

# Sistem Operasi 2009

Pertemuan 7

## Pengelolaan Memory

H u s n i

Lab. Sistem Komputer & Jaringan  
Teknik Informatika Univ. Trunojoyo

# Ikhtisar

---

- Kebutuhan manajemen memory
- Pembagian (*partitioning*) memory
- Dasar manajemen memory modern
  - *Paging*
  - *Segmentation*

# Perlunya Manajemen Memory

---

- Kini memory murah, akan lebih murah
  - Tetapi aplikasi butuh lebih dan lebih banyak memory, tidak pernah cukup!
- Manajemen memory, mencakup pertukaran (*swapping*) blok data dari *secondary storage*.
- Memory I/O lambat dibandingkan CPU
  - SO harus secara cerdas mengukur waktu *swapping* untuk memaksimalkan efisiensi CPU

# Manajemen Memory

---

*Memory perlu dialokasikan untuk menjamin suatu perbekalan yang layak bagi proses yang siap memanfaatkan waktu processor yang tersedia.*

# Kebutuhan Manajemen Memory

---

- Relocation (Relokasi)
- Protection (Proteksi)
- Sharing (Berbagi-pakai)
- Logical organisation (Organisasi Logis)
- Physical organisation (Organisasi Fisik)

# Kebutuhan: Relokasi

---

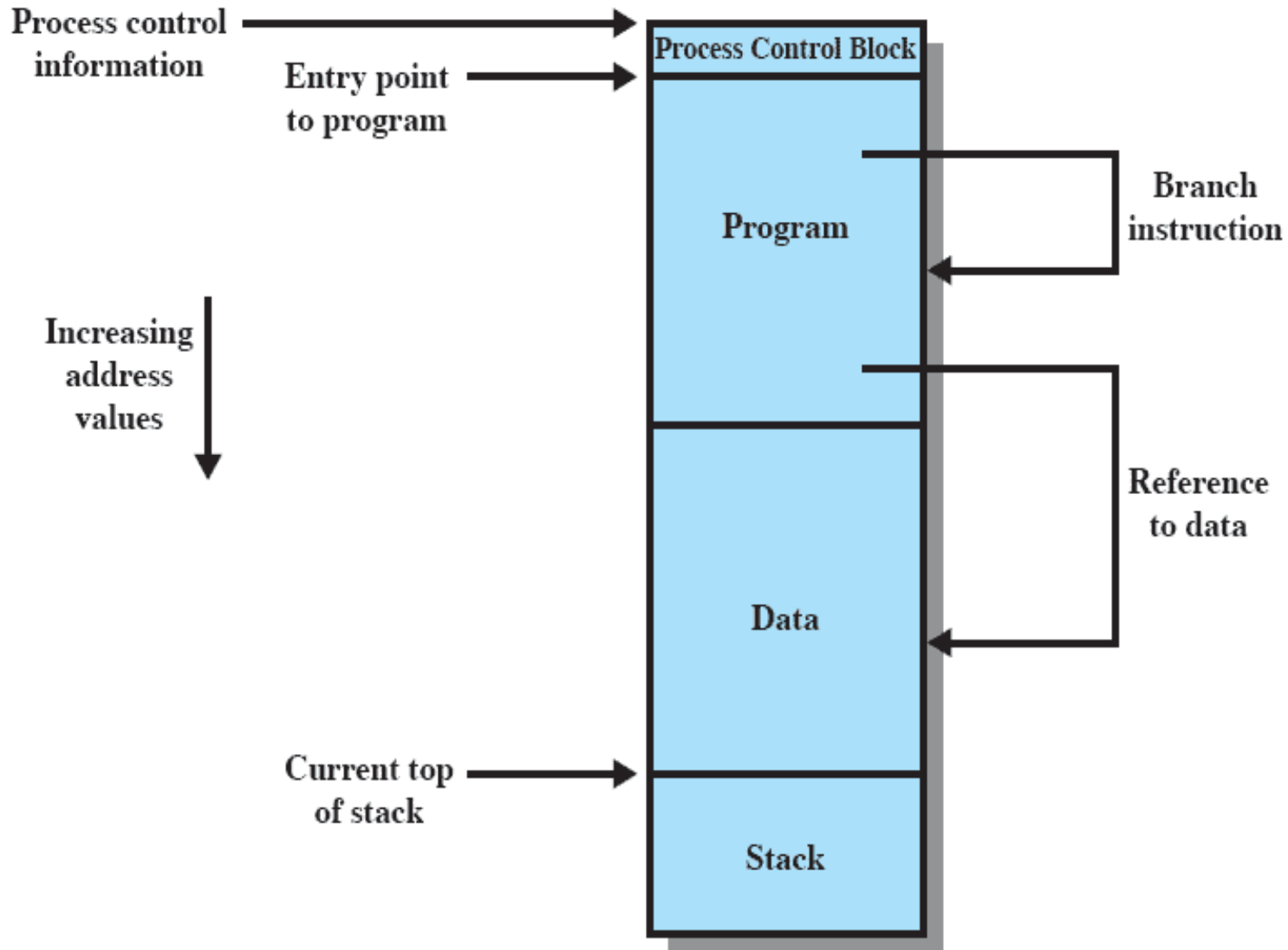
- Programmer tidak mengetahui dimana program akan ditempatkan di dalam memory ketika programnya dieksekusi,
  - Mungkin *diswap* ke disk dan dikembalikan ke memory utama pada lokasi yang berbeda (**direlokasi**)
- Referensi memory harus ditranslasikan ke alamat memory fisik aktual (sebenarnya)

# Istilah Penting

**Table 7.1 Memory Management Terms**

Istilah	Keterangan
Frame	Blok dari memory utama ( <i>Fixed-length</i> , panjang tetap)
Page	Blok data di dalam memory sekunder (disk) ( <i>Fixed-length</i> )

# Pengalamatan



# Kebutuhan: Proteksi

---

- Proses sebaiknya tidak dapat mengakses lokasi memory milik proses lain tanpa izin
- Tidak mungkin memeriksa alamat absolut saat kompilasi (*compile time*)
- Harus diperiksa pada saat program berjalan (*run time*)

# Kebutuhan: Bagi-Pakai

---

- Membolehkan beberapa proses mengakses bagian yang sama dari memory
- Lebih baik membolehkan setiap proses mengakses salinan (*copy*) yang sama dari program, bukan salinan terpisah miliknya sendiri.

# Kebutuhan: Organisasi Logis

---

- Memory diorganisasi secara linier (biasanya)
- Program ditulis dalam modul-modul
  - Modul-modul dapat ditulis dan *dcompile* secara lepas
- Derajat proteksi berbeda diberikan ke modul-modul (*read-only*, *execute-only*)
- Bagi-pakai modul antar proses
- *Segmentation* membantu di sini

# Kebutuhan: Organisasi Fisik

---

- Tidak dapat membiarkan programmer bertanggungjawab mengelola memory
- Memory tersedia bagi program + datanya mungkin tidak cukup
  - *Overlay* membolehkan berbagai modul untuk diberikan daerah yang sama dari memory tetapi ini *time consuming* untuk program
- Programmer tidak mengetahui berapa banyak ruang yang akan tersedia.

# Pemartisian

---

- Metode awal pengelolaan memory
  - Sebelum hadirnya *virtual memory*
  - Tidak banyak digunakan saat ini
- Tetapi, ini memudahkan pemahaman mengenai virtual memory nanti
  - Virtual memory dikembangkan dari metode pemartisian.

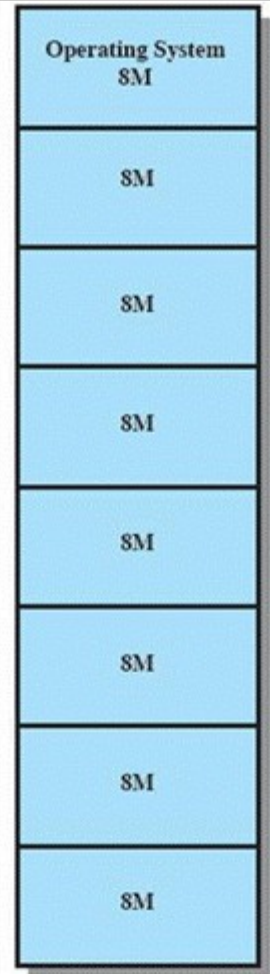
# Jenis Pemartisian

---

- Pemartisian Tetap (*fixed*)
- Pemartisian Dinamis (*dynamic*)
- Paging (Pengaturan Halaman) Sederhana
- Segmentasi Sederhana
- Paging Virtual Memory
- Segmentasi Virtual Memory

# Pemartisian Tetap

- Partisi-partisi berukuran sama
  - Proses yang berukuran kurang dari atau sama dengan ukuran partisi dapat dimuat ke dalam suatu partisi yang tersedia.
- Sistem operasi dapat *menswap* suatu proses keluar partisi
  - Jika tidak ada yang berada pada status *ready* atau *running*



(a) Equal-size partitions

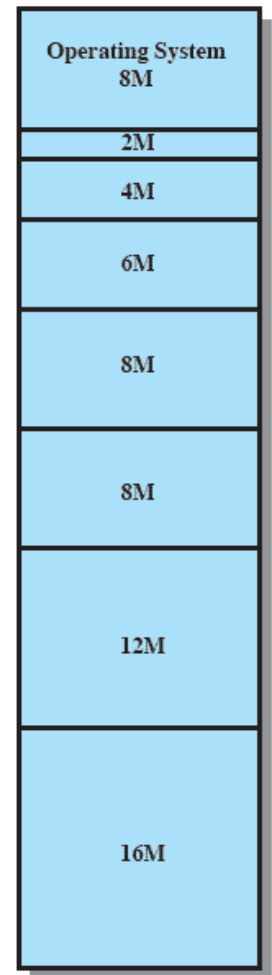
# Masalah Pemartisian Tetap

---

- Suatu program mungkin tidak muat pada suatu partisi.
  - Programmer harus merancang program dengan *overlay*.
- Pemanfaatan memory utama tidak efisien.
  - Suatu program, meskipun berukuran kecil, mendiami suatu partisi sendiri.
  - Ini mengakibatkan **fragmentasi *internal***.

# Solusi: Ukuran Partisi Tidak Sama

- Mengurangi kedua masalah
  - Tetapi tidak menyelesaikan total
- Perhatikan gambar
  - Program sampai dengan 16M dapat dimuatkan tanpa *overlay*
  - Program yang lebih kecil dapat ditempatkan dalam partisi yang lebih kecil, mengurangi fragmentasi internal



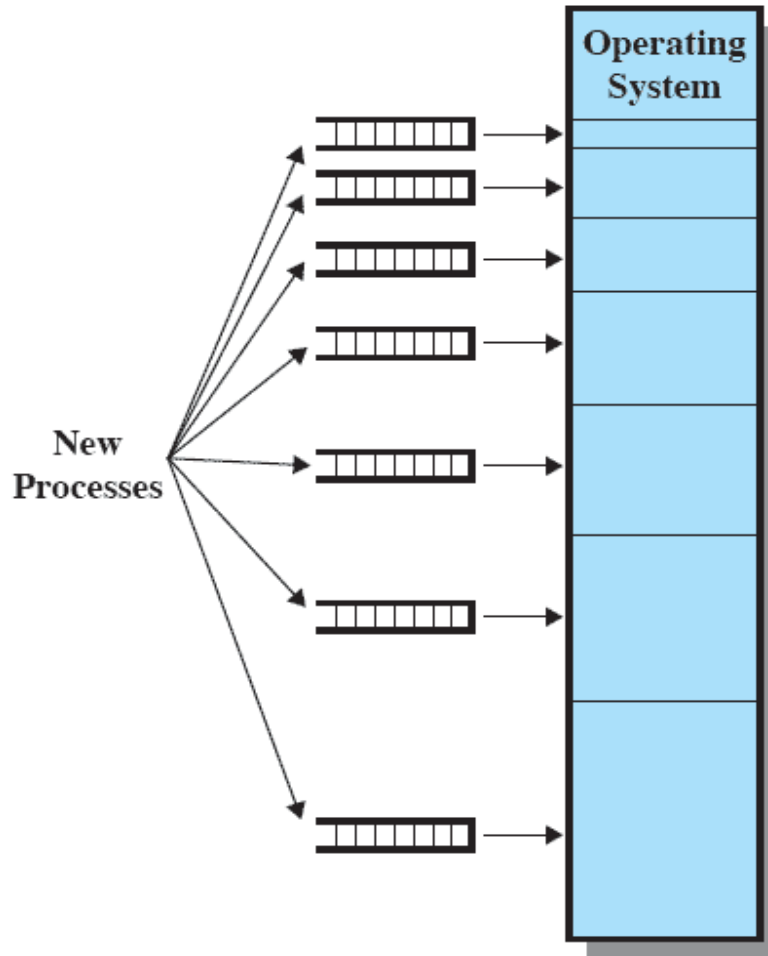
(b) Unequal-size partitions

# Algoritma Penempatan

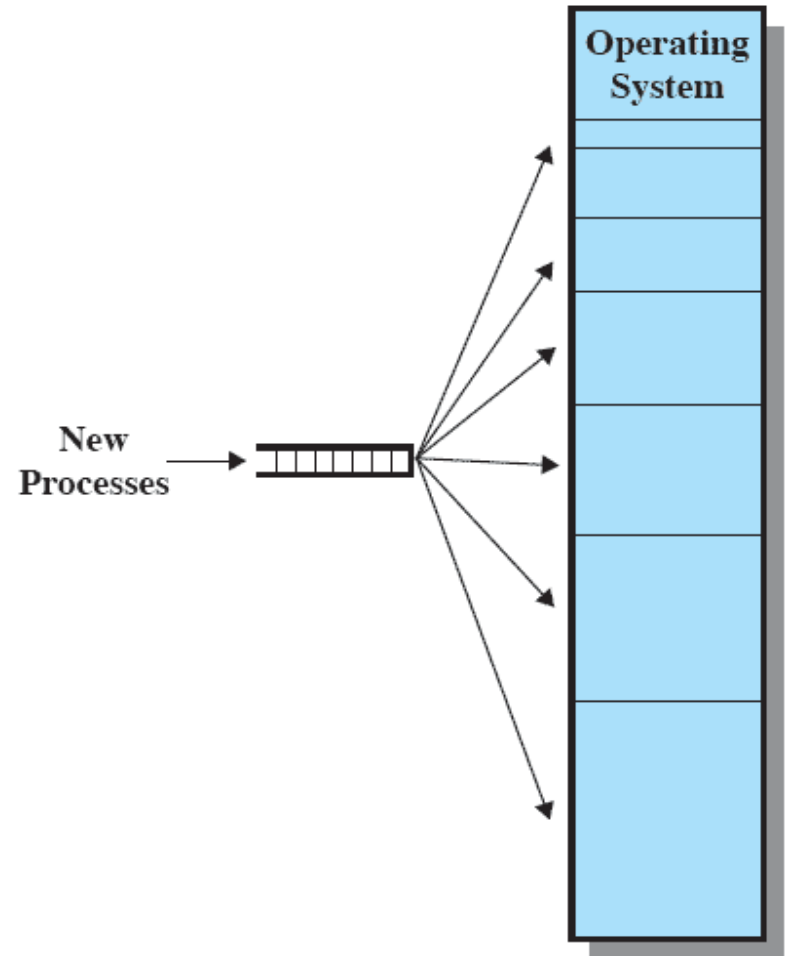
---

- Ukuran Sama
  - Penempatan *trivial* (tanpa pilihan)
- Ukuran Tidak Sama
  - Dapat memberikan setiap proses ke partisi yang paling kecil yang dapat memuatnya
  - *Queue* (antrian) bagi tiap partisi
  - Proses diberikan dengan suatu cara untuk meminimalkan memory terbuang dalam suatu partisi.

# Pemartisian Tetap



(a) One process queue per partition



(b) Single queue

# Masalah Pada Pemartisian Tetap

---

- Jumlah dari proses yang aktif dibatasi oleh sistem
  - Dibatasi oleh jumlah *pre-determined* dari partisi.
- Sejumlah besar dari proses sangat kecil akan menggunakan ruang secara tidak efisien.
  - Dalam metode partisi panjang tetap maupun variabel

# Pemartisian Dinamis (1)

---

- Panjang dan jumlah partisi bersifat variabel
- Proses dialokasikan ruang secara tepat (*exact*) sebanyak memory yang diperlukan

# Pemartisian Dinamis (2)



- ***Fragmentasi Eksternal***
- Dapat menggunakan ***compaction*** (pemadatan)
  - SO memindahkan proses-proses sehingga mereka berdampingan
  - *Time consuming* dan membuang waktu CPU

# Pemartisian Dinamis (3)

---

- SO harus memutuskan blok bebas mana untuk dialokasikan ke suatu proses
- Algoritma Best-fit
  - Pilih blok yang paling dekat dengan ukuran request tersebut
  - Secara keseluruhan, paling buruk tapi hebat
  - Karena blok terkecil ditemukan bagi proses, banyak terbentuk fragmentasi kecil
  - Pemadatan memory harus lebih sering dilakukan

# Pemartisian Dinamis (4)

---

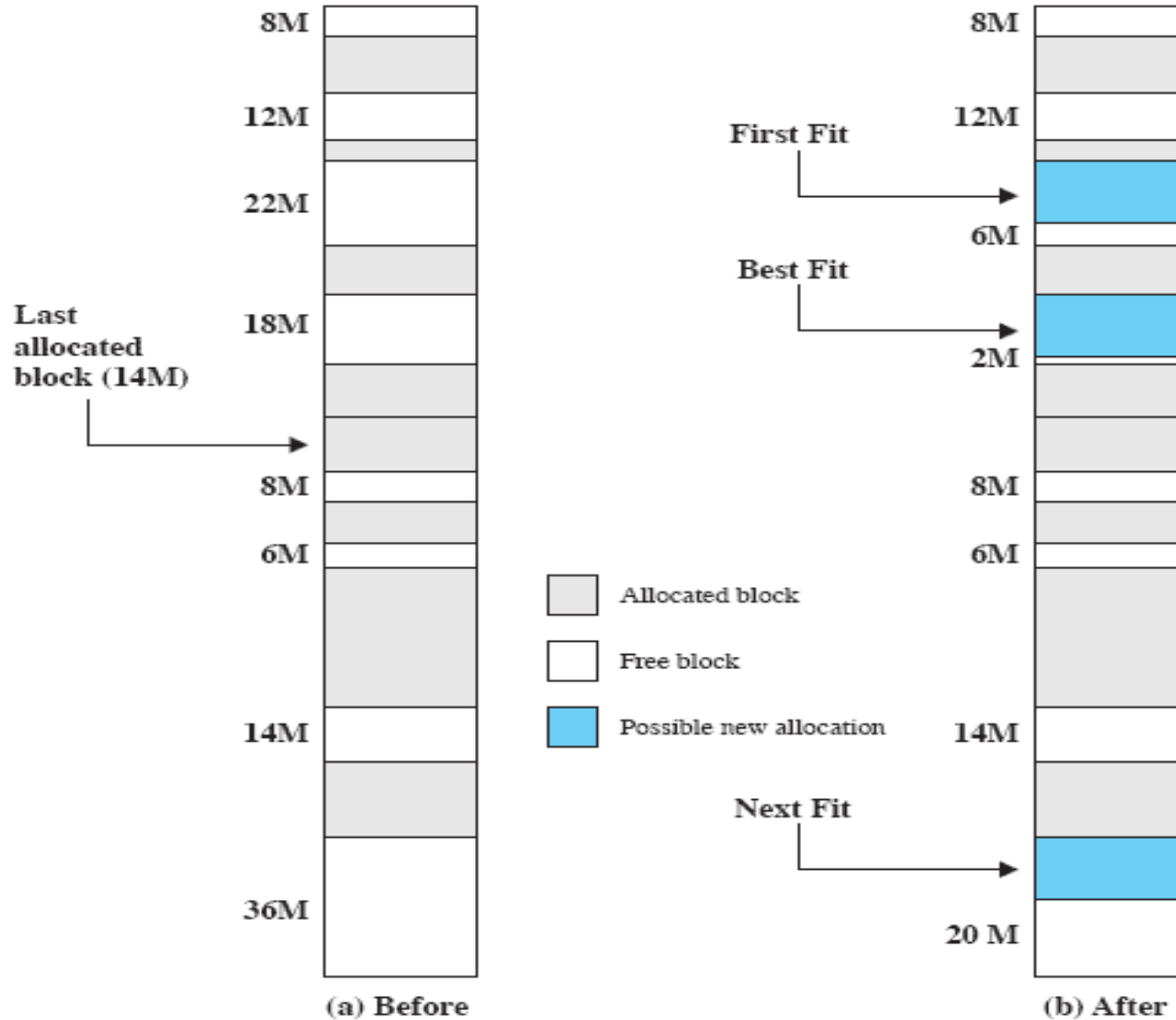
- Algoritma First-fit
  - *Scan* memory dari permulaan dan pilih blok pertama (tersedia, tidak ditempati) yang cukup besar
  - Paling cepat
  - Mungkin banyak proses dimuat pada bagian depan memory yang harus dilewati ketika mencari suatu blok bebas

# Pemartisian Dinamis (5)

---

- Algoritma Next-fit
  - *Scan* memory dari lokasi penempatan terakhir
  - Lebih sering alokasi suatu blok memory pada ujung dari memory dimana blok terbesar ditemukan
  - Blok terbesar dari memory dipecah ke dalam blok-blok yang lebih kecil
  - *Compaction* diperlukan untuk memperoleh suatu blok besar pada ujung memory

# Alokasi

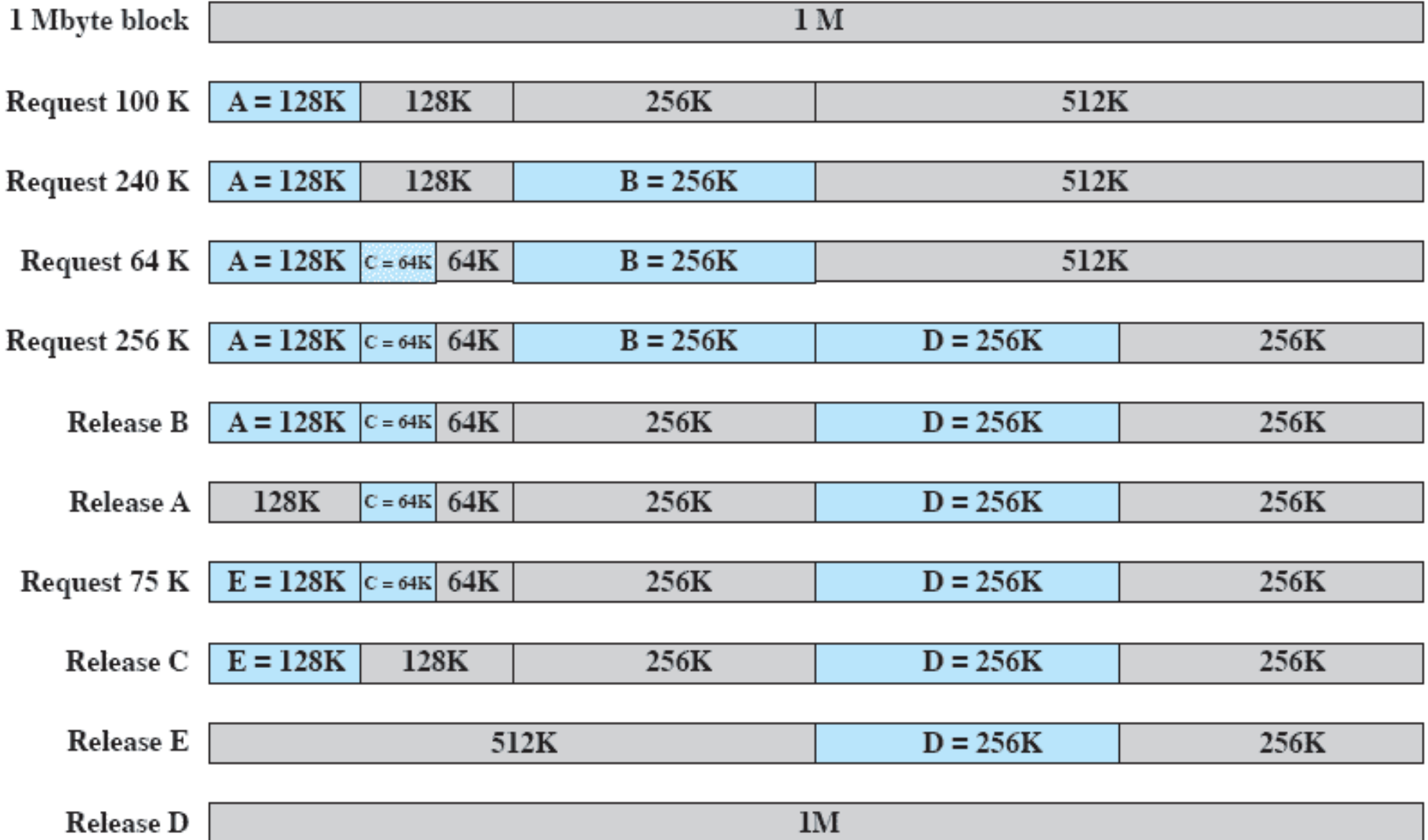


# Sistem Buddy

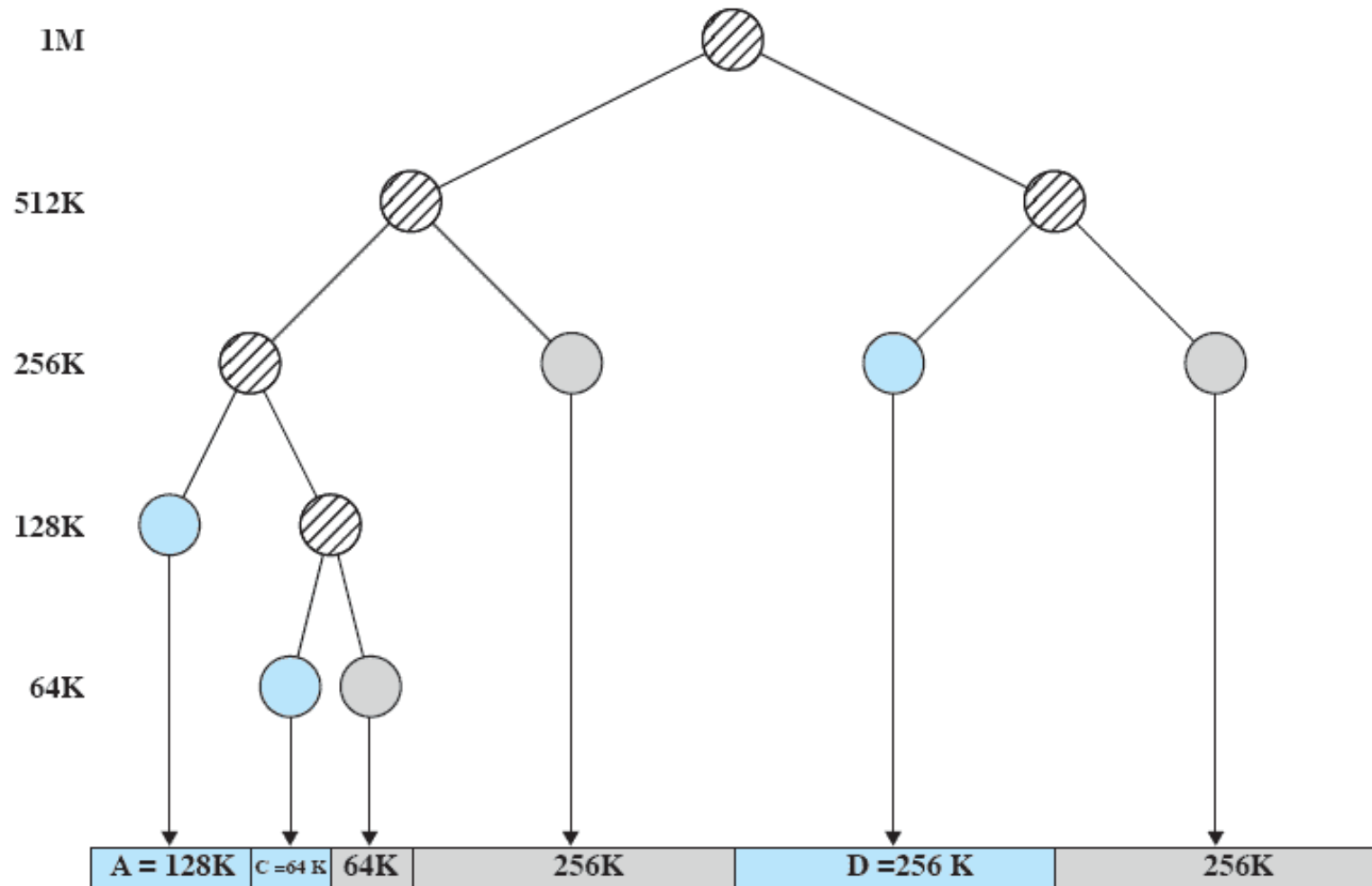
---

- Seluruh ruang tersedia diperlakukan sebagai suatu blok tunggal  $2^U$
- Jika ukuran request ( $s$ )  $2^{U-1} < s \leq 2^U$ 
  - Seluruh blok dialokasikan
- Jika tidak, blok *displit* ke dalam dua buddy yang sama besar
  - Proses berlanjut sampai blok terkecil lebih besar dari atau sama dengan  $s$  dibangkitkan.

# Contoh Sistem Buddy



# Representasi Tree Sistem Buddy



# Relokasi

---

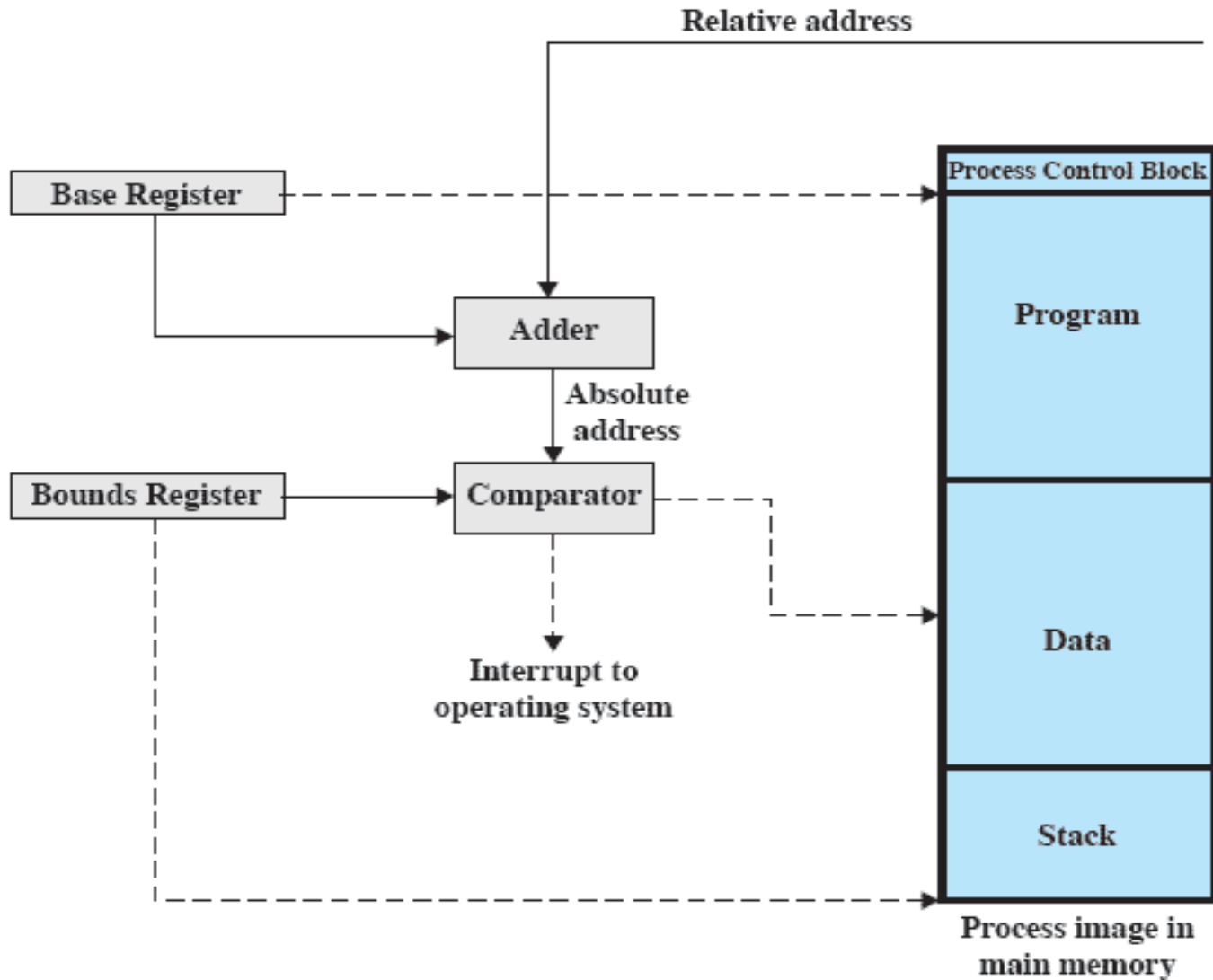
- Ketika program dimuat ke dalam memory, lokasi memory aktual (absolut) ditetapkan
- Suatu proses mungkin menempati partisi-partisi berbeda yang berarti lokasi memory absolut berbeda selama eksekusi
  - Swapping
  - Compaction

# Alamat

---

- Logis
  - Mengacu ke suatu lokasi memory terlepas dari *assignment* data terkini ke memory.
- Relatif
  - Alamat diekspresikan sebagai suatu lokasi relatif terhadap beberapa poin yang diketahui.
- Fisik atau Absolut
  - Alamat absolut atau lokasi sesungguhnya di dalam memory utama.

# Relokasi



# Register Selama Eksekusi (1)

---

- Base register
  - Alamat permulaan bagi proses
- Bounds register
  - Lokasi akhir dari proses
- Nilai-nilai ini diset ketika proses dimuat atau saat proses di-swap-in

# Register Selama Eksekusi (2)

---

- Nilai dari base register ditambahkan ke suatu alamat relatif untuk menghasilkan suatu alamat absolut
- Alamat yang dihasilkan dibandingkan dengan nilai di dalam bounds register
- Jika alamat tidak dalam bounds, suatu interupsi dibangkitkan untuk sistem operasi

# Paging (1)

---

- Membagi memory ke dalam chunks (bagian-bagian) kecil berukuran tetap dan sama, dan membagi setiap proses ke dalam chunk berukuran sama tersebut
- Chunks dari suatu proses dinamakan ***pages***
- Chunks dari memory dinamakan ***frames***

# Paging (2)

---

- SO memelihara suatu tabel page untuk setiap proses
  - Mengandung lokasi frame bagi setiap page di dalam proses
  - Alamat memory terdiri dari suatu nomor page dan offset di dalam page tersebut

# Proses dan Frame

Frame number	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	D.3
12	D.4
13	
14	

# Table Page

0	0
1	1
2	2
3	3

Process A  
page table

0	—
1	—
2	—

Process B  
page table

0	7
1	8
2	9
3	10

Process C  
page table

0	4
1	5
2	6
3	11
4	12

Process D  
page table

13
14

Free frame  
list

Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

# Segmentasi

---

- Suatu program dapat dibagi-bagi ke dalam segmen-segmen
  - Segment mungkin bervariasi panjangnya
  - Ada suatu panjang segment maksimum
- Pengalamatan terdiri dari dua bagian
  - Suatu nomor segment dan
  - Suatu offset
- Segmentasi menyerupai pemartisian dinamis

# Alamat Logis

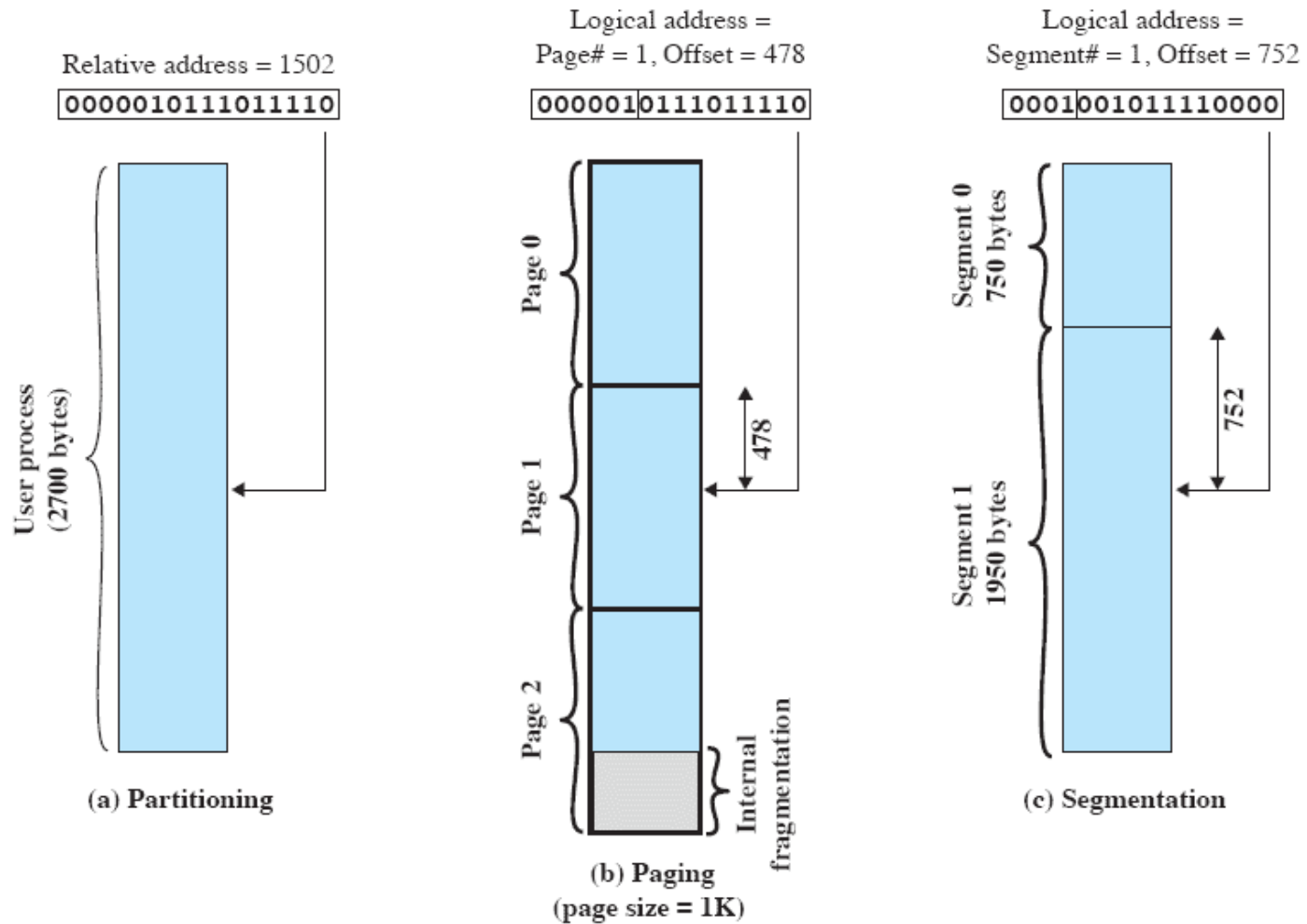
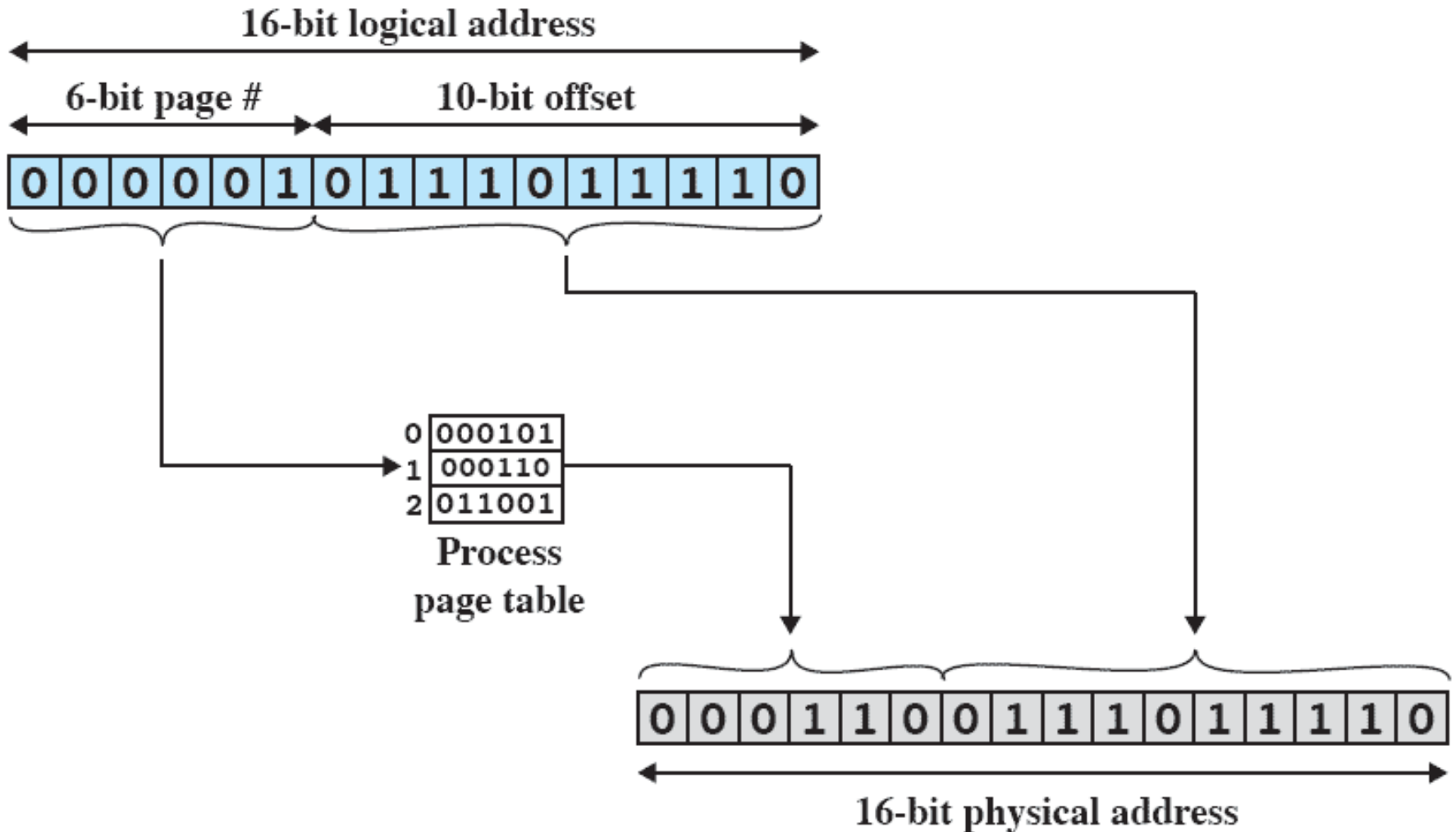


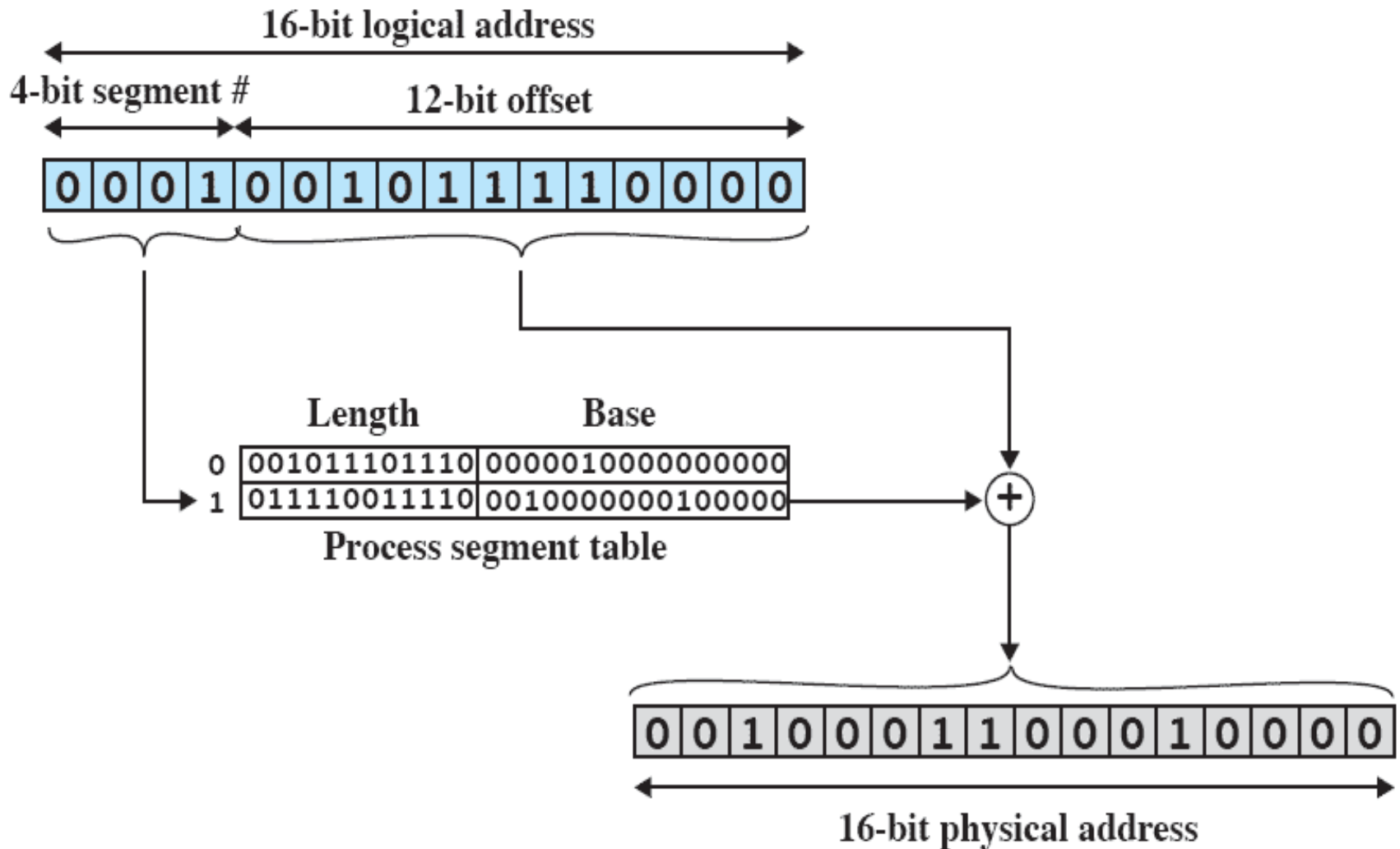
Figure 7.11 Logical Addresses

# Paging



(a) Paging

# Segmentasi



(b) Segmentation

# Tugas Pertemuan 7

---

- Jelaskan istilah pengalamatan Logical, Relative dan Physical!
- Jelaskan perbedaan Partition, Segmentation dan Paging
- Bagaimana perhitungan Alamat Absolut pada Paging dan Segmentation?