JURNAL PINTER

Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475

DOI: https://doi.org/10.21009/pinter.10.1.1

# Analisis Perbandingan Kinerja Mekanisme Paging, Segmentation, dan Kombinasinya melalui Pendekatan Simulasi

**Lita Dwi Setianingsih*, Rifdah Mayhasna Nur Allayya, Djuniadi, Alfian Ardhiansyah**

Program Studi Teknik Komputer, Fakultas Teknik Elektro, Universitas Negeri Semarang, Jl. Raya Banaran, Kota Semarang, Jawa Tengah 50229, Indonesia

*Corresponding Author. Email: litaadwiis12@students.unnes.ac.id

Received : 10 Januari 2026
Revised : 17 Mei 2026
Online : 17 Juni 2026
Edisi : 30 Juni 2026

## Abstrak

*Manajemen memori merupakan komponen penting dalam sistem operasi yang berpengaruh langsung terhadap kinerja dan stabilitas sistem, khususnya pada lingkungan multitasking. Perbedaan mekanisme manajemen memori dapat menghasilkan karakteristik kinerja yang berbeda dalam hal alokasi memori, fragmentasi, dan overhead pengelolaan sistem. Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja mekanisme paging, segmentation, serta kombinasi paging–segmentation melalui pendekatan simulasi. Metode yang digunakan adalah simulasi berbasis event-driven pada sistem operasi multitasking dengan satu unit pemrosesan (single CPU). Evaluasi kinerja dilakukan berdasarkan tingkat keberhasilan alokasi proses, fragmentasi memori, serta frekuensi page fault sebagai indikator overhead manajemen memori. Hasil simulasi menunjukkan bahwa mekanisme paging mampu mempertahankan tingkat keberhasilan alokasi proses yang tinggi pada seluruh skenario beban sistem, namun disertai peningkatan frekuensi page fault. Mekanisme segmentation menunjukkan kinerja optimal pada beban rendah, tetapi mengalami penurunan keberhasilan alokasi pada beban tinggi akibat fragmentasi eksternal. Sementara itu, mekanisme kombinasi paging–segmentation memberikan kinerja paling stabil dengan tingkat keberhasilan alokasi yang konsisten serta fragmentasi memori yang lebih terkendali. Hasil penelitian ini menunjukkan bahwa mekanisme kombinasi merupakan pendekatan yang paling seimbang untuk sistem multitasking dengan variasi beban kerja yang dinamis.*

**Kata Kunci:** manajemen memori, paging, segmentation

## Abstract

Memory management is an important component in operating systems that directly affects system performance and stability, especially in multitasking environments. Differences in memory management mechanisms can result in varying performance characteristics in terms of memory allocation, fragmentation, and system management overhead. This study aims to analyze and compare the performance of paging, segmentation, and the combination of paging-segmentation mechanisms thru a simulation approach. The method used is

Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

event-driven simulation on a multitasking operating system with a single processing unit (single CPU). Performance evaluation is conducted based on the success rate of process allocation, memory fragmentation, and page fault frequency as indicators of memory management overhead. Simulation results show that the paging mechanism can maintain a high success rate of process allocation across all system load scenarios, but with an increase in page fault frequency. The segmentation mechanism shows optimal performance under low load, but experiences a decrease in allocation success under high load due to external fragmentation. Meanwhile, the paging-segmentation combination mechanism provides the most stable performance with a consistent allocation success rate and more controlled memory fragmentation. The results of this study indicate that the combination mechanism is the most balanced approach for multitasking systems with dynamic workload variations.

**Keywords:** memory management, paging, segmentation

## Pendahuluan

Manajemen memori merupakan komponen esensial dalam sistem operasi yang berfungsi mengatur proses alokasi dan pemanfaatan memori utama oleh berbagai proses yang berjalan secara simultan. Pada lingkungan multitasking, kualitas pengelolaan memori memiliki pengaruh signifikan terhadap kinerja sistem secara keseluruhan, kestabilan eksekusi proses, serta optimalisasi penggunaan sumber daya. Konsep memori virtual berperan sebagai fondasi utama dalam menyederhanakan proses pemrograman (Chou et al., 2026) dengan menyediakan ruang alamat yang luas dan konsisten, sehingga pandangan logis aplikasi dapat dipisahkan dari keterbatasan fisik perangkat keras (Zhu et al., 2026). Alokasi dan dealokasi memori yang berlangsung secara dinamis menimbulkan tantangan tersendiri dalam menjaga keseimbangan antara kecepatan akses dan kompleksitas pengelolaan sistem. Berbagai kajian menunjukkan bahwa perbedaan rancangan unit manajemen memori serta struktur tabel halaman memiliki peranan penting dalam menentukan tingkat efisiensi sistem (Siavashi et al., 2026; Gupta et al., 2026). Selain itu, kinerja proses translasi alamat sangat dipengaruhi oleh parameter yang digunakan pada mekanisme Translation lookaside buffer (TLB) (Khan et al., 2026), di mana interaksi sistem operasi yang kurang optimal berpotensi menimbulkan polusi pada struktur internal prosesor, seperti cache dan TLB (Xiong et al., 2026). Dalam penelitian ini, komponen perangkat keras seperti MMU, TLB, dan cache tidak dimodelkan secara fisik, melainkan dijadikan sebagai kerangka konseptual untuk memahami perilaku sistem serta dampak kebijakan manajemen memori pada tingkat sistem operasi.

Untuk menangani permasalahan alokasi memori, sistem operasi mengimplementasikan berbagai mekanisme, dengan paging dan segmentation sebagai pendekatan utama. Dalam cakupan yang lebih luas, pengembangan sistem operasi juga mencakup pengelolaan memori bersama dan mekanisme pengendalian kesalahan (fault containment) guna meningkatkan keandalan pada arsitektur yang kompleks (Ali et al., 2026), serta pengaturan koherensi pada sistem memori virtual terdistribusi (Liu et al., 2026). Namun, pada konteks sistem operasi berbasis CPU tunggal, mekanisme paging dan segmentation masih menjadi solusi utama dalam pengelolaan memori. Keduanya memiliki karakteristik yang berbeda, di mana paging efektif dalam mengurangi fragmentasi eksternal melalui alokasi non-kontigu, sedangkan segmentation memungkinkan pembagian ruang alamat berdasarkan struktur logis dari program. Evaluasi terhadap efektivitas mekanisme tersebut dapat dilakukan melalui pendekatan simulasi, yang memungkinkan pengujian berbagai kebijakan dalam kondisi yang terkontrol. Sejumlah penelitian terdahulu telah memanfaatkan simulasi untuk menganalisis pemanfaatan memori dan tingkat konkurensi pada lingkungan sistem operasi dengan keterbatasan sumber daya tertentu (Alaei & Yazdanpanah, 2024). Pendekatan simulasi juga terbukti efektif dalam mengamati fenomena beban

Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

kerja intensif data yang dapat memicu kondisi oversubscription pada memori (Maas & Lorenzon, 2025). Dalam penelitian ini, simulasi digunakan secara khusus untuk mengevaluasi kebijakan manajemen memori pada tingkat sistem operasi di bawah variasi beban kerja, tanpa melibatkan pemodelan detail interaksi perangkat keras yang kompleks maupun sistem terdistribusi.

Meskipun beberapa penelitian sebelumnya telah membahas mekanisme alokasi kontinu pada arsitektur hibrida (Huang et al., 2025) serta optimasi tata letak data untuk meningkatkan efisiensi energi pada sistem tertanam (Marinelli et al., 2023), masih terdapat kebutuhan akan analisis perbandingan langsung terhadap mekanisme konvensional menggunakan simulator CPU OS yang berfokus pada logika perangkat lunak. Oleh karena itu, penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja mekanisme paging, segmentation, serta kombinasi keduanya melalui pendekatan simulasi. Evaluasi dilakukan dengan mengamati tingkat keberhasilan alokasi proses, derajat fragmentasi memori, serta frekuensi terjadinya kesalahan halaman (page fault) sebagai indikator overhead pengelolaan memori pada berbagai skenario beban sistem.

Memori virtual merupakan mekanisme yang memungkinkan sistem operasi menyediakan ruang alamat logis yang luas dan konsisten bagi setiap proses tanpa bergantung langsung pada kapasitas memori fisik. Melalui konsep ini, setiap proses dapat berjalan seolah-olah memiliki ruang memori tersendiri yang terisolasi, sehingga keamanan dan keandalan sistem dalam lingkungan multitasking dapat ditingkatkan. Mekanisme ini bekerja melalui pemetaan alamat virtual ke alamat fisik, di mana hanya data dan instruksi yang dibutuhkan yang dimuat ke memori utama, sehingga eksekusi program berukuran besar dapat dilakukan secara lebih efisien serta mendukung fleksibilitas relokasi dan pengelolaan memori oleh sistem operasi (Thyagaturu et al., 2022; Boubakri & Zouari, 2025; Zhao et al., 2026). Dalam sistem operasi modern, memori virtual menjadi landasan bagi berbagai mekanisme manajemen memori, seperti paging dan segmentation. Sejumlah penelitian menunjukkan bahwa desain memori virtual, termasuk pengorganisasian tabel halaman dan mekanisme translasi alamat, berpengaruh signifikan terhadap kinerja sistem serta overhead pengelolaan memori. Oleh karena itu, pemahaman terhadap konsep memori virtual menjadi aspek penting dalam menganalisis kinerja mekanisme manajemen memori yang dievaluasi melalui simulasi pada penelitian ini.

### Alokasi Memori Dinamis

Alokasi memori dinamis memungkinkan sistem operasi memberikan serta menarik kembali ruang memori kepada proses selama program berjalan sesuai dengan kebutuhan aktual. Mekanisme ini sangat relevan dalam sistem multitasking, mengingat kebutuhan memori setiap proses dapat berubah seiring waktu dan jenis aktivitas yang dilakukan, sehingga sistem operasi dituntut untuk mengelola ruang memori secara fleksibel dan efisien. Namun, alokasi dan dealokasi memori yang berlangsung secara berulang dapat memicu terjadinya fragmentasi memori. Fragmentasi internal muncul ketika ruang memori yang dialokasikan tidak dimanfaatkan secara optimal, sedangkan fragmentasi eksternal terjadi akibat penyebaran ruang kosong yang tidak saling berdekatan. Kedua bentuk fragmentasi tersebut berpotensi menurunkan efisiensi pemanfaatan memori serta meningkatkan risiko kegagalan dalam mengalokasikan memori untuk proses baru, meskipun secara total kapasitas memori masih mencukupi. Oleh sebab itu, sistem operasi perlu menerapkan kebijakan alokasi memori yang tepat agar dampak fragmentasi dapat diminimalkan dan kinerja sistem tetap terjaga. Analisis terhadap pengaruh fragmentasi akibat alokasi memori dinamis menjadi salah satu fokus utama dalam simulasi manajemen memori pada penelitian ini, khususnya dalam mengevaluasi perbedaan karakteristik mekanisme paging, segmentation, dan kombinasi keduanya.

### Page fault dan Overhead Kernel

Page fault merupakan kondisi yang terjadi ketika suatu proses mengakses halaman memori yang belum tersedia di memori fisik, sehingga sistem operasi harus melakukan penanganan tambahan sebelum eksekusi dapat dilanjutkan. Fenomena ini umum dijumpai pada sistem yang menerapkan memori virtual dan mekanisme paging, di mana tidak seluruh halaman proses dimuat ke memori utama secara bersamaan. Penanganan page fault melibatkan serangkaian aktivitas kernel, seperti pemrosesan interupsi, pemuatan halaman dari media penyimpanan sekunder, serta pembaruan struktur data manajemen memori. Proses tersebut menimbulkan overhead kernel yang dapat berdampak pada penurunan kinerja sistem, terutama apabila page fault terjadi dalam frekuensi yang tinggi, karena setiap kejadian page fault memerlukan waktu pemprosesan tambahan dan intervensi kernel (Alam et al., 2026). Dalam lingkungan multitasking, peningkatan jumlah proses aktif dan pola

Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

akses memori yang tidak terprediksi dapat menyebabkan frekuensi page fault semakin meningkat. Oleh karena itu, page fault beserta overhead kernel yang menyertainya digunakan sebagai indikator penting dalam mengevaluasi kinerja mekanisme manajemen memori melalui pendekatan simulasi, khususnya dalam menganalisis trade-off antara fleksibilitas alokasi memori dan overhead pengolahannya.

### Memory management unit (MMU) dan Translasi Alamat

Memory management unit (MMU) merupakan komponen yang bertanggung jawab melakukan translasi alamat virtual menjadi alamat fisik sekaligus menerapkan mekanisme proteksi memori. Melalui proses translasi alamat ini, sistem operasi dapat mengisolasi ruang alamat antar proses dan mencegah terjadinya akses memori yang tidak sah. Untuk mempercepat translasi alamat, sistem operasi memanfaatkan Translation lookaside buffer (TLB) sebagai cache yang menyimpan hasil pemetaan alamat yang sering digunakan. Efisiensi proses translasi alamat memiliki pengaruh langsung terhadap kinerja sistem, karena setiap akses memori harus melalui mekanisme pemetaan tersebut dan keterlambatan pada proses ini dapat berdampak pada penurunan performa sistem secara keseluruhan. Dalam konteks penelitian ini, konsep MMU dan translasi alamat digunakan sebagai dasar pemahaman secara konseptual mengenai mekanisme memori virtual. Simulasi yang dilakukan tidak memodelkan perangkat keras MMU secara rinci, melainkan difokuskan pada perilaku manajemen memori sistem operasi pada tingkat konseptual, sesuai dengan tujuan analisis kinerja mekanisme manajemen memori yang dikaji.

### Simulasi Manajemen Memori

Simulasi merupakan pendekatan yang digunakan untuk memodelkan serta menganalisis perilaku manajemen memori sistem operasi dalam lingkungan yang terkontrol. Melalui simulasi, berbagai mekanisme manajemen memori dapat diuji tanpa ketergantungan pada perangkat keras fisik tertentu, sehingga hasil pengujian dapat dilakukan secara konsisten dan mudah direproduksi. Pendekatan ini memberikan fleksibilitas dalam pengaturan parameter sistem, seperti jumlah proses, pola permintaan memori, dan ukuran halaman. Dengan demikian, dampak kebijakan manajemen memori terhadap kinerja sistem dapat diamati secara sistematis melalui berbagai skenario pengujian tanpa memengaruhi kestabilan sistem nyata yang digunakan. Pada penelitian ini, simulasi dimanfaatkan untuk mengevaluasi karakteristik mekanisme paging, segmentation, serta kombinasi keduanya berdasarkan metrik kinerja yang relevan. Hasil simulasi selanjutnya digunakan sebagai dasar analisis guna memahami kelebihan dan keterbatasan masing-masing mekanisme manajemen memori, sebagaimana umum dilakukan dalam penelitian sistem operasi berbasis simulasi (Jiang et al., 2026; Li, 2026).

### Fragmentasi Memori dan Dampaknya terhadap Kinerja Sistem

Fragmentasi memori merupakan kondisi ketika ruang memori tidak dimanfaatkan secara optimal akibat pola alokasi dan dealokasi yang tidak teratur, yang secara umum diklasifikasikan menjadi fragmentasi internal dan eksternal. Fragmentasi internal umumnya muncul pada mekanisme paging karena penggunaan unit memori berukuran tetap, sedangkan fragmentasi eksternal lebih sering terjadi pada mekanisme segmentation yang memerlukan alokasi memori secara kontigu. Keberadaan fragmentasi ini dapat menurunkan efisiensi pemanfaatan memori dan meningkatkan risiko kegagalan alokasi proses meskipun kapasitas memori total masih mencukupi, sehingga berpotensi memengaruhi kinerja sistem operasi pada lingkungan multitasking dengan beban tinggi. Oleh karena itu, fragmentasi memori digunakan sebagai salah satu metrik utama dalam penelitian ini untuk mengevaluasi dampak kebijakan manajemen memori melalui pendekatan simulasi, khususnya dalam membandingkan karakteristik mekanisme paging, segmentation, dan kombinasi keduanya.

## Metode

### Pendekatan dan Lingkup Penelitian

Penelitian ini menerapkan pendekatan eksperimental berbasis simulasi untuk menganalisis kinerja mekanisme manajemen memori pada sistem operasi berbasis CPU. Pendekatan simulasi dipilih karena memungkinkan

Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

pengujian berbagai kebijakan manajemen memori dalam lingkungan yang terkontrol dan dapat direproduksi, sehingga perilaku sistem dapat diamati secara sistematis tanpa bergantung pada konfigurasi perangkat keras tertentu. Lingkup penelitian difokuskan pada kebijakan manajemen memori di tingkat sistem operasi, khususnya mekanisme paging, segmentation, serta kombinasi dari keduanya. Komponen perangkat keras seperti Memory management unit (MMU), Translation lookaside buffer (TLB), dan cache prosesor tidak dimodelkan secara fisik, melainkan digunakan sebagai acuan konseptual untuk memahami implikasi kebijakan manajemen memori terhadap kinerja sistem.

## Model Sistem dan Mekanisme Manajemen Memori

Model sistem yang disimulasikan merepresentasikan sistem operasi multitasking sederhana dengan satu unit pemrosesan (single CPU). Sistem terdiri atas kernel sistem operasi, subsistem manajemen memori, serta sejumlah proses yang berjalan secara bersamaan. Setiap proses memiliki kebutuhan memori yang bersifat dinamis dan menghasilkan permintaan alokasi maupun dealokasi memori selama simulasi berlangsung. Penelitian ini mengevaluasi tiga mekanisme manajemen memori, yaitu sebagai berikut.

### Paging

Pada mekanisme paging, ruang alamat logis dan memori utama dibagi ke dalam unit berukuran tetap. Proses alokasi memori dilakukan secara non-kontigu sehingga fragmentasi eksternal dapat dihindari. Namun, penggunaan unit berukuran tetap berpotensi menimbulkan fragmentasi internal serta meningkatkan jumlah page fault, khususnya pada kondisi beban sistem yang tinggi.

### Segmentation

Mekanisme segmentation membagi ruang alamat berdasarkan struktur logis program, seperti kode, data, dan stack. Setiap segmen dialokasikan secara kontigu di memori utama. Pendekatan ini mendukung pengorganisasian memori yang lebih selaras dengan struktur program, tetapi rentan terhadap fragmentasi eksternal ketika alokasi memori dilakukan secara dinamis.

### Kombinasi Paging dan Segmentation

Mekanisme kombinasi mengintegrasikan konsep segmentation dan paging dengan memetakan setiap segmen ke dalam halaman berukuran tetap. Pendekatan ini dirancang untuk mengurangi fragmentasi eksternal sekaligus mempertahankan struktur logis ruang alamat program. Simulasi dilakukan menggunakan pendekatan event-driven, di mana setiap kejadian yang berkaitan dengan manajemen memori, seperti permintaan alokasi memori, terjadinya page fault, dan pelepasan memori, dicatat untuk keperluan analisis. Pendekatan ini memungkinkan evaluasi perilaku manajemen memori serta overhead pengelolaannya secara terukur dalam lingkungan simulasi.

## Skenario Simulasi

Pengujian dilakukan melalui beberapa skenario simulasi dengan memvariasikan beban sistem, yang direpresentasikan oleh jumlah proses aktif serta kebutuhan memori masing-masing proses. Variasi beban ini digunakan untuk mengamati perubahan perilaku mekanisme manajemen memori pada kondisi beban ringan hingga berat. Setiap skenario dijalankan dengan konfigurasi sistem yang sama untuk ketiga mekanisme manajemen memori, sehingga hasil yang diperoleh dapat dibandingkan secara adil dan konsisten. Simulasi berlangsung hingga seluruh proses selesai dieksekusi atau mengalami kegagalan alokasi memori akibat keterbatasan ruang memori yang tersedia.

## Metrik Evaluasi dan Penyajian Data

Evaluasi kinerja mekanisme manajemen memori dilakukan menggunakan beberapa metrik utama, yaitu tingkat keberhasilan alokasi proses, tingkat fragmentasi memori, serta frekuensi kesalahan halaman (page fault).

Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

Keberhasilan alokasi proses mencerminkan kemampuan sistem dalam memenuhi permintaan memori dari proses yang berjalan, sementara fragmentasi memori menunjukkan tingkat pemborosan ruang memori akibat mekanisme alokasi yang diterapkan. Frekuensi page fault digunakan sebagai indikator overhead pengelolaan memori pada sistem operasi. Data hasil simulasi diperoleh melalui pencatatan kejadian (event) selama simulasi berlangsung dan selanjutnya disajikan dalam bentuk tabel dan grafik. Tabel digunakan untuk menampilkan nilai kuantitatif dari masing-masing metrik pada setiap skenario pengujian, sedangkan grafik dimanfaatkan untuk menggambarkan tren perubahan kinerja mekanisme manajemen memori seiring dengan peningkatan beban sistem. Analisis dilakukan dengan membandingkan hasil yang diperoleh dari setiap mekanisme serta mengaitkannya dengan karakteristik teoretis yang telah dibahas pada bagian landasan teori.

## Hasil dan Analisis

### Hasil Simulasi Mekanisme *Paging*

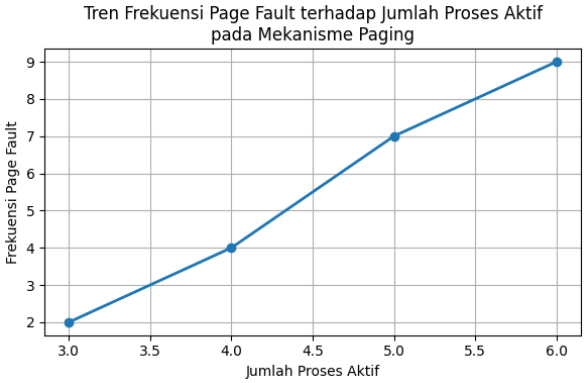
Pengujian terhadap mekanisme *paging* dilakukan melalui simulasi dengan variasi beban sistem yang direpresentasikan oleh jumlah proses aktif. Evaluasi difokuskan pada tingkat keberhasilan alokasi proses, besarnya fragmentasi internal, serta frekuensi terjadinya kesalahan halaman (*page fault*) selama simulasi berlangsung. Seluruh skenario pengujian disusun sesuai dengan metodologi.

**Tabel 1.** Hasil simulasi Mekanisme *Paging*

| Skenario | Jumlah Proses Aktif | Total Memori (KB) | Proses Berhasil Dialokasikan | Proses Gagal Dialokasikan | Kondisi memori |
|---|---|---|---|---|---|
| 1 | 3 | 32 | 3 | 2 | Stabil |
| 2 | 4 | 32 | 4 | 4 | Stabil |
| 3 | 5 | 32 | 5 | 7 | Fragmentasi internal |
| 4 | 6 | 32 | 6 | 9 | Utilisasi tinggi |

Hasil simulasi mekanisme *paging* pada berbagai tingkat beban sistem terlihat pada Tabel 1. Dalam skenario 3 hingga 6 proses aktif, mekanisme *paging* menunjukkan tingkat keberhasilan alokasi proses sebesar 100% pada seluruh skenario. Dalam skenario pertama hingga keempat, seluruh proses yang dijalankan dialokasikan berhasil ke memori, meskipun jumlah proses gagal meningkat dari 2 proses dalam skenariot 3 proses aktif menjadi 9 proses dalam skenario dengan 6 proses aktif. Peningkatan jumlah proses gagal ini menunjukkan bahwa, meskipun proses utama masih dapat dialokasikan, beban sistem memori semakin meningkat. Dengan membagi memori ke dalam unit halaman berukuran tetap, *paging* mampu memanfaatkan ruang memori yang tersebar dan menghindari fragmentasi eksternal, sehingga proses tetap dapat dialokasikan meskipun kondisi memori semakin padat. Ini menunjukkan bahwa mekanisme *paging* memiliki fleksibilitas alokasi memori yang tinggi dengan menggunakan alokasi non-kontigu. Namun, peningkatan beban sistem berdampak langsung pada jumlah kesalahan halaman yang terjadi. Dalam skenario 3 proses aktif, ada 2 kejadian kesalahan papan. Dengan beban sistem yang lebih besar, angka ini meningkat hingga mencapai 9 kesalahan papan pada skenario dengan 6 proses aktif. Peningkatan ini menunjukkan bahwa semakin banyak proses dan halaman yang dikelola, semakin banyak aktivitas sistem operasi yang diperlukan untuk translasi alamat dan pemuatan halaman ke memori utama.

JURNAL PINTER
Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1




**Gambar 1.** Tren frekuensi *page fault* terhadap peningkatan beban sistem pada mekanisme *paging*

Dari Grafik 1 diatas terlihat bahwa, dengan bertambahnya jumlah proses aktif pada mekanisme *paging*, frekuensi kesalahan halaman meningkat secara konsisten, dari 2 kejadian pada 3 proses aktif menjadi 9 kejadian pada 6 proses aktif. Pola ini menunjukkan bahwa ada korelasi positif antara beban sistem dan intensitas penanganan halaman yang dilakukan sistem operasi. Semakin banyak proses yang berjalan secara bersamaan berarti lebih banyak halaman memori yang harus dikelola dan ditranslasikan oleh sistem operasi. Akibatnya, tugas pengelolaan memori, terutama pemetaan alamat dan penanganan *page fault*, menjadi lebih sulit. Mekanisme *paging* memiliki kemampuan untuk mempertahankan keberhasilan alokasi proses pada seluruh tingkat beban sistem, tetapi tren yang ditunjukkan pada Grafik 1 menunjukkan bahwa peningkatan beban sistem tetap berdampak pada *overhead* pengelolaan memori. Dengan demikian, tren ini menunjukkan bahwa meskipun mekanisme *paging* memiliki fleksibilitas alokasi yang tinggi, konsekuensi dari peningkatan jumlah kesalahan halaman pada kondisi beban sistem.

## Hasil Simulasi Mekanisme *Segmentation*

Mekanisme *segmentation* diuji menggunakan skenario beban sistem yang sama dengan mekanisme *paging* guna memastikan validitas perbandingan hasil. Evaluasi difokuskan pada tingkat keberhasilan alokasi proses serta jenis fragmentasi yang muncul selama simulasi, khususnya fragmentasi eksternal yang menjadi karakteristik utama dari mekanisme *segmentation*.

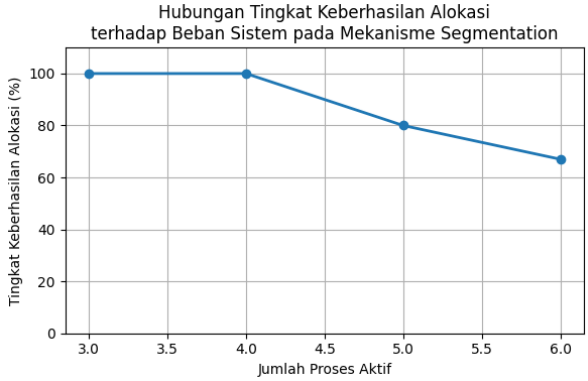
**Tabel 2.** Hasil simulasi Mekanisme *Segmentation*

| Skenario | Jumlah Proses Aktif | Total Memori (KB) | Proses Berhasil Dialokasikan | Proses Gagal Dialokasikan | Kondisi memori |
|---|---|---|---|---|---|
| 1 | 3 | 32 | 3 | 0 | Maemori masih longgar |
| 2 | 4 | 32 | 4 | 0 | Mulai terbentuk ruang kosong |
| 3 | 5 | 32 | 4 | 1 | Fragmentasi eksternal |
| 4 | 6 | 32 | 4 | 2 | Tidak ada blok kontigu |

Berdasarkan data dari Tabel 2, mekanisme *seg mentation* berhasil mencapai tingkat keberhasilan alokasi proses hingga 100% pada saat beban sistem rendah, yaitu pada skenario dengan 3 hingga 4 proses aktif. Seluruh proses dapat dialokasikan tanpa adanya kegagalan. Kondisi ini dimungkinkan karena ketersediaan ruang memori kontigu masih cukup besar, sehingga proses alokasi memori dapat dilakukan secara optimal.Namun, ketika jumlah proses aktif meningkat menjadi lima, tingkat keberhasilan alokasi menurun menjadi 80%, ditandai dengan kegagalan alokasi satu proses karena fragmentasi eksternal. Ini terjadi pada skenario dengan enam proses aktif, di mana hanya empat dari enam proses (67%) yang berhasil dialokasikan, meskipun kapasitas

JURNAL PINTER
Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

memori total belum digunakan sepenuhnya. Kondisi ini menunjukkan bahwa jumlah ruang memori kontigu yang terbatas merupakan faktor utama yang menghambat keberhasilan alokasi proses.




**Gambar 2.** Hubungan tingkat keberhasilan alokasi terhadap beban sistem pada mekanisme *segmentation*.

Berdasarkan data dari Tabel 2, Gambar 2 menunjukkan hubungan antara tingkat keberhasilan alokasi proses dan peningkatan beban sistem pada mekanisme *segmentation*. Dalam kondisi proses aktif 3 dan 4, tingkat keberhasilan alokasi 100%. Ini menunjukkan bahwa seluruh proses dapat dialokasikan dengan baik ketika ruang memori kontigu tersedia secara mencukupi. Gambar 2 menunjukkan penurunan tingkat keberhasilan alokasi hingga 80% seiring dengan meningkatnya jumlah proses aktif menjadi 5 proses. Hal ini selaras dengan hasil yang ditunjukkan pada Tabel 2, di mana satu proses gagal dialokasikan karena terbentuknya fragmentasi eksternal. Pada kondisi 6 proses aktif, penurunan ini semakin terlihat, dengan tingkat keberhasilan alokasi menurun hingga sekitar 67%, meskipun kapasitas memori total belum digunakan sepenuhnya.

Mekanisme *segmentation* sangat bergantung pada ruang memori kontigu, seperti yang ditunjukkan oleh tren penurunan pada Gambar 2. Fragmentasi eksternal yang muncul selama proses dealokasi dan alokasi memori menyebabkan ruang memori yang tersisa tidak dapat digunakan sepenuhnya untuk memenuhi kebutuhan proses baru. Hal ini diperkuat oleh grafik ini, yang menunjukkan bahwa keterbatasan ruang memori kontigu adalah faktor utama yang menghambat keberhasilan alokasi proses pada mekanisme *segmentation* ketika beban sistem meningkat.

## Hasil Simulasi Mekanism Kombinasi *Paging* dan *Segmentation*

Mekanisme kombinasi *paging* dan *segmentation* diuji untuk mengevaluasi efektivitas pendekatan hibrida dalam mengatasi keterbatasan mekanisme tunggal. Pada mekanisme ini, ruang alamat logis dibagi ke dalam segmen, yang selanjutnya dialokasikan dalam bentuk halaman pada memori fisik. Pendekatan tersebut diharapkan mampu menggabungkan fleksibilitas alokasi dari *paging* dengan keteraturan struktur logis yang dimiliki *segmentation*.

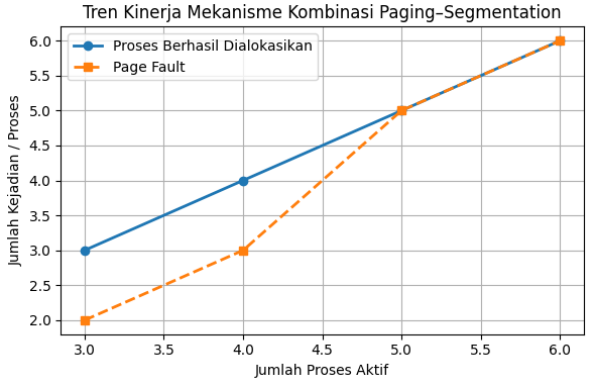
**Tabel 3.** Hasil simulasi Mekanisme *Paging* dan *Segmentation*

| Skenario | Jumlah Proses Aktif | Total Memori (KB) | Proses Berhasil Dialokasikan | *Page fault* | Kondisi memori |
|---|---|---|---|---|---|
| 1 | 3 | 32 | 3 | 2 | Stabil |
| 2 | 4 | 32 | 4 | 3 | Stabil |
| 3 | 5 | 32 | 5 | 5 | Efisien |
| 4 | 6 | 32 | 6 | 6 | Utilisasi optimal |

Berdasarkan hasil simulasi yang ditunjukkan pada Tabel 3, mekanisme kombinasi memperlihatkan tingkat keberhasilan alokasi proses yang sangat tinggi pada seluruh skenario beban sistem. Fragmentasi eksternal dapat ditekan secara signifikan melalui penerapan *paging*, sementara fragmentasi internal masih muncul tetapi dalam

JURNAL PINTER
Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

tingkat yang lebih terkendali dibandingkan dengan mekanisme *paging* murni.




**Gambar 3**. Tren kinerja mekanisme kombinasi terhadap peningkatan beban sistem.

Gambar 3 menunjukkan tren kinerja mekanisme kombinasi *paging* dan *segmentation* terhadap peningkatan beban sistem yang diwakili oleh jumlah proses aktif. Dari grafik tersebut, jumlah proses yang berhasil dialokasikan meningkat seiring dengan bertambahnya jumlah proses aktif, mulai dari 3 proses pada kondisi 3 proses aktif hingga 6 proses pada kondisi 6 proses aktif. Hal ini menunjukkan bahwa mekanisme kombinasi mampu menjaga tingkat keberhasilan alokasi proses tetap 100% pada semua skenario beban sistem, seperti yang terlihat pada Tabel 3. Sebaliknya, seiring dengan peningkatan beban sistem, frekuensi kesalahan halaman pada mekanisme kombinasi juga meningkat: dari 2 kesalahan halaman pada 3 proses aktif menjadi 6 kesalahan halaman pada 6 proses aktif. Peningkatan ini terjadi secara bertahap dan relatif stabil, tanpa lonjakan yang signifikan. Kondisi ini menunjukkan bahwa mekanisme kombinasi masih dapat mempertahankan kinerja sistem yang stabil meskipun beban pengelolaan memori meningkat.

Secara keseluruhan, tren yang ditunjukkan pada Grafik 3 menunjukkan bahwa mekanisme kombinasi *paging-segmentation* mampu menyeimbangkan *overhead* pengelolaan yang dihasilkan dan fleksibilitas alokasi memori. Penerapan *segmentation* membantu menekan dampak fragmentasi eksternal, sementara mekanisme *paging* memastikan alokasi memori tetap fleksibel. Oleh karena itu, mekanisme kombinasi dinilai efektif dalam lingkungan *multitasking* dengan variasi beban kerja yang berubah-ubah.

## Perbandingan Kinerja Antar Mekanisme

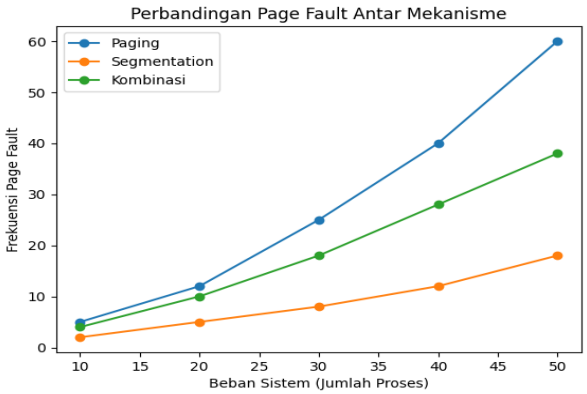
Perbandingan kinerja antar mekanisme manajemen memori dilakukan untuk mengevaluasi efektivitas relatif mekanisme *paging*, *segmentation*, dan kombinasi *paging–segmentation* berdasarkan metrik yang sama. Evaluasi ini bertujuan untuk menjawab tujuan penelitian, yaitu mengidentifikasi karakteristik kinerja masing-masing mekanisme dalam menghadapi variasi beban sistem.

**Tabel 4.** Perbandingan Kinerja Mekanisme Manajemen Memori

| Mekanisme | Keberhasilan Alokasi | Fragmentasi | *Overhead* | Mekanisme |
|---|---|---|---|---|
| *Segmentation* | Menurun pada beban tinggi | Eksternal | Rendah | *Segmentation* |
| *Paging* | Tinggi | Internal | Sedang | *Paging* |
| *Segmentation + Paging* | Sangat tinggi | Minimal | Sedang | *Segmentation + Paging* |

Berdasarkan Tabel 4, setiap mekanisme manajemen memori menunjukkan karakteristik kinerja yang berbeda dalam hal keberhasilan alokasi, jenis fragmentasi, serta *overhead* pengelolaan memori. Temuan ini mengindikasikan bahwa tidak terdapat satu mekanisme yang unggul secara mutlak pada seluruh kondisi beban sistem, melainkan terdapat *trade-off* yang dipengaruhi oleh prinsip kerja alokasi memori dan karakteristik beban kerja yang dihadapi.

JURNAL PINTER
Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1




**Gambar 4.** Perbandingan Kinerja Antar Mekanisme Manajemen Memori

Gambar 4 memperlihatkan bahwa mekanisme kombinasi *paging* dan *segmentation* menunjukkan kinerja yang paling stabil dibandingkan dua mekanisme lainnya pada berbagai tingkat beban sistem. Mekanisme *paging* unggul dalam mempertahankan keberhasilan alokasi pada beban tinggi, namun disertai dengan *overhead* pengelolaan memori yang lebih besar. Sebaliknya, mekanisme *segmentation* mengalami penurunan kinerja yang cukup signifikan pada beban tinggi akibat dominasi fragmentasi eksternal. Secara keseluruhan, hasil perbandingan menunjukkan bahwa pemilihan mekanisme manajemen memori sangat bergantung pada karakteristik beban kerja dan kebutuhan sistem. Mekanisme *segmentation* lebih sesuai untuk beban rendah dengan struktur proses yang sederhana, *paging* cocok untuk sistem *multitasking* yang memerlukan fleksibilitas alokasi memori, sedangkan kombinasi *paging* dan *segmentation* menawarkan solusi yang paling seimbang untuk berbagai kondisi beban sistem.

## Kesimpulan

Berdasarkan hasil simulasi pada berbagai skenario beban sistem, dapat disimpulkan bahwa mekanisme manajemen memori berpengaruh signifikan terhadap kinerja sistem operasi dalam lingkungan multitasking. Peningkatan jumlah proses aktif digunakan untuk merepresentasikan kenaikan beban sistem dan mengamati perbedaan karakteristik tiap mekanisme secara komparatif. Mekanisme paging mampu mempertahankan tingkat keberhasilan alokasi proses sebesar 100% pada seluruh skenario, namun disertai peningkatan frekuensi page fault dari 2 menjadi 9 kejadian akibat fragmentasi internal dan intensitas translasi alamat yang lebih tinggi. Sebaliknya, mekanisme segmentation menunjukkan keberhasilan alokasi 100% pada beban rendah, tetapi menurun menjadi 80% pada lima proses aktif dan 67% pada enam proses aktif akibat fragmentasi eksternal meskipun kapasitas memori total belum sepenuhnya terpakai. Sementara itu, mekanisme kombinasi paging dan segmentation mampu mempertahankan tingkat keberhasilan alokasi sebesar 100% dengan frekuensi page fault yang lebih terkendali pada kisaran 2–6 kejadian, sehingga dinilai paling sesuai untuk sistem multitasking dengan beban kerja dinamis karena mampu menyeimbangkan fleksibilitas alokasi memori dan overhead pengelolaan secara lebih optimal.

## Statements dan Declaration

**Konflik Kepentingan (Competing Interests).** Para penulis menyatakan bahwa tidak terdapat konflik kepentingan dalam penelitian dan publikasi artikel ini.

**Kontribusi Penulis (Author Contributions).** LDS berkontribusi dalam konseptualisasi penelitian, kajian literatur, perancangan metodologi, pengembangan model simulasi, analisis data, dan penyusunan manuskrip awal. RMNA berkontribusi dalam pengembangan dan implementasi simulasi mekanisme *paging*, *segmentation*, dan kombinasi keduanya, serta pengumpulan dan pengolahan data hasil simulasi. D berkontribusi dalam pengembangan metodologi, validasi model dan hasil simulasi, interpretasi hasil, serta penelaahan manuskrip. AA berkontribusi dalam analisis dan interpretasi data, evaluasi hasil simulasi, serta penelaahan dan penyempurnaan manuskrip. Seluruh penulis telah membaca dan menyetujui versi akhir manuskrip.

**Persetujuan Etik (Ethics Approval).** Penelitian ini tidak melibatkan partisipan manusia, hewan, maupun

Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

pengumpulan data pribadi. Penelitian dilakukan melalui pendekatan simulasi berbasis komputer untuk menganalisis dan membandingkan kinerja mekanisme *paging*, *segmentation*, dan kombinasi keduanya. Oleh karena itu, persetujuan etik tidak diperlukan.

**Persetujuan untuk Berpartisipasi (Informed Consent to Participate).** Tidak berlaku. Penelitian ini tidak melibatkan partisipan manusia.

**Persetujuan untuk Publikasi (Consent for Publication).** Tidak berlaku. Penelitian ini tidak memuat informasi pribadi atau data yang dapat mengidentifikasi individu.

**Ketersediaan Data (Data Availability).** Data dan hasil simulasi yang mendukung temuan penelitian ini tersedia dari penulis korespondensi berdasarkan permintaan yang wajar.

**Penggunaan Alat Kecerdasan Buatan (Use of Artificial Intelligence Tools).** Tidak ada alat OpenAI atau alat kecerdasan buatan generatif lainnya yang digunakan dalam pelaksanaan penelitian, pengembangan atau pelaksanaan simulasi, maupun analisis data. Jika alat pemeriksaan bahasa digunakan selama penyusunan manuskrip, penggunaannya terbatas pada pemeriksaan tata bahasa dan bahasa. Seluruh saran perbaikan bahasa ditinjau oleh para penulis, yang bertanggung jawab penuh atas isi akhir manuskrip.

## Daftar Pustaka

Alaei, M., & Yazdanpanah, F. (2024). A survey on heterogeneous CPU–GPU architectures and simulators. *Concurrency and Computation: Practice and Experience*, *37*(1). https://doi.org/10.1002/cpe.8318

Alam, F., Mifthak, M. M., Purohit, S. B., Shadab, M., Byrd, G. T., & Harfoush, K. (2026). HyperShield: An automated evaluation platform for security and performance trade-offs in virtual systems. *Journal of Cybersecurity and Privacy*, *6*(2), 56. https://doi.org/10.3390/jcp6020056

Ali, Z., Aslam, N., Marotta, A., Tiberti, W., & Cassioli, D. (2026). A systematic review of kernel-level security mechanisms, vulnerability detection and mitigation in modern operating systems. *Sensors*, *26*(8), 2452. https://doi.org/10.3390/s26082452

Boubakri, M., & Zouari, B. (2025). A survey of RISC-V secure enclaves and trusted execution environments. *Electronics*, *14*(21), 4171. https://doi.org/10.3390/electronics14214171

Chou, H., Hsieh, C.-H., & Wang, Y.-C. (2026). Towards enabling a containerized virtual desktop system. *IEEE Access*, 1–1. https://doi.org/10.1109/access.2026.3721962

Gupta, D., Sharda, D., Kaur, N., & Bansal, R. (2026). Evaluating the effectiveness of the medium of instruction on learning outcomes in technical subjects among Indian undergraduates: A quasi-experimental cross-over study. *International Journal of Educational Development*, *123*, 103589. https://doi.org/10.1016/j.ijedudev.2026.103589

Huang, B., Yu, W., Ma, M., Wei, X., & Wang, G. (2025). Artificial-intelligence-based energy management strategies for Hybrid Electric Vehicles: A comprehensive review. *Energies*, *18*(14), 3600. https://doi.org/10.3390/en18143600

Jiang, Z., Song, R., O'Neill, Z., & Dong, B. (2026). BESTOpt: A modular, physics-informed runtime environment for building energy modeling, simulation, and control optimization. *Building Simulation*. https://doi.org/10.1007/s12273-026-1472-6

Khan, M., Mushtaq, M., Pacalet, R., & Apvrille, L. (2026). Toward secure RISC-V microarchitecture: Vulnerability classes and defenses. *IEEE Access*, *14*, 51504–51519. https://doi.org/10.1109/access.2026.3679984

Li, M. (2026). Bridging theory and practice: A packet tracer-based simulation approach to computer network education. *Journal of Education and Educational Research*, *18*(2), 122–125. https://doi.org/10.54097/mhaz9p29

Liu, X., Yang, Y., Zhu, C., Hu, Y., & Zhao, W. (2026). Le os: A lightweight edge operating system for industrial internet of things under resource constraints. *Future Generation Computer Systems*, *179*, 108360. https://doi.org/10.1016/j.future.2025.108360

Maas, W., & Lorenzon, A. (2025). Exploring cloud instance options for optimal performance-cost efficiency. *Cluster Computing*, *28*(15). https://doi.org/10.1007/s10586-025-05702-5

Marinelli, T., Gómez Pérez, J. I., Tenllado, C., & Catthoor, F. (2023). COMPAD: A heterogeneous cache-

Volume 10 Nomor 1, Juni 2026
e-ISSN: 2597 - 4475
DOI: https://doi.org/10.21009/pinter.10.1.1

scratchpad CPU architecture with data layout compaction for embedded loop-dominated applications. *Journal of Systems Architecture*, *145*, 103022. https://doi.org/10.1016/j.sysarc.2023.103022

Siavashi, M., Sanaee, A., Sharifi, M., & Antichi, G. (2026). Phoenix - A novel technique for performance-aware orchestration of thread and page table placement in NUMA systems. *SN Computer Science*, *7*(5). https://doi.org/10.1007/s42979-026-05157-4

Thyagaturu, A. S., Shantharama, P., Nasrallah, A., & Reisslein, M. (2022). Operating systems and hypervisors for network functions: A survey of enabling technologies and research studies. *IEEE Access*, *10*, 79825–79873. https://doi.org/10.1109/access.2022.3194913

Xiong, C., Lin, W., Huang, H., Lin, J., & Li, K. (2026). Interference modeling and scheduling for compute-intensive batch applications. *Future Generation Computer Systems*, *179*, 108355. https://doi.org/10.1016/j.future.2025.108355

Zhao, Y., Liu, F., Hu, Y., Wang, Z., Gao, M., Mutlu, O., Xian, H., Dong, H., Jing, N., Liang, X., Guan, H., Xu, Q., Chen, C., Quan, S., Yang, T., & Jiang, L. (2026). GUMPIM: Unitary and malleable memory for processing-in-memory with guaranteed PIM Pages. *ACM Transactions on Architecture and Code Optimization*. https://doi.org/10.1145/3833426

Zhu, Y., Hu, Y., Wan, C., & Wan, Q. (2026). Neuromorphic hardware materials for intelligent artificial perception and multimodal fusion: From sensing to cognition. *Interdisciplinary Materials*. https://doi.org/10.1002/idm2.70070