AIF183201
Sistem Operasi

Pengelolaan Memori

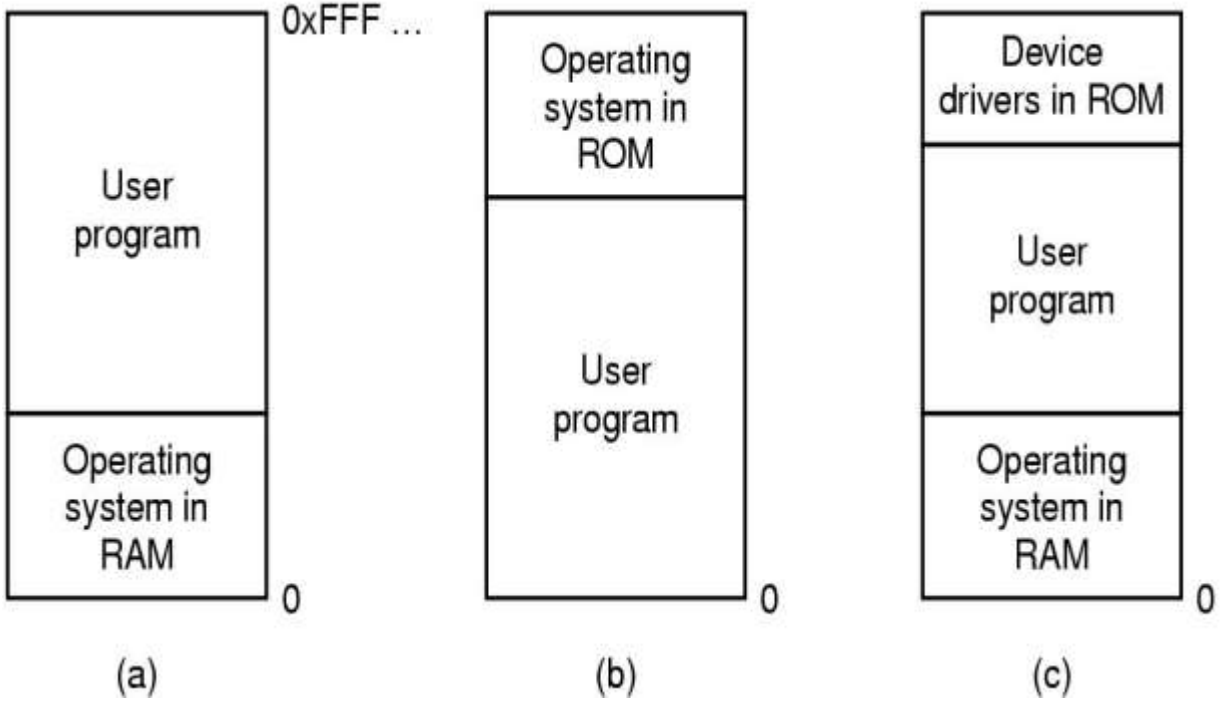
INFORMATIKA
UNPAR

Pendahuluan

- Seorang programmer menginginkan memori
  - private
  - Besarnya tidak terbatas
  - Kecepatan tinggi / tidak terbatas
  - Memori yang tidak hilang isinya jika listrik mati (non volatile)
- Fakta memori secara hierarkhi
  - Kecil, cepat tetapi mahal → cache memory
  - Lebih cepat, harga tidak terlalu mahal → main memory
  - Ukurannya besar, murah tetapi lambat → disk storage
- **Memory manager** bagian/modul dari sistem operasi yang menangani/mengelola memori

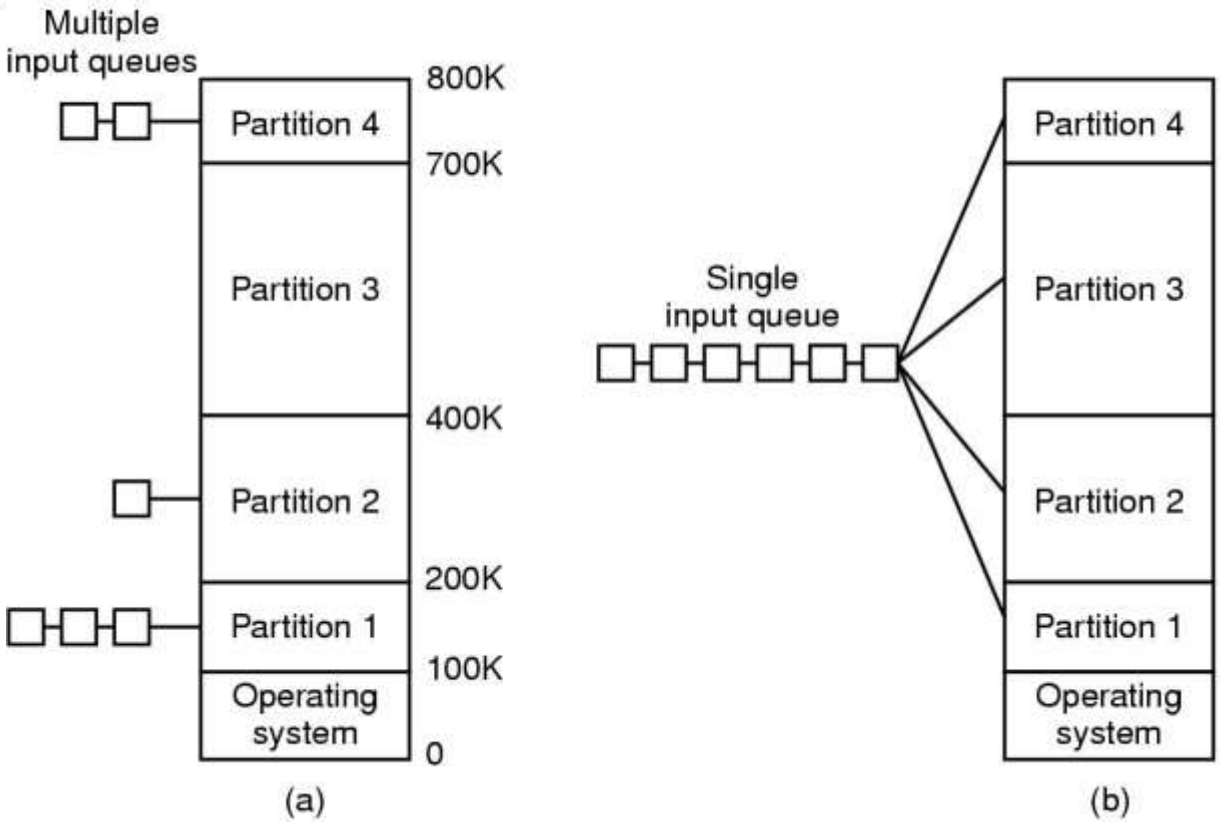
Pengelolaan Memori yang dasar

Monoprogramming tanpa swapping atau paging



Tiga cara sederhana mengorganisasi memori:
sebuah sistem operasi dengan hanya satu proses user

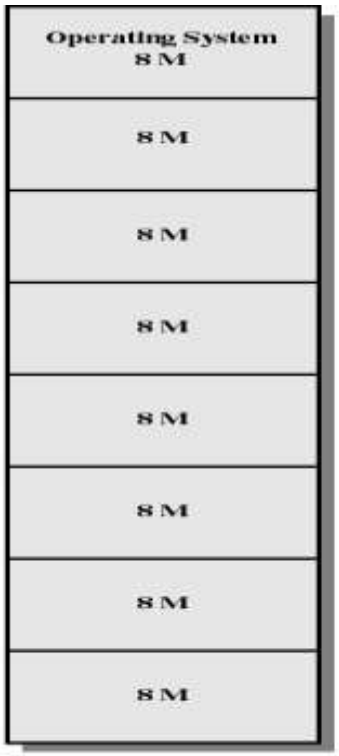
Multiprogramming dengan fixed partition



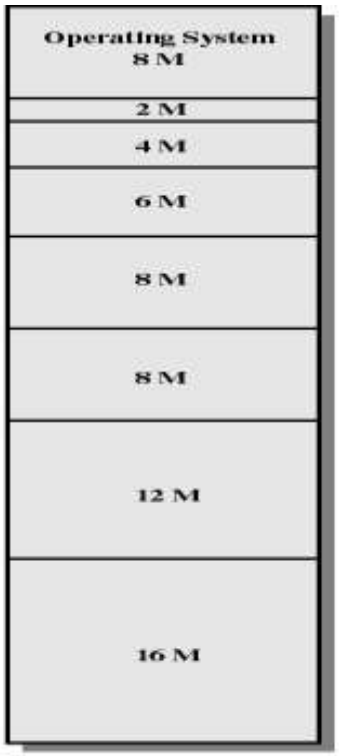
- Partisi memori yang ukurannya fix
  - Setiap partisi memiliki antrian masing-masing
  - Satu antrian untuk seluruh partisi

Multiprogramming dengan Fixed Partitions

- Multiprogramming: pada satu saat lebih dari satu proses ada di dalam memori



(a) Equal-size partitions



(b) Unequal-size partitions

Memori dibagi ke dalam n partisi dengan ukuran

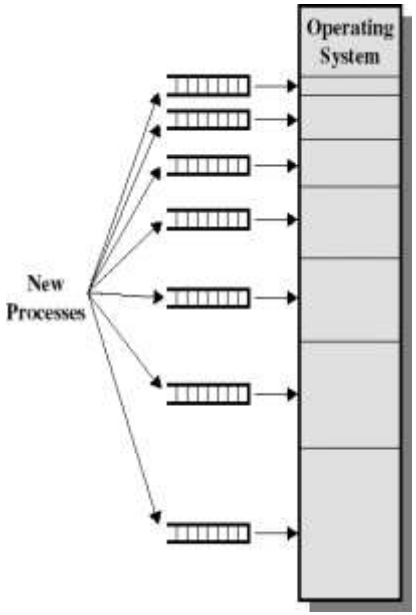
- Sama
- Tidak sama

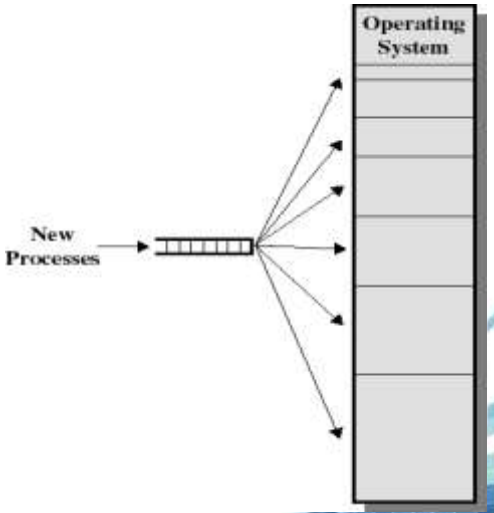
Mempartisi dengan ukuran tetap

- Partisi dengan ukuran yang sama
  - Setiap proses ang ukurannya lebih atau sama dengan ukuran partisi dapat di-load ke salah satu partisi yang kosong/tersedia
  - Jika semua partisi penuh, system operasi dapat **swap out (remove)** sebuah proses dari sebuah partisi
  - Jika sebuah program tidak muat pada partisi (ukuranya melebihi ukuran partisi), programmer harus mendesain program tersebut dengan *overlays*
  - Penggunaan memori utama tidak efisien. Setiap program sebesar berapapun, menguasai seluruhnya sebuah partisi. Ini disebut *internal fragmentation*

Algoritma penempatan dengan memori yang di partisi (1)

- Partisi ukuran sama
  - Karena semua partisi berukuran sama, penempatan proses dapat dilakukan di partisi mana pun
- Partisi dengan ukuran yang tidak sama
  - Dapat menetapkan setiap proses ke partisi yang terkecil yang tepat muat
  - antrian untuk setiap partisi
  - Proses ditetapkan pada sebuah partisi dengan tujuan: meminimalkan memori yang terbuang/tidak terpakai di dalam sebuah partisi





Persoalan pada multiprogramming

- Relocation
  - Programmer tidak tahu dimana program akan ditempatkan di memori ketika di-eksekusi
  - Sementara program dieksekusi, mungkin saja di-swap ke disk dan kembali ke memori utama pada lokasi yang berbeda (relokasi).
  - Referensi memori harus ditranslasi dalam kode ke alamat memori fisik actual
    - Lokasi alamat variable, kode tidak dapat absolute

Persoalan pada multiprogramming (2)

- Protection
  - Proses semestinya tidak dapat mengacu pada lokasi memori pada proses lain tanpa permisi
  - Tidak mungkin untuk mengecek alamat absolute di dalam program karena program dapat di-relokasi.
  - Harus di-cek selama eksekusi
    - Sistem operasi tidak dapat mengantisipasi semua referensi memori sebuah program yang akan dibuat

Relocation and Protection

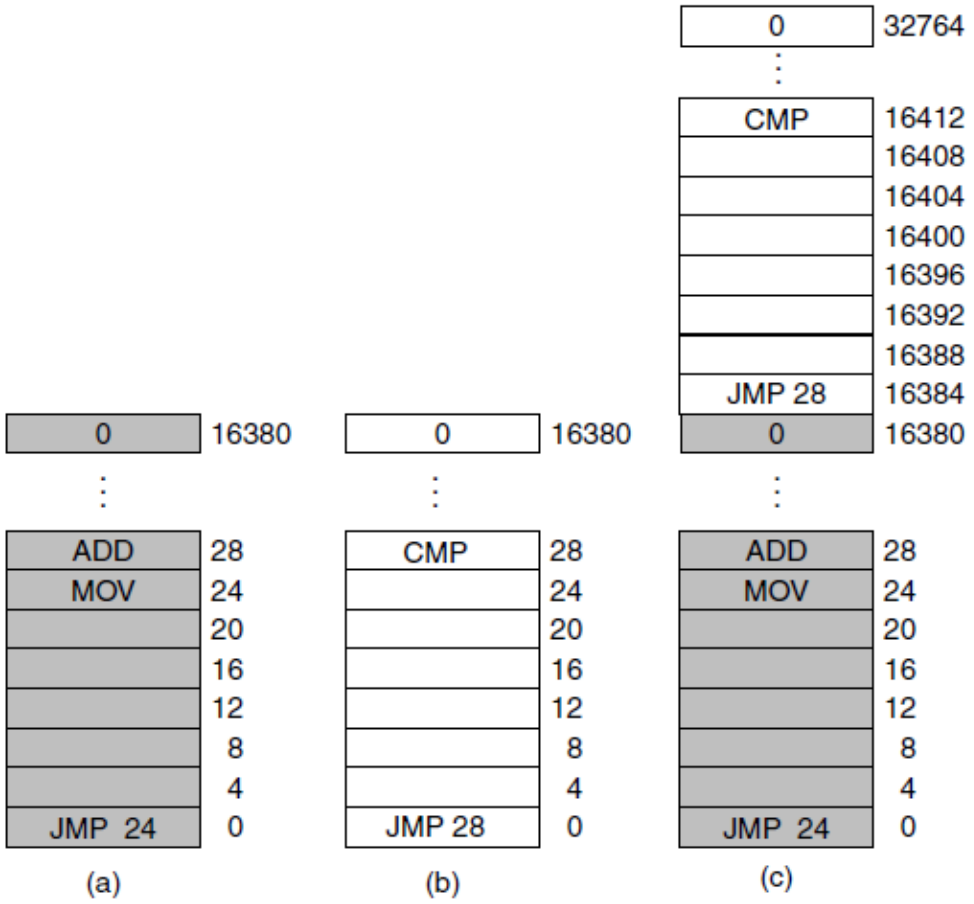
- Tidak dapat dipastikan dimana program di-load ke dalam memori
  - Alamat lokasi variable, rutin kode tidak dapat absolute
  - Harus menjaga sebuah program dari partisi program lain

- Menggunakan **base and limit values**
  - Alamat lokasi ditambahkan base value untuk memetakannya ke alamat fisik
  - Lokasi alamat yang lebih besar dari limit values adalah error

Relocation

- Ketika program di-load ke dalam memori lokasi memori actual (absolute) ditentukan
- Sebuah proses dapat saja menempati partisi yang berbeda yang berarti lokasi memori absolute yang berbeda juga selamat eksekusi (karena *swapping*)
- ***Compaction*** akan juga menyebabkan sebuah program menempati partisi yang berbeda yang berarti lokasi memori absolute yang berbeda.

Menjalankan banyak program tanpa sebuah abstraksi memori



Ilustrasi permasalahan relokasi. (a) Sebuah16-KB program. (b) Program lain 16-KB (c) Dua program di-load secara berurutan ke dalam memori

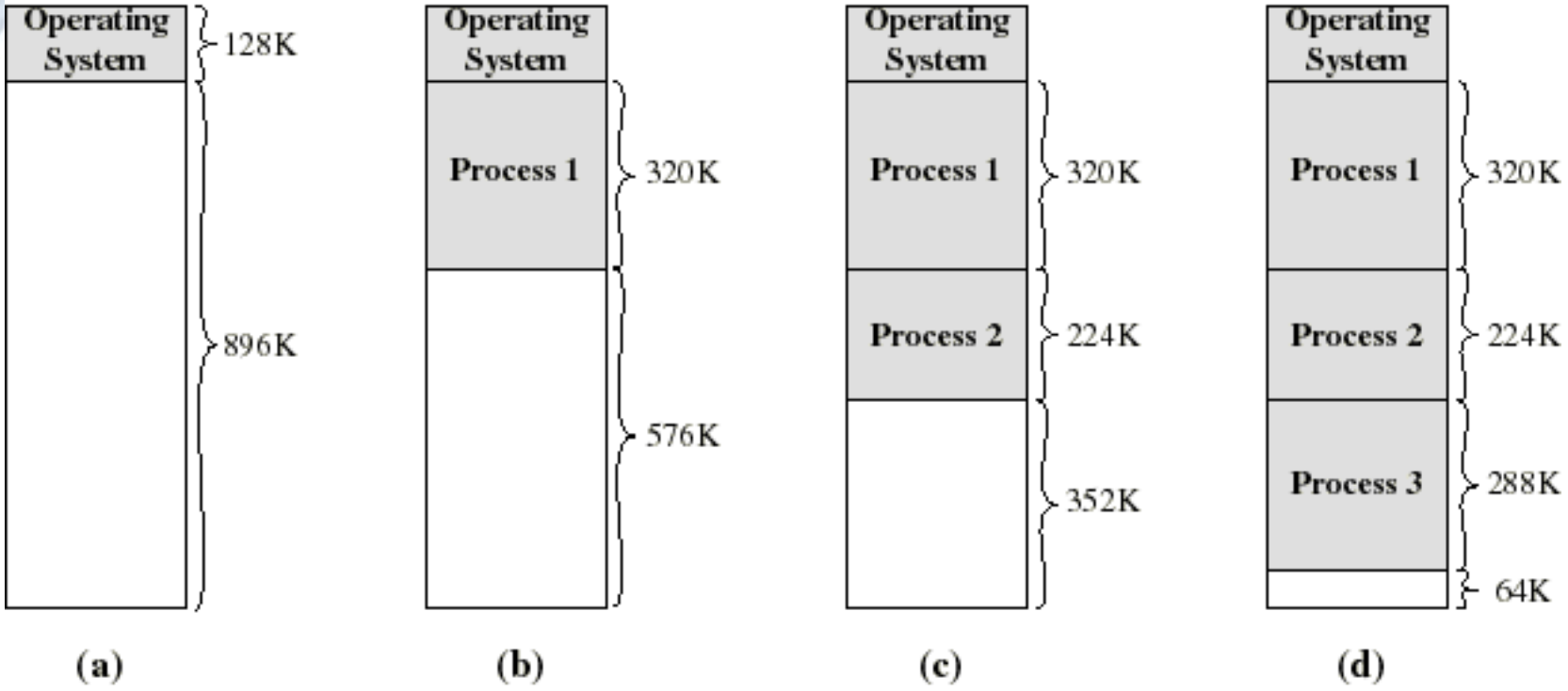
Addresses

- Logical
  - Mengacu ke lokasi memori yang bebas dari penetapannya atas data ke memori
  - Translasi harus dibuat ke alamat fisik
- Relative
  - Alamat yang diekspresikan sebagai lokasi relative terhadap sebuah posisi yang diketahui
- Physical
  - Alamat multak atau lokasi actual di memori utama

Mempartisi secara dinamis

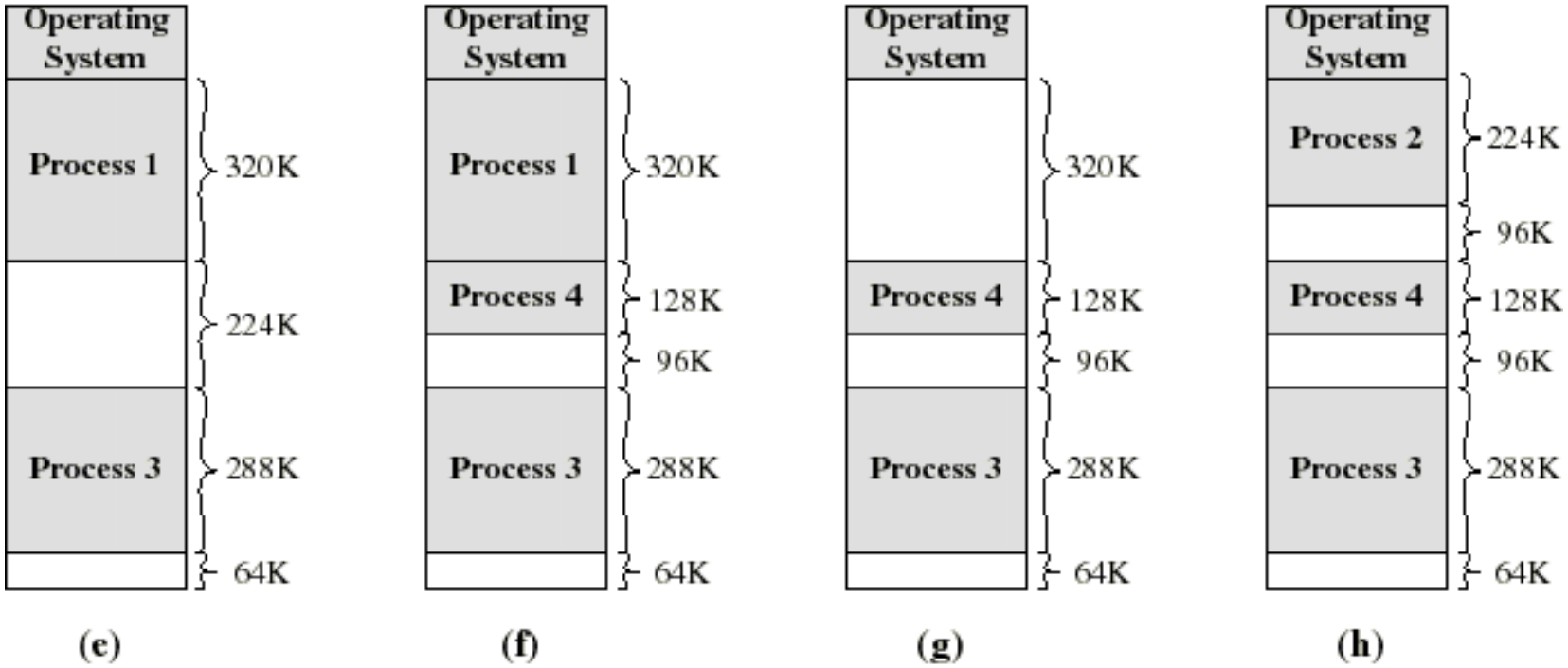
- Partisi, panjang/ukuran dan jumlahnya bervariasi
- Proses dialokasi di memori tepat sebesar yang diperlukan
- Kadang akan muncul hole-hole di dalam memori. Ini yang disebut *external fragmentation*
- Mesti menggunakan *compaction* menata kembali peletakan proses di dalam memori sehingga proses-proses teralokasi secara berurutan dan semua memori kosong berada dalam satu block

Dynamic Partitioning : example (1)



- A hole of 64K is left after loading 3 processes: not enough room for another process
- Eventually each process is blocked. The OS swaps out process 2 to bring in process 4

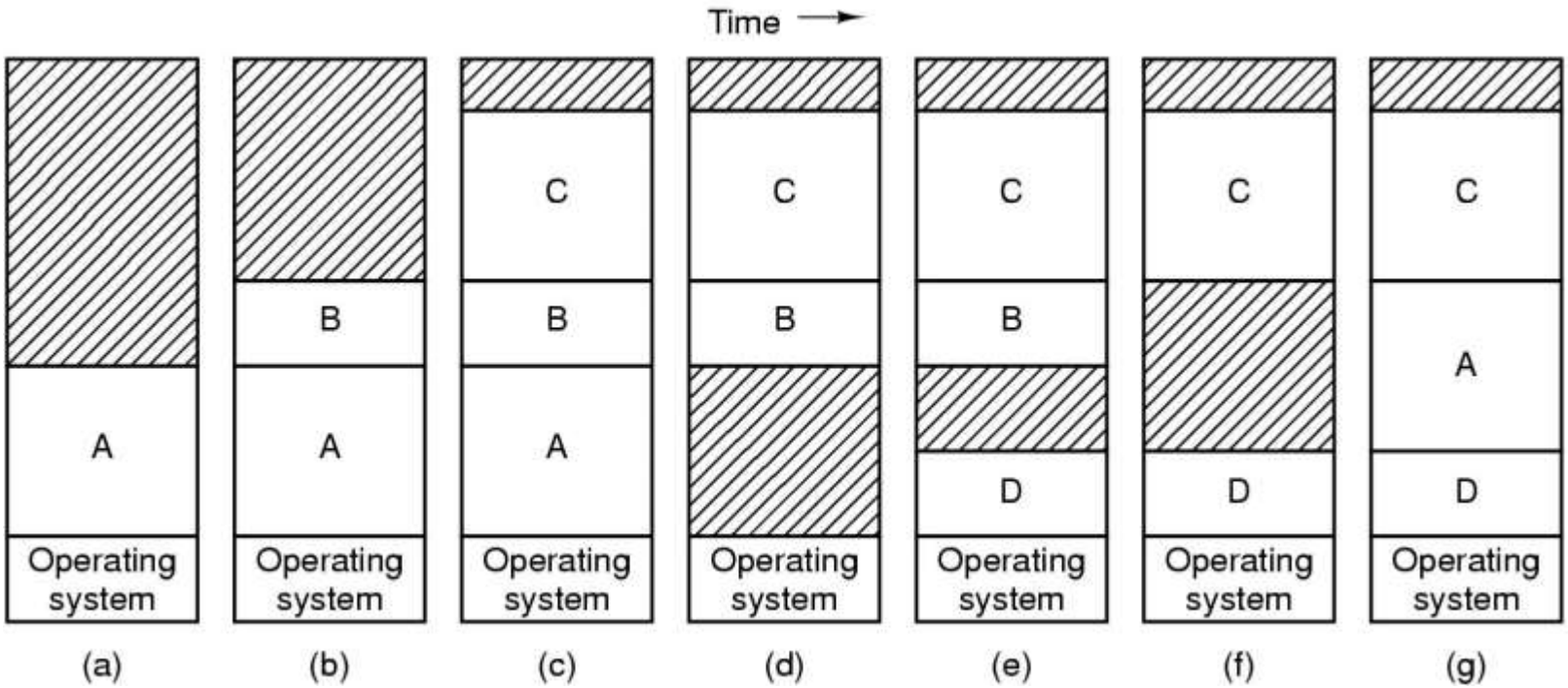
Dynamic Partitioning : example (2)



- another hole of 96K is created
- Eventually each process is blocked. The OS swaps out process 1 to bring in again process 2 and another hole of 96K is created...
- Compaction would produce a single hole of 256K

INFORMATIKA UNPAR

Swapping (1)

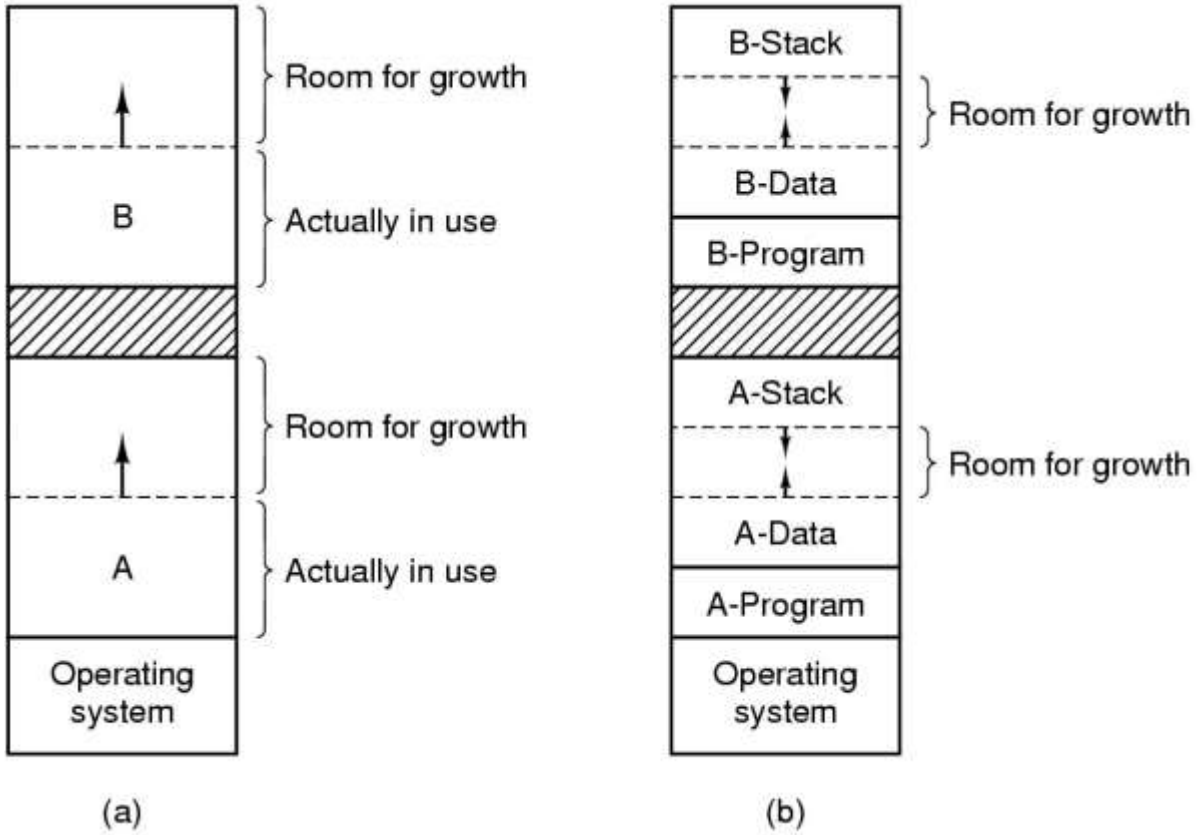


Memory allocation changes as

- processes come into memory
- leave memory

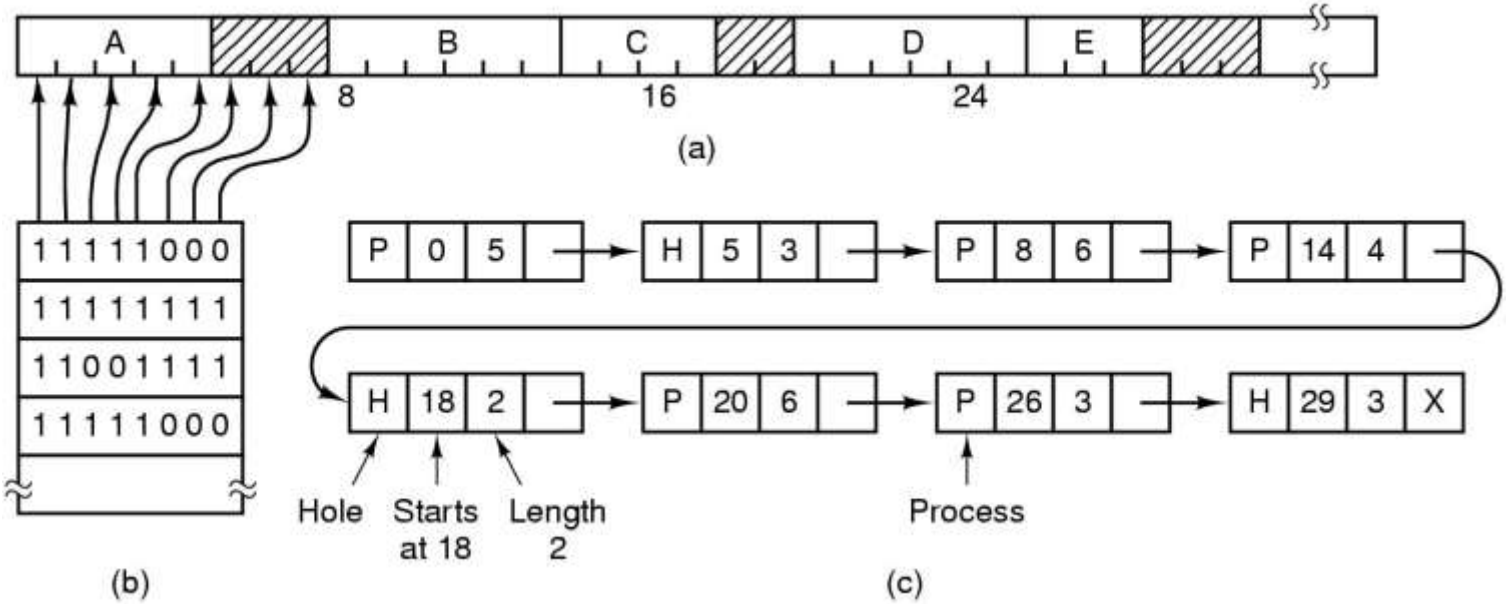
Shaded regions are unused memory

Swapping (2)



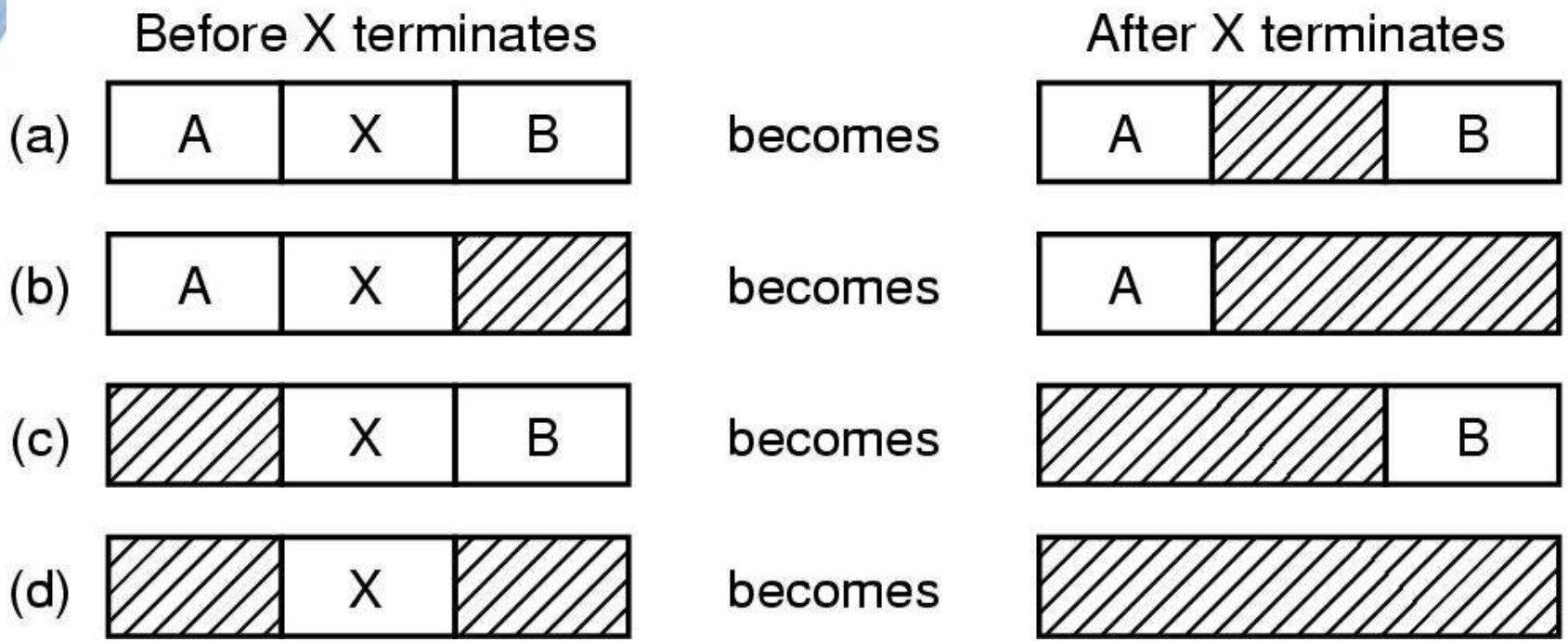
- Allocating space for growing data segment
- Allocating space for growing stack & data segment

Memory Management with Bit Maps



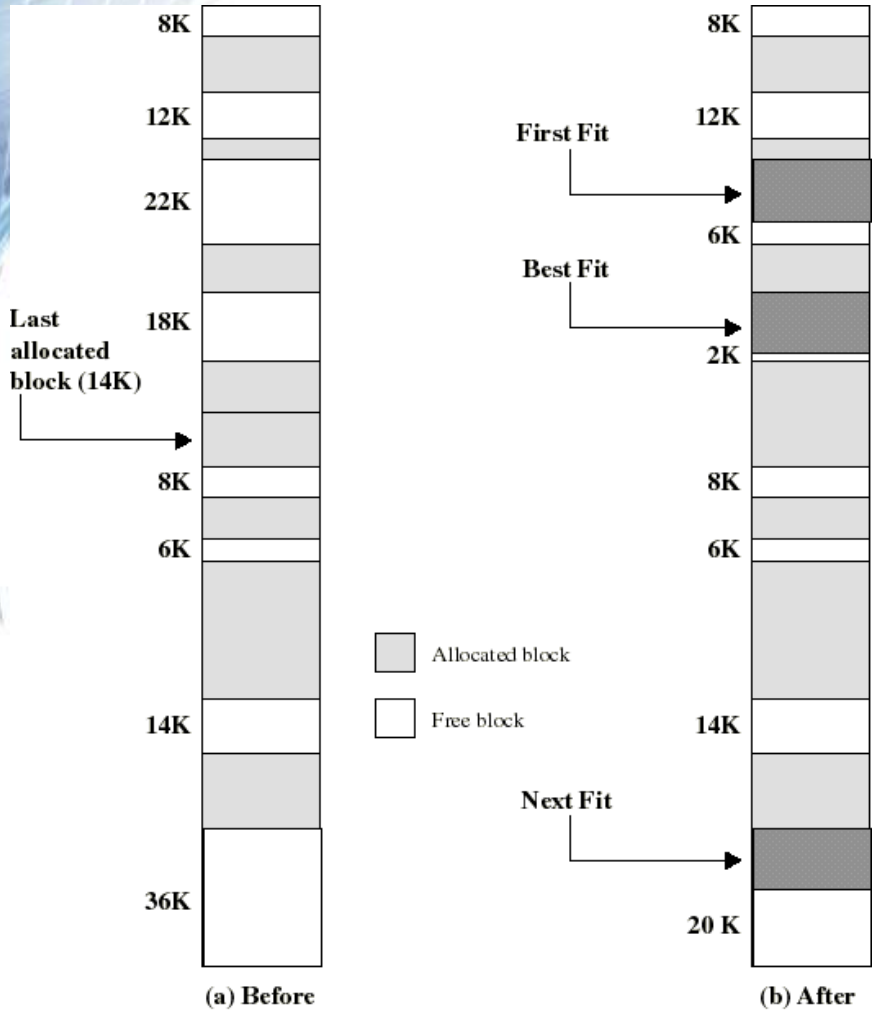
- Part of memory with 5 processes, 3 holes
  - tick marks show allocation units
  - shaded regions are free
- Corresponding bit map
- Same information as a list

Memory Management with Linked Lists



Four neighbor combinations for the terminating process X

Dynamic Partitioning Placement Algorithm (1)



Example Memory Configuration Before and After Allocation of 16 Kbyte Block

- OS must decide which free block to allocate to a process
- Goal: to reduce usage of compaction (time consuming)
- Possible algorithms:
  - Best-fit: choose smallest hole
  - First-fit: choose first hole from beginning
  - Next-fit: choose first hole from last placement
  - Worst-fit:choose largest hole
  - Quick-fit:separate list of common size

Dynamic Partitioning Placement Algorithm (2)

- Best-fit algorithm
  - Searches the entire list
  - Chooses the block that is closest in size to the request
  - Worst performer overall
  - Since smallest block is found for process, the smallest amount of fragmentation is left memory compaction must be done more often
- First-fit algorithm
  - Fastest
  - May have many process loaded in the front end of memory that must be searched over when trying to find a free block

Dynamic Partitioning Placement Algorithm (3)

- Next-fit
  - More often allocate a block of memory at the end of memory where the largest block is found
  - The largest block of memory is broken up into smaller blocks
  - Compaction is required to obtain a large block at the end of memory
- Worst-fit
  - Take the largest available hole so that the hole broken off will be big enough to be useful

Dynamic Partitioning Placement Algorithm(4)

- Quick-fit
  - Maintain a table with n entries, in which the first entry was a pointer to the head of a list of 4K holes, the second entry was a pointer to a list of 8K holes, the third entry a pointer to a list of 12K holes, and so on.
  - Holes of say, 21K, could either be put on the 20K list or special list of odd-sized holes.
  - Finding a hole of required size is extremely fast
  - Disadvantage : when process terminates or swapped out, finding its neighbors to see if a merge is possible is expensive.
  - If merging is not done, memory will quickly fragment into a large number of small holes

Memory Management : Resume

- Monoprogramming
- Multiprogramming
  - Fixed Partition
    - Equal size
    - Unequal size
      - Multiple queue
      - Single queue
    - Swapping & Overlays
    - Internal Fragmentation
  - Dynamic Partition
    - Partitions are of variable length and number ; Process is allocated exactly as much memory as required
    - External Fragmentation – Compaction
    - Placement Algorithm: First-fit, best-fit, worst-fit, next-fit, quick fit