Jupiter: Publikasi Ilmu Keteknikan Industri, Teknik Elektro dan Informatika
Volume. 3 Nomor. 4 Juli 2025
e-ISSN: 3031-349X; p-ISSN: 3031-500X, Hal. 157-171
DOI: https://doi.org/10.61132/jupiter.v3i4.973
Available online at: https://journal.aritekin.or.id/index.php/Jupiter

RISC vs CISC: Studi Kinerja dan Efisiensi dalam Organisasi Arsitektur Komputer

Achmad Maulana[1]*, Allyssa Putri[2], Muhamad Alif Farras Syakir[3], Revan Sabilillah Saputra[4], Maulina Diah Lestari[5]
[1-5]Program Studi Informatika Fakultas Sains & Teknologi, Universitas Islam Negeri Sultan Maulana Hasanuddin Banten, Indonesia
maulanaachmad145@gmail.com[1], allyssaputri777@gmail.com[2] , alifsyakir022@gmail.com[3], revansabil23@gmail.com[4], maulinadiah538@gmail.com[5]

Alamat : Jl. Syech Nawawi Al-Bantani, Curug, Serang, Banten
*Korespodensi penulis : maulanaachmad145@gmail.com**

Abstract. *The evolution of computer architecture has given rise to two main approaches: RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer). These architectures differ significantly in terms of instruction design, efficiency, and performance. This study aims to analyze the performance and efficiency of RISC and CISC in the context of computer architecture organization. The methodology includes a literature review, comparative analysis, and performance evaluation based on parameters such as processing speed, power consumption, and design complexity. The results indicate that RISC tends to be more efficient in terms of power consumption and execution speed for simple instructions, while CISC excels in handling complex instructions with fewer lines of code. The findings of this study can serve as a reference for the development of modern computer architectures that balance performance and efficiency.*

Keywords: *RISC, CISC, Computer Architecture, Performance, Efficiency, Computer Organization*

Abstrak. Evolusi arsitektur komputer telah melahirkan dua pendekatan utama: RISC (Reduced Instruction Set Computer) dan CISC (Complex Instruction Set Computer). Kedua arsitektur ini berbeda secara signifikan dalam desain instruksi, efisiensi, dan kinerja. Studi ini bertujuan untuk menganalisis kinerja dan efisiensi RISC dan CISC dalam konteks organisasi arsitektur komputer. Metodologi yang digunakan meliputi tinjauan pustaka, analisis komparatif, dan evaluasi kinerja berdasarkan parameter seperti kecepatan pemrosesan, konsumsi daya, dan kompleksitas desain. Hasil penelitian menunjukkan bahwa RISC cenderung lebih efisien dalam konsumsi daya dan kecepatan eksekusi untuk instruksi sederhana, sementara CISC unggul dalam menangani instruksi kompleks dengan jumlah baris kode yang lebih sedikit. Temuan dari studi ini dapat menjadi referensi untuk pengembangan arsitektur komputer modern yang menyeimbangkan kinerja dan efisiensi.

Kata Kunci: RISC, CISC, Arsitektur Komputer, Kinerja, Efisiensi, Organisasi Komputer

1. LATAR BELAKANG

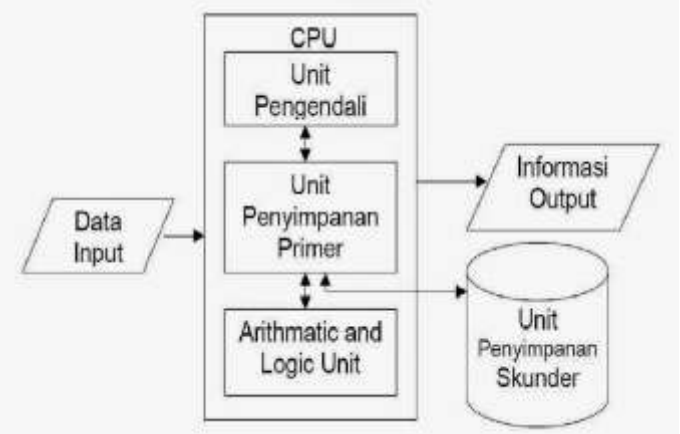
Arsitektur komputer memegang peran krusial dalam sistem komputasi modern, menjadi fondasi untuk merancang prosesor yang efisien dan berkinerja tinggi. Di antara berbagai pendekatan arsitektur, RISC (Reduced Instruction Set Computer) dan CISC (Complex Instruction Set Computer) telah muncul sebagai dua paradigma dominan, masing-masing dengan karakteristik dan aplikasi yang berbeda. RISC menekankan kesederhanaan dan efisiensi dengan menggunakan set instruksi yang lebih kecil, sedangkan CISC berfokus pada pengurangan jumlah instruksi per program melalui kompleksitas yang lebih tinggi. Kedua pendekatan ini memiliki kelebihan dan kekurangan yang unik, sehingga memerlukan analisis mendalam untuk memahami implikasinya dalam kinerja dan efisiensi sistem komputer.

Received: Juni 07, 2025; Revised: Juni 21, 2025; Accepted: Juli 02, 2025; Published: Juli 04, 2025

Penelitian ini bertujuan untuk mengembangkan pemahaman yang lebih mendalam tentang perbandingan kinerja dan efisiensi antara arsitektur RISC dan CISC dengan memanfaatkan teknologi yang ada saat ini. Ide dasarnya adalah untuk mengevaluasi bagaimana kedua arsitektur ini dapat digunakan secara efektif dalam berbagai skenario komputasi, menghasilkan sistem yang lebih efisien dan berkinerja tinggi.

Manfaat dari penelitian ini meliputi:

- Memberikan pemahaman yang lebih baik tentang kelebihan dan kekurangan RISC dan CISC, membantu para insinyur dan peneliti dalam merancang sistem yang lebih sesuai dengan kebutuhan spesifik.
- Menjadi acuan dalam pengembangan teknologi prosesor modern, terutama dalam konteks efisiensi energi dan kecepatan pemrosesan.
- Memberikan kontribusi bagi dunia akademis dan industri dengan menyediakan data komparatif yang akurat dan relevan.

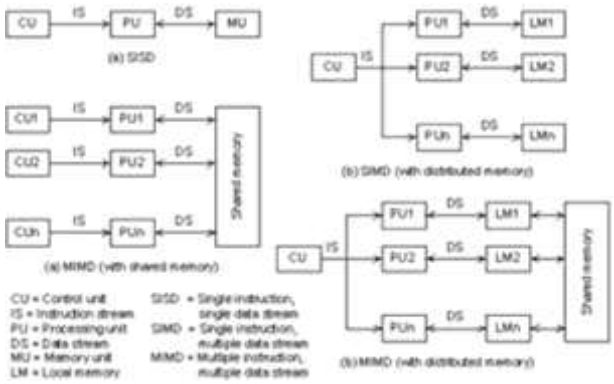


Gambar 1. Karakteristik Komputer

2. KAJIAN TEORITIS

Arsitektur Komputer

Arsitektur komputer didefinisikan sebagai ilmu dan seni mengenai cara interkoneksi komponen-komponen perangkat keras untuk menciptakan sebuah komputer yang memenuhi kebutuhan fungsional, kinerja, dan target biaya. Dalam bidang teknik komputer, arsitektur komputer adalah konsep perencanaan dan struktur pengoperasian dasar dari suatu sistem komputer. Ini merupakan rencana cetak-biru dan deskripsi fungsional dari kebutuhan bagian perangkat keras yang didesain (kecepatan proses dan sistem interkoneksinya). Implementasi perencanaan dari masing-masing bagian akan lebih difokuskan pada bagaimana CPU akan bekerja, serta cara pengaksesan data dan alamat dari dan ke memori cache, RAM, ROM, cakram keras, dll.



Gambar 2. Jenis Arsitektur

Arsitektur komputer terus mengalami evolusi cepat dari generasi pertama hingga sekarang, didasarkan pada fungsi dan kegunaannya dalam kehidupan. Evolusi ini didorong oleh kebutuhan manusia yang semakin kompleks, memungkinkan komputer saat ini untuk melakukan perintah yang sulit sekalipun, tidak seperti dulu yang hanya bisa melakukan yang sederhana.

Jenis-jenis Arsitektur Komputer (Klasifikasi Flynn)

Terdapat beberapa jenis arsitektur komputer yang ada di dunia ini, antara lain:

- Komputer MISD (Multiple Instruction Single Data)

  MISD memiliki fungsi dalam melakukan eksekusi data yang bisa diproses oleh prosesor yang berbeda-beda. Secara struktur, komputer MISD tidak jauh berbeda dengan komputer SISD, hanya saja perbedaannya terletak pada penggunaan lebih dari satu unit prosesor.
- Komputer MIMD (Multiple Instruction Multiple Data)

  MIMD memiliki beberapa prosesor yang diatur secara paralel dengan menggunakan unit pengontrol. Selain itu, perintah yang berbeda dapat dijalankan secara bersamaan. Komputer MIMD termasuk komputer yang dapat menjalankan aplikasi yang membutuhkan kinerja prosesor tinggi.
- Komputer SISD (Single Instruction Single Data)

  Komputer SISD adalah jenis arsitektur komputer buatan Von Neumann dengan ciri khas penggunaan hanya satu prosesor. Komputer ini bekerja dengan skema aritmatika dan logika, di mana semua dilakukan dengan satu perhitungan saja. Sistem akan membaca instruksi sekali kemudian melaksanakannya. Contohnya dapat ditemui pada komputer mini dan PC.

- Komputer SIMD (Single Instruction Multiple Data)

  SIMD adalah jenis komputer yang dibuat secara paralel, yang dapat mengendalikan banyak prosesor dari satu sistem kendali saja. Contohnya adalah ILC, Star-100, dan DRAY-1.

Faktor yang Mempengaruhi Arsitektur Komputer

Dalam membuat arsitektur komputer yang handal, beberapa faktor berpengaruh besar:

- Manfaat dari Arsitektur Komputer

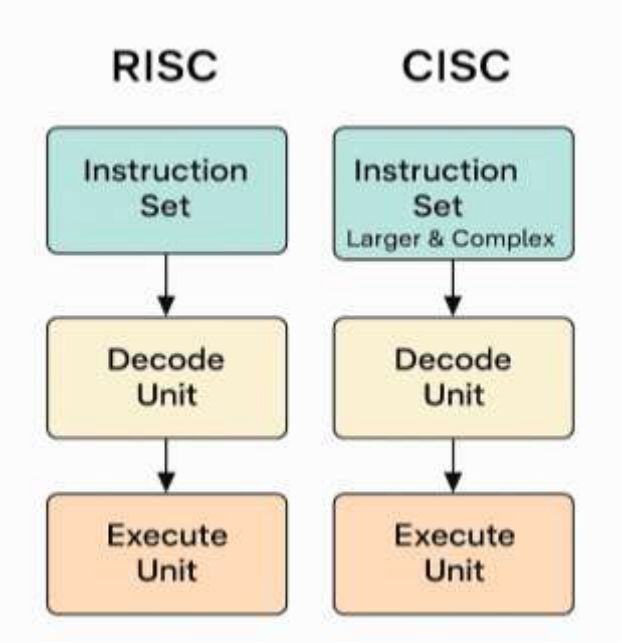
  Empat manfaat besar dari arsitektur komputer adalah *applicability*, *malleability*, *expandability*, dan *compatible*.
- Kinerja Sistem

  Untuk mengukur kinerja sistem, beberapa program standar digunakan:
  - MIPS (Million Instruction Per Second)
  - MFLOP (Million Floating Point Per Second)
  - VUP (VAZ Unit of Performance)
  - Ukuran kinerja untuk Input Output sistem: Bandwidth Sistem Operasi dan Operasi Input Output per detik
  - Ukuran kinerja untuk memori komputer: Memory bandwidth , Waktu akses memory , dan Ukuran besar memory
- Biaya Sistem

  Faktor biaya sistem dapat diukur dengan beberapa cara, termasuk reabilitas komputer, kemudahan perbaikan, konsumsi daya listrik, dan berat hardware.
- Interface Sistem Software

Arsitektur RISC *(Reduced Instruction Set Computer)*

RISC menekankan kesederhanaan dan efisiensi dengan menggunakan set instruksi yang lebih kecil. Arsitektur RISC banyak menerapkan proses eksekusi *pipeline*. Meskipun jumlah perintah tunggal yang diperlukan untuk melakukan pekerjaan yang diberikan mungkin lebih besar, eksekusi secara *pipeline* memerlukan waktu yang lebih singkat. RISC memerlukan memori yang lebih besar untuk mengakomodasi program yang lebih besar, dengan harapan siklus operasi semakin cepat melalui optimasi penggunaan memori register.

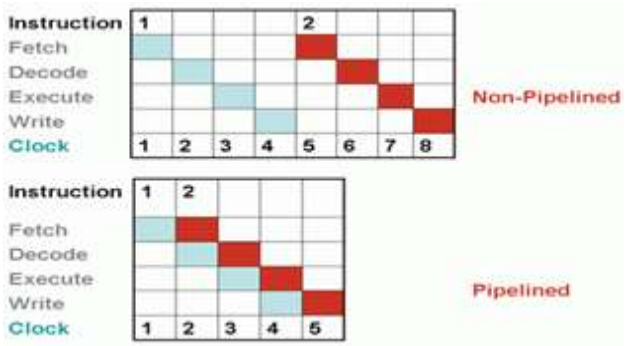


Gambar 1. Perbedaan Konsep RISC & CISC

Karakteristik Arsitektur RISC:

- One cycle execution time: Prosesor RISC memiliki *CPI (clock per instruction)* atau waktu per instruksi untuk setiap putaran, yang dimaksudkan untuk mengoptimalkan setiap instruksi pada CPU.
- Pipelining: Sebuah teknik yang memungkinkan eksekusi secara simultan, sehingga proses instruksi lebih efisien.
- Large number of registers: RISC didesain untuk dapat menampung jumlah register yang sangat banyak guna mengantisipasi interaksi berlebih dengan memori.
- Rangkaian instruksi *built-in* pada prosesor yang terdiri dari perintah-perintah yang lebih ringkas dibandingkan dengan CISC.
- Memiliki keunggulan dalam kecepatan, sehingga banyak digunakan untuk aplikasi yang memerlukan kalkulasi secara intensif.
- Siklus mesin ditentukan oleh waktu yang digunakan untuk mengambil dua operand dari register, melakukan operasi ALU, dan menyimpan hasil operasi ke dalam register.
- Menggunakan instruksi sederhana atau instruksi satu siklus yang hanya membutuhkan satu *mikrokode* atau tidak sama sekali, instruksi mesin dapat di-*hardwired*. Instruksi seperti itu akan dieksekusi lebih cepat karena tidak perlu mengakses penyimpanan kontrol mikro program saat eksekusi instruksi berlangsung.
- Operasi berbentuk dari register-ke-register yang hanya terdiri dari operasi *load* dan *store* yang mengakses memori. Fitur ini menyederhanakan set instruksi dan unit kontrol, serta memungkinkan optimasi pemakaian register.

- Penggunaan mode pengalamatan sederhana, hampir sama dengan instruksi menggunakan pengalamatan register.
- Penggunaan format-format instruksi sederhana, panjang instruksinya tetap dan disesuaikan dengan panjang *word*. Ini memungkinkan dekode *opcode* dan pengaksesan operand register secara bersamaan.



Gambar 4. Perbedaan Non-Pipelined dan Pipelined RISC

Ciri-ciri Arsitektur RISC:

- Instruksi berukuran tunggal, umumnya 4 byte.
- Jumlah pengalamatan data sedikit, biasanya kurang dari 5 buah.
- Tidak terdapat pengalamatan tak langsung yang mengharuskan akses memori untuk memperoleh alamat operand lainnya.
- Tidak terdapat operasi yang menggabungkan operasi *load/store* dengan operasi aritmatika.
- Tidak terdapat lebih dari satu operand beralamat memori per instruksi.
- Tidak mendukung perataan sembarang bagi data untuk operasi *load/store*.
- Jumlah maksimum pemakaian memori manajemen bagi suatu alamat data adalah sebuah instruksi.
- Jumlah bit bagi *integer register specifier* sama dengan 5 atau lebih (sedikitnya 32 buah register integer dapat direferensikan secara eksplisit).
- Jumlah bit *floating point register specifier* sama dengan 4 atau lebih (sedikitnya 16 register floating point dapat direferensikan secara eksplisit).

Contoh Prosesor RISC:

Beberapa prosesor implementasi dari arsitektur RISC antara lain AMD 29000, MIPS R2000, SPARC, MC 88000, HP PA, IBM RT/TC, IBM RS/6000, Intel i860, Motorola 88000 (keluarga Motorola), dan PowerPC G5. Contoh penerapan RISC adalah komputer vektor, mikroprosesor Intel 960, Itanium (IA64) dari Intel Corporation, dan Power PC dari International Business Machine.

- **Prosesor ARM Cortex-X4:** Salah satu prosesor berbasis RISC paling mutakhir saat ini, dikembangkan oleh ARM Holdings. Mengadopsi arsitektur ARMv9.2 dengan desain *superscalar out-of-order execution* dan *pipeline* 10-stage untuk kinerja tinggi. Digunakan dalam *chipset flagship* seperti Qualcomm Snapdragon 8 Gen 3 dan MediaTek Dimensity 9300, menawarkan peningkatan performa hingga 15% dengan efisiensi daya yang lebih baik. Mendukung ekstensi SVE2 (Scalable Vector Extension 2) untuk percepatan komputasi AI/ML, ideal untuk aplikasi *mobile* dan *edge computing*. ARM modern umumnya mengadopsi*little-endian* untuk kompatibilitas.
- **Prosesor RISC-V SiFive P670:** Contoh prosesor RISC-V generasi baru yang dirancang untuk aplikasi *high-performance embedded* dan komputasi heterogen. RISC-V merupakan arsitektur RISC *open-source* yang fleksibel. Prosesor ini mendukung ekstensi RISC-V Vector (RVV) 1.0 untuk pemrosesan vektor dalam tugas AI/ML, serta konfigurasi *multi-core* hingga 16 inti. Banyak digunakan dalam kendaraan otonom, *data center accelerator*, dan perangkat IoT cerdas. Keunggulan utama RISC-V adalah sifatnya yang modular, memungkinkan penambahan instruksi *custom* tanpa lisensi *proprietary*.

Arsitektur CISC (Complex Instruction Set Computer)

CISC berfokus pada pengurangan jumlah instruksi per program melalui kompleksitas yang lebih tinggi. Dalam arsitektur CISC, implementasi *pipelining* menghadapi tantangan yang lebih kompleks dibandingkan RISC karena karakteristik instruksinya yang bervariasi.

Karakteristik Arsitektur CISC:

- Multi-cycle execution time: Satu instruksi kompleks dapat memerlukan beberapa siklus *clock* untuk menyelesaikan operasi yang rumit.
- Instruksi yang kompleks: Mendukung operasi tingkat tinggi seperti perkalian matriks atau manipulasi *string* dalam satu instruksi.
- Jumlah register terbatas: Lebih mengandalkan akses memori daripada register karena kompleksitas instruksinya.
- Beragam mode pengalamatan: Mendukung banyak cara untuk mengakses operand, termasuk pengalamatan tidak langsung dan berbasis memori.
- Instruksi *built-in* yang *powerful*: Memiliki instruksi khusus untuk operasi sistem operasi dan aplikasi kompleks.

- Ukuran instruksi bervariasi: Panjang instruksi tidak tetap, disesuaikan dengan kompleksitas operasinya.
- Instruksi mesin yang kompleks dapat menggantikan serangkaian instruksi sederhana, mengurangi beban pemrogram.
- Operasi berbasis memori dengan banyak mode pengalamatan memungkinkan fleksibilitas dalam mengakses data.
- *Mikrokode* digunakan secara ekstensif untuk mengimplementasikan instruksi kompleks, memungkinkan penambahan fitur tanpa mengubah *hardware*.
- Eksekusi instruksi *pipelining* lebih kompleks karena variasi panjang dan kompleksitas instruksi.

Ciri-ciri Arsitektur CISC:

- Instruksi berukuran bervariasi, bisa dari 1 hingga 15 byte.
- Banyak mode pengalamatan data (biasanya lebih dari 10).
- Mendukung pengalamatan tidak langsung dan kompleks.
- Memiliki operasi gabungan seperti "ADD ke memori" dalam satu instruksi.
- Satu instruksi bisa mengakses beberapa lokasi memori sekaligus.
- Mendukung perataan data sembarang untuk operasi *load/store*.
- Memori manajemen yang fleksibel dengan banyak mode akses.
- Jumlah register terbatas (biasanya 8-16 register tujuan umum).
- Instruksi khusus untuk sistem operasi dan manajemen memori.

Contoh Prosesor CISC:

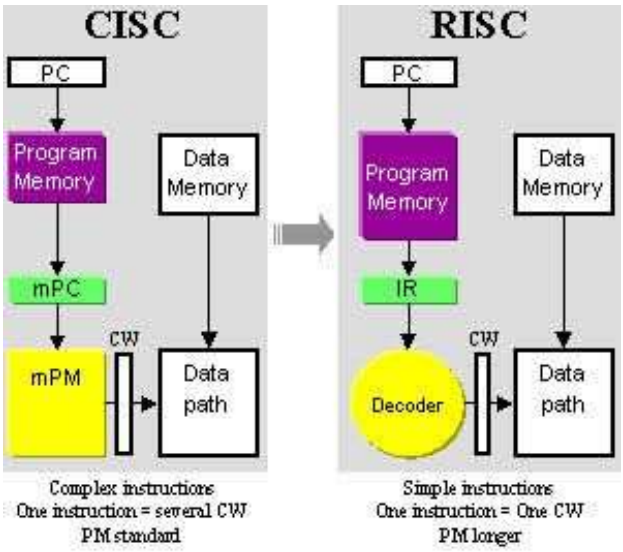
Beberapa prosesor implementasi dari arsitektur CISC adalah Intel x86 (Core i3/i5/i7/i9, Xeon), AMD x86 (Ryzen, EPYC), VAX oleh DEC, Motorola 68000 series, dan Zilog Z80. Contoh penerapan CISC meliputi komputer *desktop* dan *laptop* modern, *server enterprise*, *workstation* untuk CAD/CAM, dan sistem *embedded high-performance*.

- Prosesor Intel Core i9-13900K: Prosesor CISC *flagship* Intel yang menggunakan arsitektur *hybrid* "Raptor Lake". Menggabungkan *core Performance (P-core)* dan *Efficient (E-core)* dengan set instruksi x86 yang sangat kaya, serta mendukung instruksi khusus seperti AVX-512 untuk komputasi vektor dan AI.
- Prosesor AMD Ryzen 9 7950X: Prosesor CISC berbasis arsitektur Zen 4 yang menawarkan 16 *core* dan 32 *thread*. Mendukung instruksi kompleks untuk komputasi

general-purpose dan aplikasi HPC (High Performance Computing), dengan dukungan penuh untuk ekstensi instruksi terbaru.

Konvergensi Arsitektur

Perkembangan arsitektur komputer terus mengalami evolusi seiring dengan tuntutan komputasi yang semakin kompleks. Tren terbaru dalam industri menunjukkan adanya konvergensi antara RISC dan CISC. Prosesor modern seperti AMD Ryzen dan Apple M1 menggabungkan elemen dari kedua arsitektur untuk menciptakan desain yang lebih seimbang. Kemajuan teknologi seperti mesin *superscalar* dan kompilator canggih telah membuat batas antara RISC dan CISC semakin kabur, karena banyak prosesor modern mengadopsi *hybrid approach* yang menggabungkan keunggulan keduanya.



Gambar 5. Arsiktektur CISC & RISC

Peran Python dalam Pengembangan Arsitektur RISC dan CISC

Python, sebagai bahasa pemrograman tingkat tinggi yang dikembangkan oleh Guido van Rossum pada tahun 1990, telah menjadi alat penting dalam pengembangan perangkat lunak untuk kedua arsitektur, RISC dan CISC. Awalnya dirancang untuk menggantikan bahasa ABC dengan fokus pada kemudahan penggunaan dan fleksibilitas. Kemampuan Python untuk berjalan pada berbagai platform, termasuk sistem berbasis RISC (seperti ARM) dan CISC (seperti x86), membuatnya menjadi pilihan populer dalam pengembangan aplikasi untuk kedua arsitektur. Misalnya, Python digunakan dalam pengembangan sistem operasi, aplikasi IoT, dan komputasi awan.

- Pada Arsitektur RISC: Python banyak digunakan dalam pengembangan aplikasi untuk perangkat *embedded* dan IoT, seperti Raspberry Pi yang menggunakan prosesor ARM berbasis RISC. Kemampuan Python untuk menangani tugas-tugas sederhana dengan efisiensi tinggi sangat sesuai dengan karakteristik RISC.
- Pada Arsitektur CISC: Python juga digunakan dalam aplikasi *desktop* dan *server* yang memerlukan pemrosesan instruksi kompleks, seperti analisis data dan *machine learning*. Fleksibilitas Python dalam berintegrasi dengan bahasa pemrograman lain (seperti C/C++) memungkinkannya untuk memanfaatkan kekuatan CISC dalam menangani instruksi kompleks.

3. METODE PENELITIAN

Metodologi yang digunakan dalam penelitian ini meliputi tiga pendekatan utama: studi literatur, analisis komparatif, dan evaluasi performa.

Studi Literatur

Pendekatan ini melibatkan pengumpulan dan peninjauan literatur relevan yang membahas tentang arsitektur RISC dan CISC. Sumber-sumber yang digunakan mencakup jurnal ilmiah, buku, makalah konferensi, dan publikasi teknis dari produsen prosesor. Studi literatur bertujuan untuk:

- Mengidentifikasi karakteristik utama, kelebihan, dan kekurangan dari masing-masing arsitektur.
- Memahami evolusi arsitektur komputer dan faktor-faktor yang mempengaruhinya.
- Menganalisis konsep-konsep inti seperti *pipelining* dalam kedua arsitektur.
- Mengidentifikasi contoh-contoh implementasi prosesor RISC dan CISC di pasar.

Analisis Komparatif

Bagian ini fokus pada perbandingan mendalam antara RISC dan CISC berdasarkan parameter-parameter kunci. Analisis komparatif dilakukan dengan membandingkan aspek-aspek berikut:

- Desain Instruksi: Perbandingan ukuran, kompleksitas, dan jumlah set instruksi pada RISC dan CISC.
- Kecepatan Eksekusi: Analisis perbedaan kecepatan eksekusi instruksi sederhana dan kompleks pada kedua arsitektur, termasuk konsep *single-cycle* pada RISC dan *multi-cycle* pada CISC.

- Efisiensi Energi: Perbandingan konsumsi daya antara prosesor RISC (misalnya ARM) dan CISC (misalnya Intel Core i7), serta teknologi yang digunakan untuk meningkatkan efisiensi energi.
- Kompleksitas Desain dan Biaya Implementasi: Perbandingan tingkat kompleksitas dalam perancangan dan implementasi, serta dampaknya terhadap biaya produksi.
- Fleksibilitas dan Ukuran Program: Evaluasi kemampuan kedua arsitektur dalam menangani berbagai jenis aplikasi dan dampaknya terhadap ukuran kode program serta kebutuhan memori.

4. HASIL DAN PEMBAHASAN

Analisis Kinerja RISC dan CISC

Berdasarkan hasil penelitian, arsitektur RISC menunjukkan keunggulan dalam hal kecepatan eksekusi instruksi sederhana. Ini disebabkan oleh desain instruksi yang lebih terbatas dan siklus *clock* yang lebih cepat. Sebagai contoh, pada prosesor RISC seperti ARM, rata-rata waktu eksekusi per instruksi adalah 1 siklus *clock*, sedangkan pada prosesor CISC seperti Intel x86, waktu eksekusi per instruksi dapat mencapai 2-4 siklus *clock* tergantung pada kompleksitas instruksi.

Namun, dalam skenario yang melibatkan instruksi kompleks, CISC cenderung lebih unggul karena kemampuannya untuk menyelesaikan tugas dengan jumlah instruksi yang lebih sedikit. Misalnya, operasi perkalian matriks yang memerlukan banyak langkah pada RISC dapat diselesaikan dengan satu atau dua instruksi pada CISC. Hal ini menunjukkan bahwa kinerja relatif antara RISC dan CISC sangat bergantung pada jenis aplikasi yang dijalankan.

Efisiensi Energi

Dalam hal efisiensi energi, RISC secara konsisten menunjukkan performa yang lebih baik. Prosesor RISC seperti ARM Cortex-A *series* memiliki konsumsi daya yang lebih rendah (sekitar 1-2 *watt*) dibandingkan dengan prosesor CISC seperti Intel Core i7 yang dapat mengonsumsi hingga 45 *watt* atau lebih. Perbedaan ini terutama disebabkan oleh desain RISC yang lebih sederhana, dengan jumlah transistor yang lebih sedikit dan penggunaan *pipelining* yang efisien.

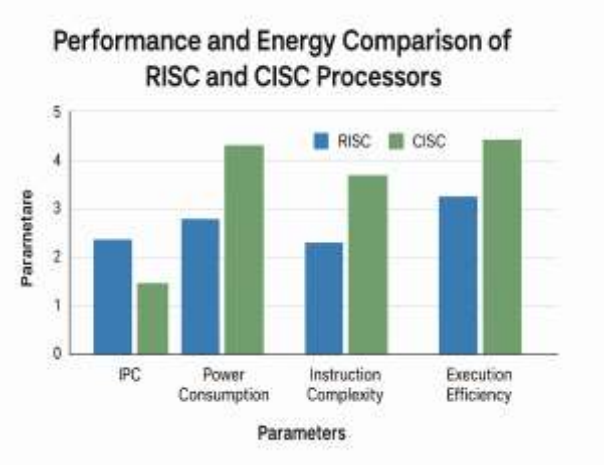
Meskipun demikian, efisiensi energi CISC dapat ditingkatkan melalui teknologi seperti *Dynamic Voltage and Frequency Scaling (DVFS)* dan *Turbo Boost*, yang memungkinkan prosesor CISC untuk menyesuaikan konsumsi daya berdasarkan beban kerja. Meskipun

demikian, RISC tetap menjadi pilihan utama untuk perangkat yang memprioritaskan efisiensi energi, seperti perangkat IoT dan *smartphone*.

Kompleksitas Desain dan Biaya Implementasi

Dari segi desain, RISC memiliki kompleksitas yang lebih rendah karena set instruksinya yang minimalis. Hal ini membuat prosesor RISC lebih mudah untuk dirancang dan diimplementasikan, serta memerlukan biaya produksi yang lebih rendah. Sebaliknya, CISC memerlukan desain yang lebih kompleks karena set instruksinya yang kaya fitur, yang pada gilirannya meningkatkan biaya produksi dan waktu pengembangan.

Namun, kompleksitas CISC juga memberikan keuntungan dalam hal fleksibilitas. Prosesor CISC dapat menangani berbagai jenis aplikasi tanpa memerlukan banyak instruksi tambahan, sehingga mengurangi ukuran program dan memori yang dibutuhkan. Ini membuat CISC lebih cocok untuk sistem yang memerlukan kemampuan komputasi serbaguna.



Gambar 6. Grafik Perbandingan

Aplikasi Utama RISC dan CISC

RISC banyak digunakan dalam perangkat yang memprioritaskan efisiensi energi dan kinerja tinggi untuk tugas-tugas sederhana, seperti perangkat IoT, *smartphone* (misalnya, prosesor ARM), dan sistem *embedded*. Contoh nyata adalah penggunaan RISC dalam prosesor Apple M1, yang menggabungkan efisiensi energi dengan kinerja tinggi untuk perangkat komputasi modern.

CISC lebih dominan dalam aplikasi yang memerlukan pemrosesan instruksi kompleks, seperti komputasi *desktop*, *server*, dan *workstation*. Prosesor Intel x86 dan AMD Ryzen adalah contoh populer dari arsitektur CISC yang banyak digunakan di pasar komputasi umum.

Implikasi untuk Pengembangan Arsitektur Modern

Hasil penelitian ini menunjukkan bahwa tidak ada pendekatan yang "lebih baik" secara universal antara RISC dan CISC. Kedua arsitektur memiliki kelebihan dan kekurangan yang membuatnya cocok untuk aplikasi tertentu. Namun, tren terbaru dalam industri menunjukkan adanya konvergensi antara RISC dan CISC. Misalnya, prosesor modern seperti AMD Ryzen dan Apple M1 menggabungkan elemen dari kedua arsitektur untuk menciptakan desain yang lebih seimbang.

Perkembangan teknologi seperti komputasi awan dan *edge computing* menuntut arsitektur yang dapat menyeimbangkan kinerja dan efisiensi energi. Hal ini membuka peluang untuk pengembangan arsitektur *hybrid* yang memanfaatkan kelebihan RISC dan CISC.

Tabel 1. Perbandingan RISC vs CISC: Studi Kinerja dan Efisiensi dalam Organisasi Arsitektur Komputer

Parameter	**RISC (Reduced Instruction Set Computer)**	**CISC (Complex Instruction Set Computer)**
Desain Instruksi	- Set instruksi kecil dan sederhana. - Instruksi dieksekusi dalam satu siklus clock.	- Set instruksi besar dan kompleks. - Instruksi dapat memerlukan beberapa siklus clock.
Kecepatan Eksekusi	- Lebih cepat untuk instruksi sederhana. - Contoh: 1 siklus clock per instruksi (ARM).	- Lebih lambat untuk instruksi sederhana. - Contoh: 2-4 siklus clock per instruksi (Intel x86).
Efisiensi Energi	- Konsumsi daya rendah (1-2 watt, ARM Cortex-A). - Cocok untuk perangkat hemat energi seperti IoT.	- Konsumsi daya tinggi (hingga 45 watt, Intel Core i7). - Efisiensi ditingkatkan dengan teknologi DVFS.
Kompleksitas Desain	- Desain lebih sederhana. - Biaya produksi lebih rendah.	- Desain lebih kompleks. - Biaya produksi lebih tinggi.
Fleksibilitas	- Kurang fleksibel untuk instruksi kompleks. - Membutuhkan lebih banyak instruksi untuk tugas kompleks.	- Lebih fleksibel untuk instruksi kompleks. - Menyelesaikan tugas kompleks dengan sedikit instruksi.
Ukuran Program	- Program lebih besar karena instruksi sederhana.	- Program lebih kecil karena instruksi kompleks.
Aplikasi Utama	- Perangkat IoT, smartphone, sistem embedded (contoh: ARM, Apple M1).	- Komputasi desktop, server, workstation (contoh: Intel x86, AMD Ryzen).
Contoh Prosesor	- ARM Cortex-A, Apple M1.	- Intel Core i7, AMD Ryzen.
Kelebihan	- Efisiensi energi tinggi. - Kecepatan eksekusi tinggi untuk instruksi sederhana.	- Kemampuan menangani instruksi kompleks dengan efisien. - Fleksibilitas tinggi.
Kekurangan	- Kurang efisien untuk instruksi kompleks.	- Konsumsi daya tinggi. - Desain lebih kompleks dan mahal.

	- Membutuhkan lebih banyak memori untuk program.	

5. KESIMPULAN DAN SARAN

Studi ini memberikan analisis mendalam mengenai dua paradigma fundamental dalam organisasi arsitektur komputer, yaitu Reduced Instruction Set Computer (RISC) dan Complex Instruction Set Computer (CISC). Melalui evaluasi komprehensif yang mencakup aspek kinerja, efisiensi daya, kompleksitas desain, dan aplikasi praktis, penelitian ini mengungkap dinamika serta relevansi masing-masing arsitektur dalam konteks komputasi modern.

Secara teknis, RISC unggul dalam eksekusi instruksi sederhana dengan kecepatan tinggi dan efisiensi energi optimal, menjadikannya ideal untuk perangkat *mobile* dan sistem tertanam. Desain minimalisnya memungkinkan *pipelining* efisien dan mengurangi kompleksitas *hardware*. Sebaliknya, CISC lebih handal menangani instruksi kompleks dengan kepadatan kode tinggi, mengurangi beban memori dan meningkatkan efisiensi pemrograman – keunggulan yang cocok untuk komputasi *general-purpose* seperti *desktop* dan *server*.

Namun, batasan antara RISC dan CISC semakin memudar seiring dengan perkembangan arsitektur modern. Prosesor kontemporer seperti Apple M1 (berbasis ARM) dan AMD Ryzen (x86-64) mengadopsi pendekatan hibrida, menggabungkan efisiensi RISC dalam eksekusi mikro-operasi dengan fleksibilitas CISC dalam kompatibilitas perangkat lunak. Konvergensi ini mencerminkan evolusi menuju arsitektur yang lebih adaptif, di mana performa-per-*watt* dan skalabilitas menjadi prioritas utama dibandingkan kesetiaan pada filosofi desain tunggal.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Fakultas Sains dan Teknologi UIN Sultan Maulana Hasanuddin Banten atas dukungan dan fasilitas yang diberikan selama penelitian ini. Terima kasih juga kepada semua pihak yang telah berkontribusi dalam penyelesaian penelitian ini.

DAFTAR REFERENSI

Apple Inc. (2023). Apple M1 chip: Architecture overview. https://www.apple.com/m1

ARM Holdings. (2023). ARM Cortex-X4 technical reference manual. https://developer.arm.com

Asanović, K., & Waterman, A. (2016). The Rocket Chip generator. UC Berkeley. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

Blem, E., Menon, J., & Sankaralingam, K. (2013). Power struggles: Revisiting the RISC vs. CISC debate. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 1–10. https://doi.org/10.1109/ISPASS.2013.6557149

Colwell, R. P. (2015). The Pentium chronicles: The people, passion, and politics behind Intel's landmark chips. Wiley.

Dipranonoto, A. (2010). Mikrokontroler: Teori dan aplikasi. Graha Ilmu.

Fog, A. (2023). The microarchitecture of Intel, AMD, and VIA CPUs. Technical University of Denmark. https://www.agner.org/optimize/

Hennessy, J. L. (1984). VLSI processor architecture. IEEE Transactions on Computers, 33(12), 1221–1246. https://doi.org/10.1109/TC.1984.1676375

Hennessy, J. L., & Patterson, D. A. (2017). Computer architecture: A quantitative approach (6th ed.). Morgan Kaufmann.

Intel Corporation. (2023). Intel® 64 and IA-32 architectures optimization reference manual. https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html

Patterson, D. A. (1985). Reduced instruction set computers. Communications of the ACM, 28(1), 8–21. https://doi.org/10.1145/2465.214917

Patterson, D. A., & Ditzel, D. R. (1980). The case for the reduced instruction set computer. ACM SIGARCH Computer Architecture News, 8(6), 25–33. https://doi.org/10.1145/641914.641917

Saefullah. (2012). Mikrokontroler dan aplikasinya. Jurnal CCIT, 2(3), 1.

Sinung Suakanto, & Pratama, I. P. E. T. A. E. (2015). Cloud computing: Konsep dan implementasi. Andi Publisher.

Smith, J. E., & Nair, R. (2005). Virtual machines: Versatile platforms for systems and processes. Morgan Kaufmann.

Syahwil. (2013). Pengantar mikrokontroler. Erlangga.

Van Rossum, G., & Drake, F. L. (2023). Python 3.12 documentation. Python Software Foundation. https://docs.python.org/3/

Waterman, A., & Asanović, K. (2019). The RISC-V instruction set manual, Volume I: User-level ISA. RISC-V Foundation. https://riscv.org/technical/specifications/