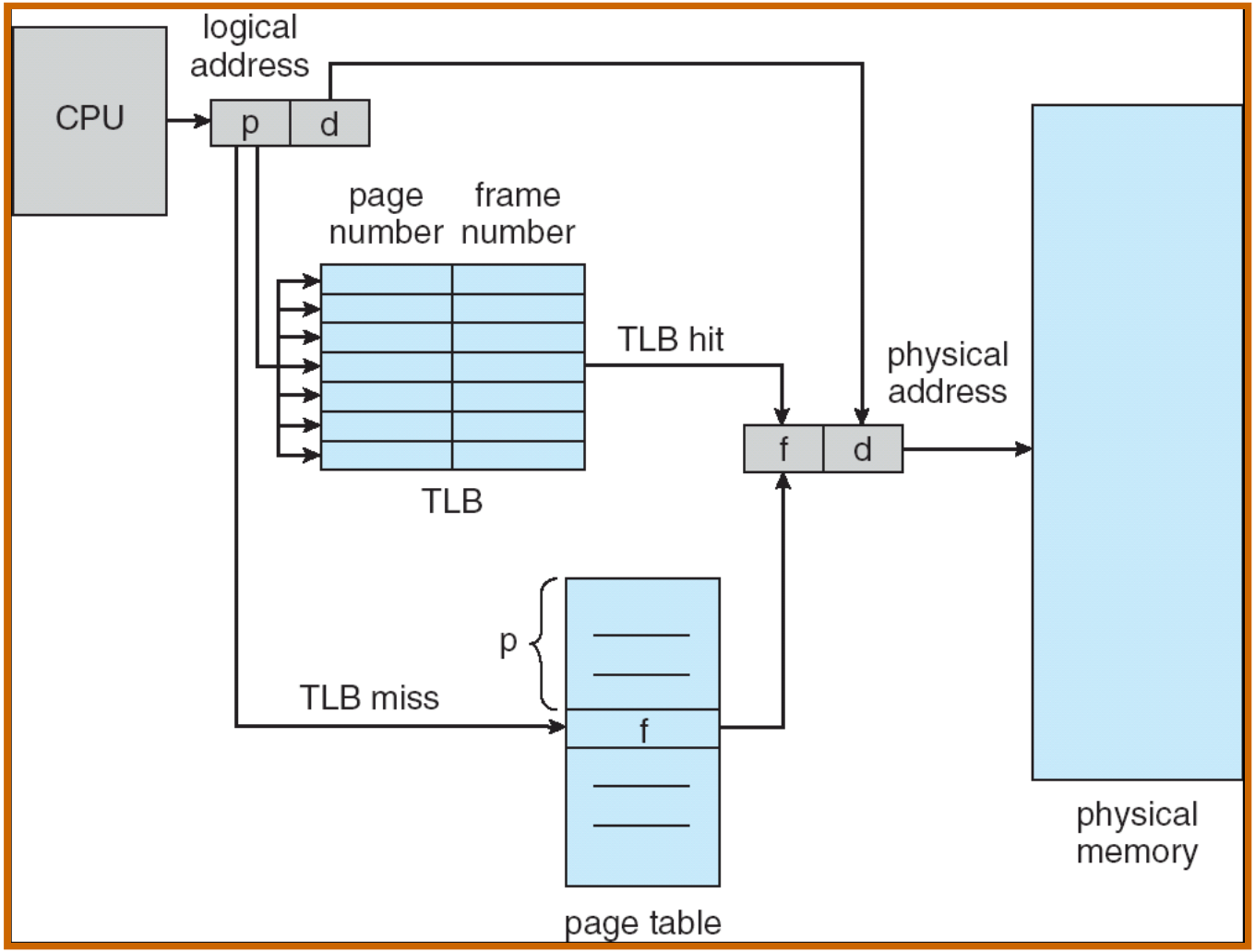
# PAGE TABLE DENGAN TLB

- Umumnya jumlah entry dalam TLB berkisar antara 64 hingga 1,024 byte
- Pergantian page pada entry TLB dapat menggunakan algoritma *Least Recently Used* atau page dipilih secara acak
- TLB dapat berisikan ASIDs (*address-space identifiers* )
  - Tanpa ASIDs, 1 TLB hanya untuk 1 proses
  - Dengan ASIDs, 1 TLB dapat digunakan oleh beberapa proses

# TLB HIT DAN TLB MISS

- TLB Hit artinya nomor page pada entri TLB ditemukan
- TLB Miss artinya
  - Nomor page yang dicari pada TLB tidak ditemukan atau
  - ASIDs suatu proses tidak sesuai dengan ASIDs pada entri TLB walau nomor page sama

# PAGE TABLE DENGAN TLB



# EFFECTIVE ACCESS TIME

Hit ratio (μ) – persentase sekumpulan page ditemukan dan diakses melalui TLB pada interval waktu tertentu

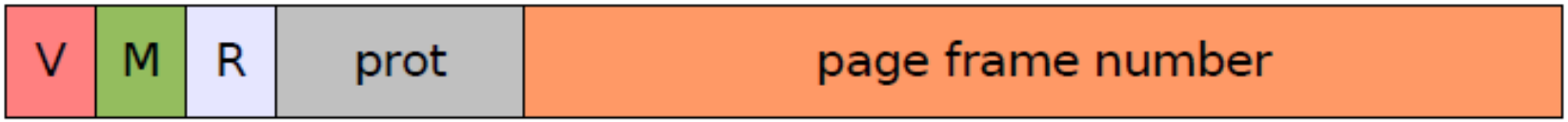
- **Effective Access Time** (EAT) TLB
- EAT= (am+atlb)μ +(am+apt+atlb)(1-μ)
- Akses ke memori fisik (am)
- Akses ke TLB (atlb)
- Akses ke Page Table (apt)

# EFFECTIVE ACCESS TIME

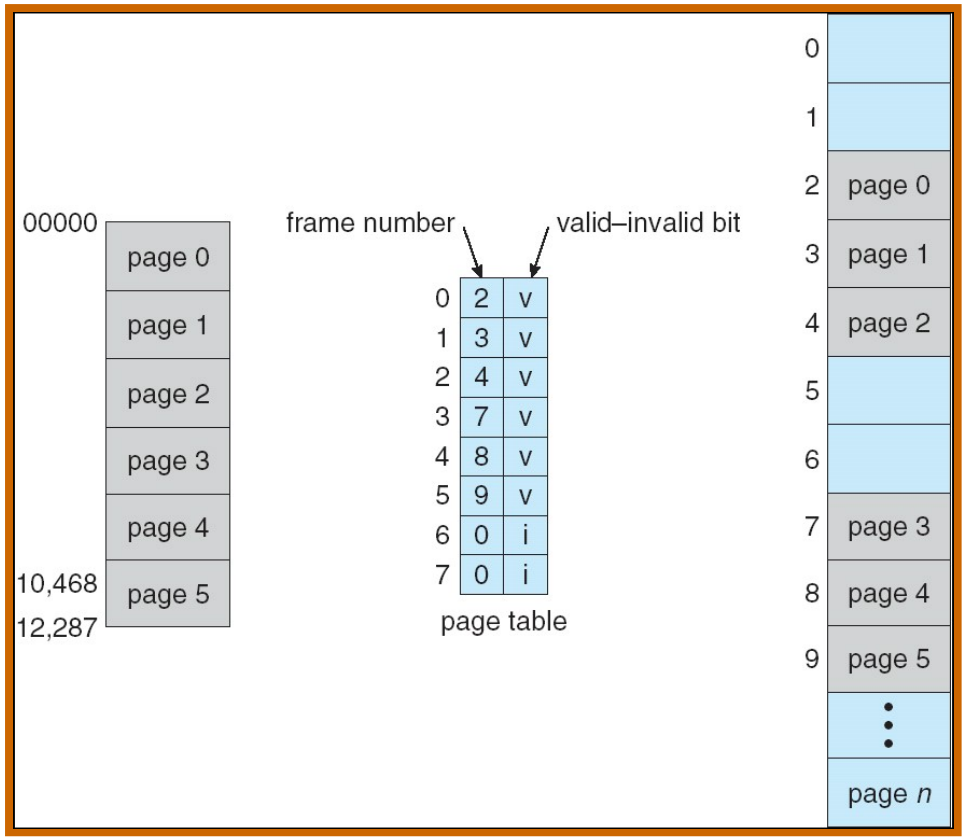
- Akses ke TLB membutuhkan waktu 20 ns
- Akses ke page table = 100 ns
- Akses ke memory = 100 ns
- Hit ratio = $\mu$ = 80%
- EAT= (20+100) $\mu$ +(20+100+100)(1-$\mu$)
- EAT= (120*0.8)+(220*0.2)=140ns

# PAGE TABLE ENTRY

- Satu entri page table terdiri dari bit untuk:
  - Page Frame Number
  - Present/Absent bit (V) :
    - Bit = 0 → invalid, page tidak ada di memory (page fault)
    - Bit = 1 → valid, page berada di memory
  - Protection bit (Prot)
    - Bit = 0 → Read/Write Page
    - Bit = 1 → Read Only Page
  - Modified bit (M)
    - Bit= 0 → page belum pernah dimodifikasi
    - Bit= 1 → page sudah pernah dimodifikasi
  - Referenced bit (R)
    - Bit= 0 → page tidak digunakan
    - Bit= 1 → page sedang digunakan (direferensi)



# BIT VALID (V) OR INVALID (I)




Valid page : 0,1,2,3,4,5

Invalid page : 6,7

# PAGE YANG DIGUNAKAN BERSAMA (SHARED PAGES)

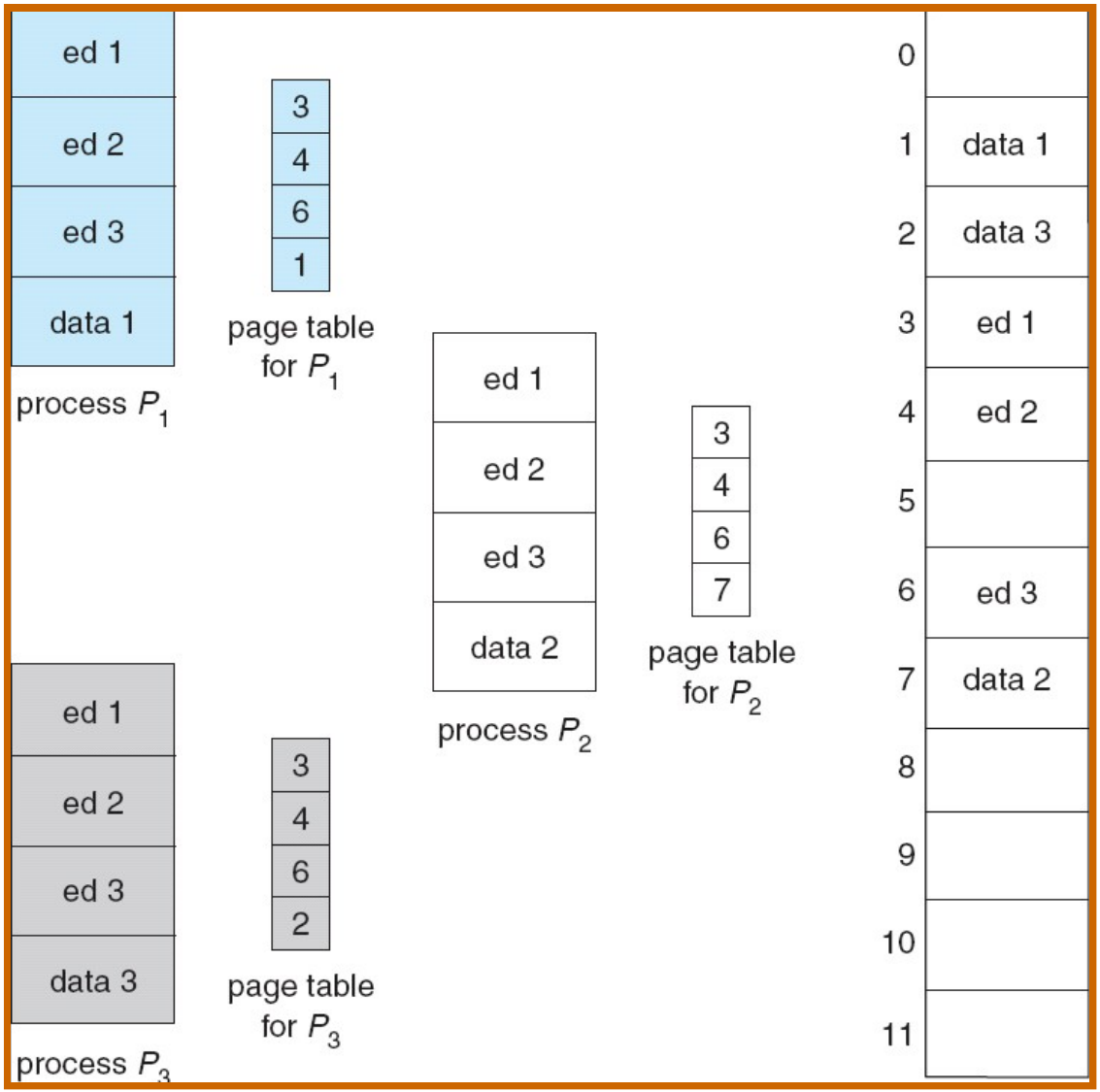
- **Shared code**
  - Kode program yang bersifat *read-only* dapat digunakan secara bersama oleh lebih dari satu proses. Contohnya: aplikasi editor teks, compiler, windows system
  - Lawan kata dari shared code → private code
- **Eksekusi dengan private code**
  - 1 user ➔150 KiB (program) + 50 KiB (data)

    ➔200KiB memory
  - 20 user ➔ 200KiB x 20 =4000KiB
- **Eksekusi dengan shared code**
  - 1 user ➔150 KiB (program) + 50 KiB (data)

    ➔200KiB memory

  20 user ➔150KiB + (50KiB x 20) = 1150KiB

# SHARED PAGES



# LATIHAN

- Dari alamat logik berikut, carilah nomor page dan offset untuk page yang berukuran 4KB.
  (a) 20000
  (b) 32768
  (c) 60000

- Sebuah mesin mempunyai ruang alamat logik sebesar 48 bit dan ruang alamat fisik 32 bit, dengan ukuran page 8KiB. Berapa jumlah entri yang ada pada sebuah *page table*?

# LATIHAN

- Jika memori logik dari sebuah proses terdiri dari 8 page dengan ukuran 1 page = 512 byte dan memori fisik terdiri dari 16 frame.
  - Berapa besar ruang alamat logik?
  - Berapa besar ruang alamat fisik?
  - Dari tabel dibawah ini, berapakah alamat fisik dari alamat logik yang mempunyai nomor page 4 dan offset 241?

| Nomor Page | Nomor Frame |
|---|---|
| 0 | 0 |
| 1 | 8 |
| 2 | 11 |
| 3 | 6 |
| 4 | 3 |
| 5 | 1 |
| 6 | 7 |
| 7 | 10 |

# LATIHAN

Diketahui sebuah sistem menggunakan mekanisme paging,

(a) Jika akses ke memori 200 nanoseconds, berapa Effective Access Time (EAT) dari akses sebuah page?

(b) Jika sistem menggunakan TLB dan 75% akses page ditemukan di TLB, berapa EATnya jika akses ke TLB adalah 10 ns

# B. STRUKTUR *PAGE TABLE*

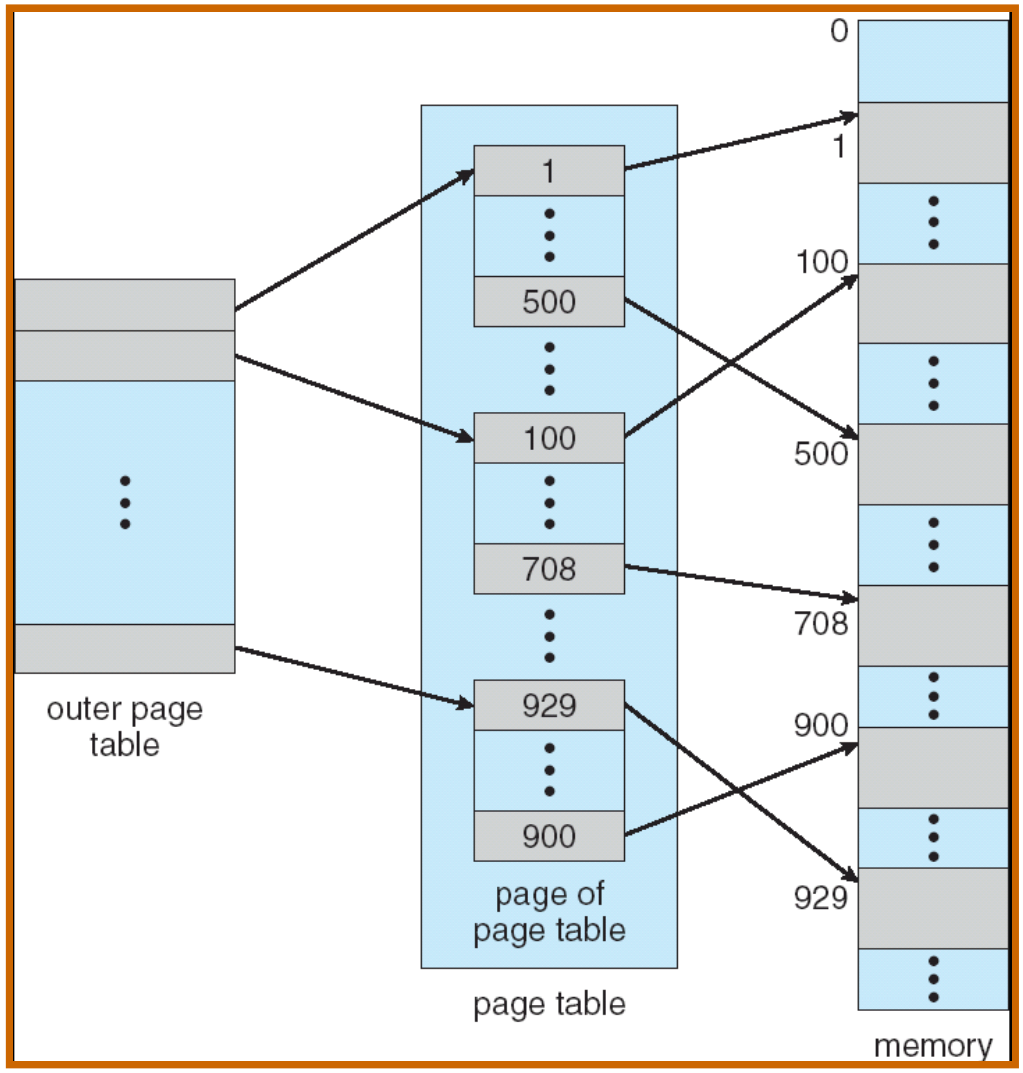
# TUJUAN PEMBELAJARAN

- Memahami teknik paging dengan  menggunakan
  - multi-level page table
  - hashed page table
  - inverted page table

# HIERARCHICAL PAGE TABLES

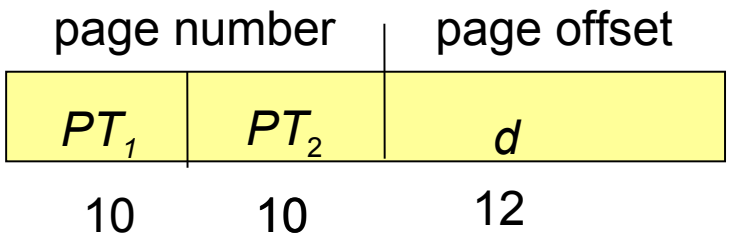
- Diketahui:
  - Ruang alamat logik = $2^{32}$ alamat
  - Ukuran page = $2^{12}$ byte
  - Ukuran 1 PTE(Page Table Entry) = 4 byte
- Berapa kapasitas Page Table untuk 1 proses?
  - Jumlah entri Page Table= $2^{32}/2^{12}=2^{20}=1048576$ entry.
  - Besar $2^{20}$ entri Page Tabel jika 1 entri @4 byte= $2^{20}$ x 4 byte= $2^{22}$ byte (4MB) →terlalu besar untuk dimuat ke memory.
- Solusinya ? Memecah page table menjadi lebih kecil

# SKEMA PAGE-TABLE DUA TINGKAT (PENTIUM)

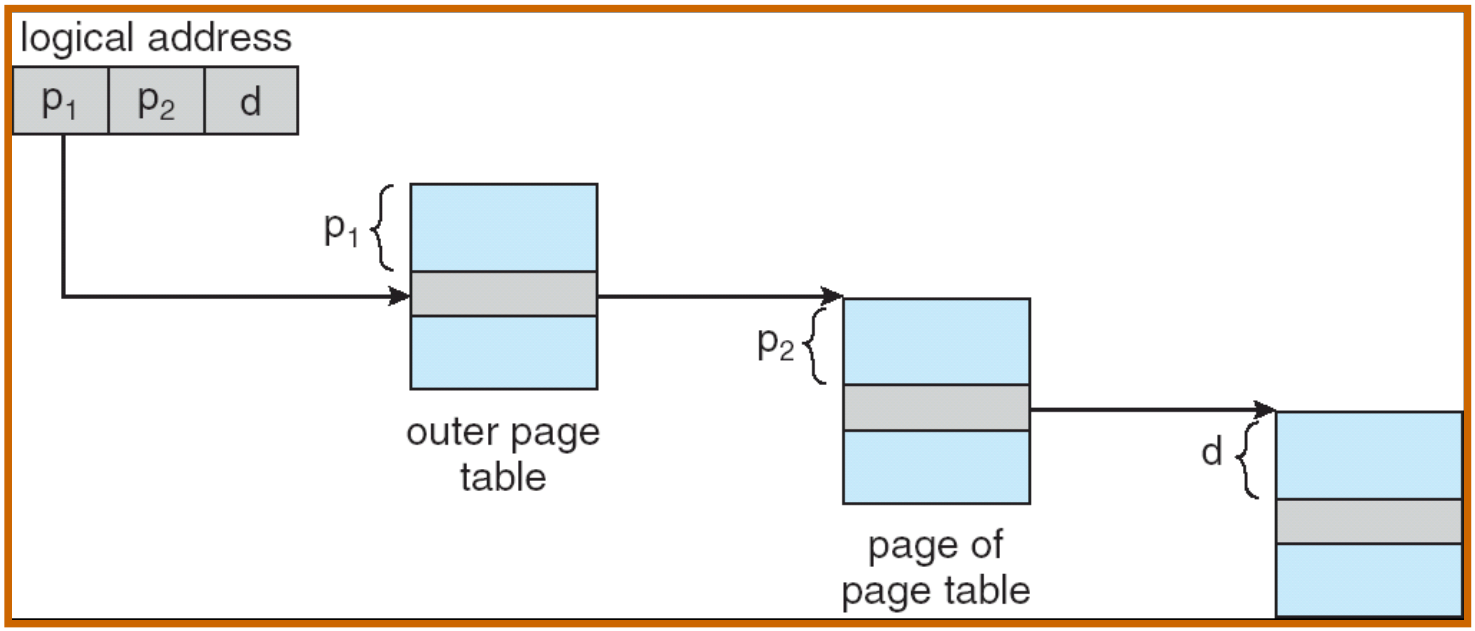


# PAGE TABLE DUA TINGKAT

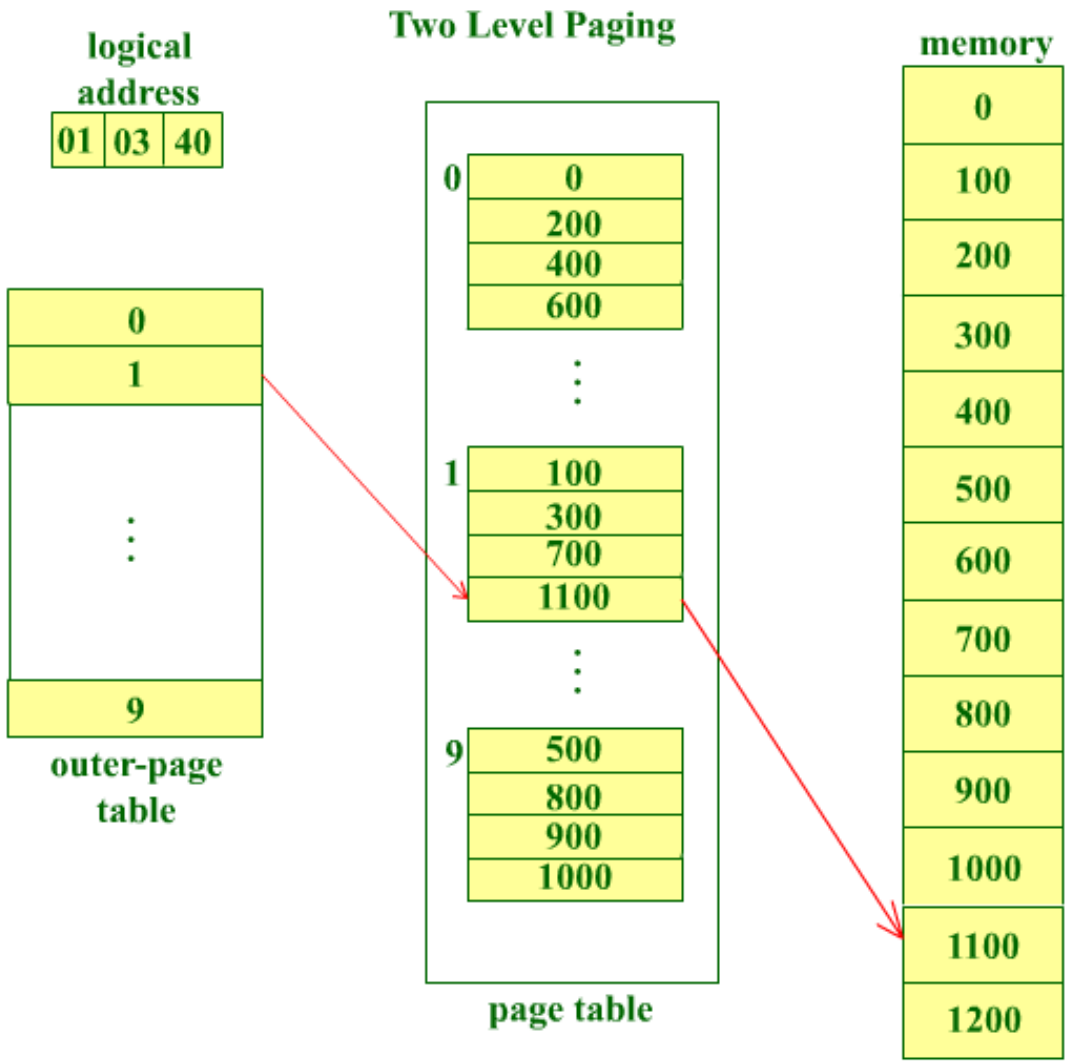
- Ruang alamat logik = $2^{32}$ alamat, ukuran page = $2^{12}$ byte,
  - page number terdiri dari 20 bit
  - page offset terdiri dari 12 bit
- Karena page table juga dipaging, page number dibagi menjadi
  - 10 bit page number
  - 10 bit page offset
- $PT_1$=Outer Page; $PT_2$=Page of Page Table



# SKEMA PENERJEMAHAN ALAMAT

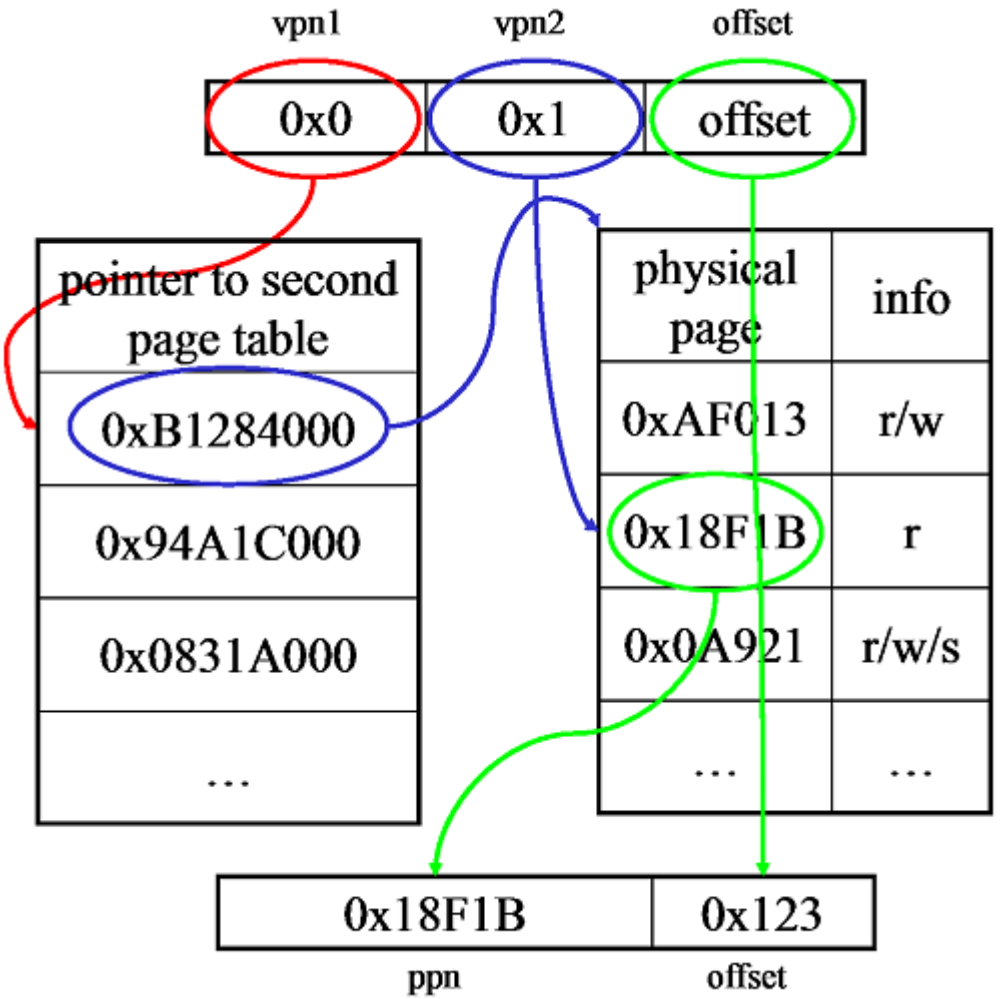


# TRANSLASI ALAMAT (DALAM DESIMAL)

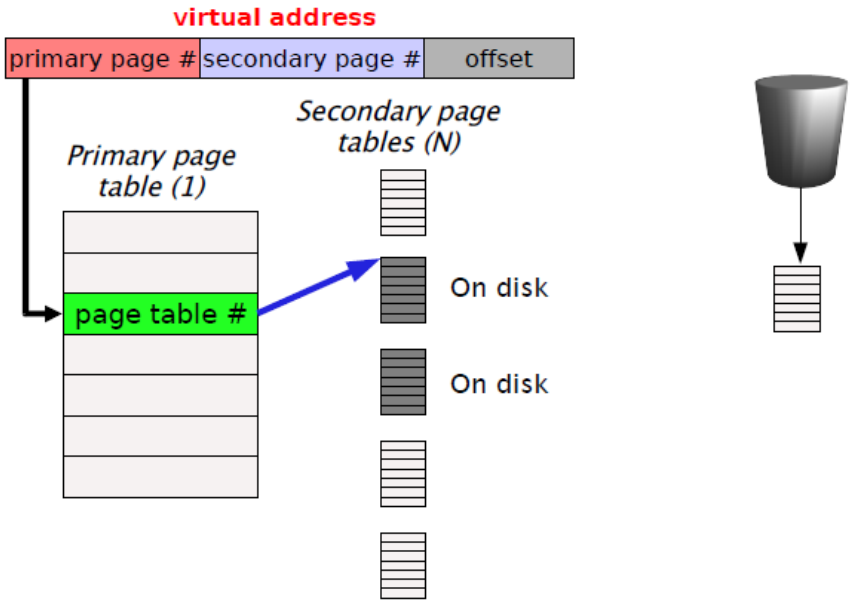


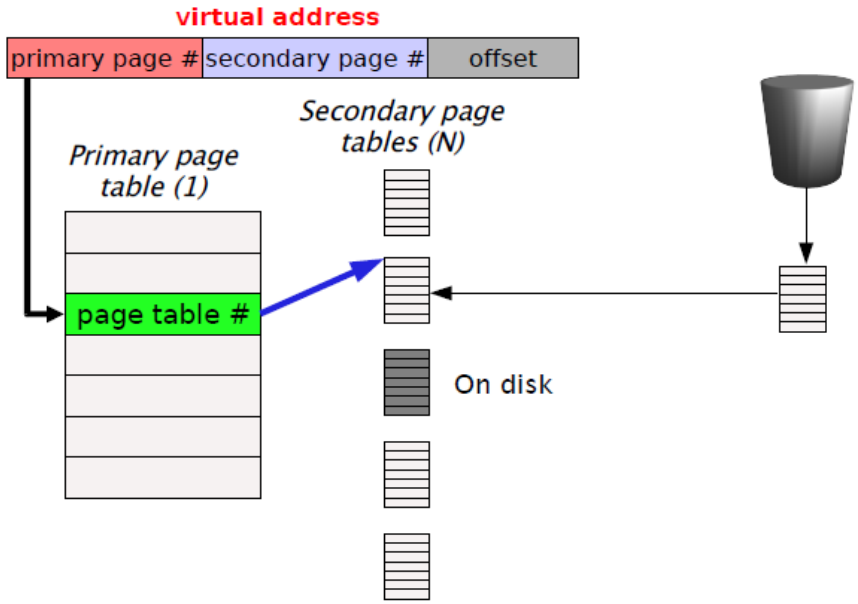

For example, if the CPU produces logical address 010340, it will reference outer page 01, which points to page 1 in the page table. Then we access offset 03 within the page to obtain frame number 1100. We then use the offset to produce physical memory address 1140.

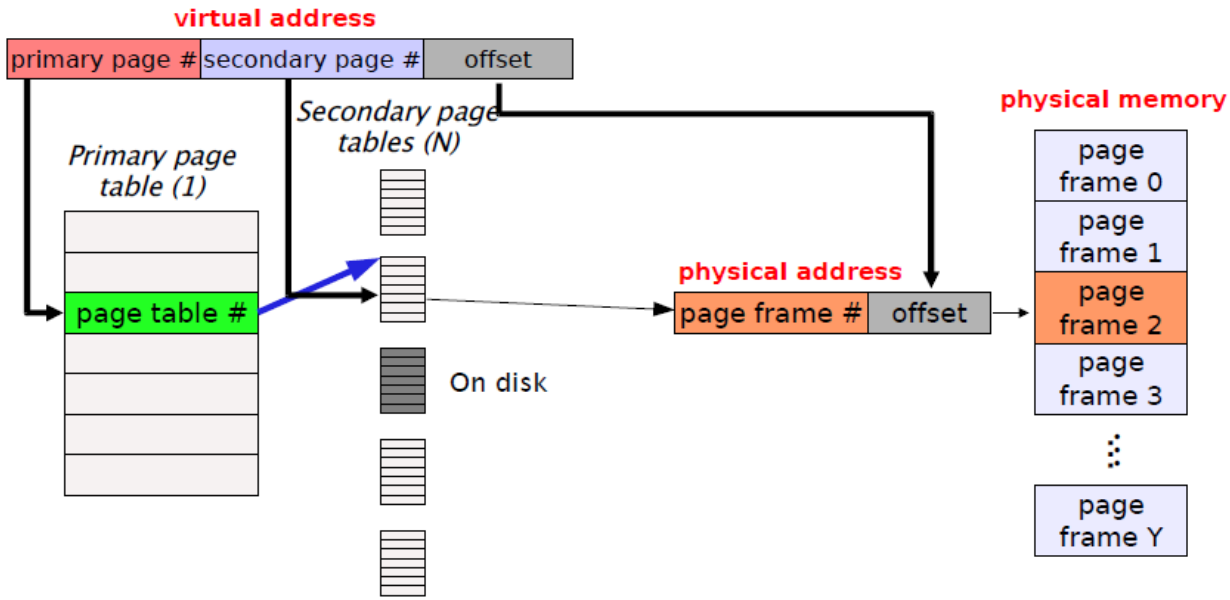
# TRANSLASI ALAMAT (DALAM HEXADESIMAL)



# ILUSTRASI TRANSLASI MULTILEVEL PAGE



# SKEMA PAGING 2 & 3 TINGKAT

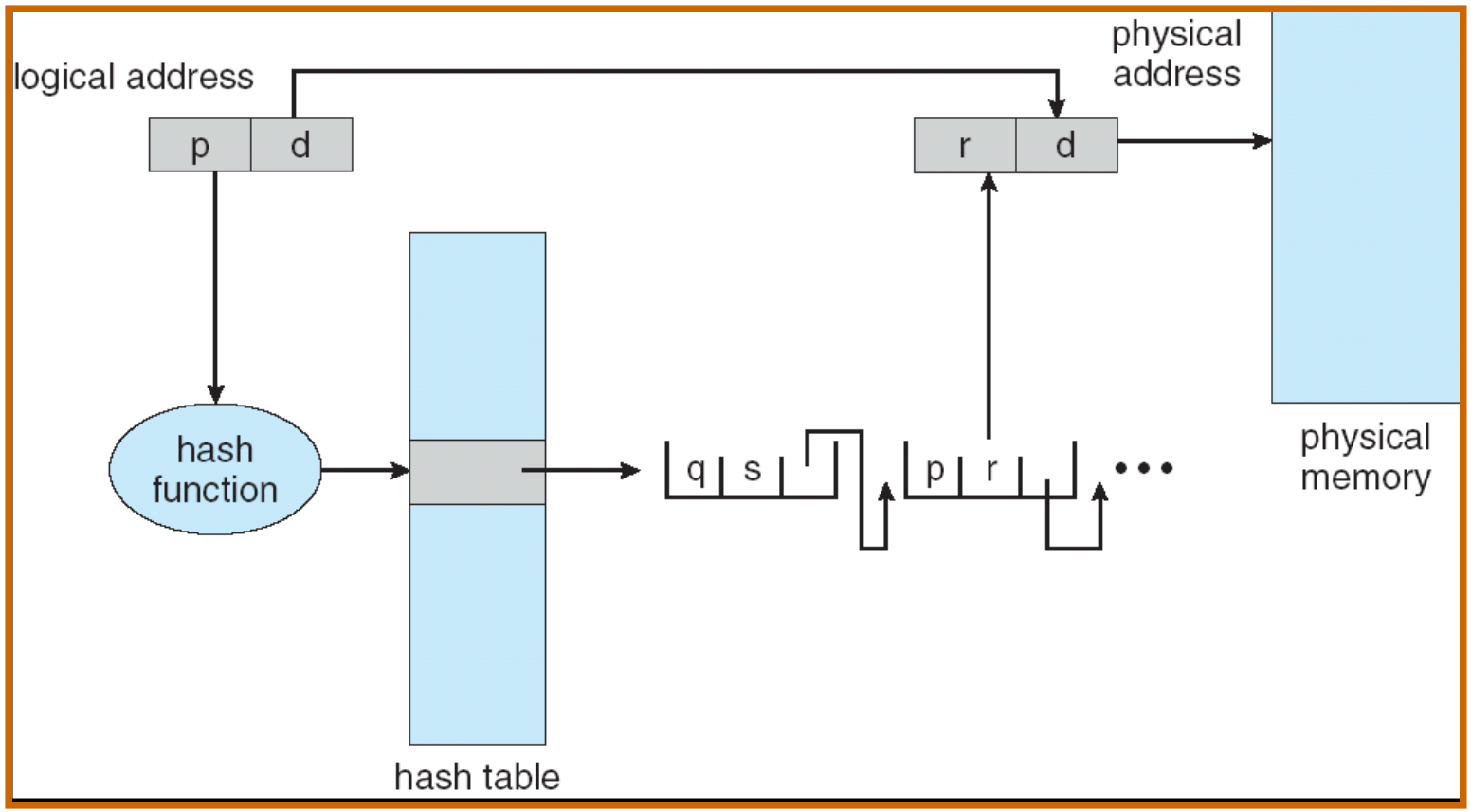
| outer page | inner page | offset |
|---|---|---|
| $p_1$ | $p_2$ | $d$ |
| 42 | 10 | 12 |

| 2nd outer page | outer page | inner page | offset |
|---|---|---|---|
| $p_1$ | $p_2$ | $p_3$ | $d$ |
| 32 | 10 | 10 | 12 |

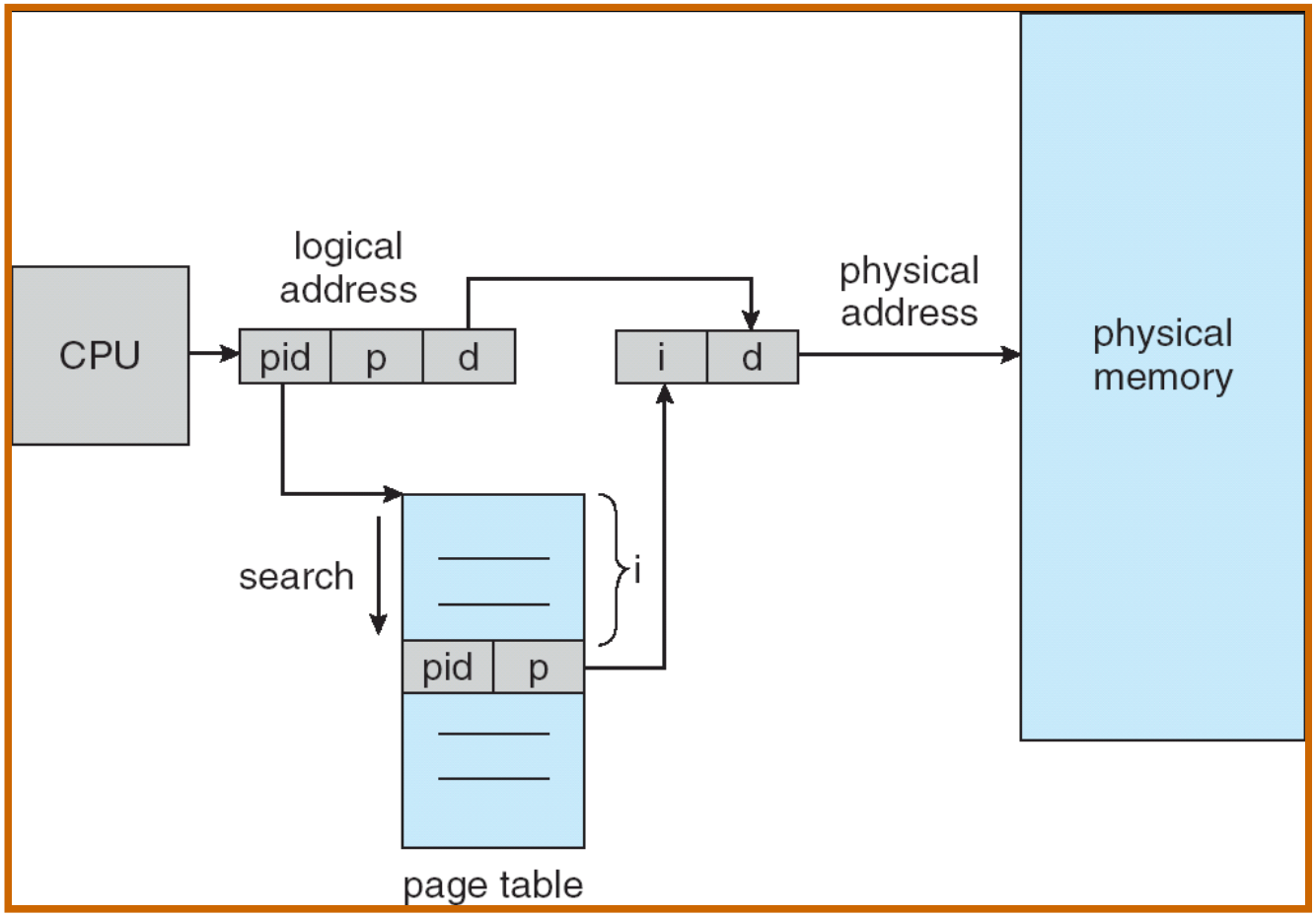
# STRUKTUR PAGE TABLE

- Hierarchical Paging
  - Paging the page table
  - 2 Level (Pentium)
  - 3 Level (SPARC)
  - 4 Level (Motorola 68030)
  - Kurang baik untuk arsitektur 64 bit
- Hashed Page Tables
  - Dapat digunakan jika ruang alamat lebih dari 32 bit
- Inverted Page Tables
  - 64-bit UltraSPARC, Power PC, IA-64

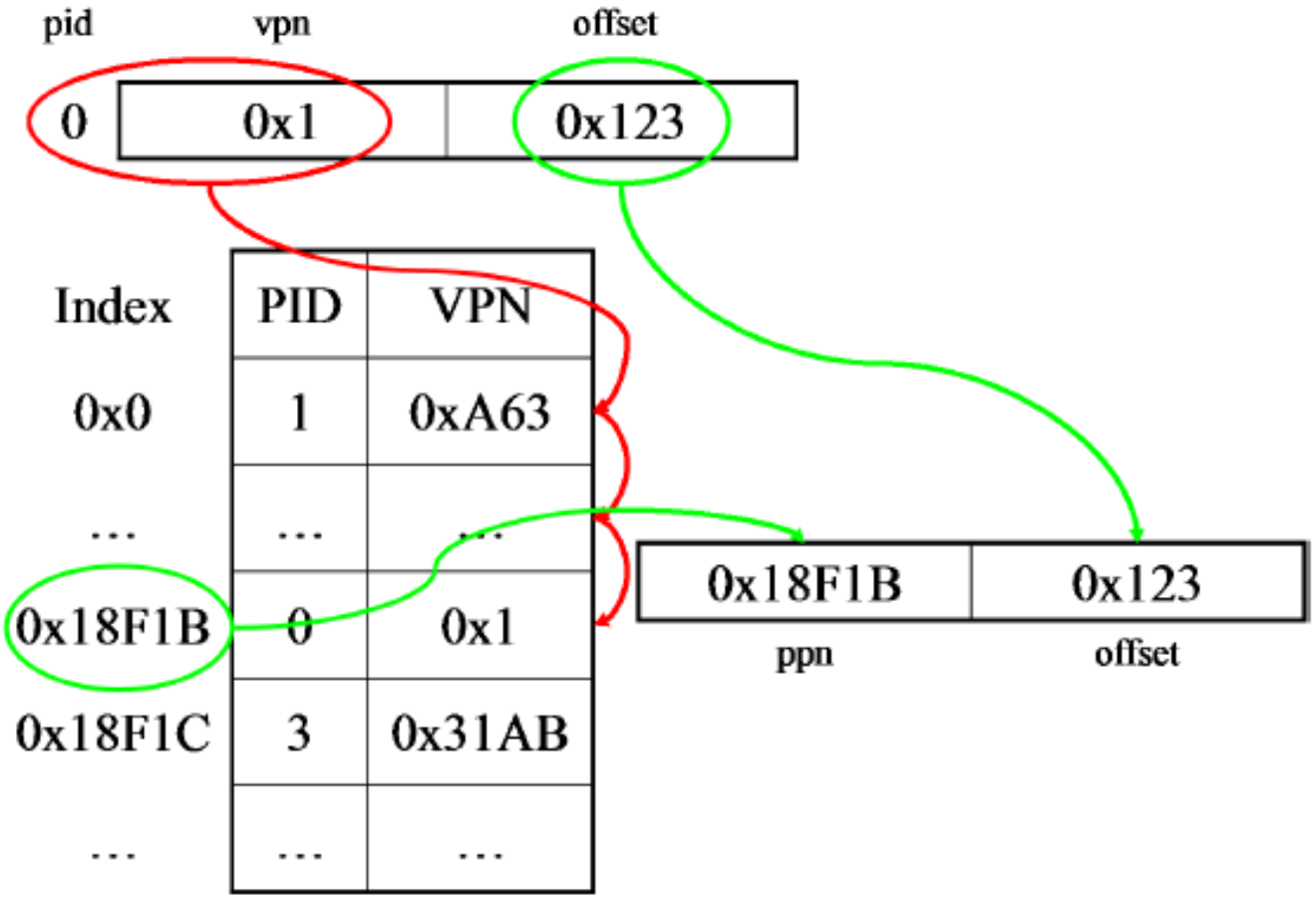
# HASHED PAGE TABLE



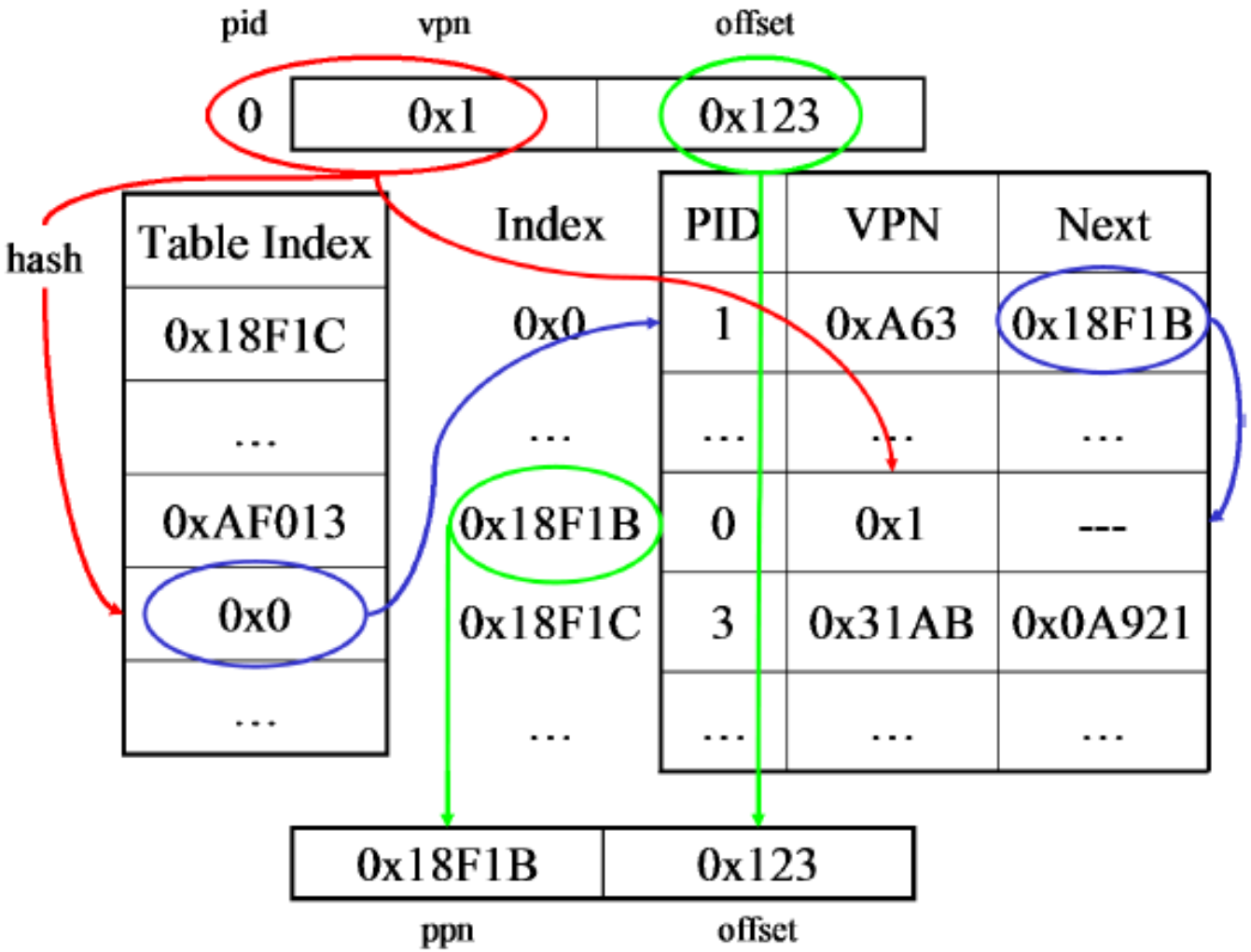
# ARSITEKTUR INVERTED PAGE TABLE



# INVERTED PAGE TABLE



# HASHED INVERTED PAGE TABLE



# LATIHAN

- Diketahui:
  - Ruang alamat logik = 32 Bit = $2^{32}$ alamat
  - Jumlah PTE Inverted Page Table (IPT) = 8
  - Ukuran page = 2MiB
  - Jumlah proses aktif = 3 dengan masing-masing PID = 1, 2 dan 3
- Pertanyaan:

a) Berapa besar memori fisik?
b) Berapa alamat fisik (dalam hexa) dari alamat logic 0x7fdd8f64 (PID 2)?
c) Berapa alamat logic dan PIDnya jika alamat fisik = 0x78e968 ?

| Valid | Process ID (PID) | Virtual Page Number |
|---|---|---|
| 1 | 1 | 0x3fe |
| 1 | 3 | 0x001 |
| 1 | 2 | 0x1ad |
| 1 | 3 | 0x7fd |
| 1 | 2 | 0x3fe |
| 1 | 1 | 0x2bf |
| 0 | 2 | 0x7fd |
| 1 | 2 | 0x0bf |

d) Jika IPT dikonversi ke Page table (PT), berapa total kapasitas PT untuk setiap proses yang aktif ? Asumsi ada tambahan 1 bit valid/invalid dan 1 bit modify/unmodified pada setiap entri PT

# JAWABAN

a) Besar memori fisik = jumlah entri IPT x ukuran 1 page = 8 page x 2MiB = 16MiB

b) Total bit untuk alamat logik= 32bit
Ukuran page = 2MiB → 21 bit (untuk offset)
Bit untuk nomor page = 32-21= 11bit
Alamat Logic = 7fdd8f64 = 0111 1111 1101 1101 1000 1111 0110 0100
No Page = 0111 1111 110 =0x3fe, berada di entri ke 4, dengan nomor frame = 100
Alamat Fisik = 1001 1101 1000 1111 0110 0100 = 0x9d8f64

c) Alamat Fisik = 0x78e968= 0111 1000 1110 1001 0110 1000, nomor entri frame = 011=3
Nomor page pada entri 3 = 0x7fd, dengan PID =3
Alamat logic = 1111 1111 1101 1000 1110 1001 0110 1000 = 0xffd8e968

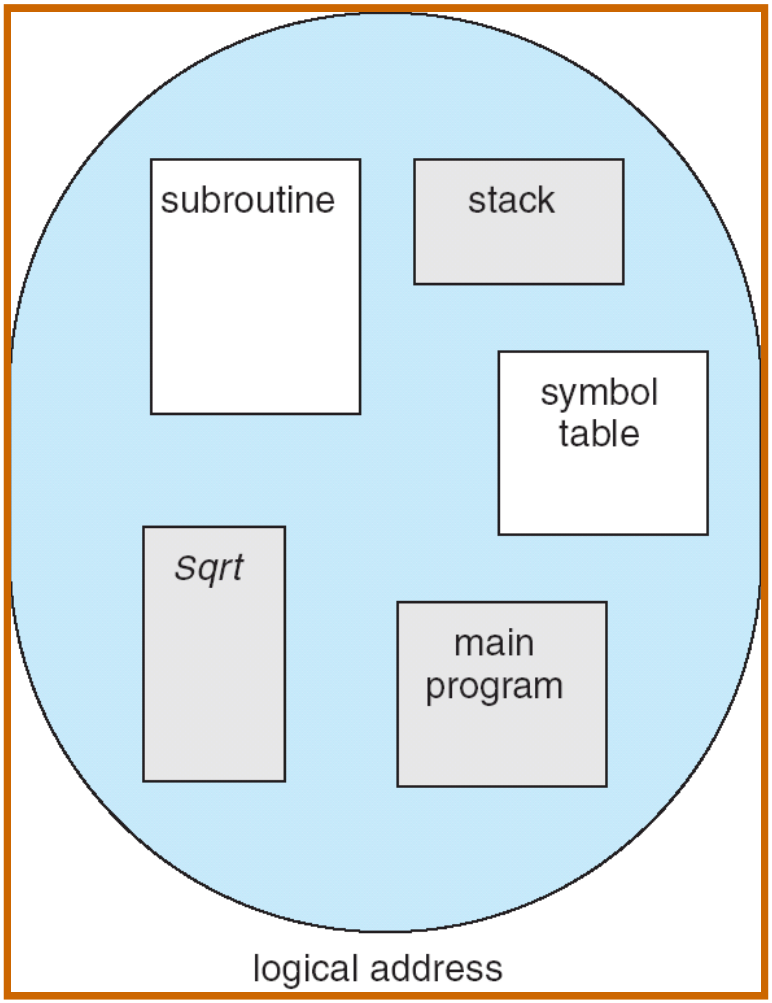
d) Total kapasitas PT = ukuran PT @proses x 3 proses
Ukuran PT @proses = jumlah entri PT x ukuran 1 entri PT
Jumlah entri PT = 32 – 21 = 11 bit = $2^{11}$ entri
Ukuran 1 entri PT = 3 bit untuk penomoran frame + 1 bit valid/invalid +1 bit modify/unmodified bit = 5 bit
Ukuran PT @proses= $2^{11}$ entri x 5 bit (ukuran 1 entri)
Total kapasitas PT = $2^{11}$ x 5 x 3 proses = 30720 bit = 3,75KiB

# C. SEGMENTASI DAN PAGING PADA INTEL PENTIUM

# TUJUAN PEMBELAJARAN

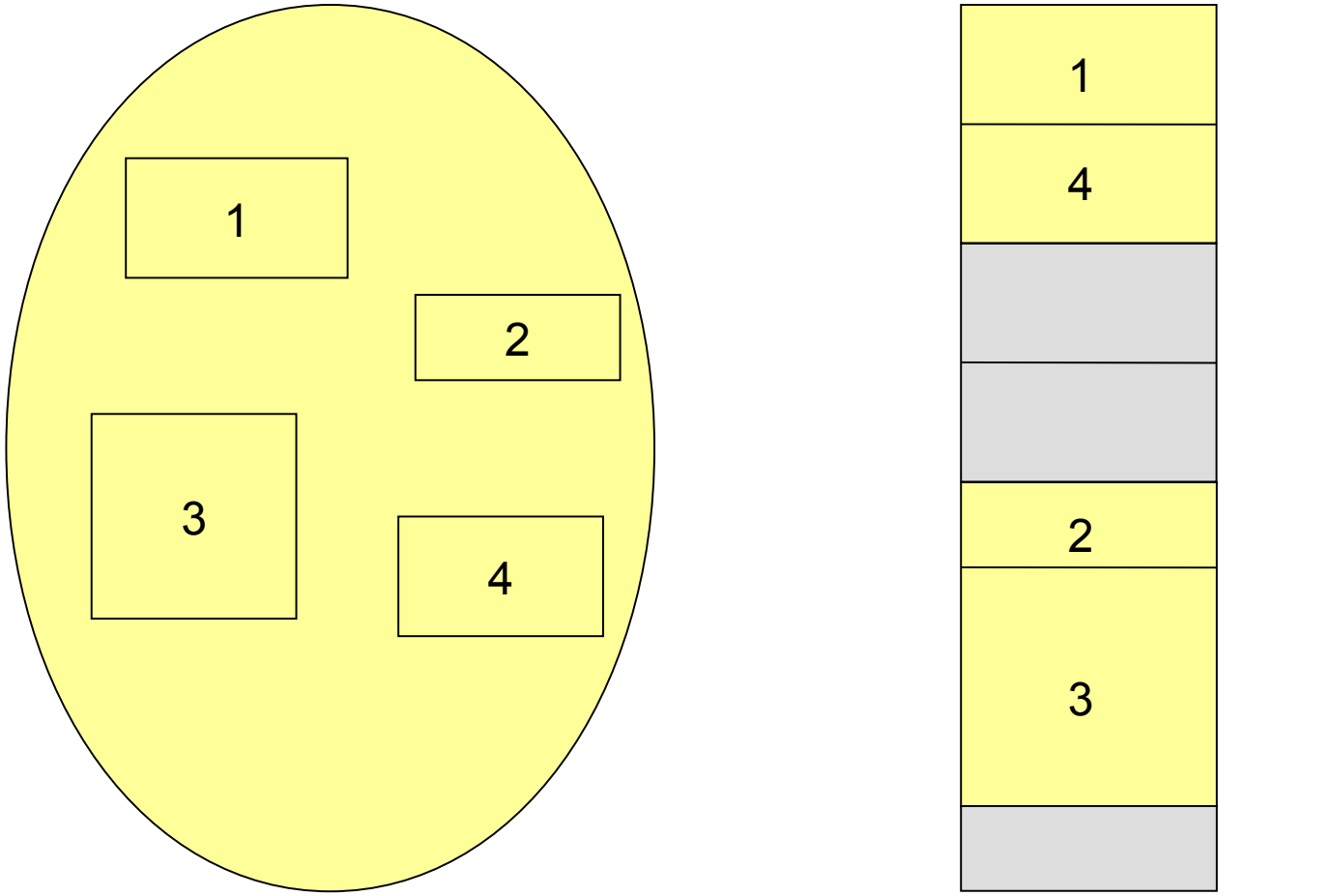
- Memahami konsep segmentasi
- Memahami implementasi segmentasi dan paging pada mesin intel pentium

# SEGMENTASI




Tampilan memori dari sisi user

# SEGMENTASI




Deskripsi logis segmentasi

# SEGMENTASI

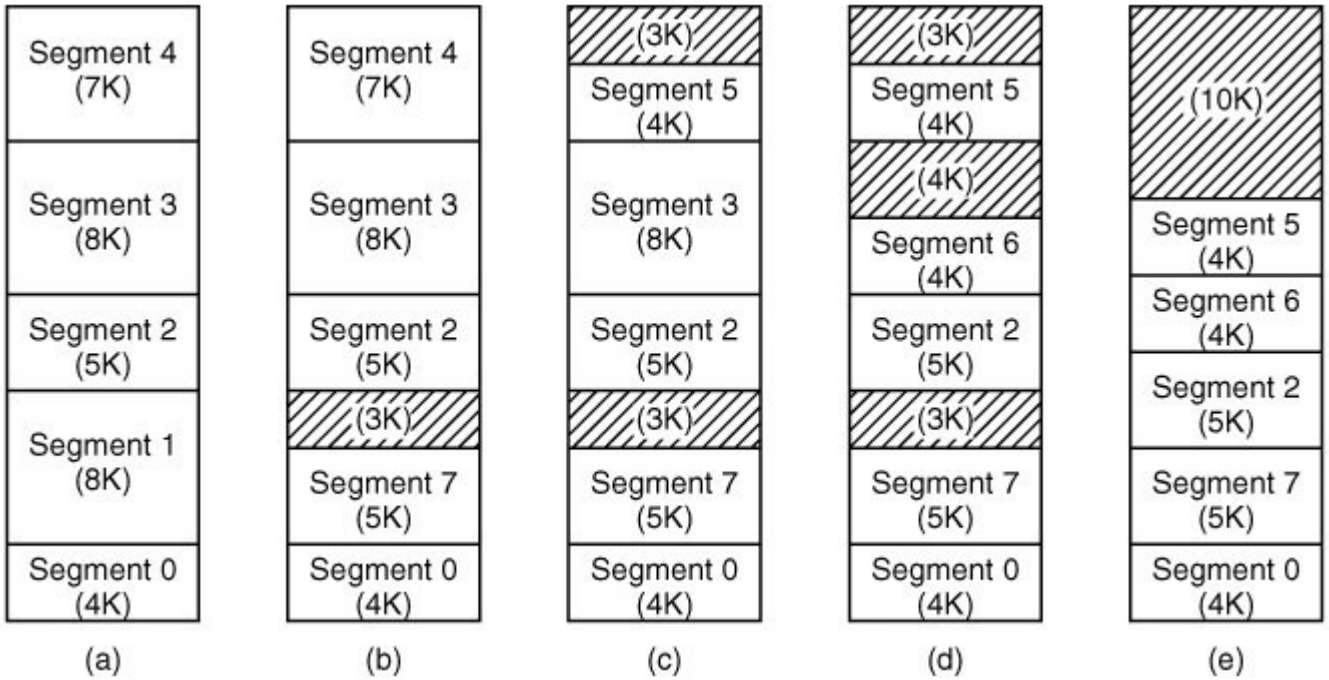
- Sebuah program dipecah menjadi beberapa segmen.
- Setiap segmen dapat berisikan  prosedur, fungsi, stack, symbol table, array, dan lain-lain.
- Karakteristik :
  - 1 proses = n blok segmen
  - besar 1 blok segmen tidak tetap

# SEGMENTASI

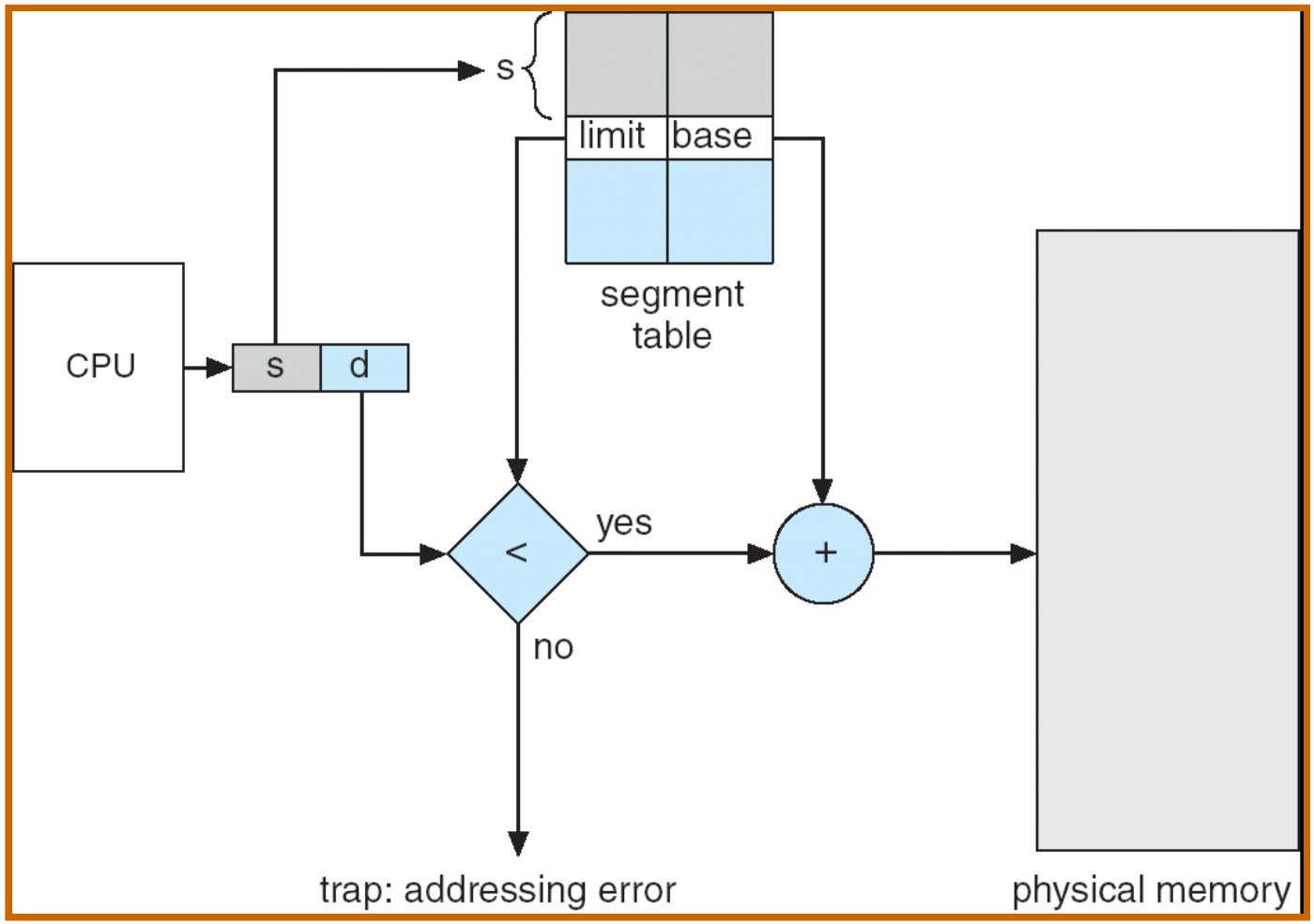
- Kekurangan ➔ Fragmentasi eksternal
- Pengalamatan:
  - Logical address terdiri dari pasangan:

    <nomor segmen, *offset*>
- Pemetaan alamat logik (<nomor-segmen, offset>) ke alamat fisik menggunakan *segment table*
- **Segment table** :
  - **base** – alamat awal segmen di memori fisik.
  - **limit** – panjang segmen

# ILUSTRASI FRAGMENTASI EKSTERNAL PADA SEGMENTASI

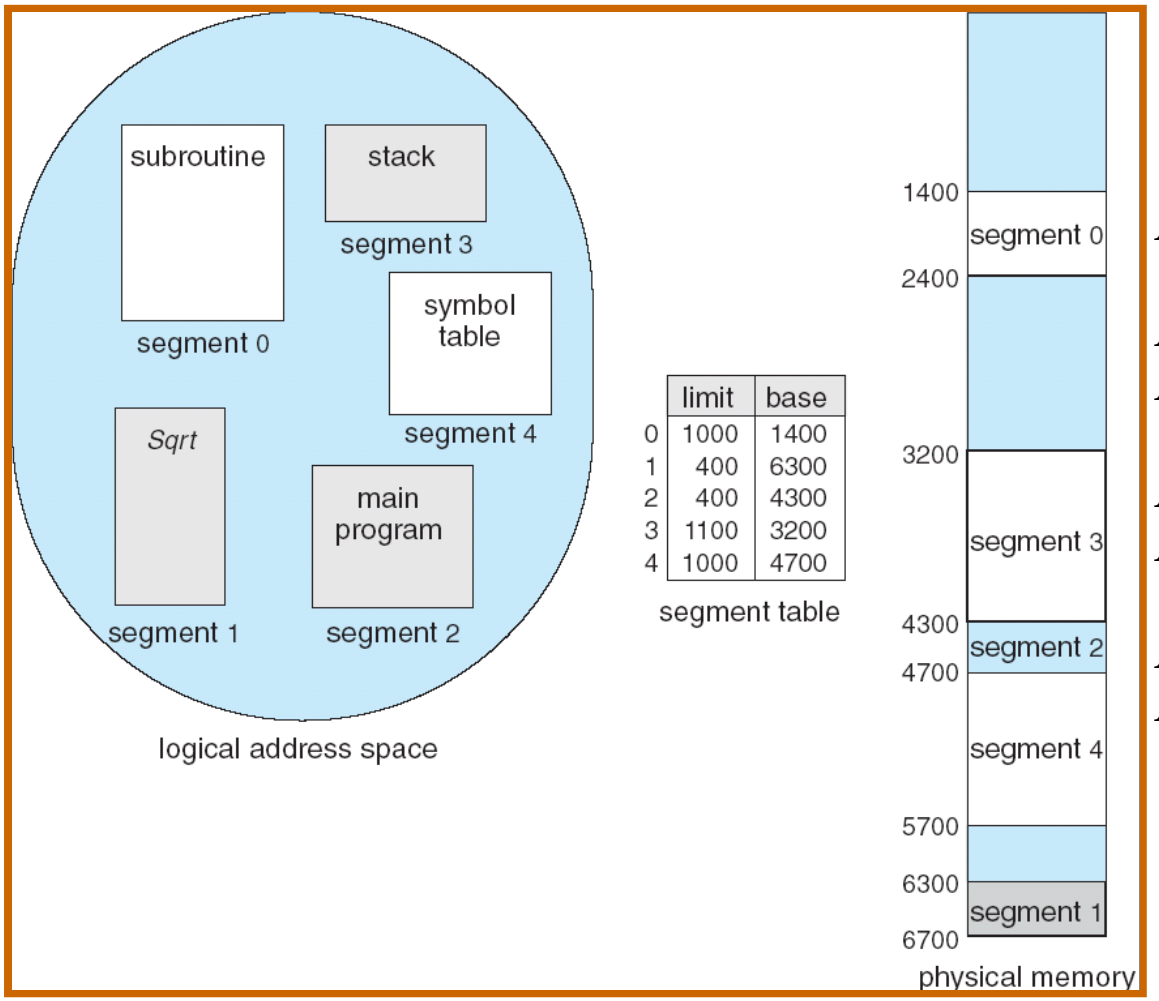


# PEMETAAN ALAMAT PADA SEGMENTASI




alur segmentasi

# SEGMENTASI




Alamat logika = <no segmen, offset>

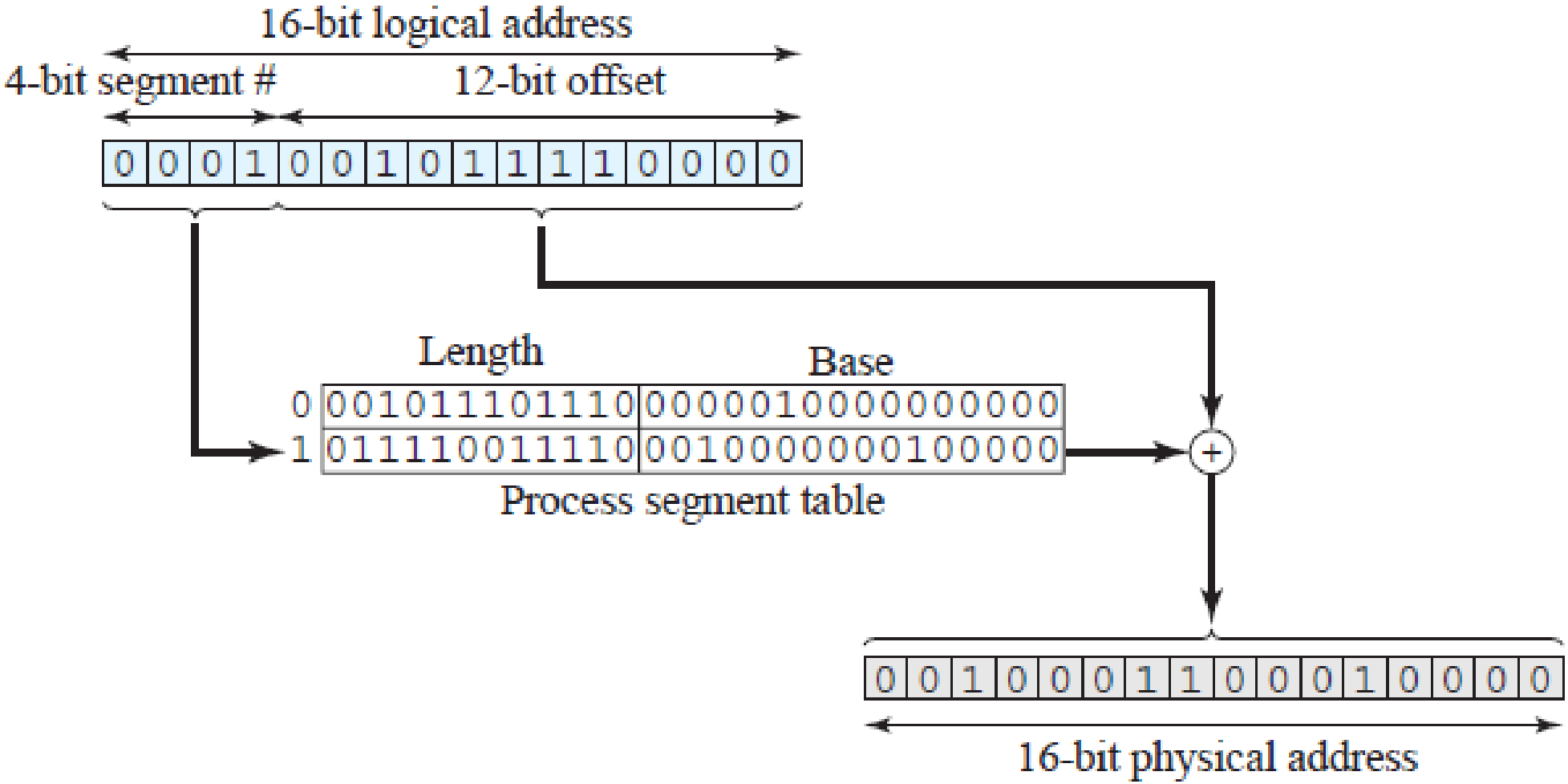
Alamat logika = <2,53>
Alamat Fisik = 4300+53 =4353

Alamat logika = <3,852>
Alamat Fisik = 3200+852 = 4052

Alamat logika = <0,1222>
Alamat Fisik = trapped !

contoh pemetaan dengan segmentasi

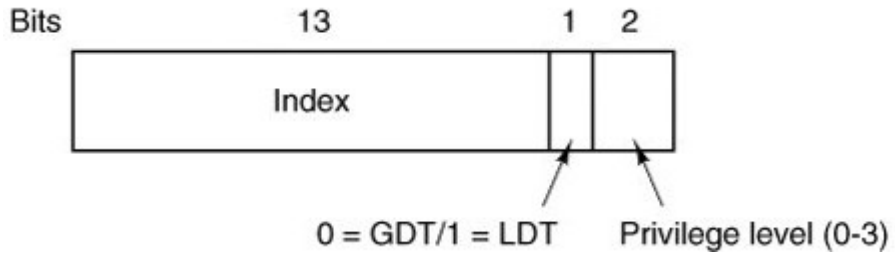
# CONTOH TRANSLASI ALAMAT



# INTEL PENTIUM

- Mendukung segmentasi, paging, dan segmentasi dengan paging
  - Windows XP & UNIX → Paging
- Jumlah maksimum segmen yang dapat dimiliki sebuah proses adalah 16384 segmen
- Ruang alamat logik maksimum untuk masing-masing segmen = 4GiB ($2^{32}$ bytes)
- Ruang alamat logik Intel Pentium dibagi menjadi dua partisi
  - Partisi pertama terdiri dari 8192 segment (private)
    - Tabel segmen → LDT (Local Descriptor Table)
  - Partisi kedua terdiri dari 8192 segment (shared)
    - Tabel segmen → GDT(Global Descriptor Table)

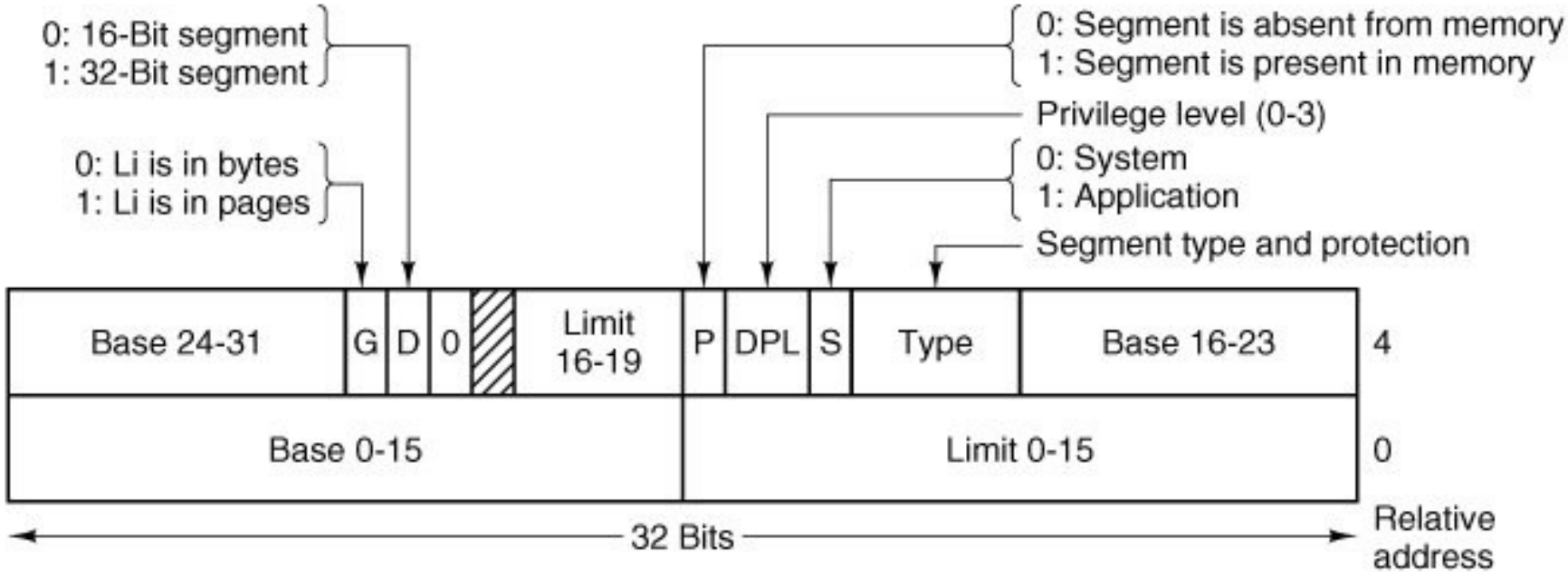
# ALUR TRANSLASI ALAMAT PADA INTEL PENTIUM



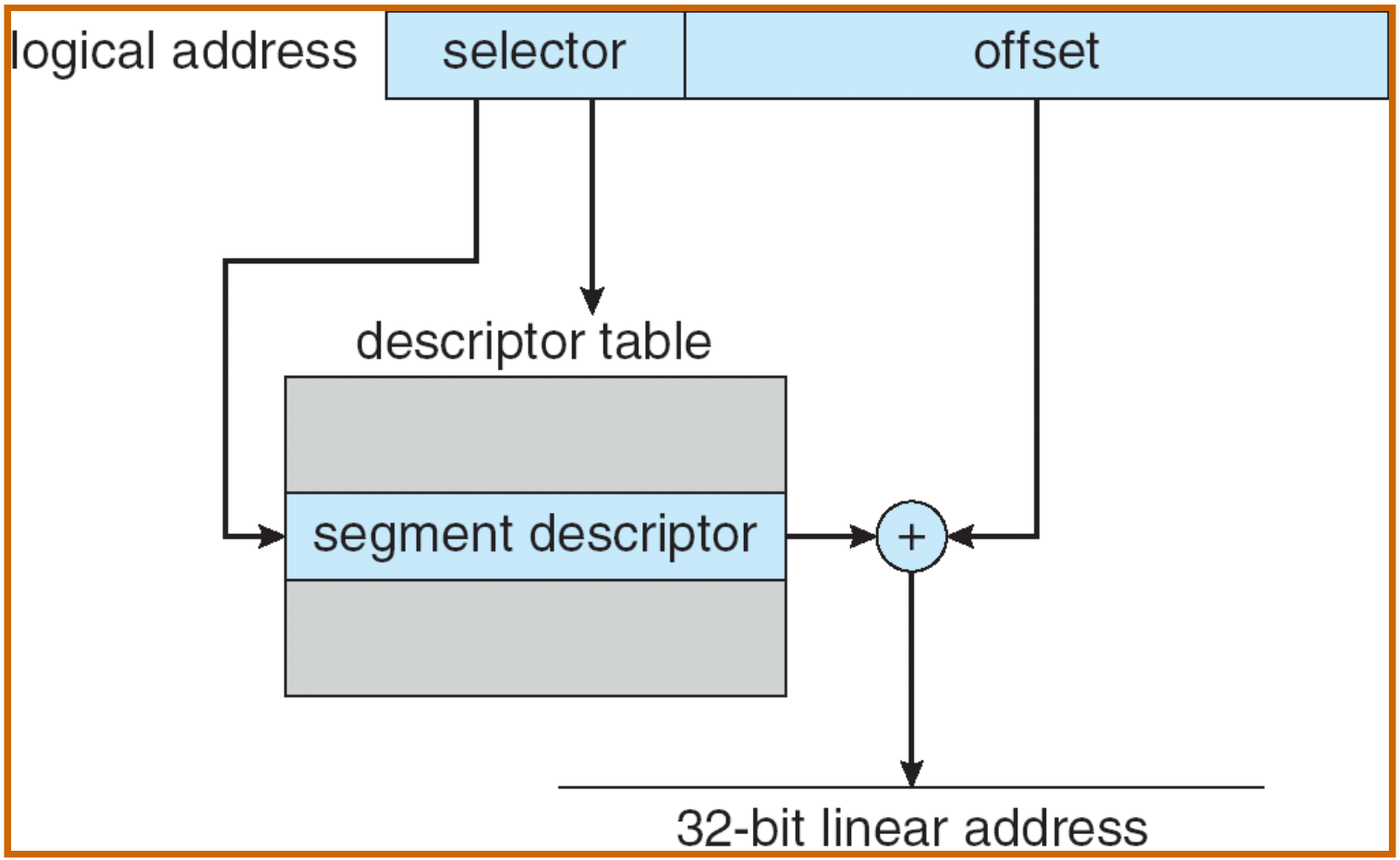

Bagan bit selector

- Alamat logik terdiri dari gabungan <selector, offset>
  - Selector (16 bit)
  - Offset (32 bit)
- Pada tabel segmen LDT dan GDT, terdapat sejumlah segment desriptor, dengan besar satu *segment descriptor* adalah 8 byte
- Informasi satu segmen disimpan pada satu *segment descriptor*. *Segment descriptor* berisikan *base location*, *base limit*, dan lainnya

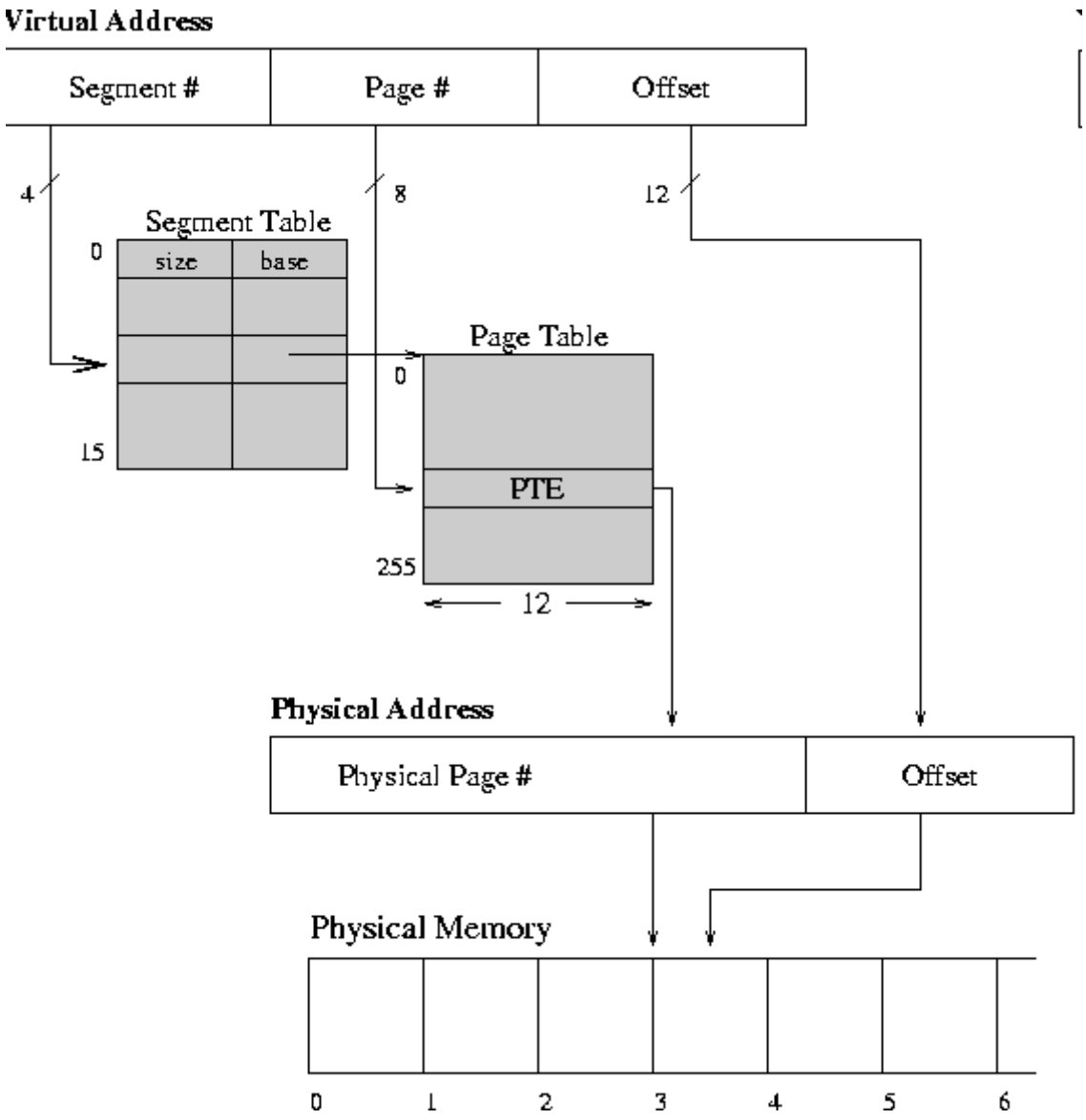
# SEGMENT DESCRIPTOR



# SEGMENTASI INTEL PENTIUM



# SEGMENTASI DENGAN PAGING

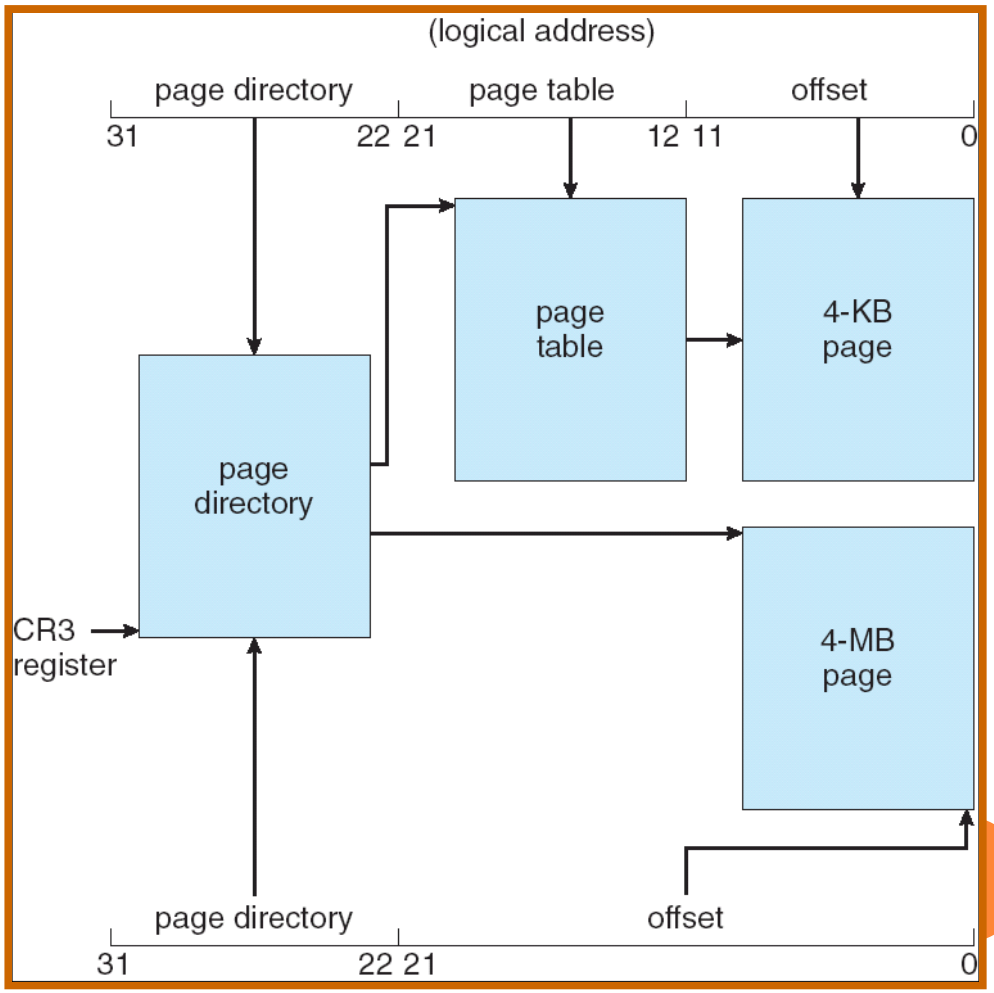


# LINUX PADA ARSITEKTUR PENTIUM

| page number | | page offset |
|---|---|---|
| $p_1$ | $p_2$ | $d$ |
| 10 | 10 | 12 |

Arsitektur Pentium memberikan pilihan ukuran page 4 KiB atau 4 MiB

Pentium dengan paging dua tingkat untuk page size 4KiB

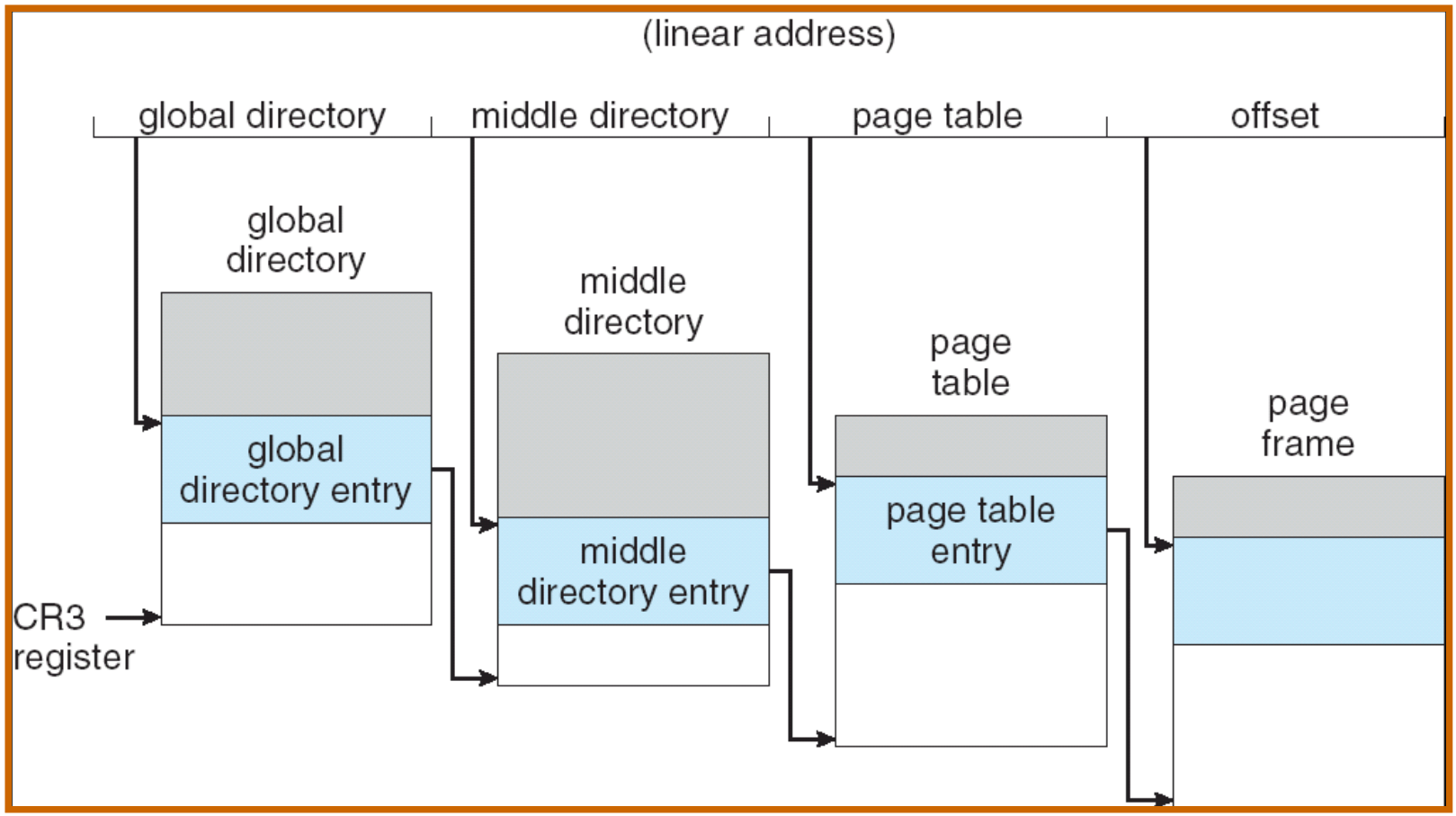


# LINUX PADA ARSITEKTUR PENTIUM

- Menggunakan 6 segmen : kernel code, kernel data, user code, user data, TSS (*taks state segment*) dan default LDT
- Menggunakan paging tiga tingkat (*three level paging*) untuk mesin 32 bit dan 64 bit
- Nilai bit untuk *middle directory*=0, jika menggunakan *two level paging* pada mesin pentium


Bagan page table tiga tingkat di linux

# LINUX PADA ARSITEKTUR PENTIUM




Paging tiga tingkat

# LATIHAN

- Jika terdapat tabel segmen sebagai berikut :

| Segment | Base | Length |
|---|---|---|
| 0 | 219 | 600 |
| 1 | 2300 | 14 |
| 2 | 90 | 100 |
| 3 | 1327 | 580 |
| 4 | 1952 | 96 |

Hitung alamat fisik dari alamat logika :

(a) 0,430

(b) 1,10

(c) 2,500

(d) 3,400

(e) 4,112

# JAWABAN

- (a) 219 + 430 = 649
- (b) 2300 + 10 = 2310
- (c) ilegal, ditrap oleh sistem operasi
- (d) 1327 + 400 = 1727
- (e) ilegal, ditrap oleh sistem operasi

Selesai