MANAJEMEN MEMORI SISTEM OPERASI

Manajemen Memori

Memori adalah pusat dari operasi pada sistem komputer modern, berfungsi sebagai tempat penyimpanan informasi yang harus diatur dan dijaga sebaik-baiknya. Memori adalah array besar dari word atau byte, yang disebut alamat. CPU mengambil instruksi dari memory berdasarkan nilai dari program counter.
Sedangkan manajemen memori adalah suatu kegiatan untuk mengelola memori komputer. Proses ini menyediakan cara mengalokasikan memori untuk proses atas permintaan mereka, membebaskan untuk digunakan kembali ketika tidak lagi diperlukan serta menjaga alokasi ruang memori bagi proses. Pengelolaan memori utama sangat penting untuk sistem komputer, penting untuk memproses dan fasilitas masukan/keluaran secara efisien, sehingga memori dapat menampung sebanyak mungkin proses dan sebagai upaya agar pemogram atau proses tidak dibatasi kapasitas memori fisik di sistem komputer.

a. Jenis Memori

• Memori Kerja
- ROM/PROM/EPROM/EEPROM
- RAM
- Cache memory

• Memori Dukung
- Floppy
- Harddisk
- CD

b. Fungsi manajemen memori :

Manajemen memori merupakan salah satu bagian terpenting dalam sistem operasi. Memori perlu dikelola sebaik-baiknya agar :
1. Utilitas CPU meningkat.
2. Data dan instruksi dapat diakses dengan cepat oleh CPU.
3. Tercapai efisiensi dalam pemakaian memori yang terbatas.
4. Transfer data dari/ke memori utama ke/dari CPU dapat lebih efisien.
5. Mengelola informasi yang dipakai dan tidak dipakai.
6. Mengalokasikan memori ke proses yang memerlukan.
7. Mendealokasikan memori dari proses telah selesai.
8. Mengelola swapping atau paging antara memori utama dan disk.

1. Isi Memori

Instruksi eksekusi yang umum, contohnya, pertama mengambil instruksi dari memori. Instruksi dikodekan dan mungkin mengambil operand dari memory. Setelah instruksi dieksekusi pada operand, hasilnya ada yang dikirim kembali ke memory. Sebagai catatan, unit memory hanya

merupakan deretan alamat memory; tanpa tahu bagaimana membangkitkan (instruction counter, indexing, indirection, literal address dan lainnya) atau untuk apa (instruksi atau data). Oleh karena itu, kita dapat mengabaikan bagaimana alamat memori dibangkitkan oleh program, yang lebih menarik bagaimana deretan alamat memori dibangkitkan oleh program yang sedang berjalan.

a. Pengikatan Alamat (Address Binding)

Pengikatan alamat adalah cara instruksi dan data (yang berada di disk sebagai file yang dapat dieksekusi) dipetakan ke alamat memori. Sebagian besar sistem memperbolehkan sebuah proses user (user process) untuk meletakkan di sembarang tempat dari memori fisik. Sehingga, meskipun alamat dari komputer dimulai pada 00000, alamat pertama dari proses user tidak perlu harus dimulai 00000. Instruksi pengikatan instruksi dan data ke alamat memori dapat dilakukan pada saat :

o Compile time

Jika lokasi memori diketahui sejak awal, kode absolut dapat dibangkitkan, apabila terjadi perubahan alamat awal harus dilakukan kompilasi ulang.

o Load time

Harus membangkitkan kode relokasi jika lokasi memori tidak diketahui pada saat waktu kompilasi.

o Execution time

Pengikatan ditunda sampai waktu eksekusi jika proses dapat dipindahkan selama eksekusi dari satu segmen memori ke segmen memori lain.

b. Dinamic Loading

Untuk memperoleh utilitas ruang memori, dapat menggunakan dynamic loading. Dengan dynamic loading, sebuah rutin tidak disimpan di memori sampai dipanggil. Semua rutin disimpan pada disk dalam format relocatable load. Mekanisme dari dynamic loading adalah program utama di-load dahulu dan dieksekusi. Bila suatu routine perlu memanggil routine lain, routine yang dipanggil lebih dahulu diperiksa apakah rutin yang dipanggil sudah di-load. Jika tidak, relocatable linking loader dipanggil untuk me-load rutin yg diminta ke memori dan meng-ubah tabel alamat.

Keuntungan dari dynamic loading adalah rutin yang tidak digunakan tidak pernah di-load. Skema ini lebih berguna untuk kode dalam jumlah besar diperlukan untuk menangani kasus-kasus yang jarang terjadi seperti error routine. Dinamic loading tidak memerlukan dukungan khusus dari sistem operasi.

c. Dinamic Linking

Sebagian besar sistem operasi hanya men-support static linking, dimana sistem library language diperlakukan seperti obyek modul yang lain dan dikombinasikan dengan loader ke dalam binary program image. Dinamic linking biasanya digunakan dengan sistem library, seperti language subroutine library. Tanpa fasilitas ini, semua program pada sistem perlu mempunyai copy dari library language di dalam executable image. Bagaimanapun, tidak seperti dynamic loading, dynamic linking membutuhkan beberapa dukungan dari sistem operasi

d. Overlay
Sebuah proses dapat lebih besar daripada jumlah memori yang dialokasikan untuk proses, teknik overlay biasanya digunakan untuk kasus ini. Teknik Overlay biasanya digunakan untuk memungkinkan sebuah proses mempunyai jumlah yang lebih besar dari memori fisik daripada alokasi memori yang diperuntukkan. Overlay tidak membutuhkan dukungan khusus dari sistem operasi. User dapat mengimplementasikannya secara lengkap menggunakan struktur file sederhana, membaca dari file ke memori dan meloncat ke memori dan mengeksekusi instruksi read yang lebih baru.

2. Ruang Alamat Logika Dan Ruang Alamat Fisik

Alamat yang dibangkitkan oleh CPU disebut alamat logika (logical address) dimana alamat terlihat sebagai uni memory yang disebut alamat fisik (physical address). Tujuan utama manajemen memori adalah konsep meletakkan ruang alamat logika ke ruang alamat fisik. Hasil skema waktu kompilasi dan waktu pengikatan alamat pada alamat logika dan alamat memori adalah sama. Tetapi hasil skema waktu pengikatan alamat waktu eksekusi berbeda. dalam hal ini, alamat logika disebut dengan alamat maya (virtual address). Himpunan dari semua alamat logika yang dibangkitkan oleh program disebut dengan ruang alamat logika (logical address space); himpunan dari semua alamat fisik yang berhubungan dengan alamat logika disebut dengan ruang alamat fisik (physical address space). Memory Manajement Unit (MMU) adalah perangkat keras yang memetakan alamat virtual ke alamat fisik. Pada skema MMU, nilai register relokasi ditambahkan ke setiap alamat yang dibangkitkan oleh proses user pada waktu dikirim ke memori.

3. Swapping

Swapping merupakan pemindahan proses dari memori utama ke disk dan kembali lagi. Sebuah proses harus berada di memori untuk dieksekusi. Proses juga dapat ditukar (swap) sementara keluar memori ke backing store dan kemudian dibawa kembali ke memori untuk melanjutkan eksekusi. Backing store berupa disk besar dengan kecepatan tinggi yang cukup untuk meletakkan copy dari semua memory image untuk semua user, sistem juga harus menyediakan akses langsung ke memory image tersebut.

4. Alokasi Berurutan

Memori utama biasanya dibagi ke dalam dua partisi yaitu untuk
o Sistem operasi biasanya diletakkan pada alamat memori rendah dengan vektor interupsi
o Proses user yang diletakkan pada alamat memori tinggi.
Alokasi proses user pada memori berupa single partition allocation atau multiple partition allocation.

a. Single Partition Allocation
Pada single partition allocation diasumsikan sistem operasi ditempatkan di memori rendah dan proses user dieksekusi di memori tinggi. Kode dan data sistem operasi harus diproteksi dari

perubahan tak terduga oleh user proses.

b. Multiple Partition Allocation
Pada multiple partition allocation, mengijinkan memori user dialokasikan untuk proses yang berbeda yang berada di antrian input (input queue) yang menunggu dibawa ke memori. Terdapat dua skema yaitu partisi tetap (fixed partition) dimana memori dibagi dalam sejumlah partisi tetap dan setiap partisi berisi tepat satu proses. Jumlah partisi terbatas pada tingkat multiprogramming. Digunakan oleh IBM OS/360 yang disebut Multiprogramming with a Fixed number of Task (MFT). Skema yang kedua adalah partisi dinamis (variable partition) merupakan MFT yang digeneralisasi yang disebut Multiprogramming with a Variable number of Tasks (MVT).

c. Fragmentasi
Fragmentasi Eksternal terjadi pada situasi dimana terdapat cukup ruang memori total untuk memenuhi permintaan, tetapi tidak dapat langsung dialokasikan karena tidak berurutan. Fragmentasi eksternal dilakukan pada algoritma alokasi dinamis, terutama strategi first-fit dan best-fit. Fragmentasi Internal terjadi pada situasi dimana memori yang dialokasikan lebih besar dari pada memori yang diminta tetapi untuk satu partisi tertentu hanya berukuran kecil sehingga tidak digunakan.

5. Paging
a. Konsep Dasar Paging
Paging merupakan kemungkinan solusi untuk permasalahan fragmentasi eksternal dimana ruang alamat logika tidak berurutan; mengijinkan sebuah proses dialokasikan pada memori fisik yang terakhir tersedia. Memori fisik dibagi ke dalam blok-blok ukuran tetap yang disebut frame.

b. Implementasi Sistem Paging
Setiap sistem operasi mempunyai metode sendiri untuk menyimpan tabel page. Beberapa sistem operasi mengalokasikan sebuah tabel page untuk setiap proses. Pointer ke tabel page disimpan dengan nilai register lainnya dari PCB. Pada dasarnya terdapat 3 metode yang berbeda untuk implementasi tabel page :
• Tabel page diimplementasikan sebagai kumpulan dari “dedicated” register.
Register berupa rangkaian logika berkecepatan sangat tinggi untuk efisiensi translasi alamat paging.
• Tabel page disimpan pada main memori dan menggunakan page table base registe” (PTBR) untuk menunjuk ke tabel page yang disimpan di main memori. Penggunakan memori untuk mengimplementasikan tabel page akan memungkinkan tabel page sangat besar (sekitar 1 juta entry).
• Menggunakan perangkat keras cache yang khusus, kecil dan cepat yang disebut associative register atau translation look-aside buffers (TLBs). Merupakan solusi standar untuk permasalahan penggunaan memori untuk implementasi tabel page.

c. Proteksi

Pada model page, proteksi memori menggunakan bit proteksi yang diasosiasikan untuk setiap frame. Biasanya bit proteksi disimpan pada tabel page. Satu bit mendifinisikan satu page untuk "read and write" atau "read-only". Setiap acuan ke memori melalui tabel page untuk menemukan nomor frame yang benar. Level proteksi yang lebih baik dapat dicapai dengan menambah jumlah bit yang digunakan.

d. Multilevel Paging
Model multilevel paging digunakan pada sistem yang mempunyai ruang alamat logika yang sangat besar yaitu antara 232 s/d 264. Pada sistem ini, tabel page akan menjadi sangat besar. Misalnya untuk sistem dengan ruang alamat logika 32 bit dan ukuran page 4K byte, maka tabel page berisi 1 juta entry (232 / 212). Solusinya yaitu dengan melakukan partisi tabel ke beberapa beberapa bagian yang lebih kecil.

e. Shared Page
Pada skema paging, dimungkinkan untuk sharing kode umum. Bentuk ini penting terutama pada lingkungan time sharing. Satu copy read-only dibagi ke beberapa proses (misalnya editor teks, compiler dan sistem window). Kode yang dibagi harus berada pada lokasi ruang alamat logika yang sama untuk semua proses.

6. Segmentasi
Segmentasi adalah skema manajemen memori yang memungkinkan user untuk melihat memori tersebut. Ruang alamat logika adalah kumpulan segmen. Setiap segmen mempunyai nama dan panjang. Spesifikasi alamat berupa nama segmen dan offset. Segment diberi nomor dan disebut dengan nomor segmen (bukan nama segmen) atau segment number. Segmen dibentuk secara otomatis oleh compiler.

a. Konsep Dasar Segmentasi
Konsep segmentasi adalah user atau programmer tidak memikirkan sejumlah rutin program yang dipetakan ke main memori sebagai array linier dalam byte tetapi memori dilihat sebagai kumpulan segmen dengan ukuran berbeda-beda, tidak perlu berurutan diantara segment tersebut. Sebuah program adalah kumpulan segmen. Suatu segmen adalah unit logika seperti program utama, prosedur, fungsi, metode, obyek, variabel lokal, variabel global, blok umum, stack, tabel simbol, array dan lain-lain
.

b. Arsitektur Segmentasi
Alamat logika terdiri dari dua bagian yaitu nomor segmen (s) dan offset (d) yang dituliskan dengan .
Pemetaan alamat logika ke alamat fisik menggunakan tabel segmen (segment table), terdiri dari :
o Segmen basis (base) berisi alamat fisik awal
o Segmen limit merupakan panjang segmen Seperti tabel page, tabel segmen dapat berupa register atau memori berkecepatan tinggi.
o Segment-table base register (STBR) digunakan untuk menyimpan alamat yang menunjuk ke

segment table.
o Segment-table length register (STLR) digunakan untuk menyimpan nilai jumlah segmen yang digunakan program.
o Untuk alamat logika (s, d), pertama diperiksa apakah segment number s legal (s < STLR), kemudian tambahkan segment number ke STBR, alamat hasil (STBR + s) ke memori dari segment table.

c. Proteksi dan Sharing
Proteksi bit dapat diletakkan pada tabel segmen. Segmen instruksi dapat diproteksi sebagai segmen read-only atau execute only, segmen data dapat diproteksi sebagai segmen read-write. Pemetaan pada perangkat keras memory akan memeriksa bit proteksi untuk mencegah akses yang illegal.