# BAB I

## PENDAHULUAN

### 1.1. Latar belakang

Arsitektur x86 telah banyak dipakai pada berbagai perangkat komputer hingga saat ini misalkan pada arsitektur prosessor *lake alder* [1]. Saat ini, penggunaan *cache memory* umum untuk memudahkan akses data. *Cache* ini memiliki ukuran kecil tetapi berfungsi sebagai penyimpanan sementara untuk data akses instan yang sering digunakan oleh prosesor [2]. *Cache* menjadi penting karena tujuan utama *cache* adalah menyimpan data yang sering digunakan oleh sistem daripada mengambil data dan instruksi dari memori utama untuk menjalankan program berikutnya. Setelah data masuk ke dalam *cache*, data tersebut diambil dari *cache* dan mempercepat proses komputasi yang pada gilirannya meningkatkan efisiensi sistem komputer.

Ada tiga jenis *cache* dalam prosesor. *Cache* pertama adalah L1 *Cache* (Level 1 *Cache*), yang merupakan *cache* tercepat dan terdekat dengan inti prosesor. L1 *Cache* menyimpan data dan instruksi yang sering digunakan. Selanjutnya, terdapat L2 *Cache* (Level 2 *Cache*), yang memiliki ukuran yang lebih besar dan biasanya digunakan bersama oleh beberapa inti prosesor. L2 *Cache* menyimpan data dan instruksi yang sering digunakan oleh inti CPU. Setelah itu, terdapat L3 *Cache* (Level 3 *Cache*), yang memiliki ukuran yang lebih besar lagi dan digunakan bersama oleh semua inti CPU dalam satu chip prosesor. L3 *Cache* membantu mengurangi beban pada memori utama (RAM) dengan menyimpan data yang sering digunakan. Dengan adanya *cache-cache* tersebut, waktu akses ke memori utama dapat dikurangi, kecepatan pemrosesan data dapat ditingkatkan, dan kinerja sistem komputer dapat dioptimalkan [3].

*Least Recently Used (LRU)* adalah sebuah kebijakan pergantian *(Replacement policy*) blok yang paling jarang diakses atau yang paling lama tidak diakses ketika *cache* penuh dan blok baru perlu dimasukkan. Ini berarti bahwa blok yang paling lama tidak diakses akan dikeluarkan terlebih dahulu saat ada blok baru yang perlu dimasukkan ke dalam *cache.* LRU masih sering digunakan dalam berbagai jenis komputer, mulai dari prosesor desktop dan server, sistem operasi dan jaringan [4]. Kebijakan LRU kesulitan dalam situasi dimana terdapat pembaruan konten yang sering, karena kebijakan ini tidak efektif beradaptasi dengan pola akses yang berubah. Dalam karya ilmiah tersebut, kebijakan LRU tidak efisien dalam menangani *cache* yang berada dalam keadaan saturasi dan tidak

saturasi yang kritis. Ketika *cache* berada dalam keadaan saturasi, artinya *cache* sudah penuh dan tidak dapat menampung konten baru.

Di sisi lain, keadaan tidak saturasi yang kritis mengacu pada situasi dimana *cache* memiliki sedikit konten dan masih memiliki kapasitas kosong yang signifikan. Kebijakan LRU mungkin tidak optimal dalam mempertahankan probabilitas hit yang tinggi karena kebijakan LRU cenderung menggantikan konten yang jarang diakses, bahkan jika *cache* sedang dalam keadaan tidak saturasi yang kritis. Hal ini dapat mengakibatkan penggantian konten yang sebenarnya masih dibutuhkan oleh pengguna, sehingga mempengaruhi probabilitas *cache hit*. Oleh karena itu, kebijakan LRU mungkin tidak efisien dalam mengelola *cache* dalam situasi dimana terdapat fluktuasi antara keadaan saturasi dan tidak saturasi yang kritis [5].

*Tree Pseudo Least Recently Used (TreePLRU)* adalah salah satu algoritma pengelolaan *cache* yang digunakan untuk memilih item mana yang harus dikeluarkan dari *cache* ketika terjadi *miss* yaitu ketika data atau instruksi yang dibutuhkan tidak ada di dalam *cache*. Tree-PLRU adalah metode varian dari metode *Least Recently Used (LRU)* yang menggunakan pohon pencarian biner untuk melacak blok *cache* yang baru diakses dalam setiap set. TreePLRU memperhitungkan pola akses yang lebih kompleks daripada algoritma pengelolaan *cache* sederhana seperti LRU. Meskipun lebih kompleks, TreePLRU tetap efisien dalam penggunaan sumber daya. Struktur pohon pada TreePLRU memungkinkan pencarian dan penggantian yang cepat dan efisien. Maka dari itu, dengan membandingkan metode TreePLRU dan LRU, dapat dilihat tingkatan efisiensi dalam pengelolaan *cache*, kecepatan pemrosesan data, dan penggunaan sumber daya secara keseluruhan antara dua metode tersebut [6].

Dalam penelitian ini, diusulkan metode TreePLRU untuk memperbaiki cara *cache* beradaptasi dengan pola akses yang berubah. Pola akses yang berubah dapat menurunkan probabilitas *cache hit*. Penelitian ini dilakukan menggunakan simulator gem5 yang digunakan untuk melakukan pemodelan dan simulasi kinerja arsitektur komputer x86. Selain itu, dalam penelitian ini akan dilakukan pengujian menggunakan perkalian matriks sebagai standar perbandingan. Pengujian ini akan mengukur tingkat keberhasilan dan kegagalan *cache*, yang dikenal juga dengan *cache hit* dan *cache miss*. Kedua parameter tersebut merupakan indikator performa *cache*.

### *1.2. Rumusan masalah*

Berdasarkan latar belakang yang dikemukakan di atas, maka rumusan masalah dalam penelitian ini sebagai berikut:

A. Bagaimana mensimulasikan metode TreePLRU pada arsitektur x86 untuk *cache replacement policy*?
B. Bagaimana performansi metode TreePLRU pada arsitektur x86 untuk *cache replacement policy?*

Berikut batasan masalah pada penelitian ini:

A. Simulator yang digunakan adalah simulator berbasis gem5.
B. Digunakan perkalian matriks berukuran 128 x 128, 256 x 256, dan 512 x 512 sebagai alat ukur perbandingan, karena perkalian matriks dapat digunakan untuk mengevaluasi keunggulan dan kelemahan masing-masing algoritma dalam berbagai skenario penggunaan *cache.*

### *1.3. Tujuan*

Adapun tujuan dari penelitian ini adalah:

*A.* Mensimulasikan metode TreePLRU pada arsitektur x86 untuk *cache replacement policy*.
*B.* Menganalisis perbandingan *cache hit* dan *cache miss* pada *cache replacement policy* antara metode TreePLRU dan LRU pada arsitektur x86.

### *1.4. Rencana Kegiatan*

A. Studi literatur

Pada tahap ini dilakukan penelusuran literatur terkait arsitektur komputer x86, metode Tree-PLRU sebagai kebijakan penggantian *(replacement policy) cache*. Tujuannya untuk memahami konsep, prinsip, dan teknik yang terkait dengan topik penelitian.

*B.* Persiapan lingkungan simulasi

Pada tahap ini dilakukan peng-instalan dan konfigurasi gem5 yang merupakan simulator arsitektur komputer yang akan digunakan untuk melakukan simulasi.

C. Skenario pengujian

Pada tahap ini dilakukan skenario pengujian yang akan dilakukan untuk menganalisis performansi Arsitektur Komputer X86 dengan menggunakan Tree-PLRU.

*D.* Analisis data

Pada tahap ini dilakukan analisis data pemrformansi yang telah dikumpulkan dari hasil simulasi.

*E.* Penyusunan laporan

Pada tahap ini dilakukan penulisan laporan penelitian yang mencakup deskripsi penelitian, metodologi, hasil analisis, dan kesimpulan yang diperoleh. Laporan ini akan menjadi dokumentasi utama dari penelitian yang dilakukan.

### *1.5. Jadwal kegiatan*

Jadwal kegiatan meliputi sebagai berikut:

Tabel 1.1 Rencana kegiatan

| **Kegiatan** | **Bulan** | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| Studi literatur | | | | | | |
| Persiapan lingkungan simulasi | | | | | | |
| Skenario Pengujian | | | | | | |
| Analisis Data | | | | | | |
| Penyusunan Laporan | | | | | | |