Modul Pertemuan 10
Memori Virtual, Paging, dan Segmentasi

MATA KULIAH : SISTEM OPERASI

25 JUNI 2025

JURUSAN TEKNIK INFORMATIKA

STMIK TAZKIA BOGOR

2025

DAFTAR ISI

BAB 1. Pendahuluan: Mengapa Komputer Butuh Memori Virtual?

1.1 Keterbatasan Memori Fisik (RAM) pada Komputer

Dalam dunia komputasi, Random Access Memory (RAM) dikenal sebagai memori utama yang sangat cepat, tempat data dan instruksi program disimpan sementara saat komputer sedang berjalan. Namun, RAM memiliki beberapa keterbatasan mendasar. Pertama, kapasitas RAM relatif terbatas dibandingkan dengan kebutuhan program modern yang semakin kompleks. Meskipun kapasitas RAM terus meningkat, harganya cenderung lebih mahal dibandingkan media penyimpanan sekunder seperti hard drive atau Solid State Drive (SSD).[1] Kedua, RAM bersifat *volatile*, yang berarti data yang tersimpan di dalamnya akan hilang seketika saat komputer dimatikan atau kehilangan daya.[3]

Keterbatasan kapasitas dan sifat *volatile* RAM ini menjadi pendorong utama pengembangan solusi manajemen memori yang lebih canggih. Tanpa adanya solusi ini, program-program besar tidak akan dapat berjalan, atau kinerja sistem akan sangat terganggu. Ini menunjukkan adanya kendala teknis dan ekonomis yang mendasari perlunya mekanisme yang dapat "memperluas" memori secara logis tanpa selalu harus menambah RAM fisik.

1.2 Konsep Dasar Memori Virtual: Definisi dan Tujuan

Memori virtual adalah sebuah strategi cerdas dalam sistem operasi yang dirancang untuk mengatasi keterbatasan memori fisik. Konsep ini memungkinkan sistem komputer untuk menggunakan sebagian ruang penyimpanan sekunder, seperti hard drive atau SSD, seolah-olah ruang tersebut adalah bagian dari memori utama (RAM).[1]

Tujuan utama dari memori virtual adalah untuk memberikan ilusi kepada setiap program bahwa ia memiliki akses ke ruang memori yang jauh lebih besar dan berurutan daripada kapasitas RAM fisik yang sebenarnya tersedia.[2] Dengan ilusi ini, program yang ukurannya melebihi kapasitas RAM fisik dapat tetap berjalan dengan lancar.[5] Memori virtual menciptakan sebuah lapisan abstraksi yang menyembunyikan detail kompleks tentang bagaimana dan di mana data program disimpan secara fisik. Hal ini memungkinkan pengembang perangkat lunak untuk fokus pada logika aplikasi mereka tanpa perlu terus-menerus memikirkan batasan memori fisik atau lokasi spesifik data di RAM. Abstraksi ini adalah pilar fundamental dalam desain sistem

operasi modern, yang pada gilirannya memfasilitasi pengembangan aplikasi yang lebih kompleks dan tangguh.

1.3 Analogi Sederhana Memori Virtual

Untuk memahami konsep memori virtual, bayangkan meja kerja Anda sebagai RAM. Meja kerja ini memiliki ruang terbatas, sehingga Anda hanya bisa meletakkan beberapa buku atau dokumen penting yang sedang Anda kerjakan secara langsung di atasnya. Dokumen-dokumen ini adalah yang paling sering Anda akses dan butuhkan dengan cepat.[2]

Di sisi lain, hard drive atau SSD komputer Anda dapat dianalogikan sebagai lemari arsip yang sangat besar. Lemari ini mampu menyimpan jauh lebih banyak dokumen dibandingkan meja kerja Anda. Namun, untuk mengambil atau menyimpan sesuatu di lemari arsip, Anda harus berjalan ke sana, membuka laci, mencari dokumen yang diinginkan, dan kemudian membawanya kembali ke meja. Proses ini membutuhkan waktu yang lebih lama dibandingkan mengambil dokumen yang sudah ada di meja.[2]

Memori virtual bekerja seperti memiliki seorang asisten yang sangat efisien. Asisten ini secara otomatis mengawasi dokumen-dokumen di meja Anda. Jika ada dokumen yang jarang Anda gunakan, asisten akan memindahkannya dari meja ke lemari arsip untuk membebaskan ruang di meja. Sebaliknya, jika Anda tiba-tiba membutuhkan dokumen yang ada di lemari arsip, asisten akan segera mengambilnya dan meletakkannya di meja Anda. Anda sebagai pengguna tidak perlu khawatir tentang di mana dokumen itu disimpan secara fisik; asisten yang mengaturnya.[2]

Analogi ini tidak hanya membantu menjelaskan konsep memori virtual, tetapi juga secara tidak langsung memperkenalkan ide *swapping* (perpindahan data antara RAM dan disk) dan *latency* (waktu yang dibutuhkan untuk mengakses data). Perpindahan dokumen dari lemari arsip ke meja (atau sebaliknya) membutuhkan waktu, yang mencerminkan perbedaan kecepatan akses antara hard drive/SSD dan RAM. Ini adalah dasar untuk memahami mengapa memori virtual, meskipun sangat berguna, juga memiliki potensi dampak pada kinerja sistem.

BAB 2. Memori Virtual: Cara Kerja dan Manfaatnya

2.1 Fungsi Utama Memori Virtual dalam Sistem Operasi

Memori virtual memainkan peran krusial dalam sistem operasi modern, melampaui sekadar "memperbesar" kapasitas memori. Fungsi-fungsi utamanya meliputi:

- **Meningkatkan Kapasitas Memori Sistem:** Dengan mengalokasikan sebagian ruang penyimpanan disk sebagai memori virtual, sistem dapat menampung lebih banyak data dan instruksi program daripada hanya mengandalkan RAM fisik. Ini secara efektif memperluas kapasitas memori yang tersedia untuk aplikasi.[1]
- **Mengurangi Beban RAM Fisik:** Data atau bagian program yang jarang digunakan dapat dipindahkan ke memori virtual di disk, sehingga membebaskan lebih banyak ruang di RAM fisik untuk data yang sering diakses. Ini meningkatkan efisiensi dan kinerja sistem secara keseluruhan.[1]
- **Mendukung Multitasking Aplikasi Secara Bersamaan:** Memori virtual memungkinkan sistem untuk menjalankan banyak aplikasi secara bersamaan, bahkan jika total kebutuhan memori gabungan dari aplikasi-aplikasi tersebut melebihi kapasitas RAM fisik yang tersedia. Setiap aplikasi mendapatkan alokasi memori yang cukup, memungkinkan pengguna untuk beralih antar aplikasi dengan lebih lancar.[1]
- **Memberikan Isolasi Memori Antar Proses:** Salah satu fungsi vital memori virtual adalah menciptakan ruang alamat virtual yang terisolasi untuk setiap proses. Ini berarti satu program tidak dapat secara tidak sengaja atau sengaja mengakses atau memodifikasi memori yang dialokasikan untuk program lain. Isolasi ini adalah pilar penting dalam menjaga stabilitas dan keamanan sistem operasi modern, mencegah *crash* sistem yang disebabkan oleh satu aplikasi yang bermasalah atau aktivitas berbahaya.[2]

Fungsi-fungsi ini secara kolektif menunjukkan bahwa memori virtual bukan hanya tentang penambahan kapasitas, tetapi juga tentang manajemen sumber daya yang efisien dan keamanan sistem yang kokoh. Kemampuan untuk mengisolasi proses adalah aspek fundamental yang memungkinkan sistem operasi modern beroperasi dengan stabil dan aman, mencegah satu kesalahan dalam satu program merusak seluruh sistem.

2.2 Mekanisme Swapping: Memindahkan Data antara RAM dan Disk

Mekanisme inti yang memungkinkan memori virtual beroperasi adalah *swapping*. Swapping adalah proses dinamis di mana sistem operasi memindahkan data atau program antara RAM fisik dan ruang penyimpanan sekunder (disk).[5]

Ketika RAM penuh atau sistem membutuhkan ruang lebih untuk proses lain, sistem operasi akan memilih data atau bagian program yang saat ini ada di RAM tetapi jarang digunakan, lalu memindahkannya ke area khusus di hard disk yang disebut *swap space* atau *paging file*. Proses pemindahan ini dikenal sebagai *swap out*.[11] Sebaliknya, ketika data atau bagian program yang telah di-*swap out* ini dibutuhkan kembali oleh CPU untuk dieksekusi, sistem operasi akan memindahkannya kembali dari *swap space* di disk ke RAM. Proses ini disebut *swap in*.[5]

Swapping adalah bagian krusial dari implementasi memori virtual karena inilah cara fisik bagaimana ilusi memori yang besar dicapai. Namun, mekanisme ini juga secara langsung menunjukkan potensi *bottleneck* kinerja. Kecepatan akses data di RAM diukur dalam nanodetik, sementara akses ke disk diukur dalam milidetik. Perbedaan kecepatan yang sangat besar ini berarti bahwa *swapping* yang berlebihan dapat secara signifikan memperlambat kinerja sistem. Hal ini menimbulkan kebutuhan akan teknik manajemen memori yang lebih canggih, seperti *paging*, untuk meminimalkan perpindahan data yang tidak perlu ke dan dari disk.

2.3 Keuntungan Memori Virtual

Penerapan memori virtual membawa berbagai keuntungan signifikan bagi sistem komputer:

- **Multitasking yang Lebih Baik:** Salah satu manfaat paling nyata adalah kemampuan untuk menjalankan banyak aplikasi secara bersamaan dengan lebih lancar. Memori virtual memungkinkan sistem untuk mengalokasikan memori yang cukup untuk setiap aplikasi atau program yang berjalan, bahkan jika total kebutuhan memori mereka melebihi kapasitas RAM fisik. Hal ini memberikan pengalaman pengguna yang lebih mulus saat beralih antar aplikasi.[1]
- **Efisiensi Biaya:** Dengan memanfaatkan ruang penyimpanan disk sebagai ekstensi memori, kebutuhan untuk membeli RAM fisik yang lebih banyak dapat dikurangi. Hard drive dan SSD jauh lebih murah per gigabyte dibandingkan RAM, sehingga memori virtual menawarkan solusi yang lebih efisien secara biaya untuk

"memperluas" memori sistem.[1]

- **Fleksibilitas:** Pengguna atau sistem operasi dapat menyesuaikan ukuran *swap space* sesuai dengan kebutuhan spesifik. Ini memberikan fleksibilitas dalam mengelola kapasitas memori virtual agar sesuai dengan beban kerja sistem.[1]
- **Penyederhanaan Pengembangan Program:** Bagi para programmer, memori virtual sangat menyederhanakan proses pengembangan aplikasi. Programmer tidak perlu lagi khawatir tentang batasan memori fisik yang sebenarnya atau di mana data mereka akan disimpan secara fisik. Mereka dapat memprogram seolah-olah ada blok memori yang sangat besar dan berurutan yang tersedia, membuat proses pengkodean lebih mudah dan memungkinkan pengembangan aplikasi yang lebih kompleks dan canggih.[10]

Keuntungan-keuntungan ini secara kolektif menunjukkan bahwa memori virtual adalah solusi holistik yang mengatasi berbagai masalah teknis, ekonomi, dan pengembangan perangkat lunak secara bersamaan. Ini bukan hanya fitur teknis, tetapi juga enabler penting bagi ekosistem komputasi modern, memungkinkan komputer menjadi lebih kuat, lebih mudah digunakan, dan lebih terjangkau bagi banyak pengguna.

2.4 Tantangan Memori Virtual

Meskipun memori virtual menawarkan banyak keuntungan, ada beberapa tantangan dan potensi kelemahan yang perlu diperhatikan:

- **Kecepatan Akses yang Lebih Lambat:** Tantangan utama adalah perbedaan kecepatan antara RAM dan disk. Akses ke memori virtual melibatkan proses pembacaan dan penulisan ke media penyimpanan (hard disk atau SSD), yang jauh lebih lambat dibandingkan akses ke RAM fisik. Dampaknya, penggunaan memori virtual dapat menurunkan kinerja sistem, terutama jika sistem seringkali harus mengakses data dari *swap space*.[1]
- **Thrashing:** Ini adalah kondisi kritis di mana sistem menghabiskan sebagian besar waktunya untuk memindahkan data antara RAM dan disk (melakukan *swapping*) daripada benar-benar menjalankan instruksi program. Terlalu seringnya *page fault* (permintaan data yang tidak ada di RAM) dapat memicu *thrashing*, menyebabkan kinerja sistem menurun drastis dan terasa sangat lambat atau "macet".[5]
- **Konsumsi Daya Lebih Tinggi:** Ketika ruang penyimpanan digunakan sebagai memori virtual, aktivitas baca/tulis ke disk akan meningkat secara signifikan.

Aktivitas disk yang lebih tinggi ini dapat menyebabkan peningkatan konsumsi daya, yang menjadi perhatian khusus pada perangkat portabel seperti laptop.[1]

Tantangan-tantangan ini menunjukkan bahwa memori virtual adalah sebuah kompromi. Manfaatnya dalam memperluas kapasitas dan mendukung multitasking datang dengan biaya kinerja yang harus dikelola secara cermat oleh sistem operasi. Konsep *thrashing* adalah mode kegagalan kritis yang harus dihindari oleh desainer sistem operasi, yang mendorong pengembangan algoritma manajemen memori yang cerdas (seperti algoritma penggantian halaman) untuk meminimalkan interaksi disk yang lambat dan menjaga responsivitas sistem.

BAB 3. Paging: Memori Virtual dengan Halaman

3.1 Konsep Dasar Paging: Pembagian Memori menjadi Halaman (Pages) dan Bingkai (Frames)

Paging adalah salah satu skema manajemen memori yang paling umum digunakan untuk mengimplementasikan memori virtual. Konsep dasarnya adalah membagi memori menjadi blok-blok berukuran tetap. Memori fisik (RAM) dibagi menjadi unit-unit berukuran tetap yang disebut *frame*.[8] Sementara itu, ruang alamat logika (ruang memori yang dilihat oleh program) juga dibagi menjadi blok-blok berukuran sama yang disebut *page*.[8] Ukuran *page* ini biasanya ditentukan oleh sistem operasi dan perangkat keras komputer, dengan ukuran standar yang umum adalah 4 Kilobyte (KB).[21]

Pembagian memori menjadi blok-blok berukuran tetap ini adalah fitur kunci yang membedakan paging dari metode alokasi memori sebelumnya dan dari segmentasi. Desain ini secara langsung bertujuan untuk mengatasi masalah *fragmentasi eksternal*. Dengan memiliki blok memori yang seragam, setiap *frame* kosong di RAM dapat mengakomodasi *page* mana pun dari suatu program, menyederhanakan proses alokasi dan dealokasi memori, serta meningkatkan efisiensi penggunaan memori dengan mencegah terbentuknya celah-celah besar yang tidak dapat digunakan.

3.2 Bagaimana Paging Mengatasi Fragmentasi Memori

Paging secara efektif mengatasi masalah fragmentasi eksternal, yang merupakan salah satu tantangan besar dalam manajemen memori tradisional. Fragmentasi eksternal terjadi ketika ada cukup total memori bebas untuk memenuhi permintaan, tetapi memori tersebut tersebar dalam blok-blok kecil yang tidak berurutan, sehingga tidak dapat dialokasikan untuk satu proses besar.[18]

Dalam paging, sebuah proses tidak perlu dialokasikan pada *frame* memori fisik yang berurutan. Bagian-bagian (pages) dari sebuah program dapat tersebar di berbagai *frame* yang tersedia di RAM, tanpa harus berdekatan satu sama lain.[18] Ini menghilangkan masalah fragmentasi eksternal karena setiap *frame* kosong dapat digunakan, terlepas dari lokasinya.

Meskipun paging sangat baik dalam mengatasi fragmentasi eksternal, ia masih dapat menyebabkan *fragmentasi internal*. Fragmentasi internal terjadi ketika *page* terakhir dari sebuah program tidak terisi penuh, menyisakan sedikit ruang yang tidak terpakai di dalam *frame* yang dialokasikan.[18] Misalnya, jika ukuran

page adalah 4KB dan sebuah program hanya membutuhkan 3KB untuk *page* terakhirnya, maka 1KB sisanya di dalam *frame* tersebut akan terbuang dan tidak dapat digunakan oleh program lain. Keberadaan fragmentasi internal ini menunjukkan bahwa setiap solusi manajemen memori memiliki *trade-off* dan tidak ada yang benar-benar "sempurna". Paging memitigasi satu masalah besar (fragmentasi eksternal) dengan memperkenalkan masalah lain yang lebih kecil (fragmentasi internal) yang umumnya lebih mudah dikelola.

3.3 Translasi Alamat: Peran Tabel Halaman (Page Table) dan MMU (Memory Management Unit)

Proses kunci dalam paging adalah translasi alamat, yaitu mengubah alamat logika (virtual) yang dihasilkan oleh CPU menjadi alamat fisik yang sebenarnya di RAM. Setiap alamat yang dihasilkan oleh CPU dibagi menjadi dua bagian: *page number* (nomor halaman) dan *page offset* (offset dalam halaman).[13]

- **Page Number (p):** Bagian ini digunakan sebagai indeks untuk mencari entri yang sesuai di dalam sebuah struktur data yang disebut *page table* (tabel halaman).[13]
- **Page Table:** Tabel ini berisi alamat basis (*base address*) dari setiap *page* di memori fisik, yang sebenarnya adalah nomor *frame* di mana *page* tersebut berada.[8]
- **Page Offset (d):** Bagian ini menunjukkan posisi spesifik data di dalam *page* tersebut. *Page offset* kemudian dikombinasikan dengan alamat basis (nomor *frame*) yang ditemukan di *page table* untuk membentuk alamat memori fisik yang lengkap dan sebenarnya.[15]

Seluruh proses translasi alamat ini dilakukan oleh perangkat keras khusus yang disebut **Memory Management Unit (MMU)**. MMU seringkali terintegrasi langsung ke dalam CPU.[2] Peran MMU sangat penting karena translasi alamat adalah operasi yang sangat sering dan kritis. Jika proses ini dilakukan sepenuhnya oleh perangkat lunak,

overhead kinerja akan sangat signifikan, membuat memori virtual menjadi tidak praktis. Oleh karena itu, MMU sebagai komponen *hardware* memastikan bahwa

translasi alamat dapat dilakukan dengan kecepatan tinggi, memungkinkan efisiensi memori virtual.

3.4 Analogi Sederhana Paging

Untuk memahami paging, bayangkan sebuah buku yang sangat besar (program komputer Anda) yang tidak mungkin diletakkan seluruhnya di meja kerja Anda (RAM) karena keterbatasan ruang.[22] Buku ini telah dibagi menjadi halaman-halaman kecil yang berukuran sama (ini adalah *pages*). Meja kerja Anda memiliki slot-slot kosong yang juga berukuran sama (ini adalah *frames*), tempat Anda bisa meletakkan halaman-halaman buku tersebut.[10] Anda tidak perlu meletakkan seluruh buku di meja. Anda hanya perlu meletakkan halaman-halaman yang sedang Anda baca atau yang akan Anda baca sebentar lagi. Untuk mengetahui halaman nomor berapa ada di slot meja nomor berapa, Anda memiliki daftar isi khusus (ini adalah *page table*). Jika Anda ingin membaca halaman tertentu dan halaman itu tidak ada di meja, Anda harus pergi ke tumpukan buku di lemari arsip (disk) untuk mengambil halaman tersebut dan meletakkannya di slot meja yang kosong.[22]

Analogi ini secara intuitif menjelaskan beberapa konsep penting: *demand paging* (hanya memuat halaman yang dibutuhkan), *page fault* (ketika halaman yang dicari tidak ada di RAM), dan *page table* sebagai "peta" yang menerjemahkan lokasi virtual ke lokasi fisik. Ini membantu memahami bagaimana sistem menjadi efisien dengan hanya memuat bagian-bagian program yang diperlukan, alih-alih memuat seluruh program sekaligus.

3.5 Penanganan Page Fault dan Konsep Penggantian Halaman

Page fault adalah peristiwa yang terjadi ketika CPU mencoba mengakses sebuah *page* yang merupakan bagian dari ruang alamat virtual program, tetapi *page* tersebut saat ini tidak berada di memori fisik (RAM), melainkan masih tersimpan di disk.[7] Ini bukan berarti ada kesalahan dalam program, melainkan sinyal bagi sistem operasi untuk melakukan tindakan.

Ketika *page fault* terjadi, alur penanganannya adalah sebagai berikut:

1. Sistem operasi akan menghentikan sementara proses yang sedang berjalan yang menyebabkan *page fault*.
2. Sistem operasi kemudian mencari *page* yang dibutuhkan di disk.
3. Setelah *page* ditemukan, sistem operasi akan memuatnya ke dalam *frame* kosong

yang tersedia di RAM.

4. *Page table* yang sesuai akan diperbarui untuk mencerminkan lokasi *page* yang baru dimuat di RAM.
5. Akhirnya, proses yang terinterupsi akan dilanjutkan seolah-olah tidak ada yang terjadi.[8]

Namun, bagaimana jika tidak ada *frame* kosong di RAM saat *page fault* terjadi? Dalam situasi ini, sistem operasi harus memilih salah satu *page* yang sudah ada di RAM untuk "dikorbankan" dan dipindahkan kembali ke disk (*page replacement*). Setelah *frame* tersebut kosong, barulah *page* yang baru dapat dimuat.[7] Proses *page replacement* ini sangat penting untuk mengelola ruang RAM yang terbatas secara efektif. Meskipun *page fault* adalah bagian normal dari operasi *demand paging*, frekuensinya harus dijaga agar tidak menyebabkan kondisi *thrashing* yang dapat menurunkan kinerja sistem secara drastis. Ini menunjukkan bagaimana sistem operasi secara aktif dan dinamis mengelola sumber daya memori untuk mengoptimalkan kinerja.

BAB 4. Segmentasi: Memori Virtual dengan Segmen Logis

4.1 Konsep Dasar Segmentasi: Pembagian Memori berdasarkan Struktur Logis Program

Segmentasi adalah skema manajemen memori lain yang digunakan dalam sistem operasi, yang memiliki pendekatan berbeda dari paging. Alih-alih membagi memori menjadi blok-blok berukuran tetap, segmentasi mendukung cara pandang programmer terhadap memori.[8] Dalam segmentasi, ruang alamat logika dibagi menjadi unit-unit logis yang disebut *segmen*, dan setiap segmen memiliki panjang yang bervariasi.[8]

Segmen-segmen ini merepresentasikan bagian-bagian logis dari sebuah program, seperti:

- Program utama (main program)
- Prosedur atau fungsi
- Blok data (misalnya, variabel global, *array*)
- *Stack* (untuk panggilan fungsi dan variabel lokal)
- *Heap* (untuk alokasi memori dinamis)
- Tabel simbol [19]

Perbedaan utama segmentasi dari paging adalah fokusnya pada struktur logis program dan ukuran variabel. Ini mencerminkan upaya untuk menjembatani jurang antara cara manusia (programmer) berpikir tentang program dan cara komputer (memori fisik) menyimpannya. Dengan membagi memori berdasarkan unit-unit logis ini, sistem operasi dapat memberikan perlindungan dan *sharing* memori yang lebih intuitif dan kuat pada tingkat semantik.

4.2 Pandangan Programmer terhadap Segmentasi

Dalam model segmentasi, seorang programmer melihat memori sebagai kumpulan segmen yang independen satu sama lain. Setiap segmen memiliki nama dan panjangnya sendiri.[9] Alamat dalam segmentasi tidak lagi berupa satu angka linier, melainkan diartikan sebagai pasangan dua dimensi: (nomor segmen, *offset* dalam segmen).[13]

Pandangan programmer ini sangat penting karena memengaruhi bagaimana program dirancang dan bagaimana sistem operasi dapat memberikan perlindungan dan *sharing* yang lebih granular. Misalnya, segmen yang berisi kode program dapat ditandai sebagai "hanya baca" dan "dapat dieksekusi", sementara segmen data dapat ditandai sebagai "baca-tulis". Ini mencegah modifikasi kode yang tidak disengaja atau eksekusi data sebagai instruksi. Selain itu, segmen kode yang umum (misalnya, pustaka fungsi) dapat dibagi pakai (*shared*) antar beberapa proses dengan mudah, karena setiap proses hanya perlu menunjuk ke segmen yang sama. Ini adalah contoh bagaimana desain sistem operasi dapat memfasilitasi pengembangan perangkat lunak yang lebih modular, aman, dan efisien.

4.3 Translasi Alamat: Peran Tabel Segmen (Segment Table) dan MMU

Seperti halnya paging, segmentasi juga memerlukan mekanisme translasi alamat untuk mengubah alamat logika (nomor segmen, *offset*) menjadi alamat fisik yang sebenarnya di RAM.[13] Pemetaan ini dilakukan menggunakan struktur data yang disebut *segment table* (tabel segmen).[13]

Setiap entri di *segment table* berisi dua informasi penting untuk setiap segmen:

- **Alamat Basis (Base Address):** Ini adalah alamat fisik awal di RAM tempat segmen tersebut dimuat.
- **Batas (Limit):** Ini menunjukkan panjang atau ukuran total segmen tersebut.[13]

Proses translasi alamat ini juga dilakukan oleh Memory Management Unit (MMU). Ketika CPU menghasilkan alamat logika (nomor segmen, *offset*), MMU akan melakukan langkah-langkah berikut:

1. MMU menggunakan nomor segmen sebagai indeks untuk mencari entri yang sesuai di *segment table*.
2. Sebelum menambahkan *offset* ke alamat basis, MMU akan melakukan pemeriksaan batas (*limit check*). MMU akan memverifikasi apakah *offset* yang diminta berada dalam batas yang diizinkan oleh panjang segmen. Jika *offset* melebihi batas segmen, MMU akan menghasilkan kesalahan (*segmentation fault*), mencegah program mengakses memori di luar segmennya.[15]
3. Jika pemeriksaan batas berhasil, MMU akan menambahkan *offset* ke alamat basis segmen untuk mendapatkan alamat fisik yang sebenarnya di RAM.[15]

Translasi alamat pada segmentasi lebih kompleks dibandingkan paging murni karena

melibatkan pemeriksaan batas untuk setiap akses memori. Meskipun ini berkontribusi pada *overhead* kinerja, pemeriksaan batas ini merupakan mekanisme penting untuk *proteksi memori*, memastikan bahwa program hanya dapat mengakses area memori yang memang dialokasikan untuknya.

4.4 Analogi Sederhana Segmentasi

Untuk memahami segmentasi, bayangkan lemari arsip Anda (memori) tidak hanya dibagi menjadi laci-laci berukuran sama. Sebaliknya, lemari ini dibagi menjadi rak-rak dengan label kategori yang berbeda, seperti "Prosedur", "Data", "Stack", dan "Variabel Global".[24]

Setiap rak ini (yang merepresentasikan sebuah *segmen*) memiliki panjang yang berbeda-beda, disesuaikan dengan jumlah dokumen atau "buku" yang ada di dalamnya. Misalnya, rak "Prosedur" mungkin lebih panjang jika program memiliki banyak fungsi, sementara rak "Stack" akan tumbuh dan menyusut secara dinamis sesuai dengan panggilan fungsi.[24]

Sebagai pengguna, Anda bisa langsung tahu di rak mana Anda harus mencari "data" atau "prosedur" tertentu karena setiap rak memiliki label logis. Anda juga tahu batas-batas setiap rak, sehingga Anda tidak akan secara tidak sengaja mencari dokumen di luar batas rak yang seharusnya.[24]

Analogi ini menekankan aspek logis dan ukuran variabel dari segmentasi, serta bagaimana hal itu memfasilitasi organisasi dan proteksi memori yang lebih baik dari sudut pandang pengguna atau programmer. Ini adalah cara yang lebih "manusiawi" dalam mengorganisir memori dibandingkan dengan pembagian blok tetap yang dilakukan oleh paging.

4.5 Keuntungan dan Kekurangan Segmentasi

Segmentasi menawarkan serangkaian keuntungan dan kekurangan yang berbeda dibandingkan dengan paging:

Keuntungan Segmentasi:

- **Pandangan Programmer yang Alami:** Segmentasi sangat sesuai dengan cara programmer memandang program sebagai kumpulan unit logis (seperti kode, data, stack, atau prosedur). Ini membuat pengembangan program lebih intuitif

dan pengelolaan memori lebih mudah dari perspektif programmer.[8]

- **Proteksi dan Sharing yang Lebih Baik:** Karena memori dibagi berdasarkan unit logis, segmentasi memungkinkan proteksi dan *sharing* yang lebih granular antar segmen. Misalnya, segmen kode yang sering digunakan (seperti pustaka) dapat dengan mudah dibagi pakai (*shared code*) di antara beberapa proses, dan setiap segmen dapat memiliki izin akses yang berbeda (misalnya, hanya baca, baca-tulis, atau eksekusi).[8]
- **Tidak Ada Fragmentasi Internal (Idealnya):** Karena segmen dapat berukuran variabel dan dialokasikan sesuai dengan kebutuhan aktual program, tidak ada ruang yang terbuang di dalam segmen itu sendiri (kecuali jika sistem mengalokasikan sedikit lebih besar dari yang diminta). Ini merupakan keunggulan dibandingkan paging yang selalu memiliki potensi fragmentasi internal.[33]
- **Fleksibilitas:** Segmentasi dapat menampung struktur data berukuran variabel yang dinamis, seperti *stack* dan *heap*, yang ukurannya dapat berubah selama eksekusi program.[32]

Kekurangan Segmentasi:

- **Fragmentasi Eksternal:** Ini adalah kelemahan utama segmentasi. Karena segmen memiliki ukuran yang bervariasi, ketika segmen-segmen dimuat dan dihapus dari memori, ruang kosong yang tersisa dapat menjadi terfragmentasi menjadi potongan-potongan kecil yang tersebar. Meskipun total memori bebas mungkin cukup, tidak ada satu blok pun yang cukup besar untuk menampung segmen baru yang lebih besar, menyebabkan pemborosan memori.[9]
- **Kompleksitas Manajemen Memori:** Mengelola segmen dengan ukuran yang bervariasi lebih kompleks dibandingkan mengelola *page* dengan ukuran tetap. Sistem operasi harus melacak ukuran dan lokasi setiap segmen, serta menemukan blok memori bebas yang sesuai untuk alokasi.[8]
- **Overhead Tabel Segmen:** Meskipun tabel segmen mungkin lebih kecil dibandingkan tabel halaman dalam beberapa kasus, tetap ada *overhead* yang terkait dengan pemeliharaan tabel segmen untuk setiap proses dan akses ke tabel tersebut dapat menambah waktu yang dibutuhkan untuk operasi memori.[33]

Keuntungan dan kekurangan ini menunjukkan bahwa segmentasi adalah solusi yang kuat untuk masalah logis dalam manajemen memori dan proteksi, tetapi memperkenalkan kembali masalah fisik fragmentasi eksternal yang diatasi oleh paging. Ini adalah *trade-off* yang berbeda, yang menjelaskan mengapa sistem operasi modern seringkali menggabungkan kedua teknik ini.

BAB 5. Paging dan Segmentasi: Perbandingan dan Kombinasi

Paging dan segmentasi adalah dua teknik manajemen memori yang fundamental, masing-masing dengan filosofi dan karakteristiknya sendiri. Memahami perbedaan di antara keduanya sangat penting untuk mengapresiasi desain sistem operasi modern.

5.1 Perbedaan Kunci antara Paging dan Segmentasi

Berikut adalah tabel perbandingan yang merangkum perbedaan utama antara paging dan segmentasi:

Tabel 1. Perbedaan Kunci dan Paging dan Segmentasi

Kriteria	Paging	Segmentasi
Ukuran Blok	Blok berukuran tetap (*fixed-size*) (pages/frames) [8]	Blok berukuran variabel (*variable-size*) (segmen) [8]
Pandangan Memori	Fisik/Mekanis (tidak peduli struktur logis program) [8]	Logis (sesuai pandangan programmer terhadap program) [8]
Fragmentasi	Mengatasi fragmentasi eksternal, dapat menyebabkan fragmentasi internal [18]	Mengatasi fragmentasi internal, dapat menyebabkan fragmentasi eksternal [9]
Keterlibatan Programmer	Programmer tidak perlu tahu tentang detail paging [9]	Programmer terlibat (misalnya, mendefinisikan segmen logis) [9]
Proteksi	Proteksinya tidak terpisah (per *page*), tidak ada *shared code* secara langsung [9]	Proteksinya terpisah (per segmen), mendukung *shared code* [8]
Ruang Alamat Linier	Hanya terdapat satu ruang alamat linier [9]	Terdapat banyak ruang alamat linier (satu per segmen) [9]

Perbedaan-perbedaan mendasar ini menunjukkan bahwa paging dan segmentasi adalah pendekatan yang saling melengkapi untuk masalah manajemen memori. Paging dirancang untuk efisiensi penggunaan memori fisik, sementara segmentasi dirancang untuk organisasi logis dan proteksi program. Masing-masing mengatasi jenis masalah yang berbeda, sehingga tidak ada satu pun yang secara universal "lebih baik" dari yang lain.

5.2 Kapan Masing-masing Teknik Digunakan

Pemilihan teknik manajemen memori bergantung pada tujuan desain sistem operasi:

- **Paging:** Umumnya digunakan untuk mendapatkan ruang alamat linier yang besar tanpa perlu memori fisik lebih. Ini memungkinkan program yang ukurannya lebih besar dari RAM fisik untuk dieksekusi secara dinamis dengan memuat bagian-bagian yang diperlukan saja. Paging adalah dasar dari implementasi memori virtual modern yang memungkinkan efisiensi memori fisik yang tinggi dan mengatasi masalah fragmentasi eksternal.[13]
- **Segmentasi:** Digunakan untuk mengizinkan program dan data dipecah menjadi ruang alamat mandiri berdasarkan unit-unit logis. Ini sangat berguna untuk mendukung *sharing* dan proteksi yang lebih baik antar bagian-bagian program, karena setiap segmen dapat memiliki izin akses yang berbeda dan dapat dibagi pakai di antara proses-proses.[13]

Penggunaan yang berbeda ini memperjelas bahwa pilihan teknik manajemen memori adalah cerminan dari prioritas desain. Paging adalah pendekatan yang berpusat pada sistem, mengoptimalkan pemanfaatan sumber daya fisik. Segmentasi adalah pendekatan yang berpusat pada pengguna/programmer, mencerminkan bagaimana perangkat lunak secara alami terstruktur.

5.3 Sekilas tentang Kombinasi Paging dan Segmentasi

Mengingat keuntungan unik dan kekurangan masing-masing teknik, banyak sistem operasi modern mengimplementasikan memori virtual dengan mengombinasikan paging dan segmentasi. Pendekatan ini dikenal sebagai *paged segmentation* atau *segmented paging*.[15]

Tujuan dari kombinasi ini adalah untuk mendapatkan manfaat dari kedua teknik secara

bersamaan:

- **Fleksibilitas segmentasi** dalam mengorganisasi program secara logis, memberikan pandangan memori yang lebih alami bagi programmer, serta mendukung proteksi dan *sharing* yang granular.
- **Efisiensi paging** dalam mengelola memori fisik, mengatasi fragmentasi eksternal, dan memungkinkan program untuk dimuat secara non-kontigu di RAM.[15]

Kombinasi ini adalah evolusi alami dalam desain sistem operasi, yang menunjukkan bahwa solusi yang paling efektif seringkali adalah hibrida yang mengambil kekuatan dari beberapa pendekatan untuk mengatasi keterbatasan masing-masing. Dengan menggabungkan keduanya, sistem operasi dapat menawarkan manajemen memori yang kuat, fleksibel, dan efisien, yang merupakan fondasi bagi kinerja dan stabilitas komputasi modern.

6. Kesimpulan dan Poin-Poin Penting

Ringkasan Konsep Utama

Manajemen memori adalah salah satu fungsi inti dari sistem operasi, dan konsep memori virtual, paging, serta segmentasi adalah pilar utamanya.

- **Memori Virtual** adalah sebuah ilusi yang diciptakan oleh sistem operasi, memberikan kesan kepada program bahwa mereka memiliki akses ke ruang memori yang jauh lebih besar daripada RAM fisik yang sebenarnya. Tujuan utamanya adalah memungkinkan program yang lebih besar dari RAM untuk berjalan dan mendukung multitasking yang efisien.
- **Paging** adalah teknik manajemen memori yang membagi memori fisik menjadi *frame* berukuran tetap dan ruang alamat logika menjadi *page* berukuran sama. Ini secara efektif mengatasi masalah fragmentasi eksternal dan merupakan dasar bagi implementasi *demand paging* dan penanganan *page fault*.
- **Segmentasi** adalah teknik yang membagi memori berdasarkan unit-unit logis program (seperti kode, data, stack) yang memiliki ukuran bervariasi. Ini mendukung pandangan alami programmer terhadap memori, memungkinkan proteksi dan *sharing* yang lebih baik antar bagian-bagian program.
- **Memory Management Unit (MMU)** adalah perangkat keras krusial yang bertanggung jawab untuk menerjemahkan alamat logika (virtual) menjadi alamat fisik di RAM, baik dalam skema paging maupun segmentasi.

Pentingnya Manajemen Memori dalam Sistem Operasi Modern

Manajemen memori yang efisien bukan hanya fitur teknis, melainkan inti dari kinerja, stabilitas, dan keamanan sistem operasi modern.[5] Tanpa teknik canggih seperti memori virtual yang diimplementasikan melalui paging dan/atau segmentasi, kemampuan komputer untuk melakukan multitasking yang lancar dan menjalankan program yang kompleks akan sangat terbatas.[1]

Adanya memori virtual memungkinkan pengguna untuk menjalankan banyak aplikasi secara bersamaan tanpa khawatir kehabisan RAM fisik. Teknik-teknik ini juga memastikan bahwa setiap program beroperasi dalam ruang memorinya sendiri yang terisolasi, mencegah konflik antar aplikasi dan meningkatkan keamanan sistem secara keseluruhan. Dengan demikian, pemahaman tentang memori virtual, paging, dan segmentasi adalah fundamental untuk mengerti bagaimana sistem operasi modern

bekerja dan mengapa pengalaman komputasi kita saat ini begitu mulus dan stabil.

Works cited

1. Pengertian Virtual RAM: Cara Kerja, Fungsi, Kelebihan, dan ..., accessed June 24, 2025, https://indonews.id/artikel/334545/Pengertian-Virtual-RAM-Cara-Kerja-Fungsi-Kelebihan-dan-Kekurangan/
2. ELI5: What is virtual memory and how is it different than RAM? : r/explainlikeimfive - Reddit, accessed June 24, 2025, https://www.reddit.com/r/explainlikeimfive/comments/142dd8t/eli5_what_is_virtual_memory_and_how_is_it/
3. Buku Ajar SISTEM OPERASI - OSF, accessed June 24, 2025, https://osf.io/6yh4e/download
4. Apa perbedaan antara memori dan penyimpanan? - Kingston Technology, accessed June 24, 2025, https://www.kingston.com/id/blog/pc-performance/difference-between-memory-storage
5. Pengertian Manajemen Memori Sistem Operasi - DomaiNesia, accessed June 24, 2025, https://www.domainesia.com/berita/manajemen-memori-sistem-operasi/
6. Implementasi Virtual Memory | PPT - SlideShare, accessed June 24, 2025, https://www.slideshare.net/SalsabelaMaulina1/implementasi-virtual-memory-246153979
7. Daniel Muliawan - danielmuliawan, accessed June 24, 2025, https://danielmuliawan.blog.binusian.org/author/danielmuliawan/
8. Virtual memory: paging and segmentation | Intro to Computer ..., accessed June 24, 2025, https://library.fiveable.me/introduction-computer-architecture/unit-5/virtual-memory-paging-segmentation/study-guide/Wrwj8wS7SGJDi02g
9. Do modern OS's use paging and segmentation? - Stack Overflow, accessed June 24, 2025, https://stackoverflow.com/questions/24358105/do-modern-oss-use-paging-and-segmentation
10. Virtual Memory in Operating System - GeeksforGeeks, accessed June 24, 2025, https://www.geeksforgeeks.org/operating-systems/virtual-memory-in-operating-system/
11. The Ins and Outs of Virtual Memory Swap - Alibaba Cloud, accessed June 24, 2025, https://www.alibabacloud.com/tech-news/a/memory_virtualization/guvey4dak6-the-ins-and-outs-of-virtual-memory-swap
12. Chapter 10: Virtual Memory, accessed June 24, 2025, https://www.cs.hunter.cuny.edu/~sweiss/course_materials/csci340/slides/chapter10.pdf
13. Sistem Operasi Sistem Paging Dan Segmentasi | PDF - Scribd, accessed June 24, 2025, https://id.scribd.com/document/504890858/Sistem-Operasi-Sistem-Paging-dan-Segmentasi

14. Kelompok 1 Memori Virtual | PDF | Bisnis - Scribd, accessed June 24, 2025, https://id.scribd.com/document/648629149/kelompok-1-memori-virtual
15. Memory management unit - Wikipedia, accessed June 24, 2025, https://en.wikipedia.org/wiki/Memory_management_unit
16. Explain the concept of swapping in virtual memory. - TutorChase, accessed June 24, 2025, https://www.tutorchase.com/answers/a-level/computer-science/explain-the-concept-of-swapping-in-virtual-memory
17. Memori Virtual, accessed June 24, 2025, https://arna.lecturer.pens.ac.id/Diktat_SO/8.Virtual%20Memory.pdf
18. Manajemen Memori, accessed June 24, 2025, https://arna.lecturer.pens.ac.id/Diktat_SO/7.Manajemen%20Memory.pdf
19. Sistem Paging Dan Segmentasi CONTOH!!!!!!!! - Scribd, accessed June 24, 2025, https://id.scribd.com/document/415283304/Sistem-Paging-dan-Segmentasi-CONTOH-docx
20. MANAJEMEN MEMORI – Sistem Operasi - Blog UNNES, accessed June 24, 2025, https://blog.unnes.ac.id/bimaaaa/manajemen-memori/
21. Paging and Memory Management: Enhance Computing Speed ..., accessed June 24, 2025, https://www.lenovo.com/us/en/glossary/paging/
22. Paging In Operating System: Complete Explanation - PW Skills, accessed June 24, 2025, https://pwskills.com/blog/paging-in-operating-system-complete-explanation/
23. November 2010 - WordPress.com, accessed June 24, 2025, https://komputersederhana.files.wordpress.com/2018/06/pert-18-ch08-hardware-virtual-memory-20101109.pdf
24. Difference Between Paging And Segmentation - Shiksha Online, accessed June 24, 2025, https://www.shiksha.com/online-courses/articles/difference-between-paging-and-segmentation/
25. Paging vs Segmentation Explained with Clothes Analogy | OS Memory Management in 60 Seconds - YouTube, accessed June 24, 2025, https://www.youtube.com/watch?v=VOgfVA7drbQ
26. System Paging Dan Segmentasi | PDF | Metode & Bahan Ajar | Komputer - Scribd, accessed June 24, 2025, https://id.scribd.com/doc/149623397/System-Paging-dan-Segmentasi
27. Virtual Memory: Sistem Operasi | PDF - Scribd, accessed June 24, 2025, https://id.scribd.com/document/666524304/06
28. Understanding Memory in Computing: From Physical to Virtual - Jyotiprakash's Blog, accessed June 24, 2025, https://blog.jyotiprakash.org/understanding-memory-in-computing-from-physical-to-virtual
29. MODERN OPERATING SYSTEMS Third Edition ANDREW S ..., accessed June 24, 2025,

https://repository.unikom.ac.id/60425/1/MOS-3e-03%20%5BMemory%20Management%5D.pdf
30. www.shiksha.com, accessed June 24, 2025, https://www.shiksha.com/online-courses/articles/paging-in-operating-system-real-life-analogy/#:~:text=Paging%20is%20a%20memory%20management,memory%20from%20the%20secondary%20memory.
31. Virtual Memory – Bagas - Blog UNNES, accessed June 24, 2025, https://blog.unnes.ac.id/bagas/virtual-memory/
32. The Ultimate Guide to Segmentation in Operating Systems - Uncodemy, accessed June 24, 2025, https://uncodemy.com/blog/segmentation-in-operating-systems-guide/
33. Segmentation in Operating System - GeeksforGeeks, accessed June 24, 2025, https://www.geeksforgeeks.org/operating-systems/segmentation-in-operating-system/
34. OS42 - Memory Management Unit | MMU - YouTube, accessed June 24, 2025, https://www.youtube.com/watch?v=agU0UnLiqLo
35. Paged Segmentation and Segmented Paging - GeeksforGeeks, accessed June 24, 2025, https://www.geeksforgeeks.org/operating-systems/paged-segmentation-and-segmented-paging/
36. DESAIN MEMORI VIRTUAL PADA MIKROARSITEKTUR POWERPC, MIPS, DAN X86 - OoCities.org, accessed June 24, 2025, https://www.oocities.org/transmisi_eeundip/kuspriyanto.pdf
37. Tipe manajemen memori pada sistem operasi | PPT - SlideShare, accessed June 24, 2025, https://www.slideshare.net/slideshow/tipe-manajemen-memori-pada-sistem-operasi/138532008