Ketiak proses-proses bekerja sama dengan komunikasi, beragam proses berpartisipasi dalam suatu usaha dengan menghubungkan semua proses. Komunikasi menyediakan cara untuk sinkronisasi atau koordinasi beragam aktivitas. Komunikasi dicirikan dengan berisi pesan-pesan dengan suatu urutan. Primitif untuk mengirim dan menerima pesan disediakan sebagai bagian bahasa pemrograman atau disediakan kernel sistem operasi. Karena tak ada sesuatu yang dipakai bersama diantara proses-proses itu dalam melewatkan pesan-pesan, tak ada masalah mutual exclusion. Tetapi masalah deadlock dan startvation dapat muncul.

5.11. Pokok penyelesaian masalah kongkurensi

Pada dasarnya penyelesaian masalah kongkurensi terbagi menjadi dua, yaitu :
a. Mengasumsikan adanya memori yang digunakan bersama.
b. Tidak mengasumsikan adanya memori yang digunakan bersama.
Adanya memori bersama lebih mempermudah penyelesaian masalah kongkurensi. Metode penyelesaian ini dapat dipakai untuk sistem singleprocessor ataupun multiprocessor yang mempunyai memori bersama. Penyelesaian ini tidak dapat digunakan untuk multiprocessor tanpa memori bersama ataupun untuk sistem tersebar.

Bab 6

MANAJEMEN MEMORI

Bagian operating sistem yang mengatur memori disebut dengan memory manager. Pemakaian memori (manajemen memori dan organisasi) perlu dilakukan karena hal tersebut sangat mempengaruhi kinerja komputer, sehingga memiliki fungsi dan tugas penting dan kompleks yaitu berkaitan dengan :

a. Memori utama sebagai sumber daya yang harus dialokasikasikan dan dipakai bersama di antara sejumlah proses yang aktif, sehingga dapat memanfaatkan pemroses dan fasilitas masukan/keluaran secara efisien, sehingga memori dapat menampung sebanyak mungkin proses.
b. Upaya agar pemogram atau proses tidak dibatasi kapasitas memori fisik di sistem komputer.

6.1. Manajemen memori

Sistem manajemen memori dapat dibagi kedalam dua kelas, yaitu : pemindahan proses (back and forth) diantara memori utama dengan disk selama eksekusi (swapping and paging) dan tidak ada pemindahan proses.

Mempunyai beberapa fungsi, antara lain :

a. Mengelola informasi memori yang dipakai dan tidak dipakai.
b. Mengalokasikan memori ke proses yang memerlukan.
c. Mendealokasikan memori dari proses yang telah selesai.
d. Mengelola swapping antara memori utama dan disk.

6.2. Manajenen memori pada sistem multiprogramming

Untuk sistem komputer yang berukuran besar (bukan small computers), membutuhkan pengaturan memori, karena dalam multiprogramming akan melibatkan banyak pemakai secara simultan sehingga di memori akan terdapat lebih dari satu proses bersamaan. Oleh karena itu dibutuhkan sistem operasi yang mampu mendukung dua kebutuhan tersebut, meskipun hal tersebut saling bertentangan, yaitu :

a. Pemisahan ruang-ruang alamat.
b. Pemakaian bersama memori.

Manajer memori harus memaksakan isolasi ruang-ruang alamat tiap proses agar mencegah proses aktif atau proses yang ingin berlaku jahat mengakses dan merusak ruang alamat proses lain. Manajer memori di lingkungan multiprogramming sekalipun melakukan dua hal, yaitu :

a. Proteksi memori dengan isolasi ruang-ruang alamat secara disjoint.

c. Pemakaian bersama memori.

Memungkinkan proses-proses bekerja sama mengakses daerah memori bersama.

Ketika konsep multiprogramming digunakan, pemakaian CPU dapat ditingkatkan.

Sebuah model untuk mengamati pemakaian CPU secara probabilistic :

CPU utilization = 1 - p n

Dengan :

* N menunjukkan banyaknya proses pada suatu saat, sehingga kemungkinan bahwa semua n proses akan menunggu menggunakan I/O (masalah CPU menganggur) adalah sebesar p n. Fungsi dari n disebut sebagai degree of multiprogramming.
* P menunjukkan besarnya waktu yang digunakan sebuah proses

Contoh analisisnya :

Diketahui :

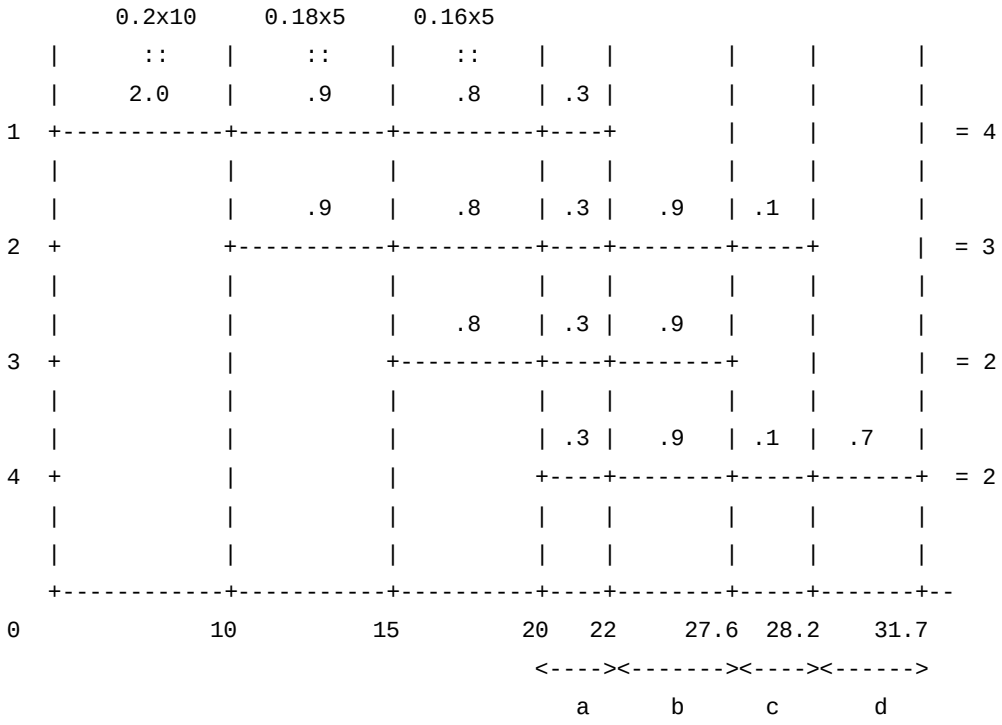
Job	Arrival time	CPU minutes needed
1	10:00	4
2	10:10	3
3	10:15	2
4	10:20	2

Bila semua job bersifat 80% I/O wait. Tentukan kapan waktu selesainya masing-masing job !

Jawab :

Process

```
                  1        2        3        4
-------------------------------------------------
CPU idle .80      .64      .51      .41
CPU busy .20      .36      .49      .59
CPU/process       .20      .18      .16      .15
-------------------------------------------------
```



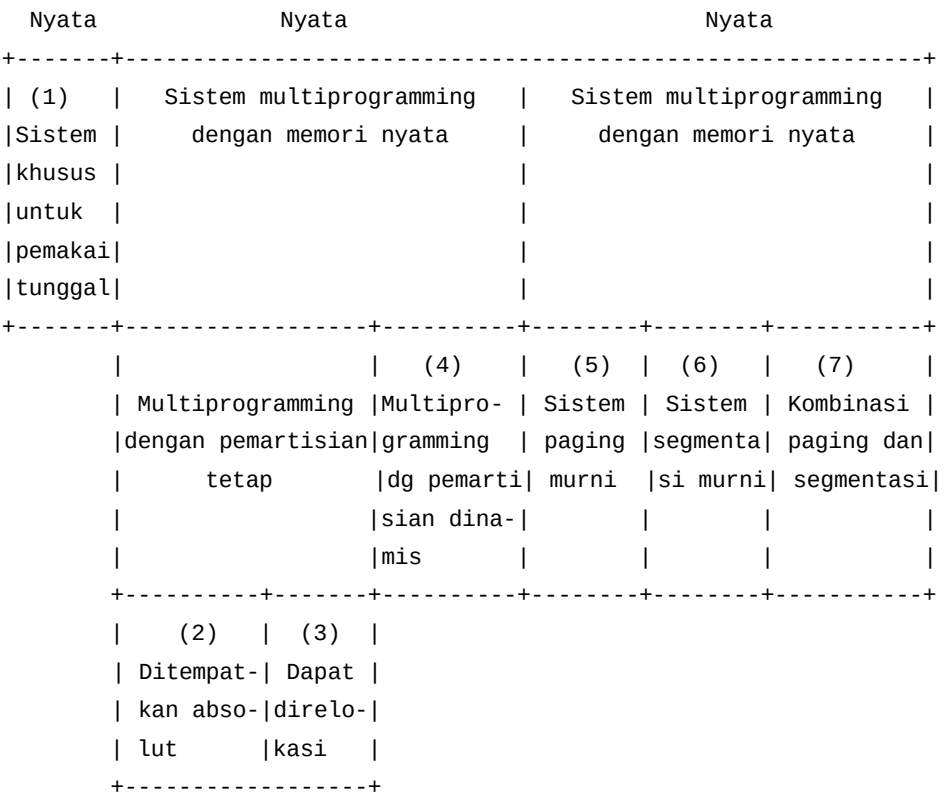
Keterangan :

```
ax0.15=0.3   bx0.16=0.9   cx0.18=0.1   dx0.2=0.7
     a=2          b=5.6        c=0.6       d=3.5
```

6.3. Klasifikasi manajemen memori.

Klasifikasi manajemen memori diberikan Deitel (DEI-90).

Gambar 6.1 menunjukkan skema klasifikasi manajamen memori.



Gambar 6.1 : Klasfikasi manajemen memori

Teknik-teknik manajemen memori (1), (2), (3), (4) merupakan pengelolaan untuk dengan kapasitas memori sebatas memori fisik yang tersedia.

Teknik-teknik ini tidak dapat digunakan untuk memuat program-program lebih besar dibanding kapasitas fisik memori yang tersedia.

Teknik-teknik manajemen memori (5), (6), (7) dapat digunakan untuk mengakali kapasitas memori yang terbatas sehingga dapat dijalankan program yang lebih besar dibanding kapasitas memori fisik yang tersedia.

6.4. Manajemen memori berdasarkan keberadaan swapping

Manajemen memori berdasarkan keberadaan swapping terbagi menjadi dua, yaitu :

1. Manajemen tanpa swapping.

Manajemen memori tanpa pemindahan citra proses antara memori utama dan disk selama ekseskusi.

2. Manajemen dengan swapping.

Manajemen memori dengan pemindahan citra proses antara memori utama dan disk selama ekseskusi.

6.5. Manajemen memori berdasar alokasi memori

Manajemen memori berdasar alokasi memori terbagi dua, yaitu :

1. Alokasi memori berurutan (kontigu).

Adalah tiap-tiap proses menempati satu blok tunggal lokasi memori yang berturutan.

Keunggulan :

a. Sederhana.
b. Tidak akan terbentuk lubang-lubang memori bersebaran.
c. Karena berurutan, proses dapat dieksekusi dengan cepat.

Kelemahan :

a. Dapat memboroskan memori.
b. Tidak dapat memuatkan proses bila tidak ada satu blok memori yang mencukupi.

2. Alokasi memori tak berurutan (non-kontinyu).

Program dibagi menjadi beberapa blok atau segmen. Blok-blok program ditempatkan di memori dalam potongan-potongan tanpa perlu saling berdekatan. Teknik biasa digunakan pada sistem memori maya sebagai alokasi page-page dilakukan secara global.

Keuntungan :

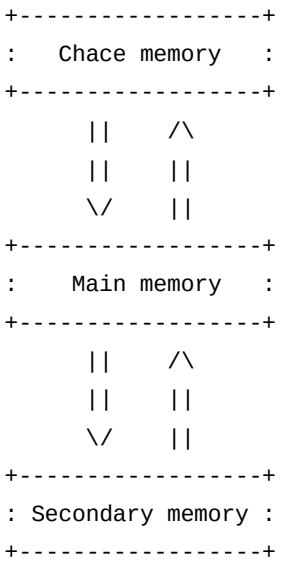
a. Sistem dapat memanfaatkan memori utama secara lebih efisien.
b. Sistem operasi masih mampu memuatkan proses bila jumlah total lubang-lubang memori cukup untuk memuat proses yang akan dieksekusi.

Kelemahan :

a. Memerlukan pengendalian yang lebih rumit dan sulit.
b. Memori dapat menjadi banyak lubang tersebar (memori tak terpakai bersebaran).

6.6. Hirarki memori

Pemakaian memori dua tingkat, menggunakan cache memory yang dapat meningkatkan kinerja dan utilisasi memori secara dinamik. Chace memory merupakan penyimpan berkecepatan tinggi lebih cepat dibanding memori utama.
Chace memory lebih mahal dibanding memori utama, sehingga kapasitas cache relatif kecil.



Gambar 6.2 : Hubungan chace memori, memori utama dan memori sekunder.

Gambar 6.2 memperlihatkan hubungan antara chace memory, memori utama dan penyimpan sekunder. Dengan cache memory, bagian program yang akan digunakan (dieksekusi atau diacu) dikopi dulu ke chace sebelum dieksekusi. Di chace memory, instruksi dapat dieksekusi dengan lebih cepat dibanding di memori utama. Penggunaan chace atau memori antara yang lebih cepat mempunyai alasan yang dikemukakan oleh Denning, yaitu ekseksui program biasanya pada suatu

interval waktu mengumpul di satu lokasi kecil. Prinsip ini disebut lokalitas. Lokalitas dapat berupa lokalitas waktu dan ruang. Prinsip lokalitas berkembang menjadi konsep working set model.

6.7. Manajemen memori tanpa swapping

Manajemen memori tanpa swapping terdiri dari :

a. Monoprogramming.

Monoprogramming sederhana tanpa swapping merupakan manajemen memori paling sederhana, sistem komputer hanya mengijinkan satu program/pemakai berjalan pada satu waktu. Semua sumber daya sepenuhnya dkuasi proses yang sedang berjalan.

Manajemen memori monoprogramming sederhana mempunyai ciri-ciri berikut :

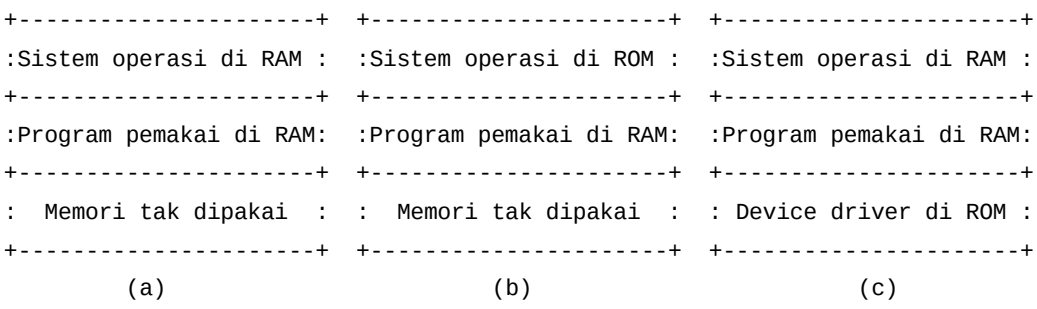
a. Hanya terdapat satu proses pada satu saat, sehingga proses baru akan menimpa proses lama yang sudah selesai eksekusi.
b. Hanya satu proses mengunakan semua memori.
c. Pemakai memusatkan program keseluruh memori dari disk atau tape.
d. Program mengambil kendali seluruh mesin.

Karena hanya terdapat satu proses dan menguasai seluruh sistem, maka eksekusi memori dilakukan secara berurutan.

Teknik ini digunakan sampai sekitar 1960, ditinggalkan bahkan untuk komputer pribadi karena tiap proses harus berisi device driver perangkat I/O yang digunakan.

Gambar 6.3 menunjukkan tiga organisasi memori menjalankan satu proses tunggal :

c. Gambar 6.3(a) menunjukkan seluruh kebutuhan (sistem operasi, device driver dan proses driver dapat ditempatkan di sistem operasi atau di setiap proses pemakai, bergantung perancang sistem operasi.
d. Gambar 6.3(b) menunjukkan sistem operasi ditempatkan di ROM, sedang program pemakai di RAM.
e. Gambar 6.3(c) menunjukkan device driver di ROM. Device driver di ROM biasa disebut ROM-BIOS (Read Only Memory - Basic Input Output Systems).



Gambar 6.3 : Tiga cara organisasi memori untuk satu proses tunggal

Embedded system

Teknik monoprogamming masih dipakai untuk sistem kecil yaitu sistem tempelan (embedded system) yang menempel atau terdapat disistem lain. Sistem-sistem tempelan menggunakan mikroprosesor kecil, seperti Intel 8051, dan sebagainya. Sistem ini biasanya untuk mengendalikan satu alat sehingga menjadi bersifat intelejen (intelegent devices) dalam menyediakan satu fungsi spesifik. Karena hanya satu fungsi spesifik, dapat diprogram di mikroprosesor dengan memori kecil (1-64 Kb).

Sistem tempelan telah banyak digunakan, misalnya sistem tempelan di mobil antar lain untuk :

a. Pengendalian pengapian.
b. Pengendalian pengeluaran bahan bakar.
c. Pengendalian pengereman.
d. Pengendalian suspensi.
e. Pengendalian kemudi.
f. Dan sebagainya.

Pada mobil mewah terdapat lebih dari 50 mikroprosesor, masing-masing mengendalikan satu fungsi spesifik.

Proteksi pada monoprogramming sederhana.

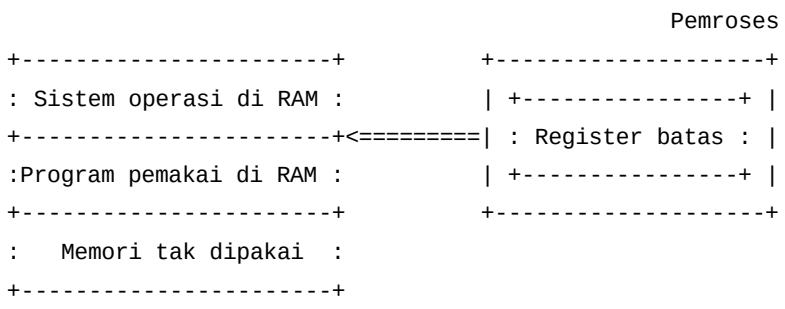
Pada monoprogramming, pemakai mempunyai kendali penuh terhadap seluruh memori utama. Memori terbagi menjadi tiga bagian, yaitu :

a. Bagian yang berisi rutin-rutin sistem operasi.

b. Bagian yang berisi program pemakai.
c. Bagian yang tidak digunakan.

Masalah proteksi di monoprogramming adalah cara memproteksi rutin sistem operasi dari penghancuran program pemakai. Program pemakai dapat tersesat sehingga memanipulasi atau menempati ruang memori rutin sistem operasi. Aktivitas program pemakai ini dapat merusak sistem operasi.
Sistem operasi harus diproteksi dari modifikasi program pemakai. Proteksi ini diimplementasikan menggunakan satu registe batas (boundary register) dipemroses.Setiap kali program pemakai mengacu alamat memori dibandingkan register batas untuk memastikan proses pemakai tidak merusak sistem operasi, yaitu tidak melewati nilai register batas.

Register batas berisi alamat memori tertinggi yang dipakai sistem operasi. Jika program pemakai mencoba memasuki sistem operasi, instruksi diintersepsi dan job diakhiri dan diberi pesan kesalahan. Untuk memperoleh layanan sistem operasi, program pemakai harus menggunakan instruksi spesifik meminta layanan sistem operasi. Integritas sistem operasi terjaga dan program pemakai tidak merusak bagian sistem operasi.



Gambar 6.4 : Proteksi pada monoprogramming

Gambar 6.4 menunjukkan skema proteksi menggunakan register batas.
Register batas menunjuk alamat terakhir sistem operasi. Bila program pemakai mengacu ke alamat daerah sistem operasi, pemroses menjadi fault menyatakan terjadinya pelanggaran pengaksesan oleh proses pemakai.

b. Multiprogramming dengan pemartisian statis.

Terdapat beberapa alasan kenapa multiprogramming digunakan, yaitu :

a. Mempermudah pemogram.
   Pemogram dapat memecah program menjadi dua proses atau lebih.
b. Agar dapat memberi layanan interaktif ke beberapa orang secara simultan.
   Untuk itu diperlukan kemampuan mempunyai lebih dari satu proses dimemori agar memperoleh kinerja yang baik.
c. Efisiensi penggunaan sumber daya.
   Bila pada multiprogramming maka proses tersebut diblocked (hanya DMA yang bekerja) dan proses lain mendapat jatah waktu pemroses, maka DMA dapat meningkatkan efisiensi sistem.
d. Eksekusi lebih murah jika proses besar dipecah menjadi beberapa proses kecil.
e. Dapat mengerjakan sejumlah job secara simultan.

Multiprogramming dapat dilakukan dengan pemartisian statis, yaitu memori dibagi menjadi beberapa sejumlah partisi tetap. Pada partisi-partisi tersebut proses-proses ditempatkan. Pemartisian statis berdasarkan ukuran partisi-partisinya terbagi dua, yaitu :

1. Pemartisian menjadi partisi-partisi berukuran sama, yaitu ukuran semua partisi memori adalah sama.

Beberapa proses yang ukurannya kurang atau sama dengan ukuran partisi dimasukkan ke sembarang partisi yang tersedia.

Kelemahan :

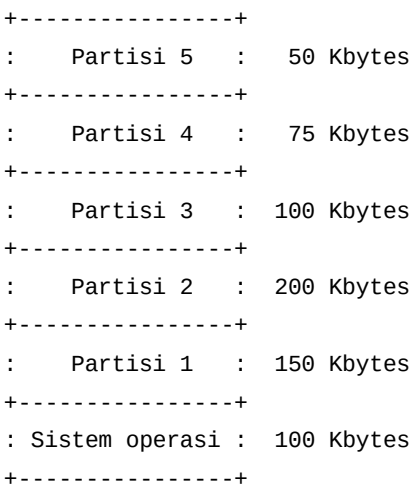
* Bila program berukuran lebih besar dibanding partisi yang tersedia, maka tidak dapat dimuatkan, tidak dapat dijalankan. Pemogram harus mempersiapkan overlay sehingga hanya bagian program yang benar-benar dieksekusi yang dimasukkan ke memori utama dan saling bergantian. Untuk overlay diperlukan sistem operasi yang mendukung swapping.
* Untuk program yang sangat kecil dibanding ukuran partisi yang ditetapkan, maka banyak ruang yang tak dipakai yang diboroskan, disebut fragmentasi internal.

Kelemahan ini dapat dikurangi dengan partisi-partisi tetap berukuran berbeda.

2. Pemartisian menjadi partisi-partisi berukuran berbeda, yaitu ukuran semua partisi memori adalah berbeda.

Gambar 6.5 menunjukkan skema multiprogramming pemartisian tetap berukuran

berbeda.



Gambar 6.5 : Multiprogramming dengan pemartisian tetap berukuran sama

6.8. Strategi penempatan program ke paritisi

a. Strategi penempatan pada pemartisian menjadi partisi-partisi berukuran sama. Penempatan proses ke memori dilakukan secara mudah karena dapat dipilih sembarang partisi yang kosong.
b. Strategi penempatan pada pemartisian menjadi partisi-partisi berukuran berbeda.

Terdapat dua strategi penempatan program ke partisi, yaitu :

a. Satu antrian untuk tiap partisi (banyak antrian untuk seluruh partisi).

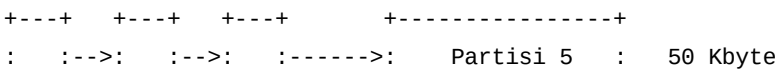
Proses ditempatkan ke partisi paling kecil yang dapat memuatnya.

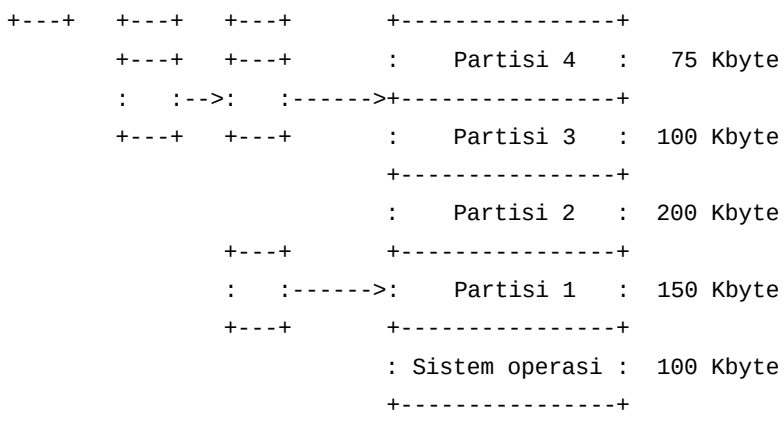
Keuntungan :

Teknik ini adalah meminimalkan pemborosan memori.

Kelemahan :

Dapat terjadi antrian panjang disuatu partisi sementara antrian partisi-partisi lain kosong. Teknik ini diperlihatkan pada gambar 6.6.





Gambar 6.6 : Multiprogramming dengan pengisian pemartisian tetap dengan banyak antrian.

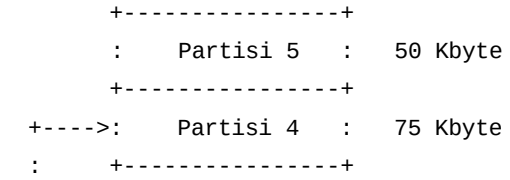
b. Satu antrian untuk seluruh partisi.

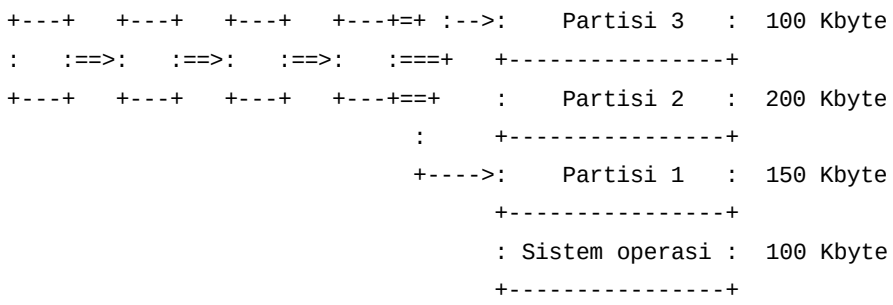
Proses-proses diantrikan di satu antrian tunggal untuk semua partisi.
Proses segera ditempatkan di partisi bebas paling kecil yang dapat memuat.

Keunggulan :
Lebih fleksibel serta implementasi dan operasi lebih minimal karena hanya mengelola satu antrian.

Kelemahan :
Proses dapat ditempatkan di partisi yang banyak diboroskan, yaitu proses kecil ditempatkan di partisi sangat besar.

Teknik ini diperlihatkan pada gambar 6.7.





Gambar 6.7 : Multiprogramming dengan pengisian pemartisian tetap dengan satu antrian.

Kelemahan ini dapat diatasi dengan prosedur pemindahan. Pemindahan dilakukan bila proses besar akan masuk memori tetapi hanya tersedia partisi kecil sementara proses kecil menempati partisi besar. Proses kecil di swap ke partisi kecil yang sedang bebas kemudian proses besar di antrian menempati partisi besar yang ditinggal proses kecil.

Pemartisian memori menjadi partisi-partisi secara statis mempunyai dua masalah, yaitu :

a. Relokasi.

Adalah masalah penempatan proses sesuai alamat fisik sehubungan alamat partisi memori dimana proses ditempatkan. Proses dapat ditempatkan pada partisi-partisi berbeda menurut keadaan sistem saat itu. Pengalamatan fisik secara absolut untuk proses tidak dapat dilakukan.

Solusi pertama :

Sistem operasi menambahkan alamat awal partisi dimana proses ditempatkan ke setiap alamat yang diacu proses. Pada saat proses kompilasi, linker harus memasukkan satu daftar atau bit map biner pada program memberitahu word program yang alamat-alamatnya direlokasi. Linker harus mencatat opcode, konstanta, dan item-item yang tak perlu direlokasi.

Masalah yang ditimbulkan :

Solusi ini menimbulkan masalah proteksi terhadap memori. Program tak terkendali selalu mampu membangun instruksi baru dan meloncati.

Tak ada cara untuk menghentikan jika program membaca atau menulis word di memori partisi lain (yang bukan hak-nya). Masalah relokasi dan proteksi tidak dapat dipisahkan, diperlukan satu solusi tunggal mengatas kedua masalah tersebut.

b. Proteksi.

Masalah proteksi pada banyak partisi dengan banyak proses di satu sistem secara bersamaan dikhawatirkan proses menggunakan atau memodifikasi daerah yang dikuasai proses lain (yang bukan haknya). Bila kejadian ini terjadi, maka proses lain dapat terganggu dan hasil yang diperolehnya dapat menjadi kacau.

Solusi IBM 360 :

Pada komputer IBM 360 membagi memori menjadi blok-blok, tiap blok ditambahi 4 bit kode proteksi. Blok berukuran 2 Kb. Proses jua mempunyai PSW (Program Status Word) yang antara lain berisi status proteksi. Status proteksi ini terdiri dari 4 bit (sama dengan bit kode proteksi untuk blok memori), merupakan kunci dalam pengaksesan memori. Proses hanya diijinkan mengakses blok-blok memori yang berkode proteksi sama dengan kode proteksi yang dimiliki PSW proses. Jika proses mengakses blok memori berkode proteksi berbeda dengan kunci PSW-nya, terjadi trap. Trap ini memberitahu sistem operasi bahwa telah terjadi pelanggaran memori, yaitu terdapat pengaksesan ke blok memori yang bukan wewenang proses yang menyebabkan trap.

Solusi menggunakan base register dan limit register :

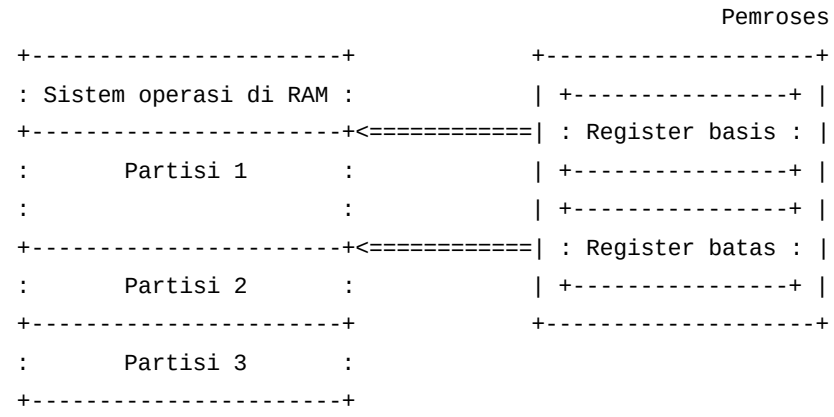
Solusi lain adalah menggunakan dua register yaitu base register dan limit register. Base register diisi alamat awal partisi dan limit register diisi panjang partisi. Setiap alamat yang dihasilkan secara otomatis ditambah dengan nilai base register. Instruksi yang mengacu pada alamat yang melebihi limit register akan menimbulkan trap yang memberitahu sistem operasi bahwa telah terjadi pelanggaran pengaksesan memori.

Teknik ini lebih unggul dibanding teknik pada IBM 360 karena sangat lebih efisien. Teknik ini tidak perlu menempatkan 4 bit proteksi di tiap blok memori. Teknik inipun lebih fleksibel.

Keuntungan :

a. Alamat tidak perlu dimodifikasi.
b. Setiap instruksi dapat diperiksa agar tidak meloncati batas limit register.
c. Program dapat dipindah walau sedang dieksekusi.
   Pemindahan dilakukan hanya dengan mengganti nilai base register.

Gambar 6.8 menunjukkan skema proteksi dan relokasi menggunakan register basis dan register batas. Register basis menunjuk alamat awal proses sedang register batas menunjuk alamat akhir proses. Bila proses mengacu alamat lebih dari alamat yang ditunjuk register batas maka pemroses mengirim sinyal fault yang memberitahu terjadinya pelanggaran pengaksesan memori.



Gambar 6.8 : Skema relokasi dan proteksi menggunakan register basis dan register batas.

6.9. Fragmentasi pada pemartisian statis.

Fragmentasi yaitu penyiaan/pemborosan memori akan terjadi pada setiap organisasi penyimpanan.

Fragmentasi pada pemartisian tetap terjadi adalah :

a. Fragmentasi internal.
   Proses tidak mengisi penuh partisi yang telah ditetapkan untuk proses.
b. Fragmentasi ekstenal.
   Partisi dapat tidak digunakan karena ukuran partisi lebih kecil dibanding ukuran proses yang menunggu di antrian, sehingga tidak digunakan.

Untuk sistem-sistem tanpa swapping (pemindahan lokasi proses), maka fragmentasi-fragmentasi tidak dapat dikurangi. Pada sistem-sistem dengan swapping, sistem lebih intelijen karena dapat melakukan beberapa altenatif mengatasi fragmentasi eksternal.

6.10. Multiprogramming dengan swapping

Pada sistem batch, organisasi memori dengan pemartisian tetap telah efektif. Selama jumlah proses yang terseduan dapat membuat pemroses sibuk, tak ada alasaan menggunakanan teknik lebih rumit. Pada sistem timesharing, situasinya berbeda, umumnya terdapat lebih banyak proses dibanding memori yang tersedia untuk memuat seluruh proses. Dengan demikian perlu menyimpan proses-proses yang tidak termuat ke disk. Untuk menjalankan proses-proses yang akan dieksekusi, proses-proses itu harus telah masuk memori utama.

Pemindahan proses dari memori utama ke disk dan sebaliknya di sebut swapping. Dengan swapping, multiprogramming pada sistem time sharing dapt ditingkatkan kinerjanya yaitu dengan memindah proses-proses blocked ke disk dan hanya memasukkkan proses-proses ready ke memori utama. Beragam masalah harus diatasi multiprogramming dengan swapping, antara lain :

a. Pemartisian secara dinamis.
b. Strategi pencatatan pemakaian memori.
c. Algoritma penempatan proses ke memori.
d. Strategi penempatan ruang swap pada disk.

6.11. Multiprogramming dengan pemartisian dinamis

Pemartisian statis tidak menarik karena terlalu banyak diboroskan proses-proses yang lebih kecil dibanding partisi yang ditempatinya. Dengan pemartisian dinamis maka jumlah, lokasi dan ukuran proses di memori dapat beragam sepanjang waktu secara dinamis. Proses yang akan masuk ke memori segera dibuatkan paritisi untuknya sesuai kebutuhannya. Teknik ini meningkatkan utilitasi memori.

Kelemahan pemartisian dinamis adalah :

a. Dapat terjadi lubang-lubang kecil memori di antara partisi-partisi yang dipakai.

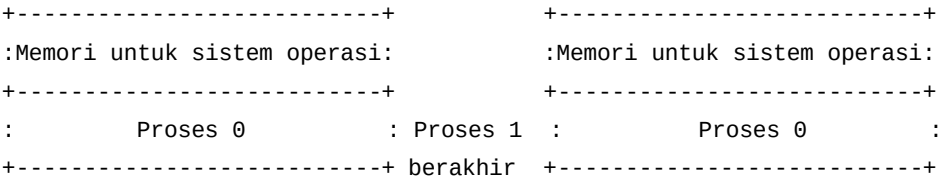
b. Merumitkan alokasi dan dealokasi memori.

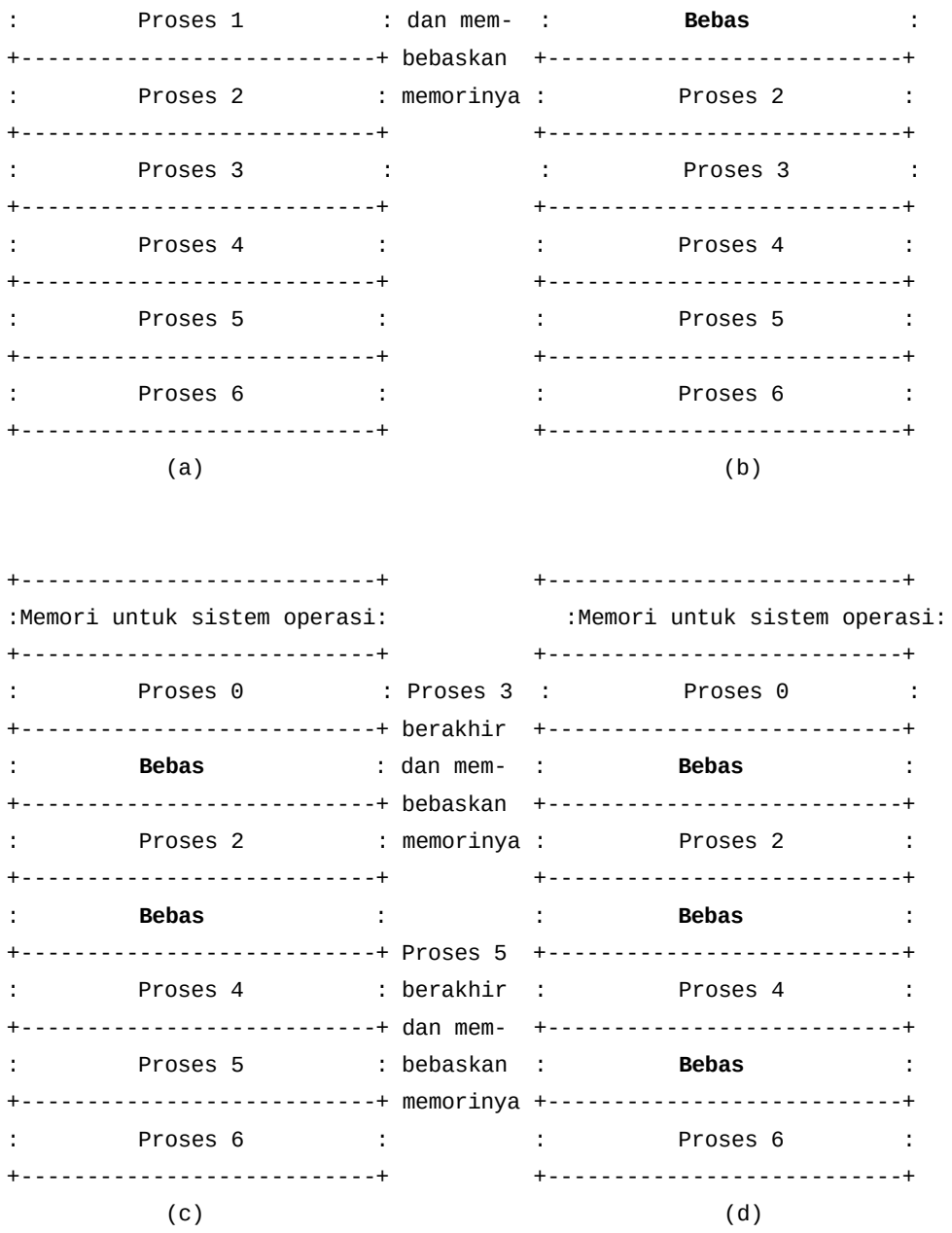
6.12. Terjadi lubang-lubang kecil memori

Contoh terjadinya lubang-lubang di antara partisi-partisi adalah gambar 6.9 :

* Gambar 6-9(a) adalah konfigurasi awal, terdapat 7 proses di ruang memori. Setelah proses 1 berakhir, konfigurasi memori menjadi gambar 6-9(b).
* Begitu proses 3 berakhir, konfigurasi memori menjadi gambar 6-9(c).
* Proses 5 berakhir, menghasilkan konfigurasi gambar 6-9(d). Memori dipenuhi lubang-lubang memori yang tak terpakai.

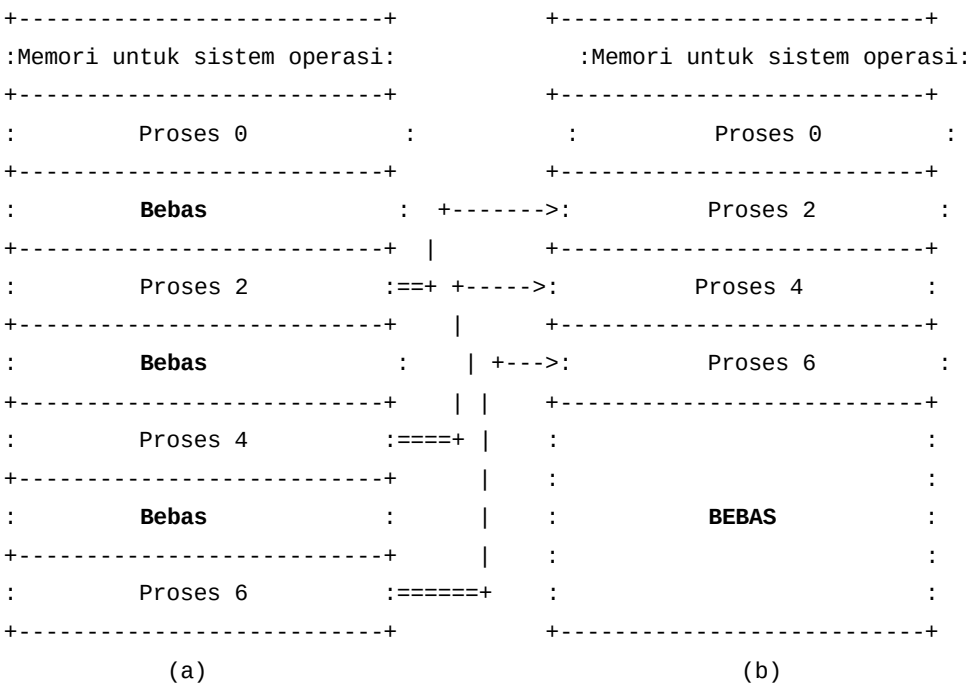
Lubang-lubang (yaitu kelompok blok-blok memori yang tidak digunakan) kecil di antara blok-blok memori yang digunakan dapat diatasi dengan pemadatan memori (memory compaction). Pemadatan memori adalah operasi menggabungkan semua lubang kecil menjadi satu lubang besar dengan memindahkan semua proses agar saling berdekatan.





Gambar 6.9 : Alokasi memori secara dinamis.

Gambar 6-10 menunjukkan skema pemadatan memori. Proses 2, 4, dan 6 dipindahkan agar menampati ruang-ruang berturutan dengan proses 0 sehingga diperoleh lubang memori besar. Lubang memori besar ini dapat ditempati proses yang akan masuk.



Gambar 6.10 : Lubang-lubang memori dan pemadatan memori.

Kelemahan utama teknik pemadatan memori :

- ☑ Memerlukan waktu yang sangat banyak.
- ☑ Sistem harus menghentikan sementara semua proses selagi melakukan pemadatan. Hal in meningkatkan waktu tanggapan di sistem interaktif dan tak mungkin digunakan di sistem waktu nyata real.

6.13. Proses tumbuh berkembang

Masalah lain pada pemartisian dinamis adalah proses dapat tumbuh berkembang.

Segmen data proses dapat tumbuh, misalnya karena :

* Heap untuk data dinamis berkembang.

* Stack untuk pemanggilan prosedur dan variabel lokal.

Solusi masalah ini adalah bila proses bersebelahan dengan lubang memori tak dipakai, proses tumbuh memakai lubang itu. Masalah menjadi parah bila proses bersebelahan dengan proses-proses lain.

Peringkat alternatif penyelesaian adalah :

- ☑ Bila masih terdapat lubang besar yang dapat memuat proses, maka proses dipindah ke lubang memori yang cukup dapat memuat.
- ☑ Satu proses atau lebih di swap ke disk agar memberi lubang cukup besar untuk proses yang berkembang.
- ☑ Jika proses tidak dapat tumbuh di memori dan daerah swap di disk telah penuh, proses harus menunggu atau disingkirkan.

6.14. Pencatatan pemakain memori

Memori yang tersedia harus dikelola, dilakukan dengan pencatatan pemakaian memori. Terdapat dua cara utama pencatatan pemakaian memori, yaitu :

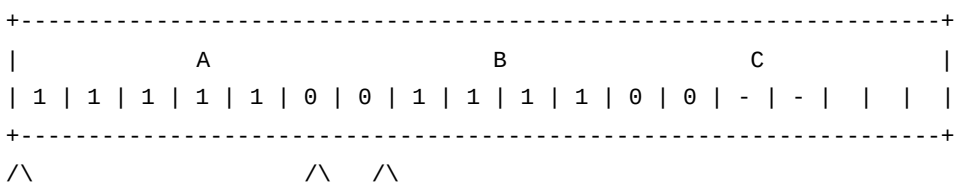
1. Pencatatan memakai peta bit.

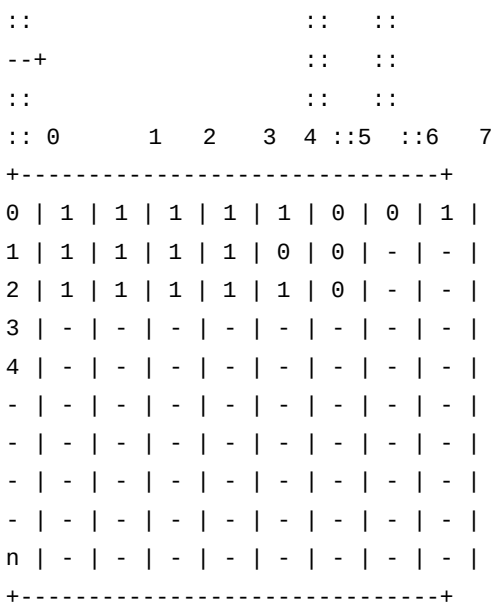
Memori dibagi menjadi unit-unit alokasi,berkorespondensi dengan tiap unit alokasi adalah satu bit pada bit map.

* Nilai 0 pada peta bit berarti unit itu masih bebas.

* Nilai 1 berarti unit digunakan.

Gambar 6-11 menunjukkan skema peta bit untuk pencatatan pemakaian memori. Elemen peta bit bernilai 1 menunjukkan blok tersebut telah digunakan oleh proses dan bernilai 0 yang berarti belum digunakan oleh proses.





Gambar 6-11 : Peta bit untuk pengelolaan pemakaian memori.

Masalah

Masalah pada peta bit adalah penetapan mengenai ukuran unit alokasi memori, yaitu :

- ☑ Unit lokasi memori berukuran kecil berarti membesarkan ukuran peta bit.
- ☑ Unit alokasi memori n berukuran besar berarti peta bit kecil tapi memori banyak disiakan pada unit terakhir jika ukuran proses bukan kelipatan unit alokasi.

Keunggulan :

* Dealokasi dapat dilakukan secara mudah, hanya tinggal menset bit yang berkorespondensi dengan unit yang telah tidak digunakan dengan 0.

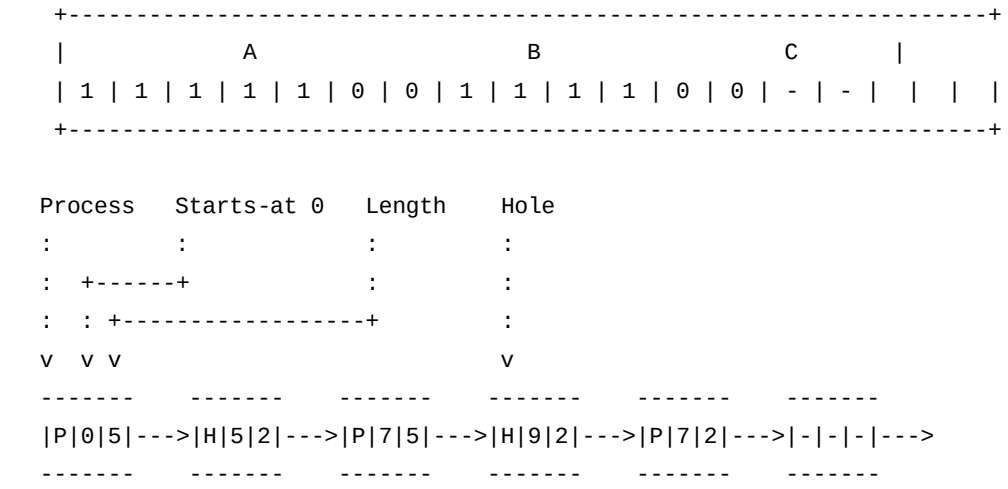
Kelemahan :

* Harus dilakukan penghitungan blok lubang memori saat unit memori bebas.
* Memerlukan ukutan bit map besar untuk memori yang besar.

2. Pencatatan memakai senarai berkait.

Sistem operasi mengelola senarai berkait (linked list) untuk segmen-segmen memori yang telah dialokasikan dan bebas. Segmen memori menyatakan memori

untuk proses atau memori yang bebas (lubang). Senarai segmen diurutkan sesuai alamat blok.



Gambar 6-12 : Pengelolaan pemakaian memori dengan senarai berkait.

Keunggulan :

* Tidak harus dilakukan perhitungan blok lubang memori karena sudah tercatat di node.
* Memori yang diperlukan relatif lebih kecil.

Kelemahan :

* Dealokasi sulit dilakukan karena terjadi berbagai operasi penggabungan node-nude di senarai.

6.15. Strategi alokasi memori

Terdapat berbagai strategi alokasi proses ke memori. Alokasi harus mencari sekumpulan blok memori yang ukurannya mencukupi memuat proses yaitu lubang kosong yang sama atau lebih besar dibanding ukuran memori yang diperlukan proses.

Beragam algoritma itu antara lain :

*** First-fit algorithm.**

Strategi in dapat dilakukan pada pencatatan memori dengan bit map maupun senarai berkait. Manajer memori menscan sampai menemukan lubang besar yang mencukupi penempatan proses. Lubang dibagi dua, untuk proses dan lubang tak digunakan, kecuali ketika lubang tersebut tepat sama dengan ukuran yang diperlukan proses.

Keunggulan :

- Algoritma ini akan menemukan lubang memori paling cepat dibanding algoritma-algoritma lain.

*** Next-fit algorithm.**

Strategi ini dapat dilakukan pada pencatatan memori dengan bit-map maupun senarai berkait. Mekanisme algoritma ini sama dengan algoritma first fit algorithm, hanya tidak dimulai di awal tapi dari posisi terakhir kali menemukan segmen paling cocok.

Simulasi oleh Bays (1977) menunjukkan next-fit algorithm berkinerja lebih buruk dibanding first-fit algorithm.

*** Best-fit algorithm.**

Strategi ini dapat dilakukan pada pencatatan memori dengan bit-map maupun senarai berkait. Algoritma mencari sampai akhir dan mengambil lubang terkecil yang dapat memuat proses. Algoritma ini mencoba menemukan lubang yang mendekati ukuran yang diperlukan.

*** Worst-fit algorithm.**

Strategi ini dapat dilakukan pada pencatatan memori dengan bit-map maupun senarai berkait. Selalu mencari lubang besar yang tersedia sehingga lubang dapat dipecah menjadi cukup besar, agar berguna untuk proses-proses berikutnya. Simulasi menunjukkan worst-fit algorithm bukan gagasan yang bagus.

*** Quick-fit algorithm.**

Strategi ini hanya untuk pencatatan memori dengan senarai berkait.

Keempat algoritma dapat dipercepat dengan mengelola dua senarai, yaitu :

- Senarai untuk proses.
- Senarai untuk lubang memori.

Dengan cara ini, saat alokasi hanya perlu menginspeksi senarai lubang, tidak perlu senarai proses.

Keunggulan :

· Teknik ini mempercepat pencarian lubang atau penempatan proses.

Kelemahan :

· Kompleksitas dealokasi memori bertambah dan melambatkan dealokasi memori karena memori yang dibebaskan harus dipindahkan dari senarai proses ke senarai lubang.

*** Quick fit.**

Cara diatas dapat diperluas, algoritma mengelola sejumlah senarai lubang memori dengan beragam ukuran yang paling sering diminta.

Contoh :

Algoritma mengelola senarai lubang sebagai berikut :

· Senarai 8 Kb.
· Senarai 12 Kb.
· Senarai 20 Kb.
· Senarai 40 Kb.
· Senarai 60 Kb.
· Dan seterusnya.

Senarai mencatat lubang-lubang memori sesuai ukuran lubang. Lubang-lubang memori dimuat di senarai sesuai ukuran terdekat, misalnya lubang memori 42 dimuat pada senarai 40 Kb. Dengan beragam senarai maka alokasi memori dapat dilakukan dengan cepat yaitu tinggal mencarai senarai terkecil yang dapat menampung proses tersebut.

Keunggulan :

· Algoritma ini sangat cepat dalam alokasi proses.

Kelemahan :

· Dealokasi sulit dilakukan.

Ketika proses berakhir atau dipindah keluar (swap-out) maka menemukan tetangga-tetangga memori yang dipakai proses untuk penggabungan adalah

sangat mahal/lama. Jika penggabungan tidak dilakukan, memori akan segera menjadi banyak lubang kecil yang tak berguna.

6.16. Sistem Buddy

Sistem buddy adalah algoritma pengelolaan memori yang memanfaatkan kelebihan penggunaan bilangan biner dalam pegalamatan memori. Karakteristik bilangan biner digunakan untuk mempercepat penggabungan lubang-lubang berdekatan ketika proses terakhir atau dikeluarkan.
Manajer memori mengelola senarai blok-blok bebas berukuran 1, 2, 4, 8, 16 byte dan seterusnya sampai kapasita memori. Pada komputer dengan 1 Mbyte memori maka dapat terdapat 21 senarai yaitu dari 1 byte sampai 1 Mbyte.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	lubang
Semula																	1
Minta 85 kb	A																-
Minta 45 kb	A	B															-
Minta 75 kb	A	B	C														-
A didealokasi		B	C														-
Minta 55 kb		B	D	C													-
B didealokasi			D	C													-
D didealokasi				C													-
C didealokasi																	-

Gambar 6.13 : Pengelolaan memori dengan sistem Buddy.

Mekanisme pengelolaan :

- Awalnya semua memori adalah bebas dan hanya satu senarai 1 Mbyte yang terisi berisi satu isian tunggal satu lubang 1 Mbyte. Senarai-senarai lain adalah kosong.

Misalnya proses baru berukuran 85 Kbyte mekanisme yang dijalankan adalah sbb :

- ☑ Karena hanya terdapat senarai berisi 2k, maka permintaan 85 kb dialokasikan ke yang terdekat yaitu berarti 128 kb, 2k terkecil yang mampu memuat.
- ☑ Karena tidak tersedia blok berukuran 128 kb, atau 256 kb atau 512 kb, maka blok 1 Mb dipecah menjadi dua blok 512 kb. Blok-blok pecahan ini disebut buddies. Satu beralamat mulai dari 0 dan lainnya mulai alamat 512 k.
- ☑ Salah satu blok 512 kb yang beralamat 0 dipecah lagi menjadi dua blok buddies 256 kb. Satu beralamat mulai dari 0 dan lainnya mulai alamat 256 kb.
- ☑ Blok 256 pada alamat 0, kemudian dipecah menjadi 2 blok buddies 128 kb.
- ☑ Blok yang pertama dialokasikan ke proses yang baru.

Keunggulan :

- ☑ Sistem buddy mempunyai keunggulan dibanding algoritma-algoritma yang mengurutkan blok-blok berdasarkan ukuran. Ketika blok berukuran 2k dibebaskan, maka manajer memori hanya mencari pada senarai lubang 2k untuk memeriksa apakah dapat dilakukan penggabungan. Pada algoritma-algoritma lain yang memungkinkan blok-blok memori dipecah dalam sembarang ukuran, seluruh senarai harus dicari.
- ☑ Dealokasi pada sistem buddy dapat dilakukan dengan cepat.

Kelemahan :

· Utilisasi memori pada sistem buddy sangat tidak efisien.

Masalah ini muncul dari dari kenyataan bahwa semua permintaan dibulatkan ke 2k terdekat yang dapat memuat. Proses berukuran 35 kb harus dialokasikan di 64 kb, terdapat 29 kb yang disiakan. Overhead ini disebut fragmentasi internal karena memori yang disiakan adalah internal terhadap segmen-segmen yang dialokasikan.

6.17. Alokasi ruang swap pada disk

Strategi dan algoritma yang dibahas adalah untuk mencatat memori utama.
Ketika proses akan dimasukkan ke memori utama (swap-in), sistem dapat menemukan ruang untuk proses-proses itu.

Terdapat dua strategi utama penempatan proses yang dikeluarkan dari memori utama (swap-out) ke disk, yaitu :

- **Ruang disk tempat swap dialokasikan begitu diperlukan.**

  Ketika proses harus dikeluarkan dari memori utama, ruang disk segera dialokasikan sesuai ukuran proses. Untuk itu diperlukan algoritma untuk mengelola ruang disk seperti untuk mengelola memori utama. Ketika proses dimasukkan kembali ke memori utama segera ruang disk untuk swap didealokasikan.

- **Ruang disk tempat swap dialokasikan lebih dulu.**

  Saat proses diciptakan, ruang swap pada disk dialokasikan. Ketika proses harus dikeluarkan dari memori utama, proses selalu ditempatkan ke ruang yang telah dialokasikan, bukan ke tempat-tempat berbeda setiap kali terjadi swap-out. Ketika proses berakhir, ruang swap pada disk didealokasikan.