MAKALAH ORGANISASI ARSITEKTUR KOMPUTER

RISC :
REDUCED INSTRUCTION SET COMPUTING

DOSEN PENGAMPU : Brylian Hendrasuryawan, S.Kom., M.T.I.

DISUSUN OLEH :

Muhammad Riza Zaidaan	(L0223031)
Piero Muharoja Anantra	(L0223035)
Siroj Munir	(L0223050)

PROGRAM STUDI SAINS DATA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2023

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala rahmat-Nya sehingga makalah berjudul "RISC : *Reduced Instruction Set Computing*" dapat tersusun hingga selesai. Tidak lupa kami mengucapkan terima kasih atas bantuan para pihak yang berkontribusi dengan membantu pencarian data untuk makalah ini.

Penyusunan makalah ini bertujuan untuk memenuhi nilai tugas mata kuliah Organisasi Arsitektur Komputer. Selain itu, pembuatan makalah juga memiliki tujuan agar menambah wawasan dan pengetahuan bagi penulis maupun pembaca.

Karena keterbatasan pengetahuan maka kami yakin makalah ini masih banyak kekurangan. Oleh karena itu, kami mengharapkan kritik dan saran agar makalah semakin lebih baik. Akhir kata, semoga makalah ini dapat berguna, baik bagi kami selaku penulis ataupun yang membacanya.

Surakarta, 19 Oktober 2023

PENDAHULUAN

Bayangkan komputer sebagai dapur. Ada dua jenis metode memasak, yang pertama adalah menggunakan banyak alat berbeda untuk setiap tugas kecil, seperti pisau untuk memotong, sendok untuk mengaduk, dan lain-lain. Ini disebut "Komputasi Instruksi Kompleks" atau CISC (*Complex Instruction Set Computing*).

Namun sekarang bayangkan dapur yang sangat cerdas dimana kita hanya menggunakan alat yang sangat sederhana dan serbaguna seperti pisau tajam untuk memotong semuanya dan sendok yang bagus untuk mengaduk bahan makanan. Membuat segalanya lebih mudah yang sejalan dengan konsep RISC (*Reduced Instruction Set Computing*) di dunia komputer.

RISC ibarat memilih alat yang sangat bagus dan sederhana untuk menjalankan tugas komputer agar komputer dapat berjalan lebih cepat dan efisien. Semua instruksi komputasi yang dijalankan oleh komputer RISC merupakan instruksi yang sangat sederhana dan lugas, namun komputer dapat mengeksekusinya dengan sangat cepat karena instruksi tersebut dirancang untuk bekerja secara efisien.

LATAR BELAKANG

Di era perkembangan teknologi informasi yang pesat, inovasi dalam desain arsitektur komputer sangat penting untuk memenuhi kebutuhan kinerja yang terus meningkat.
Salah satu keberhasilan penting dalam perkembangan arsitektur komputer adalah munculnya konsep RISC (*Reduced Instruction Set Computing*).

RISC adalah kunci untuk mengatasi beberapa kendala yang dihadapi oleh arsitektur sebelumnya dan berkontribusi terhadap komputasi yang lebih efisien. Arsitektur RISC terbukti menjadi inovasi yang mengubah paradigma melalui prinsip desain perangkat keras mendasar seperti instruksi sederhana, penggunaan saluran pipa, dan fokus pada kinerja tinggi.

PEMBAHASAN

BAB 1: KONSEP DASAR RISC

Pengertian RISC

Reduced Instruction Set Computing atau yang biasa disingkat RISC merupakan sebuah rancangan arsitektur komputer yang menyederhanakan instruksi set pada komputer. Instruksi-intruksi sederhana tersebut akan mengerjakan tugas-tugas dari komputer. Pada CISC atau Complex Instruction Set Computing sebuah instruksi dapat digunakan untuk melakukan sebuah tugas tertentu, namun pada RISC dibutuhkan lebih banyak instruksi-intruksi untuk melakukan sebuah tugas. Hal ini dikarenakan RISC ditulis dalam kode-kode yang lebih sederhana daripada CISC yang lebih kompleks.

Tujuan RISC ditulis dengan kode-kode sederhana adalah untuk meningkatkan kecepatan instruksi pada komputer, khususnya pada implementasi pipe instruction atau pipa instruksi yang lebih sederhana dengan instruksi yang lebih sederhana. Dengan adanya pipe instruction yang sederhana komputer dapat melakukan proses pengerjaanya yang lebih sederhana

Komputer RISC biasanya memiliki banyak register tujuan berkecepatan tinggi, biasanya 16 atau 32, dan memiliki arsitektur muat dan simpan di mana kode untuk instruksi register-register (untuk melakukan tes dan aritmatika) terpisah dari instruksi yang memberikan akses ke memori utama komputer. Komputer RISC memiliki mode pengalamatan yang mudah dan waktu instruksi yang dapat diprediksi karena desain CPU, yang menyederhanakan desain sistem secara keseluruhan.

Pada awal tahun 1980an, perancang arsitektur komputer menyadari potensi manfaat dari teknologi instruction set architecture (ISA) yang lebih sederhana. Meskipun ISA yang lebih sederhana menghasilkan kumpulan instruksi yang lebih kecil, tujuan utama ISA bukan sekadar mengurangi jumlah instruksi secara mekanis, tetapi juga memberikan keuntungan besar dalam desain komputer.

Dengan membuat istilah “RISC”, desainer ingin menunjukkan bahwa mereka berkonsentrasi pada penyederhanaan instruksi untuk mencapai beberapa tujuan utama dalam pengembangan komputer. Pertama, instruksi yang lebih sederhana dapat dieksekusi dengan lebih cepat karena memerlukan waktu yang lebih sedikit dalam siklus jam. Ini meningkatkan kinerja komputer, terutama untuk aplikasi yang membutuhkan instruksi yang dieksekusi dengan cepat.

Selain itu, instruksi yang lebih sederhana memungkinkan penggunaan sumber daya perangkat keras yang lebih efisien. Ini memungkinkan perangkat keras dirancang untuk melakukan eksekusi instruksi dengan lebih efisien, mengurangi beban kompleksitas perangkat keras, dan meningkatkan efisiensi energi. Selain itu, RISC mengurangi kompleksitas eksekusi instruksi dengan menghindari mode pengalamatan bersyarat yang rumit.

Sejarah perkembangan Reduced Instruksi Set Computing (RISC) sangat panjang dan berhasil dalam arsitektur komputer. Banyak produsen dan lembaga penelitian mulai mengembangkan RISC komersial setelah konsepnya diperkenalkan pada awal tahun 1980an. Prosesor RISC seperti IBM POWER, HP PA-RISC, dan beberapa implementasi arsitektur SPARC Sun Microsystems mulai banyak digunakan di server dan workstation. Pada tahun 1990an dan 2000an, muncul mikroprosesor berbasis RISC yang digunakan di berbagai perangkat, seperti ponsel pintar dan perangkat Internet of Things. Dengan waktu, RISC terus meningkatkan kinerja, efisiensi, dan fleksibilitas, menjadikannya salah satu arsitektur RISC yang paling populer.

Perbandingan RISC dengan CISC

a. Register Set
    - RISC: Dalam arsitektur RISC, biasanya hanya ada satu set register umum. Hal ini menyebabkan penggunaan register menjadi efisien, namun juga membatasi jumlah data yang dapat disimpan secara langsung.
    - CISC: Dalam arsitektur CISC, seringkali terdapat beberapa kumpulan register yang berbeda, seperti register umum dan register khusus untuk fungsi tertentu. Hal ini memungkinkan fleksibilitas yang lebih besar dalam penyimpanan dan pengambilan data.
b. Register Operand
    - RISC: Dalam RISC, operasi aritmetika biasanya melibatkan operasi antara register-register, operand akan diambil dari regrister menggunakan instruksi.
    - CISC: CISC memungkinkan instruksi untuk mengambil operand dari memori langsung, tanpa melibatkan register. Hal ini dapat memudahkan penulisan kode, tetapi juga dapat menambah kompleksitas perangkat keras.

c. Efisiensi pada chip register
    - RISC: RISC sering mendukung mekanisme pengiriman parameter melalui register on-chip, yang memungkinkan transfer parameter yang lebih efisien antara fungsi.
    - CISC: CISC juga dapat mendukung transfer parameter melalui register, tetapi karena instruksi lebih kompleks, pengiriman parameter mungkin lebih bervariasi.
d. Siklus Instruksi
    - RISC: Instruksi RISC cenderung dieksekusi dalam satu siklus clock, yang berarti waktu eksekusi instruksi lebih singkat.
    - CISC: Instruksi CISC seringkali membutuhkan beberapa siklus clock untuk dieksekusi, terutama karena kompleksitas instruksi.
e. Sistem Control
    - RISC: Arsitektur RISC lebih cenderung menggunakan kontrol yang bersifat keras (hardwired control), yang memungkinkan eksekusi instruksi yang lebih cepat.
    - CISC: Arsitektur CISC kadang-kadang menggunakan kontrol yang bersifat mikroprogram (microprogrammed control) untuk mengatur instruksi kompleks. Ini mungkin memerlukan lebih banyak siklus clock.
f. Pipelined
    - RISC: Arsitektur RISC meiliki pipeline yang banyak karena memecah instruksi yang akan dieksekusi menjadi banyak dan tersegmendtasi.
    - CISC: Arsitektur CISC meiliki pipeline yang lebih sedikit karena membutuhkan lebih sedikit intruksi pada pipeline.
g. Instruction
    - RISC: RISC memiliki instruksi yang relatif sederhana dan spesifik. Ini memungkinkan efisiensi eksekusi instruksi.
    - CISC: CISC memiliki banyak instruksi yang kompleks, yang dapat mencakup banyak tugas dalam satu instruksi. Ini memungkinkan penulisan kode yang lebih singkat, tetapi dapat memperlambat eksekusi.
h. Intruction Length
    - RISC: Instruksi RISC umumnya memiliki Intruction Length yang fixed, yang mempermudah pengolahan dan eksekusi instruksi.

- CISC: Instruksi CISC dapat memiliki Intruction Length yang bervariasi, tergantung pada kompleksitasnya.

i. Complexity
    - RISC: RISC menempatkan sebagian kompleksitas pada compiler, yang harus menghasilkan kode yang efisien dengan instruksi yang sederhana.
    - CISC: CISC seringkali memindahkan kompleksitas ke level mikrokode dalam perangkat keras, yang memungkinkan instruksi yang lebih kompleks.
j. Addressing
    - RISC: Pada pengalamatan RISC memiliki pengalamatan yang sederhana sehingga tidak memerlukan memori yang sederhana.
    - CISC: CISC memiliki pengalamatan yang lebih kompleks yang mendukung intruksi-intruksi yang kompleks.

Keunggulan RISC

Arsitektur Reduced Instruction Set Computing (RISC) telah menjadi lebih populer daripada Complex Instruction Set Computing (CISC) seiring berjalannya waktu karena sejumlah alasan yang didukung oleh penelitian dan pengembangan dalam ilmu komputer.

a. Efisiensi Eksekusi Instruksi: RISC memiliki instruksi yang sederhana dan efisien, yang memungkinkan eksekusi instruksi dalam waktu yang lebih singkat. Hal ini menghasilkan kinerja yang lebih cepat dalam berbagai aplikasi komputasi.
b. Pipelining yang Efisien: Arsitektur RISC seringkali mendukung pipelining yang efisien, memungkinkan beberapa instruksi untuk dieksekusi secara bersamaan dalam tahap yang berbeda. Ini menghasilkan peningkatan kinerja yang signifikan.
c. Efisiensi Energi: RISC memiliki keunggulan dalam efisiensi energi karena instruksi sederhana memerlukan lebih sedikit daya untuk dieksekusi. Ini penting dalam aplikasi yang memerlukan daya rendah, seperti perangkat mobile dan perangkat IoT.
d. Sederhana dan Terstruktur: RISC memiliki set instruksi yang sederhana dan terstruktur, yang memudahkan perancang perangkat keras dan kompilator untuk menghasilkan kode yang efisien.
e. Fokus pada Optimasi Kompilator: RISC memindahkan sebagian kompleksitas ke kompilator, yang dapat menghasilkan kode yang lebih efisien. Hal ini memungkinkan fleksibilitas dalam optimasi kompilator untuk meningkatkan kinerja.

f. Skalabilitas: Arsitektur RISC dapat dengan mudah diubah dan ditingkatkan dengan menambahkan lebih banyak register atau instruksi, membuatnya lebih mudah untuk disesuaikan dengan kebutuhan yang berkembang.
g. Kompatibilitas dengan Perkembangan Teknologi: RISC telah mampu beradaptasi dengan perubahan teknologi seperti mikroprosesor terintegrasi (SoC) yang digunakan dalam perangkat pintar. Ini telah menjadikannya pilihan yang kuat dalam berbagai aplikasi.
h. Kinerja yang Konsisten: Kinerja instruksi RISC relatif konsisten, yang membuatnya mudah diprediksi dalam aplikasi real-time dan aplikasi yang memerlukan kinerja yang stabil.
i. Keberlanjutan: RISC telah mengalami perkembangan dan peningkatan yang konsisten selama beberapa dekade, menjadikannya arsitektur yang dapat diandalkan dalam jangka panjang.

BAB 2: ARSITEKTUR RISC

Memori

Sebuah arsitektur komputer dibuat untuk melakukan tugasnya sefisien mungkin. Kecepatan menyelesaikan tugas yang tinggi dan tanpa membebani komputer menjadi tujuan para arsitek komputer membuat desain terbaik mereka. Mengatur memori pada komputer dapat membuat proses yang dilakukanya menjadi cepat dan efisien.

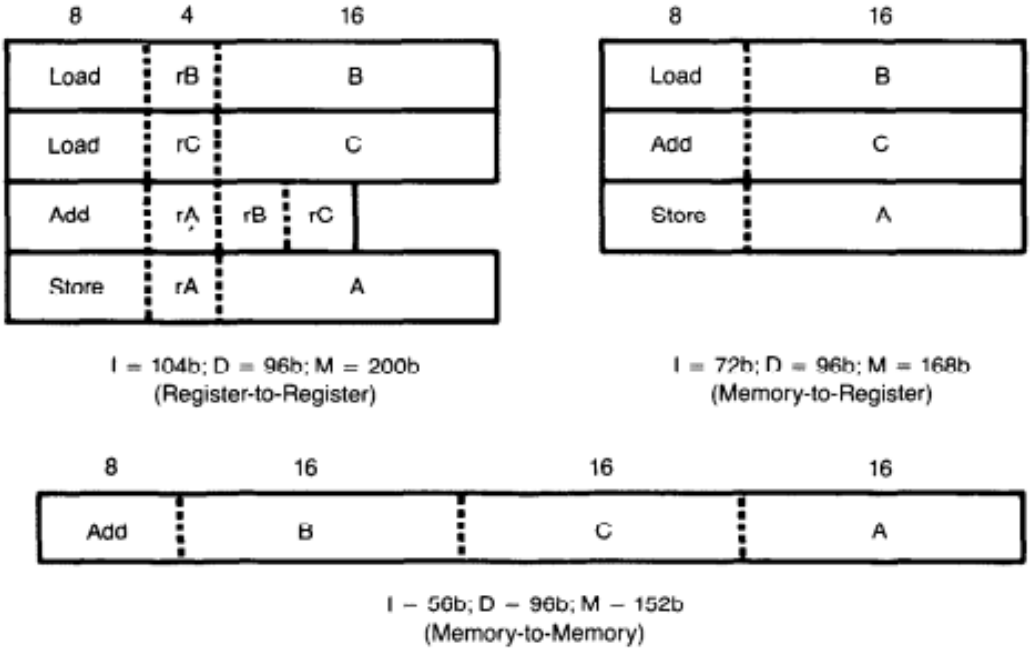
Pada awalnya pengembangan CISC dimaksudkan agar komputer dapat melakukan tugasnya dengan cepat. Daripada menggunakan banyak sekali instruksi dan pipeline yang panjang, CISC memberikan sebuah instruksi yang kompleks dan membuat pipeline menjadi lebih pendek. Hal tersebut membuat komputer yang tadinya mengambil banyak intruksi untuk melakukan tugas menjadi sedikit. Dengan banyaknya instruksi yang kompleks diharapkan komputer dapat mengkompilasi dengan lebih mudah dan cepat.

Dengan cepatnya perkembangan IC(Integrated Circuit) pada 1970-an peneliti arsitek komputer memiliki beberapa prinsip dalam membangun arsitektur komputer :

a. Memori yang digunakan pada microprograming berkembang sangat cepat, yang membuat microprogam hanya memiliki sedikit dampak atau tidak sama sekali
b. Semenjak instruksi mikro menjadi lebih cepat daripada instruksi pada biasanya,

software yang dikembangkan menggunakan microcode agar dapat lebih cepat dan dapat melakukan fungsinya dengan lebih baik.

c. Teknik arsitektur dibuat menjadi bagian-bagian yang lebih kecil sesuai dengan bahwa proses eksekusi sesuai dengan besarnya program yang dibuat
d. Arsitek komputer harus mengembangkan set instruksi yang didasarkan oleh register karena instruksi stack atau memori ke memori menjadi superior untuk dijadikam model eksekusi.



Gambar 2.1 Register dan memori komputer

Setiap data kata membutuhkan 32 bit untuk menyimpan datanya dan 16 bit untuk pengalamatanya. Dari gambar diatas dapat disimpulkan bahwa arsitektur memory-to-memory menjadi arsitektur terbaik karena size data yang dibutuhkan paling sedikit diatara ketiganya yaitu 152 bit, dilajut oleh arsitektur memory-to-register yang memerlukan 168 bit, dan yang terakhir arsitektur register-to-register dengan membutuhkan 200 bit.

Instruksi dan Pipelining

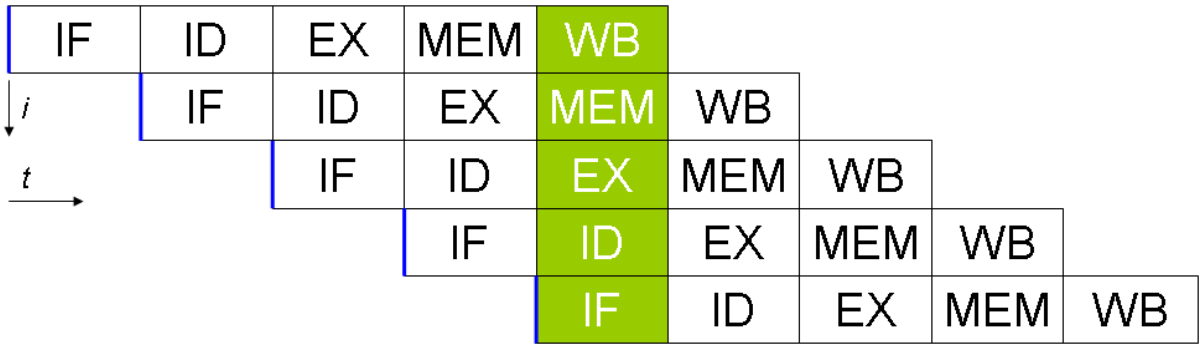
Instruksi-instruksi pada RISC dibuat sesederhana mungkin sehingga proses yang dilakukan setiap tahapan menjadi lebih cepat. Akan tetapi hal tersebut akan membuat pipelining menjadi sangat pancajang karena banyaknya tahapan.

Salah satu ciri dari RISC adalah instruksi operasi yang digunakan. Pada operasi register-to-register RISC hanya menggunakan operasi LOAD dan STORE dalam mengakses data yang diperlukan. Dengan menyederhanakan operasi yang digunakan siklus waktu proses data menjadi lebih singkat. dengan adanya siklus yang singkat proses-proses yang akan dikerjakan komputer akan menjadi lebih fleksibel.

Instruksi operasi yang sederhana pada RISC membuat pengalamatan data yang dibutuhkan menjadi lebih sederhana dan tidak membebani memori lebih banyak. Ketika pengalamatan data menjadi kompleks maka memori yang dibutuhkan akan menjadi lebih besar untuk suatu siklus data.

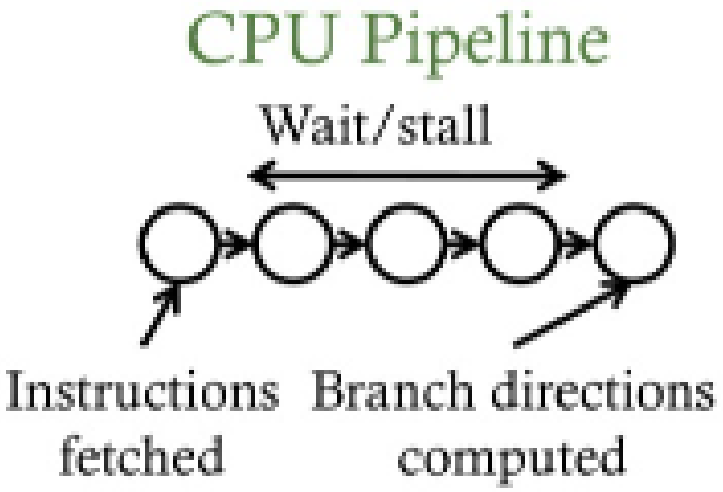
Pipelining merupakan sebuah konsep komputasi yang ada di dalam prosessor komputer untuk meningkatkan kinerja eksekusi instruksi. Teknik ini digunakan dalam desain mikroprosessor dan digunakan untuk memberikan efisisensi waktu dalam mengeksekusi serangkaian instruksi pada sebuah program yang sedang dijalankan.

Pada teknik pipelining instruksi instruksi yang dibutuhkan untuk menjalankan sebuah program dibagi menjadi beberapa tahapan lalu dirakit menjadi satu untuk dieksekusi. Dengan dibaginya instruksi-intruksi tersebut CPU dapat memproses data-data yang diberikan dengan lebih cepat dan mempercepat siklus CPU.



Gambar 2.2 Pipelining

Pada RISC pipelining ini akan menjadi sangat panjang apabila dibandingkan dengan CISC. Hal tersebut dikarenakan instruksi-instruksi pada RISC dibuat sesederhana mungkin yang membuat instruksi yang dirakit menjadi lebi banyak. Walaupun pipelining menjadi lebih panjang, pipelining RISC memiliki siklus proses yang lebih cepat sehingga prosesse yang terjadi pada CPU menjadi lebih fleksiblel



Gambar 2.3 CPU pipelining

Pada pipelining ini terdapat beberapa tahapan utama saat pemrosessan data. Tahapan yang pertama yaitu Fetch, pada tahap ini instruksi yang akan dikerjakan diambil dari memori dan dimuat pada piplining. Tahapan yang kedua yaitu Decode, Instruksi yang sudah diambil dipecah menjadi beberapa mikro operasi agar dapat diproses dengan mudah oleh CPU. Yang ketiga Excecute, instruksi-intruksi tersebut diproses dan dioperasikan dan menghasilkan sebuah output data. Dan yang terakhir write back, pada tahapan ini hasil data dari proses pada CPU ditulis ulang di register.

Dengan sederhananya instruksi set yang ada membuat siklus proses menjadi lebih singkat dan cepat. Hal ini memungkinkan pipelining menjadi lebih fleksibel sesuai dengan tugas yang diberikan ke CPU. Dengan siklus proses yang cepat serta pipelining yang fleksibel membuat waktu yang dibutuhkan untuk menyelesaikan instruksi-instruksi dari program menjadi lebih cepat dan efisien.

BAB 3 : IMPLEMENTASI RISC DI PROSESOR

Contoh Prosesor RISC

Prosesor RISC adalah jenis arsitektur prosesor yang didesain dengan seperangkat instruksi yang relatif sederhana dan memiliki eksekusi yang cepat. Ini bertujuan untuk meningkatkan kinerja prosesor dengan mengurangi kompleksitas set instruksi dan memungkinkan instruksi-instruksi tersebut dieksekusi dengan cepat. Di bawah ini adalah beberapa contoh prosesor RISC terkenal :

1. **ARM**

   Prosesor ARM adalah salah satu contoh paling terkenal dari arsitektur RISC. Prosesor ini digunakan dalam berbagai perangkat, mulai dari ponsel cerdas hingga perangkat IoT (*Internet of Things*) dan komputer pribadi. ARM memiliki seperangkat instruksi yang relatif sederhana dan efisien.

2. **MIPS**

   Prosesor MIPS (*Microprocessor without Interlocked Pipeline Stages*) adalah salah satu proyektor RISC awal yang populer. Ini digunakan dalam komputer bekerja, workstation, dan bahkan konsol permainan seperti PlayStation awal.

3. **SPARC**

   SPARC (*Scalable Processor Architecture*) adalah arsitektur prosesor yang dikembangkan oleh Sun Microsystems (sekarang bagian dari Oracle). Ini digunakan dalam server dan workstation Sun.

4. **PowerPC**

   PowerPC adalah arsitektur RISC yang dikembangkan bersama oleh IBM, Apple, dan Motorola (sekarang *Freescale Semiconductor*). Ini digunakan dalam produk Apple Macintosh selama beberapa tahun dan juga digunakan dalam konsol permainan seperti Xbox 360.

5. **RISC-V**

   RISC-V adalah arsitektur RISC terbuka yang menjadi semakin populer. Ini memiliki seperangkat instruksi yang dapat disesuaikan dan dapat digunakan dalam berbagai aplikasi, dari mikrokontroler hingga superkomputer.

Analisis Kinerja

Desain RISC meningkatkan kinerja dalam berbagai aplikasi dengan pendekatan yang berfokus pada keefisienan instruksi dan eksekusi. Prosesor RISC menggunakan seperangkat instruksi yang sederhana dan spesifik, yang memungkinkan setiap instruksi dieksekusi dengan cepat. Selain itu, arsitektur RISC sering mendukung eksekusi paralel, yang memungkinkan beberapa instruksi dieksekusi secara bersamaan, meningkatkan *throughput* prosesor. Penyediaan banyak *register* internal dalam prosesor RISC mengurangi ketergantungan pada memori utama, mempercepat akses data yang dibutuhkan oleh instruksi-instruksi.

Desain ini juga mengoptimalkan kinerja untuk kompiler, yang dapat menghasilkan kode yang lebih efisien. Terakhir, latensi instruksi yang tetap dalam RISC memungkinkan prediksi jalur instruksi yang lebih baik, mengurangi *overhead* dalam eksekusi instruksi. Semua ini membuat prosesor RISC menjadi pilihan yang efisien dan kuat untuk berbagai aplikasi, mulai dari komputer pribadi hingga perangkat bergerak dan sistem *embedded*.

BAB 4 : PEMROGRAMAN PADA ARSITEKTUR RISC

Panduan Pemrograman

Pemrograman pada arsitektur RISC (*Reduced Instruction Set Computing*) memiliki beberapa teknik dan prinsip yang berbeda dibandingkan dengan arsitektur lain seperti CISC (*Complex Instruction Set Computing*). Berikut adalah beberapa teknik dan prinsip pemrograman yang efektif pada arsitektur RISC:

1. **Pemanfaatan Register**

   RISC memiliki banyak *register* umum yang dapat digunakan oleh pemrogram. Oleh karena itu, pemrogram harus memaksimalkan penggunaan *register* ini untuk menyimpan data sementara dan hasil perhitungan. Hindari penggunaan memori lebih sering, karena mengakses memori bisa lebih lambat daripada mengakses *register*.
2. **Instruksi Sederhana**

   RISC didesain dengan instruksi yang sangat sederhana dan berukuran tetap. Oleh karena itu, pemrogram perlu memecah tugas kompleks menjadi serangkaian instruksi sederhana. Ini memerlukan pemrogram untuk lebih berpikir tentang manajemen memori dan perhitungan secara eksplisit.
3. **Load-Store Architecture**

   Arsitektur RISC umumnya mengikuti prinsip "*load-store*," yang berarti hanya instruksi khusus yang dapat mengakses memori, sedangkan instruksi lain harus bekerja pada data yang ada di *register*. Ini memungkinkan akses ke memori menjadi lebih efisien dan membatasi jumlah instruksi yang bekerja pada memori.
4. **Pipelining**

   RISC sering menggunakan *pipelining*, yang memungkinkan beberapa instruksi untuk dieksekusi secara bersamaan dalam tahap-tahap yang berbeda dari *pipelining*. Pemrogram harus memahami bagaimana *pipelining* bekerja dan menghindari konflik data atau instruksi dalam *pipelining*.
5. **Optimisasi Kinerja**

   Karena instruksi pada RISC cenderung sederhana, pemrogram harus fokus pada pengoptimalan kinerja dengan memanfaatkan *pipelining*, prediksi percabangan yang akurat, dan teknik lainnya. Mengoptimalkan perangkat lunak agar sesuai dengan arsitektur hardware sangat penting.

6. **Pemahaman Terhadap Cache**

   Memahami cara kerja *cache* pada arsitektur RISC adalah penting. Pemrogram harus berusaha untuk meminimalkan *cache misses* dan memaksimalkan penggunaan *cache*, karena akses ke memori utama bisa menjadi *bottleneck*.

7. **Prediksi Percabangan**

   Instruksi percabangan (*branch instructions*) dapat mempengaruhi kinerja secara signifikan. Pemrogram harus berusaha untuk mengurangi percabangan yang sering dan memanfaatkan teknik prediksi percabangan yang efisien.

8. **Komunikasi dengan Perangkat Keras**

   RISC seringkali memerlukan komunikasi yang lebih dekat dengan perangkat keras daripada CISC. Pemrogram harus memahami cara kerja *register*, *cache*, dan pengelolaan memori agar dapat mengoptimalkan kode dengan baik.

9. **Pemrograman Paralel**

   Banyak arsitektur RISC mendukung pemrograman paralel, seperti SIMD (*Single Instruction, Multiple Data*) atau vektorisasi. Pemrogram harus memanfaatkan fitur-fitur ini untuk meningkatkan kinerja aplikasi.

10. **Pemecahan Masalah Berbasis Instruksi**

    Pemrogram harus mampu memecahkan masalah dalam batasan instruksi sederhana yang tersedia dalam arsitektur RISC. Hal ini memerlukan pemahaman yang mendalam tentang instruksi yang ada dan kreativitas dalam merancang solusi yang efisien.

Pemrograman pada arsitektur RISC dapat menjadi lebih efektif jika pemrogram memahami prinsip-prinsip dasar ini dan mengoptimalkan kode mereka dengan mempertimbangkan karakteristik arsitektur RISC yang sederhana namun efisien.

BAB 5: STUDI KASUS

Penerapan RISC dalam Perangkat Modern

1. **Advanced RISC Machine (ARM)**

   ARM singkatan dari Advanced RISC Machine adalah arsitektur processor 32-bit yang dikembangkan oleh ARM Limited, dan banyak digunakan oleh perangkat genggam. Berkat fitur hemat energinya, CPU ARM sangat dominan di pasar perangkat elektronik genggam. Saat ini, processor ARM menguasai sekitar 75 persen pasar elektronik genggam.

2. **Scalable Processor Architecture (SPARC)**

Gambar 2.3 CPU pipelining

SPARC (Scalable Processor Architecture) adalah microprocessor berarsitektur RISC (Reduced Instruction Set Computing) yang didesain oleh Sun Microsystems sejak tahun 1985. Pengembangan SPARC didahului dengan tekad Sun yang memusatkan bisnisnya dibidang workstation dan tidak ingin terlalu bergantung pada pemasok prosesor Motorola yang saat itu mensuplai seri prosesor 680x0 kepada Sun, disamping juga terhadap dominasi Intel dengan arsitektur i386-nya.

Sejatinya, keistimewaan dari arsitektur SPARC adalah spesifikasi dari beberapa prosesornya yang menganut kaidah Open Source. Sun mengawalinya pada tahun 2005 yang membuka spesifikasi UltraSPARC T1 dibawah lisensi bebas GPL. Spesifikasi yang dibebaskan, juga untuk T2 dapat ditemukan di situs opensparc.net. Prosesor 32-bit dengan arsitektur V8 dari generasi awal ternyata masih aktual dan produknya masih diperoleh sampai sekarang dari beberapa pembuat prosesor, termasuk yang telah diintegrasikan dalam sistem embedded.

KESIMPULAN

RISC singkatan dari Reduced Instruction Set Computing. Merupakan bagian dari arsitektur mikroprosessor, berbentuk kecil dan berfungsi untuk negeset istruksi dalam komunikasi diantara arsitektur yang lainnya. Reduced Instruction Set Computing (RISC) atau "Komputasi set instruksi yang disederhanakan"

Implementasi pertama arsitektur power PC yaitu 601 memiliki rancangan yang sangat mirip dengan rancangan RS 6000, model PowerPC berikutnya mempunyai konsep superscalar. Kelebihan arsitektur RISC yang berkaitan dengan kinerja dapat ditunjukan dengan sejumlah "Sircumstansial Evidence". Optimasi kompiler yang lebih efektif dan dapat dikembangkan. Sebagian besar instruksi yang dihasilkan oleh kompiler relatif sederhana.

Berkaitan dengan penggunaan pipelining instruksi yang diterapkan secara lebih efektif terhadap RISC. Program-program RISC harus lebih responsife terhadap interrupt.

Berkaitan dengan implementasi VLSI (Very Large Scale Integration). Apabila digunakan rancangan dan implementasi CPU akan berubah, artinya dimungkinkan untuk menaruh CPU keseluruhan pada keping tunggal. Waktu yang dibutuhkan untuk implementasi dan perancangan karena prosessor VLSI cukup sulit dibuat sehingga para perancang harus membuat rancangan rangkaian, tata letak dan pemodelan pada tingkat perangkat, dengan menggunakan pemodelan RISC proses tersebut akan lebih mudah selain apabila kinerja keping RISC ekuivalen dengan mikroprosessor CISC (Pentium) yang setara maka keuntungan dengan memakai pendekatan RISC akan terasa sekali.

DAFTAR PUSTAKA

Hennessy, J. L., & Patterson, D. A. (1990). "Computer Architecture: A Quantitative Approach." Journal of Systems and Software, 43(3), 199-216.

Patterson, D. A., & Ditzel, D. R. (1980). "The Case for the Reduced Instruction Set Computer." ACM SIGARCH Computer Architecture News, 8(6), 25-33.

Hennessy, J. L., & Patterson, D. A. (2002). "Computer Organization and Design: The Hardware/Software Interface." ACM Transactions on Computational Logic, 33(1), 104-116.

Asanović, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., ... & Williams, S. W. (2006). "The Landscape of Parallel Computing Research: A View from Berkeley." Communications of the ACM, 52(2), 56-67.

Brown, A. (2015). Teknik Optimasi Kinerja pada Arsitektur RISC. Jurnal Teknologi Informatika, 8(2), 45-60.