Ir. Heru Nurwarsito, M.Kom
Barlian Henryranu P, ST, MT
Eko Saksi Pramukantoro, S.Kom, M.Kom

14

SELF-PROPAGATING ENTREPRENEURIAL EDUCATION DEVELOPMENT

1. PENDAHULUAN

1.1 Pengantar

Processor superscalar merupakan sebuah tipe processor yang dapat mengeksekusi beberapa set instruksi sekaligus dalam waktu yang sama. Tipe processor ini dapat terdiri dari beberapa sub-unit yang bertugas untuk mengontrol tipe fungsi-fungsi dasar tertentu. Sementara processor lain juga memiliki unit ini, processor superscalar dapat mengirimkan informasi secara langsung pada unit-unit tersebut untuk diproses sementara processor utama melakukan pekerjaan lainnya. Processor superscalar merupakan titik tengah dari tiga tipe processor utama.

Processor komputer dibagi menjadi tiga kategori utama, yaitu scalar, superscalar dan vector.Processor scalar merupakan tipe processor yang paling umum dikenali dan digunakan oleh mayoritas pengguna komputer. Processor ini menerima satu perintah dalam satu waktu dan mengeksekusinya secara berurutan atau sesuai dengan prioritasnya.Seperti yang telah disinggung sebelumnya, processor scalar merupakan tipe processor yang banyak digunakan pada komputer rumah dan juga komputer bisnis.

1.2 Tujuan

Tujuan dibuatnya modul ini adalah :
Adapun yang menjadi tujuan pokok penulisan modul ini adalah:
Untuk mengetahui pemahaman umum tentang prosesor superscalar
Untuk menganalisa jenis-jenis dan karakteristik dari prosesor superscalar

1.3 Latar Belakang

Kebanyakan dari komputer saat ini menggunakan mekanisme superscalar ini.Standar pipeline yang digunakan adalah untuk pengolahan bilangan matematika integer (bilangan bulat, bilangan yang tidak memiliki pecahan), kebanyakan CPU juga memiliki kemampuan untuk pengolahan untuk data floating point (bilangan berkoma).Pipeline yang mengolah integer dapat juga digunakan untuk mengolah data bertipe floating point ini, namun untuk aplikasi tertentu, terutama untuk aplikasi keperluan ilmiah CPU yang memiliki kemampuan pengolahan floating point dapat meningkatkan kecepatan prosesnya secara dramatis.

Superscalar ini mampu menjlankan Instruction Level Parallelism dengan satu prosesor.Superscalar dapat diaplikasikan di RISC dan CISC, tapi pada umumnya RISC.

Peristiwa menarik yang bisa dilakukan dengan metoda superscalar ini adalah dalam hal memperkirakan pencabangan instruksi (brach prediction) serta perkiraan eksekusi perintah (speculative execution).Peristiwa ini sangat menguntungkan buat program yang membutuhkan pencabangan dari kelompok intruksi yang dijalankankannya.

Program yang terdiri dari kelompok perintah bercabang ini sering digunakan dalam pemrograman. Contohnya dalam menentukan aktivitas yang dilakukan oleh suatu sistem berdasarkan umur seseorang yang sedang diolahnya, katakanlah jika umur yang

bersangkutan lebih dari 18 tahun, maka akan diberlakukan instruksi yang berhubungan dengan umur tersebut, anggaplah seseorang tersebut dianggap telah dewasa, sedangkan untuk kondisi lainnya dianggap belum dewasa. Tentu perlakuannya akan dibedakan sesuai dengan sistem yang sedang dijalankan.

2. Superskalar

Sedangkan processor vector merupakan processor yang dapat menerima beberapa perintah dalam satu waktu melalui sistem array.Serangkaian perintah masuk secara simultan pada inti utama processor (main core).Perintah ini dianggap sebagai bagian tunggal dari perintah besar oleh processor dan dieksekusi secara simultan (bersamaan).
Di antara processor scalar dan vector, terdapatlah processor superscalar. Pada processor superscalar, processor utama hanya dapat menerima satu perintah dalam satu waktu, serupa dengan processor scalar. Namun di sisi lain, terdapat koneksi langsung pada sistem sekunder processor, teknologi yang tidak muncul pada bentuk processor lainnya.

Mangapa menggunakan Prosesor Superscalar

Superscalar (superskalar) adalah arsitektur prosessor yang memungkinkan eksekusi yang bersamaan (parallel) dari instruksi yang banyak pada tahap pipeline yang sama sebaik tahap pipeline yang lain.

Merupakan salah satu rancangan untuk meningkatkan kecepatan CPU.Kebanyakan dari komputer saat ini menggunakan mekanisme superscalar ini.Standar pipeline yang digunakan adalah untuk pengolahan bilangan matematika integer (bilangan bulat, bilangan yang tidak memiliki pecahan), kebanyakan CPU juga memiliki kemampuan untuk pengolahan untuk data floating point (bilangan berkoma).Pipeline yang mengolah integer dapat juga digunakan untuk mengolah data bertipe floating point ini, namun untuk aplikasi tertentu, terutama untuk aplikasi keperluan ilmiah CPU yang memiliki kemampuan pengolahan floating point dapat meningkatkan kecepatan prosesnya secara dramatis.

Superscalar ini mampu menjlankan Instruction Level Parallelism dengan satu prosesor.Superscalar dapat diaplikasikan di RISC dan CISC, tapi pada umumnya RISC.

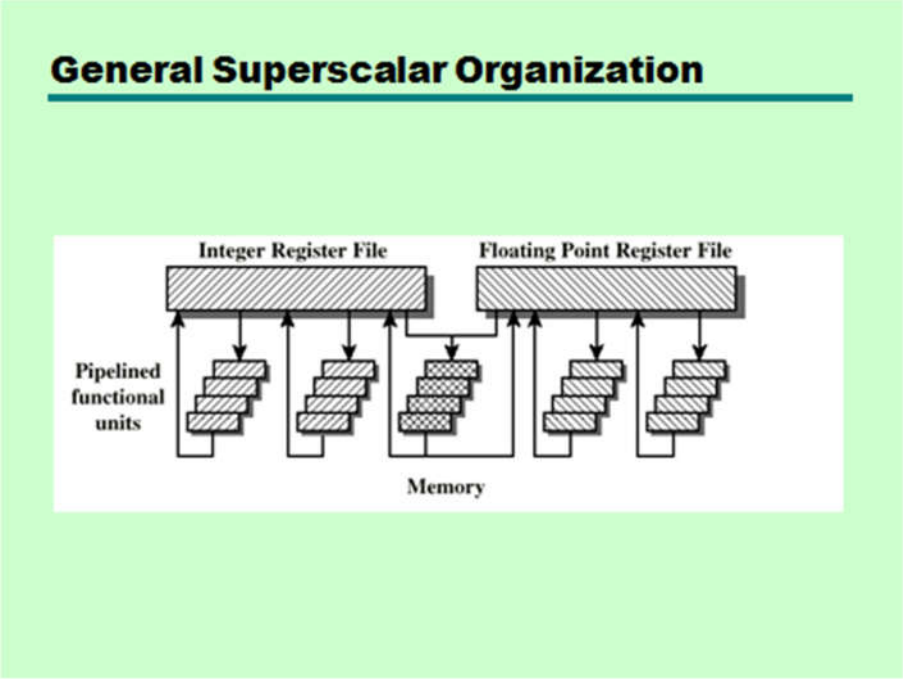
Peristiwa menarik yang bisa dilakukan dengan metoda superscalar ini adalah dalam hal memperkirakan pencabangan instruksi (brach prediction) serta perkiraan eksekusi perintah (speculative execution).Peristiwa ini sangat menguntungkan buat program yang membutuhkan pencabangan dari kelompok intruksi yang dijalankankannya.

Program yang terdiri dari kelompok perintah bercabang ini sering digunakan dalam pemrograman. Contohnya dalam menentukan aktivitas yang dilakukan oleh suatu sistem berdasarkan umur seseorang yang sedang diolahnya, katakanlah jika umur yang bersangkutan lebih dari 18 tahun, maka akan diberlakukan instruksi yang berhubungan dengan umur tersebut, anggaplah seseorang tersebut dianggap telah dewasa, sedangkan untuk kondisi lainnya dianggap belum dewasa. Tentu perlakuannya akan dibedakan sesuai dengan sistem yang sedang dijalankan.

Organisasi Superscalar secara umum

Alasan desain Superscalar sebagian besar operasi menggunakan besaran/nilai skalar Operasi ini memungkinkan peningkatan kinerja sistem hingga level tertentu. Superscalar Implementation

Proses fetch dari beberapa instruksi secara bersamaan. Logika untuk menentukan ketergantungan sebenarnya yang meliputi nilai register Mekanisme untuk mengkomunikasikan nilai tersebut. Mekanisme untuk menginisialisasi instruksi paralel.Tersedianya sumber untuk eksekusi paralel dari beberapa instruksi.Mekanisme processing instruksi dengan urutan yg sesuai.



Organisasi prosesor superscalar secara umum

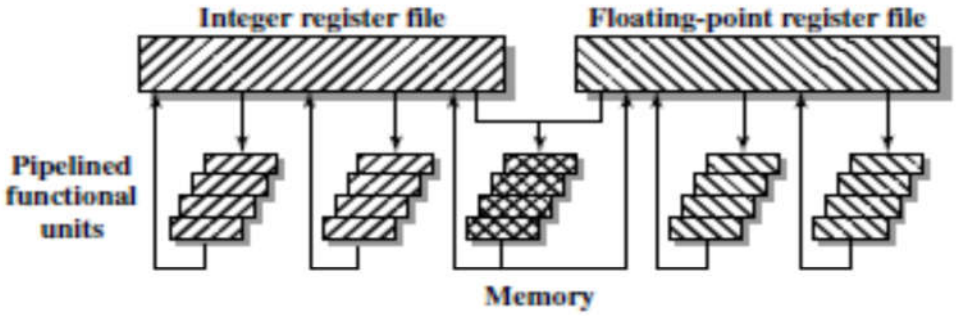
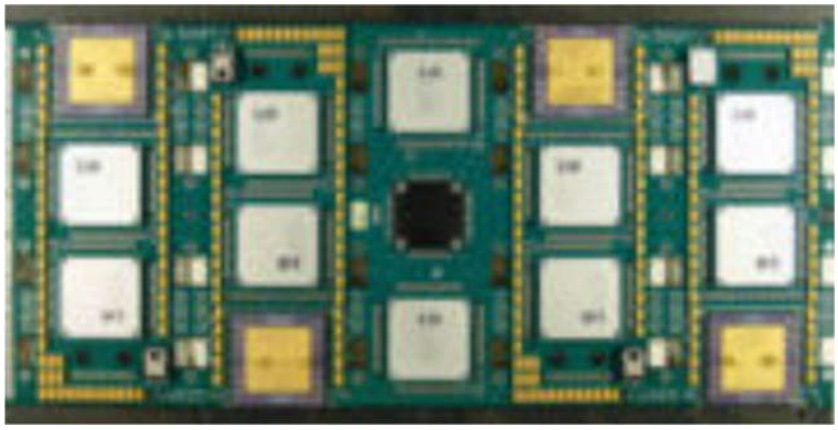


Figure 14.1 General Superscalar Organization

Table 14.1 Reported Speedups of Superscalar-Like Machines

Reference	Speedup
[TJAD70]	1.8
[KUCK77]	8
[WEIS84]	1.58
[ACOS86]	2.7
[SOHI90]	1.8
[SMIT89]	2.3
[JOUP89b]	2.2
[LEE91]	7

Processor superscalar



Processor board of a CRAY T3e parallel computer with four *superscalar* Alpha processors

Superpipeline

Teknologi pipeline yang digunakan pada komputer bertujuan untuk meningkatkan kinerja dari komputer. Secara sederhana, pipeline adalah suatu cara yang digunakan untuk melakukan sejumlah kerja secara bersamaan tetapi dalam tahap yang berbeda yang dialirkan secara kontiniu pada unit pemrosesan. Dengan cara ini, maka unit pemroses selalu bekerja.

Teknik pipeline ini dapat diterapkan pada berbagai tingkatan dalam sistem komputer. Bisa pada level yang tinggi, misalnya program aplikasi, sampai pada tingkat yang rendah, seperti pada instruksi yang dijalankan oleh microprocessor.

Teknik pipeline yang diterapkan pada microprocessor, dapat dikatakan sebuah arsitektur khusus.Ada perbedaan khusus antara model microprocessor yang tidak menggunakan arsitektur pipeline dengan microprocessor yang menerapkan teknik ini.

Pada microprocessor yang tidak menggunakan pipeline, satu instruksi dilakukan sampai selesai, baru instruksi berikutnya dapat dilaksanakan.Sedangkan dalam microprocessor yang menggunakan teknik pipeline, ketika satu instruksi sedangkan diproses, maka instruksi yang berikutnya juga dapat diproses dalam waktu yang bersamaan. Tetapi, instruksi yang diproses secara bersamaan ini, ada dalam tahap proses yang berbeda. Jadi, ada sejumlah tahapan yang akan dilewati oleh sebuah instruksi.

Misalnya sebuah microprocessor menyelesaikan sebuah instruksi dalam 4 langkah.Ketika instruksi pertama masuk ke langkah 2, maka instruksi berikutnya diambil untuk diproses pada langkah 1 instruksi tersebut.Begitu seterusnya, ketika instruksi pertama masuk ke langkah 3, instruksi kedua masuk ke langkah 2 dan instruksi ketiga masuk ke langkah 1.

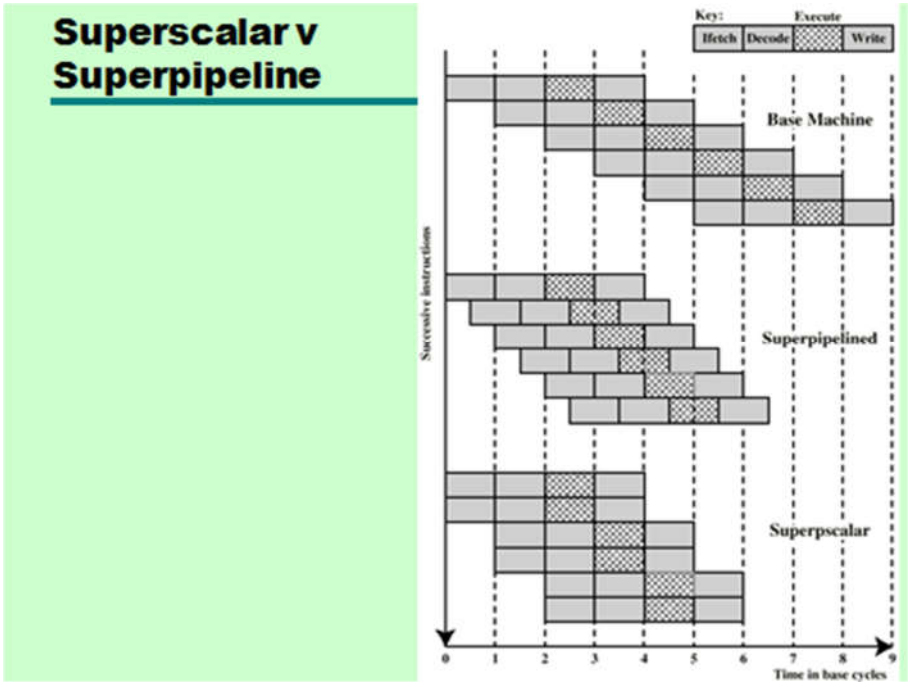
Dengan penerapan pipeline ini pada microprocessor akan didapatkan peningkatan dalam unjuk kerja microprocessor. Hal ini terjadi karena beberapa instruksi dapat dilakukan

secara parallel dalam waktu yang bersamaan. Secara kasarnya diharapkan akan didapatkan peningkatan sebesar K kali dibandingkan dengan microprocessor yang tidak menggunakan pipeline, apabila tahapan yang ada dalam satu kali pemrosesan instruksi adalah K tahap.

Teknik pipeline ini menyebabkan ada sejumlah hal yang harus diperhatikan sehingga ketika diterapkan dapat berjalan dengan baik. Tiga kesulitan yang sering dihadapi ketika menggunakan teknik pipeline ini adalah : Terjadinya penggunaan resource yang bersamaan, Ketergantungan terhadap data, Pengaturan Jump ke suatu lokasi memori.

Karena beberapa instruksi diproses secara bersamaan ada kemungkinan instruksi tersebut sama-sama memerlukan resource yang sama, sehingga diperlukan adanya pengaturan yang tepat agar proses tetap berjalan dengan benar. Sedangkan ketergantungan terhadap data, bisa muncul, misalnya instruksi yang berurutan memerlukan data dari instruksi yang sebelumnya. Kasus Jump, juga perlu perhatian, karena ketika sebuah instruksi meminta untuk melompat ke suatu lokasi memori tertentu, akan terjadi perubahan program counter, sedangkan instruksi yang sedang berada dalam salah satu tahap proses yang berikutnya mungkin tidak mengharapkan terjadinya perubahan program counter.

Dengan menerapkan teknik pipeline ini, akan ditemukan sejumlah perhatian yang khusus terhadap beberapa hal di atas, tetapi tetap akan menghasilkan peningkatan yang berarti dalam kinerja microprocessor. Ada kasus tertentu yang memang sangat tepat bila memanfaatkan pipeline ini, dan juga ada kasus lain yang mungkin tidak tepat bila menggunakan teknologi pipeline.



Superscalar VS Superpipelined

Superscalar adalah salah satu jenis dari arsitektur, dimana superscalar adalah sebuah uniprocessor yang dapat mengeksekusi dua atau lebih operasi scalar dalm bentuk parallel, Merupakan salah satu rancangan untuk meningkatkan kecepatan CPU.

Sedangkan, teknologi pipeline yang digunakan pada komputer bertujuan untuk meningkatkan kinerja dari komputer. Secara sederhana, pipeline adalah suatu cara yang digunakan untuk melakukan sejumlah kerja secara bersamaan tetapi dalam tahap yang berbeda yang dialirkan secara kontiniu pada unit pemrosesan. Dengan cara ini, maka unit pemroses selalu bekerja.

Pipelineini memiliki empat tahap: instruksi fetch, decode operasi, pelaksanaan operasi,dan hasil menulis back. Tahap eksekusi crosshatched untuk kejelasan. Perhatikan bahwa meskipun beberapa instruksi mengeksekusi secara bersamaan, hanya satu instruksi dalam yang eksekusi tahap pada satu waktu. Bagian selanjutnya dari diagram menunjukkan implementasi superpipelined yang mampu melakukan tahapan pipeline dua persiklus clock. Sebuah cara alternatif melihat ini adalah bahwa fungsi yang dilakukan dalam setiap tahap dapat dibagi menjadi dua nonoverlapping dan masing-masing dapat mengeksekusi dalam setengah jam implementasi cycle. Superperpipeline yang berperilaku dengan cara inidikatakanderajat2. Akhirnya, terendah bagian dari diagram menunjukkan implementasi superscalar mampu melaksanakan dua contoh masing-masing tahap secara paralel. Tinggi derajat superpipelinedan implementasi superscalar tentu saja mungkin. Baik superpipeline dan implementasi superscalar yang digambarkan dalam Gambar14.2 memiliki jumlah yang sama mengeksekusi instruksi pada waktu yang sama diprosesor superpipelined stabilstate. Prosesor superpipeline tertinggal jauh di belakang prosesor superscalar diawal program dan pada setiap target cabang.

Limitations

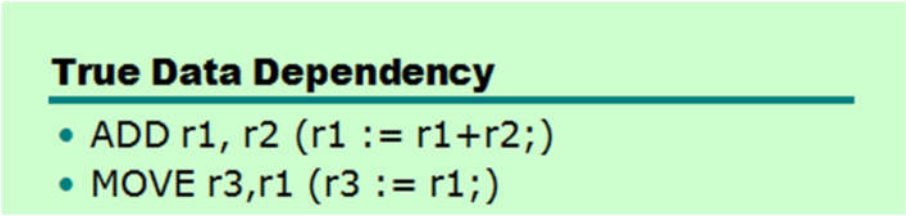
Pendekatan superscalar tergantung pada kemampuan untuk mengeksekusi beberapa instruksi secara paralel. Paralelisme instruksi-level merujuk pada sejauh mana, rata-rata, instruksi dari sebuah program dapat dieksekusi dalam kombinasi parallel. Kompiler berbasis optimasi dan teknik perangkat keras dapat digunakan untuk memaksimalkan instruksi-level parallelisme. Sebelum meneliti teknik desain yang digunakan dalam mesin superscalar untuk meningkatkan instruksi-level parallelisme, kita perlu melihat keterbatasan mendasar untuk paralelisme dengan yang harus diatasi oleh sistem. Daftar lima keterbatasan:

- True data dependency
- Procedural dependency
- Resource conflicts
- Output dependency
- Antidependency

Kami membahas tiga pertama dari keterbatasan dalam sisa bagian ini. Sebuah diskusi dua terakhir harus menunggu beberapa perkembangan pada bagian berikutnya.

TRUE DATA DEPENDENCY

Perhatikan urutan berikut:



Instruksi kedua dapat diambil dan dikodekan tetapi tidak dapat mengeksekusi sampai mengeksekusi instruksi pertama. Alasannya adalah bahwa instruksi kedua membutuhan data yang dihasilkan oleh instruksi pertama. Situasi ini disebut sebagai ketergantungan data yang benar (juga disebut ketergantungan aliran atau menulis setelah membaca). Gambar14.3 mengilustrasikan ketergantungan ini dalam mesin superscalar derajat2. Dengan tidak ada ketergantungan, dua instruksi dapat diambil dan dieksekusi secara paralel. Jika ada Data ketergantungan antara instruksi pertama dan kedua, maka instruksi kedua yang tertunda saat siklus clock sebanyak yang diperlukan untuk menghapus ketergantungan. Secara umum, setiap instruksi harus ditunda sampai semua nilai input telah diproduksi. Dalam sebuah pipeline yang sederhana, seperti diilustrasikan pada bagian atas Gambar14.2, yang urutan instruksi tersebut akan menyebabkan penundaan. Namun, pertimbangan berikut, di mana salah satu beban adalah dari memori bukan dari register:

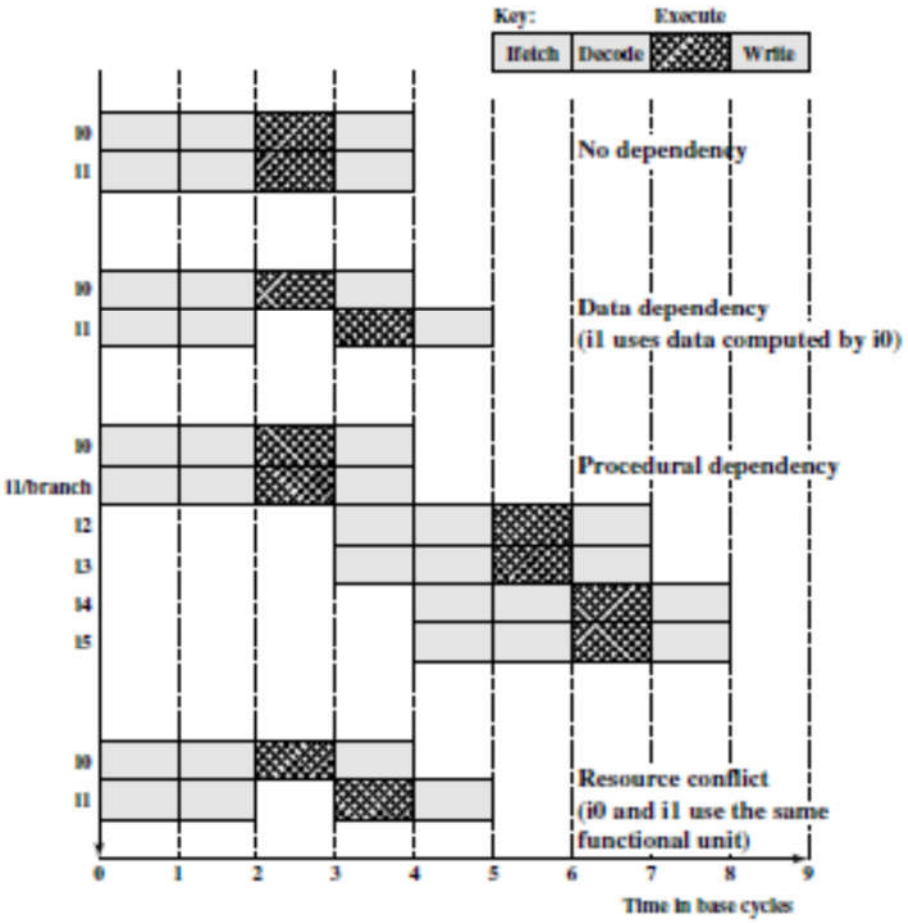


Figure 14.3 Effect of Dependencies

MOV EAX, eff ;load register EAX with the contents of effective memory address eff
MOV EBX, EAX ;load EBX with the contents of EAX

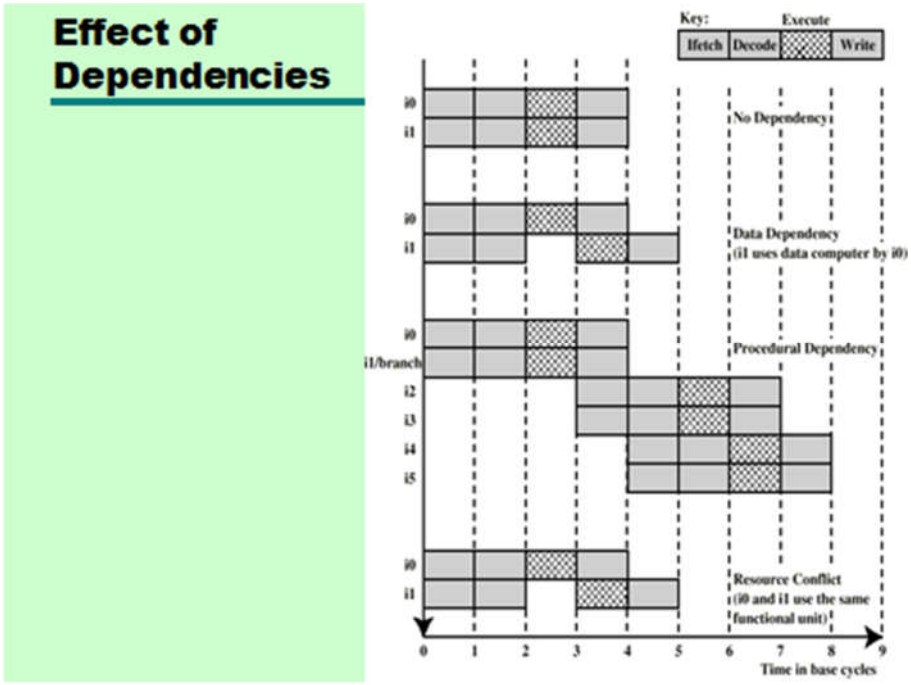
Sebuah prosesor RISC yang khas mengambil dua atau lebih siklus untuk melakukan beban dari memori. Ini dapat mengambil puluhan atau bahkan ratusan siklus untuk cache miss pada semua tingkat cache ketika beban akses cache, karena keterlambatan akses memori off-chip. Salah satu cara untuk mengkompensasi delay ini dengan compiler untuk menyusun ulang instruksi sehingga bahwa satu atau lebih instruksi berikutnya yang tidak bergantung pada beban memori dapat mulai mengalir melalui skema pipeline. ini kurang efektif dalam kasus superscalar pipeline: Instruksi independen dieksekusi selama beban memiliki kecenderungan akan dieksekusi pada siklus pertama dari beban, hal ini meyebabkan prosesor berada pada keadaan idle sampai beba tersebut selesai dilakukan.

PROCEDURAL DEPENDENCIES

PROCEDURAL DEPENDENCIES seperti yang telah dibahas dalam Bab 12, kehadiran cabang di suatu urutan instruksi instruksi mempersulit operasi pipeline. Instruksi-instruksi yang mengikuti suatu percabangan (diambil atau tidak diambil) memiliki ketergantungan prosedural pada cabang dan tidak dapat dilaksanakan sampai cabang tersebut dieksekusi. Gambar14.3 mengilustrasikan efek dari cabang pada pipeline superscalar derajat 2. Sebagaimana telah kita lihat, jenis ketergantungan prosedural juga mempengaruhi skalarpipeline. Konsekuensi untuk pipeline superscalar lebih buruk, karena l kesempatan yang ebih besar hilang pada delay masing-masing. Jika variabel-panjang instruksi yang digunakan, maka muncul ketergantungan prosedural yang lain. Karena panjang setiap instruksi tertentu tidak diketahui, maka harus setidaknya sebagian diterjemahkan sebelum instruksi berikut dapat diambil. Hal ini untuk mencegah mengambil simultan diperlukan dalam pipeline superscalar. Hal ini salah satu alasan bahwa teknik superscalar lebih mudah diterapkan pada RISC atau seperti arsitektur yang menyerupai RISC, dengan panjang instruksi tetap.

Resource Conflict

Sebuah konflik sumber daya adalah kompetisi dari dua atau lebih instruksi untuk sumber daya yang sama pada waktu yang sama. Contoh sumber daya termasuk memori, cache, bus, register file port, dan unit fungsional (misalnya, ALU adder). Dalam kaitannya dengan pipeline, konflik sumber daya memperlihatkan perilaku yang mirip dengan data ketergantungan (Gambar 14.3). Ada beberapa perbedaan, namun. Untuk satu hal, konflik sumber daya bisa diatasi dengan duplikasi sumber daya, sedangkan ketergantungan data yang sebenarnya tidak bisa dihilangkan. Juga, ketika operasi membutuhkan waktu lama untuk menyelesaikan, konflik sumber dapat diminimalkan dengan pipelining unit fungsional yang sesuai.



DESIGN ISSUES

Membuat perbedaan penting antara dua konsep yang berhubungan dengan instruksi-level parallelism dan paralelisme mesin. Instruksi-level parallelism ada saat instruksi secara berurutan yang independen dan dengan demikian dapat dilakukan dalam paralel dengan cara saling tumpang tindih. Sebagai contoh dari konsep-instruksi level parallelism, pertimbangkan hal berikut dua kode fragmen:

Load R1 ←R2 Add R3 ←R3, "1"

Add R3 ←R3, "1" Add R4 ←R3, R2

Add R4 ←R4, R2 Store [R4] ←R0

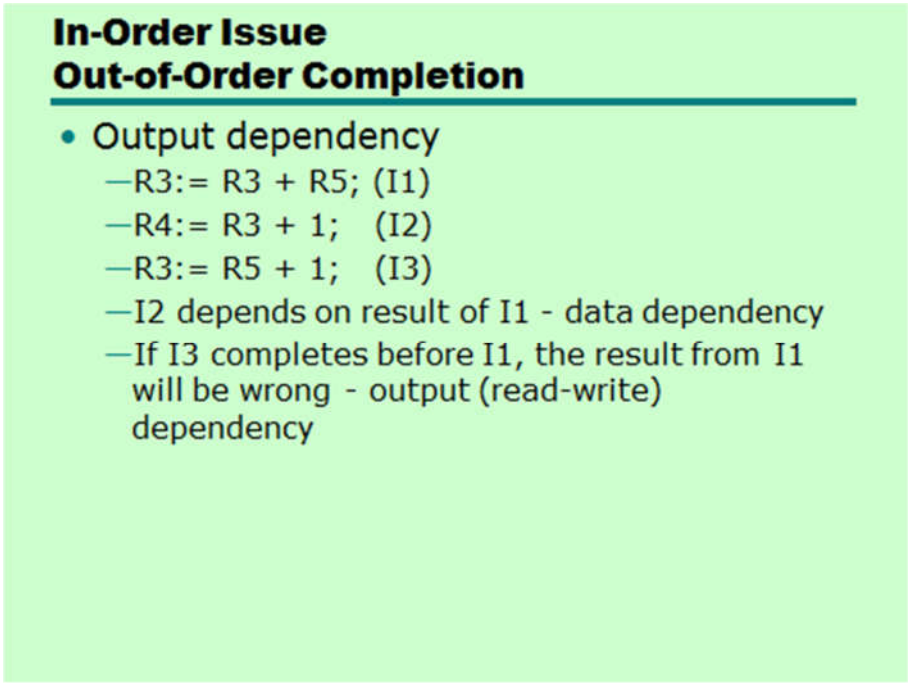
Tiga instruksi di sebelah kiri adalah independen, dan dalam teori ketiga dapatdieksekusi secara paralel. Sebaliknya, tiga instruksi di sebelah kanan tidak bisa dijalankansecara paralel karena instruksi kedua menggunakan hasil pertama, daninstruksi ketiga menggunakan hasil dari yang kedua. Tingkat-tingkat paralelisme instruksi ditentukan oleh frekuensiData yang benar ketergantungan dan dependensi prosedural dalam kode. Faktor-faktor ini, digilirannya, tergantung pada arsitektur set instruksi dan pada aplikasi. Instruksi-level parallelism juga ditentukan oleh apa [JOUP89a] mengacu pada saat operasilatency: waktu sampai hasil dari instruksi yang tersedia untuk digunakan sebagai operan dalam latency instruction. selanjutnya menentukan berapa banyak penundaan data yang atau ketergantungan prosedural akan menyebabkan. Paralelisme mesin adalah ukuran dari kemampuan prosesor untuk mengambil keuntungan instruksi-level parallelism. Paralelisme mesin ditentukan

oleh jumlahinstruksi yang dapat diambil dan dieksekusi pada waktu yang sama (jumlahpipeline paralel) dan oleh kecepatan dan kecanggihan mekanisme yangprosesor digunakan untuk menemukan petunjuk independen. Kedua instruksi-level dan paralelisme mesin merupakan faktor penting dalam meningkatkankinerja. Suatu program mungkin tidak memiliki cukup instruksi-level parallelismuntuk mengambil keuntungan penuh dari paralelisme mesin. Penggunaan instruksi tetap-panjangarsitektur set, seperti dalam RISC, meningkatkan instruksi-level parallelism. Di sisi laintangan, paralelisme mesin terbatas akan membatasi kinerja tidak peduli apa sifatprogram.

Instruction Issue Policy

Seperti telah disebutkan, paralelisme mesin tidak hanya masalah memiliki beberapa contohdari masing-masing prosesor stage.The pipeline juga harus mampu mengidentifikasi instructionlevelparalelisme dan mengatur yang mengambil, decoding, dan eksekusi instruksisecara paralel. [JOHN91] menggunakan isu instruksi istilah untuk merujuk pada proses memulaieksekusi instruksi dalam unit fungsional prosesor dan istilahinstruksi isu kebijakan untuk merujuk ke protokol yang digunakan untuk mengeluarkan instruksi. Secara umum,kita dapat mengatakan bahwa masalah terjadi ketika instruksi instruksi bergerak dari decodetahap pipeline ke tahap pertama mengeksekusi pipeline.Pada dasarnya, prosesor ini mencoba untuk melihat ke depan dari titik saat eksekusiuntuk menemukan instruksi yang dapat dibawa ke dalam pipeline dan executed.Ada tiga jenis pemesanan yang penting dalam hal ini:

- In-order issue with in-order completion
- In-order issue with out-of-order completion
- Out-of-order issue with out-of-order completion



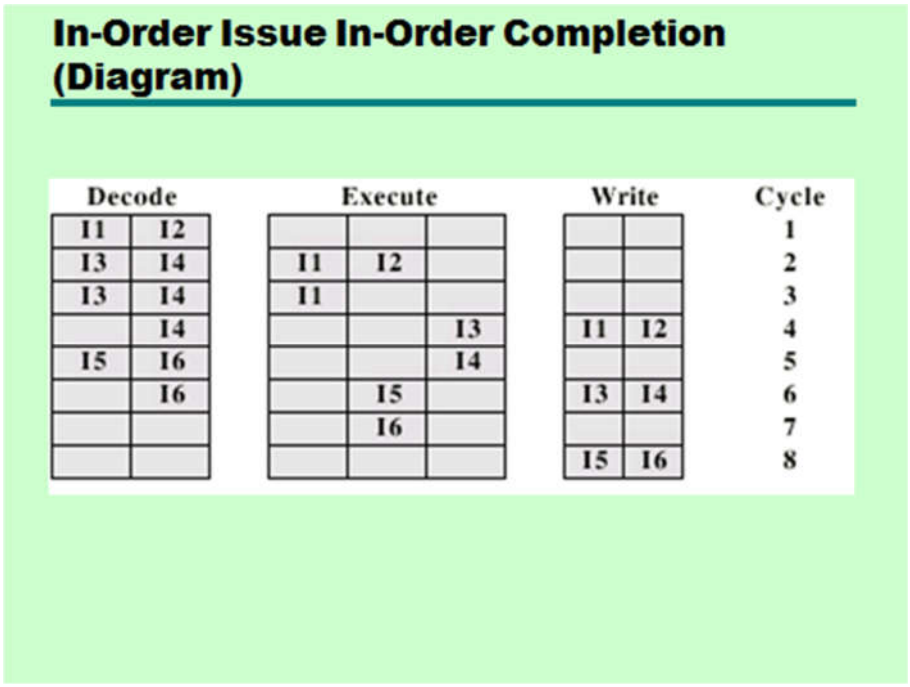
IN-ORDER ISSUE WITH IN-ORDER COMPLETION

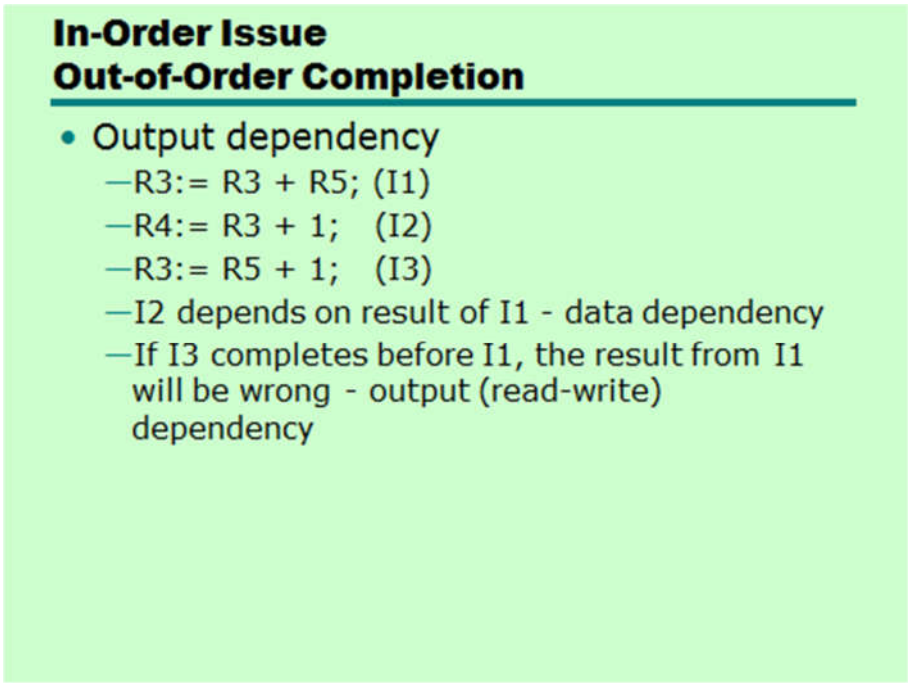
Masalah Sederhana Instruksi Kebijakan untuk mengeluarkan instruksi dalam urutan yang tepat yang akan dicapai dengan berurutan eksekusi (in-order issue) dan untuk menulis hasil dalam urutan yang sama (in-order completion). Bahkan pipaskalar mengikutikebijakan seperti berpikiran sederhana. Namun,hal ini berguna untuk mempertimbangkan kebijakan ini sebagai dasar untuk membandingkan pendekatan yang lebih canggih. Gambar14.4a memberi contoh policy. Kita mengasumsikan pipa yang superscalar mampu mengambil dan decoding dua instruksi pada satu waktu, memiliki tiga terpisah unit fungsional (misalnya, dua bilangan bulat aritmetika dan satu aritmatika floating-point), dan memiliki dua contoh contoh stage. Themenulis-kembali pipa mengasumsikan berikut kendala pada fragmen kode enam-instruksi:

• I1 membutuhkan dua siklus untuk mengeksekusi.
• I3 dan I4 konflik untuk unit fungsional yang sama.
• I5 tergantung pada nilai yang dihasilkan oleh I4.
• I5 dan I6 konflik untuk unit fungsional.

Instruksi yang diambil dua pada satu waktu dan diteruskan ke unit decode. Karenainstruksi yang diambil berpasangan, dua instruksi berikutnya harus menunggu sampai pasangandecode pipeline tahapan memiliki jaminan cleared.To di-order selesai, ketika adakonflik untuk unit fungsional atau ketika sebuah unit fungsional membutuhkan lebih dari satusiklus untuk menghasilkan Akibatnya, mengeluarkan

instruksi sementara warung.Dalam contoh ini, waktu yang telah berlalu dari decoding instruksi pertama untuk menulis hasil terakhir adalah delapan siklus.





In-Order Issue Out-of-Order Completion

Setelah Out-of-order completion selesai digunakan dalam prosesor RISC skalar untuk meningkatkan kinerja instruksi yang memerlukan beberapa siklus. Gambar14.4b mengilustrasikan penggunaannya pada prosesor superscalar. Instruksi I2 yang diperbolehkan

untuk berjalan sampai selesai sebelum I1. Hal ini memungkinkan i3 akan selesai sebelumnya, dengan hasil bersih tabungan satu siklus. Dengan out-of-order selesai, sejumlah instruksi berada pada waktu manapun, sampai tingkat maksimum paralelisme mesin di semua unit fungsional. Menerbitkan Instruksi perlambatan oleh konflik sumber daya, data ketergantungan, atau ketergantungan prosedural. Selain keterbatasan tersebut, ketergantungan baru, yang kita disebut untuk sebelumnya sebagai ketergantungan output (juga disebut menulis setelah menulis ketergantungan), potongan kode berikut ini mengilustrasikan Ketergantungan ini (*op* represents any operation):

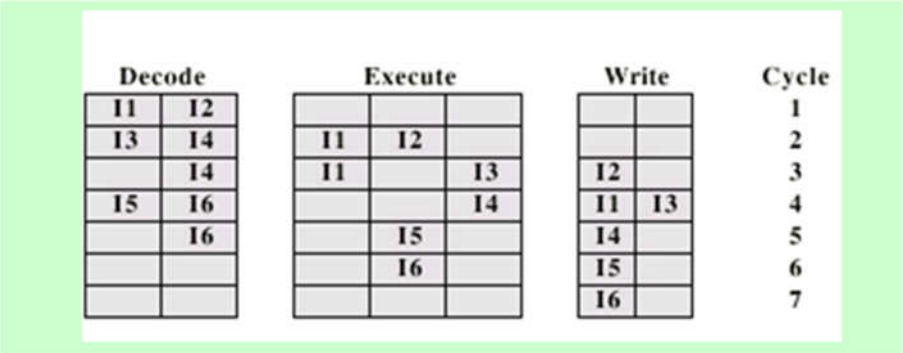
I1: R3 ←R3 op R5

I2: R4 ←R3 + 1

I3: R3 ←R5 + 1

I4: R7 ←R3 op R4

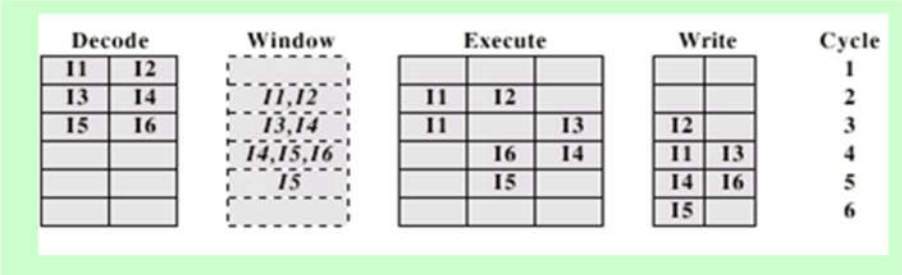
Instruksi I2 tidak dapat dijalankan sebelum instruksi I1, karena itu perlu hasilnya dalam register R3 diproduksi di I1, ini adalah contoh dari dependensi data yang benar, seperti yang dijelaskan dalam Bagian 14.1. Demikian pula, I4 harus menunggu i3, karena menggunakan hasil yang dihasilkan oleh i3. Bagaimana hubungan antara I1 dan I3? Tidak ada data ketergantungan di sini, seperti telah kita definisikan. Namun, jika i3 mengeksekusi sampai selesai sebelum untuk I1, maka nilai yang salah dari isi R3 akan diambil untuk pelaksanaanI4. Akibatnya, I3 harus menyelesaikan setelah I1 untuk menghasilkan output nilai yang benar. Untuk memastikan hal ini, dikeluarkannya instruksi ketiga harus terhenti jika hasilnya kemudian mungkin ditimpa oleh sebuah instruksi yang lebih tua yang membutuhkan waktu lebih lama untuk menyelesaikan



Out-of-Order Issue Out-of-Order Completion

Out-of-order completion memerlukan logika instruksi yang lebih kompleks dari pada di-order completion. Selain itu, lebih sulit untuk menangani interupsi instruksi

dan pengecualian. Ketika interrupt terjadi, instruksi eksekusi pada saat ini titik ditangguhkan, akan dilanjutkan lagi kemudian. Prosesor harus menjamin bahwa resumption harus diperhitungkan pada saat interupsi, instruksi yang berada di depan instruksi yang menyebabkan interrupt telah selesai dieksekusi.



Antidependency

Masalah out-of-order, kebijakan out-of-order dipengaruhi oleh kenala yang sama oleh kebijakan sebelumnya. Sebuah instruksi tidak dapat dikeluarkan jika ia melanggar suatu ketergantungan atau konflik. Perbedaannya adalah ada lebih banyak instruksi yang bisa dikeluarkan, yang kan mengurangi kemungkinan terjadinya keterlambatan suatu tahapan pipeline. Selain itu, terdapat suatu ketergantungan yang baru, yang telah kita kenal sebelumnya sebagai antiketergantungan (juga dikenal sebagai read-write). Potongan kode yang telah diberikan sebelumnya dapat dipakai untuk menjelaskan ketergantungan ini.

I1: R3 ← R3 op R5

I2: R4 ← R3 + 1

I3: R3 ← R5 + 1

I4: R7 ← R3 op R4

Register Renaming

Ketika keluar-of-order mengeluarkan instruksi dan /atau out-of-order instruksi penyelesaian diperbolehkan, kita telah melihat bahwa ini menimbulkan kemungkinan dependensi WAW dan dependens. WAR ini berbeda dari dependensi data RAW dan konflik sumber daya, yang mencerminkan aliran data melalui program dan eksekusi. Urutan WAW dependensi dan dependensi WAR, di sisi laintangan, muncul karena nilai-nilai dalam register mungkin tidak lagi mencerminkan urutan nilai ditentukan oleh aliran program. Ketika instruksi yang dikeluarkan secara berurutan dan lengkap secara berurutan, adalah mungkin untuk menentukan isidari register masing-masing pada setiap titik dalam execution.

Ketika keluar dari urutan instruksi yang digunakan, nilai-nilai dalam register tidak dapat sepenuhnya diketahui pada setiap titik dalam waktu hanya dari pertimbangan dari urutan instruksi yang ditentukan oleh program. Akibatnya, nilai-nilai dalam konflik untuk penggunaan register, dan prosesor harus menyelesaikan konflik itu dengan sesekali mengulur tahap pipa. Antidependencies dan dependensi output kedua contoh konflik penyimpanan. Beberapa instruksi bersaing untuk penggunaan lokasi register yang sama, menghasilkan kendala yang menghambat pipa performance. Masalah dibuat lebih akut ketika mendaftar teknik optimas iyang digunakan (seperti dibahas dalam Bab13), karena teknik kompiler mencoba untuk memaksimalkan penggunaan register, maka memaksimalkan jumlah konflik penyimpanan. Salah satu metode untuk mengatas idengan jenis konflik penyimpanan didasarkan pada sumber daya-konflik tradisional solusi: duplikasi sumber daya. Dalam konteks ini, teknik ini disebut sebagai mendaftar berganti nama. Pada intinya, register dialokasikan dinamis oleh hardware prosesor, dan merekay ang terkait dengan nilai-nilai dibutuhkan oleh instruksi pada berbagai titik dalam time. Ketika nilai mendaftar baru dibuat (yaitu, ketika mengeksekusi instruksi yang memiliki tujuan mendaftar sebagai operan), daftar baru dialokasikan untuk nilai tersebut. Instruksi yang berikutnya mengakses nilai bahwa sebagai operan sumber dalam register yang harus melalui mengganti nama Proses: referensi mendaftar dalam instruksi tersebut harus direvisi untuk merujuk pada berisi daftar nilai yang dibutuhkan. Dengan demikian, daftar referensi yang sama asli diinstruksi yang berbeda dapat merujuk ke register yang sebenarnya berbeda, jika berbeda nilai dimaksudkan. Mari kita mempertimbangkan bagaimana mendaftar mengubah nama dapat digunakan pada fragmen kode yang kami telah meneliti:

I1: R3b ← R3a op R5a

I2: R4b ← R3b + 1

I3: R3c ← R5a + 1

I4: R7b ← R3c op R4b

Referensi mendaftar tanpa subscript mengacu pada referensi mendaftar logis ditemukan dalam instruksi. Referensi mendaftar dengan subscript yang mengacu pada hardware mendaftar dialokasikan untuk mengadakan value. Ketika baru alokasi baru dibuat untuk daftar logister tentu, referensi instruksi selanjutnya untuk yang mendaftar logis sebagai sumber operan dibuat untuk merujukke register perangkat keras yang paling baru dialokasikan (baru-baru ini dalam hal urutan instruksi program). Dalam contoh ini, penciptaan mendaftar R3c dalam instruksi i3 menghindari RAW ketergantungan pada instruksi kedua dan ketergantungan output pada instruksi pertama, dan tidak mengganggu dengan nilai yang benar sedang diakses oleh I4. Para Hasilnya adalah bahwa i3 dapat

dikeluarkan segera, tanpa mengubah nama, i3 tidak bisa dikeluarkan sampai instruksi pertama selesai dan instruksi kedua dikeluarkan.

Dalam arsitektur komputer, Branch predictor adalah sebuah bagian dari sebuah prosesor yang menentukan apakah kondisional brach melewati sebuah aliran instruksi dari sebuah program yang mungkin bisa diambil atau tidak.

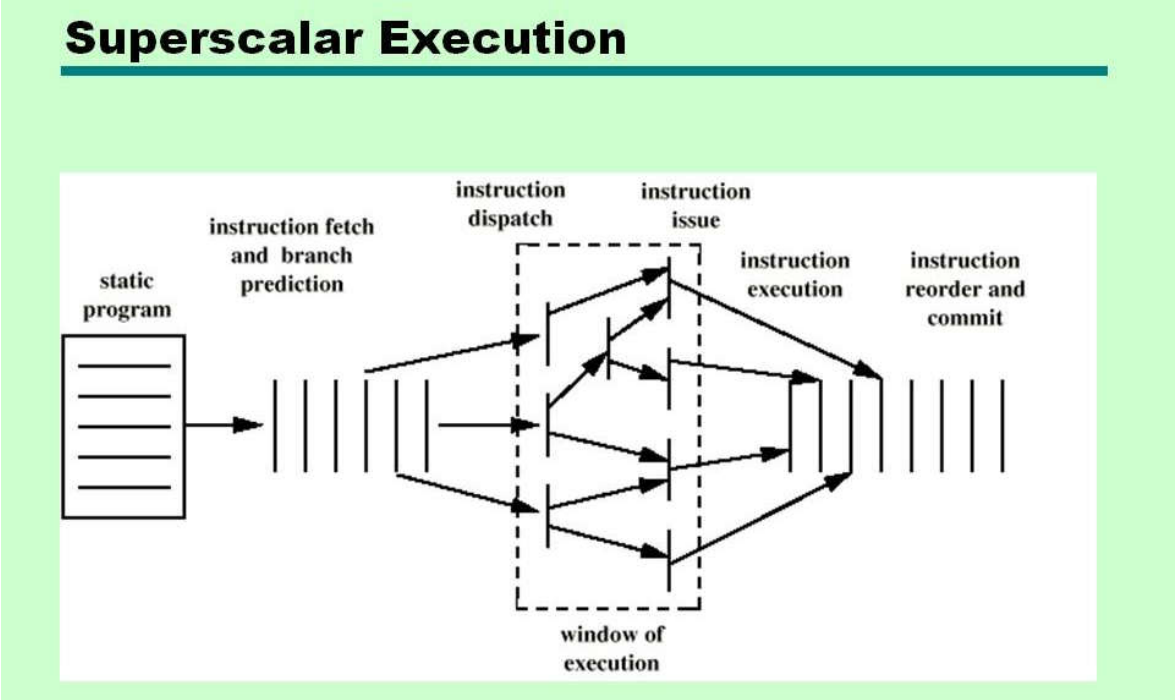
Tujuan dari brach predictor ini adalah untuk meningkatkan aliran dalam sebuah instruksi pipeline.

Pada slide ini dijelaskan bahwa 80486 menggunakan kedua instruksi sequensial selanjutnya setelah branch dan instruksi sasaran target. Branch predictor ini juga memberi jeda 2 cycle jika branch diambil.

RISC – branch berjeda (cabang yang memiliki penundaan)

Menghitung hasil dari sebuah cabang sebelum instruksi yang tidak berguna terambil Selalu mengeksekusi sebuah instruksi tunggal sesegera mungkin mengikuti sebuah cabang Menjaga pipeline agar tetap terisi selama proses pengambilan sebuah aliran instruksi. Tidak sebagus superscalar.

Beberapa instruksi butuh untuk dieksekusi pada slot penundaan Instruksi tergantung pada masalah yang dihadapi Kembali pada branch prediction



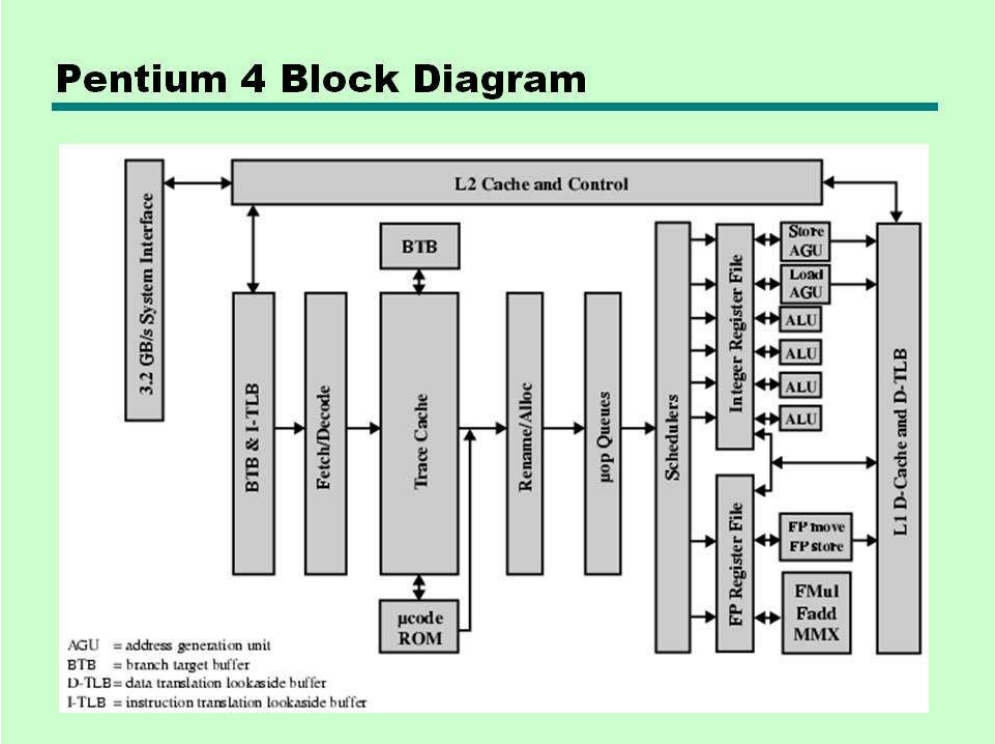
Pada slide 28 dijelaskan bahwa sebuah static program disusun untuk kemudian di kirim ke instruksi fetch (pengambilan instruksi) dan branch prediction untuk menentukan apakah

kondisional brach melewati sebuah aliran instruksi dari sebuah program yang mungkin bisa diambil atau tidak. Setelah itu, sekumpulan instruksi ini dikirim menuju tempat eksekusi pada sebuah jendela eksekusi sesuai persoalan / instruksi yang dihadapi. Setelah melewati eksekusi instruksi akhirnya keseluruhan proses instruksi ini diulang kembali (reorder) dan dijalankan.

Implementasi superscalar

Proses fetch dari beberapa instruksi secara bersamaan. Logika untuk menentukan ketergantungan sebenarnya yang meliputi nilai register. Mekanisme untuk mengkomunikasikan nilai tersebut. Mekanisme untuk menginisialisasi instruksi paralel. Tersedianya sumber untuk eksekusi paralel dari beberapa instruksi. Mekanisme processing instruksi dengan urutan yang sesuai. Pada penjelasan diatas bisa diterangkan bahwa untuk superscalar dapat digunakan untuk berbagai keperluan dan dapat diimplementasikan pada perangkat prosessor seperti Pentium 4.

Pada Pentium, implementasi superscalar dapat dijabarkan sebagai berikut :
80486 – CISC. Pentium - ada beberapa komponen superscalar. 2 unit eksekusi integer yang terpisah.Yang dimaksudkan di sini adalah, terdapat dua mesin yang bisa melakukan eksekusi secara terpisah, tidak hanya 1 eksekusi, sehingga hasilnya lebih cepat. Pentium Pro – Full superscalar. Memperhalus models subsequent & Meningkatkan design superscalar.

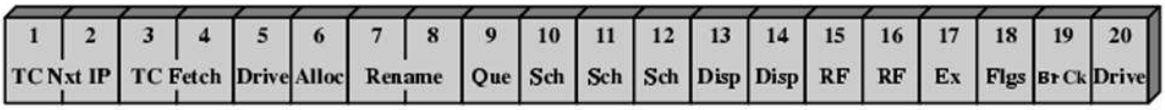


Berikut ini adalah merupakan gambaran tentang diagram blok pada Pentium 4.

Pengambilan instructions pada memory dari urutan static program.

Menterjemahkan instruction menjadi satu atau lebih instruksi RISC dengan panjang yg tetap (micro-operations). Meng-eksekusi micro-ops pada pipeline superscalar.micro-ops boleh dieksekusi tanpa berurutan. Memasukan hasil dari micro-ops ke register set dalam urutan orisinil program. Penggabungan CISC (bagian terluar) dengan RISC (bagian terdalam). Pada pipeline RISC terdapat 20 tahapan. Beberapa micro-ops memerlukan banyak tahapan eksekusi. Pipeline terpanjang. Pada x86 hingga Pentium ada 5 tahapan pipeline.

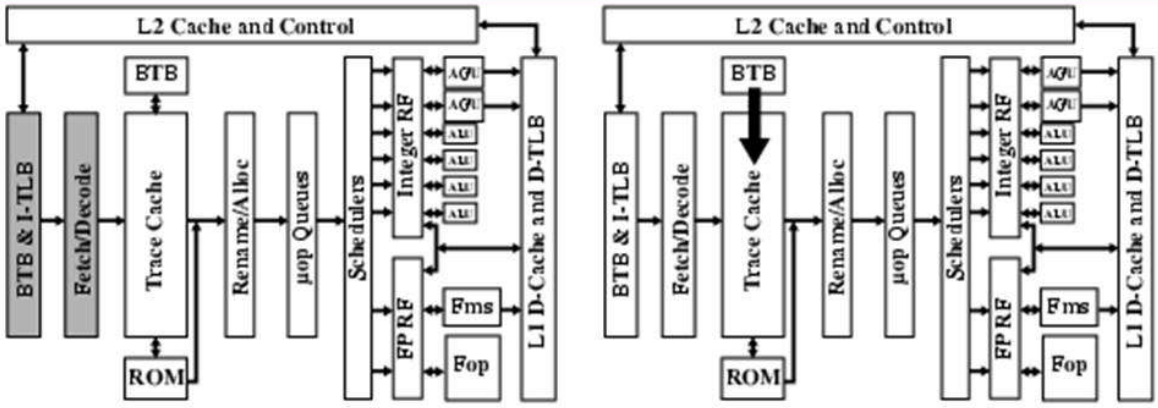
Pentium 4 Pipeline



TC Next IP = trace cache next instruction pointer
TC Fetch = trace cache fetch
Alloc = allocate
Rename = register renaming
Que = micro-op queuing
Sch = micro-op scheduling
Disp = Dispatch
RF = register file
Ex = execute
Flgs = flags
Br Ck = branch check

Berikut ini adalah gambar tentang pipeline pada Pentium 4 beserta keterangan isi setiap blok.

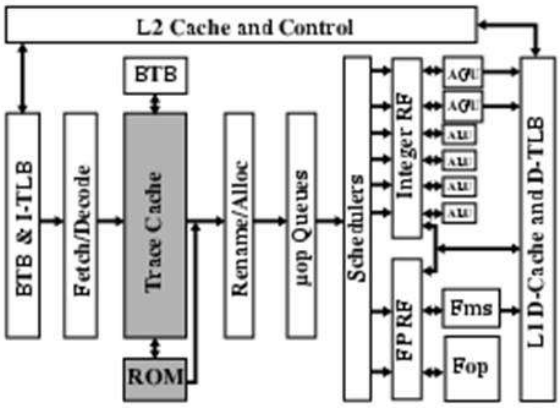
Pentium 4 Pipeline Operation (1)



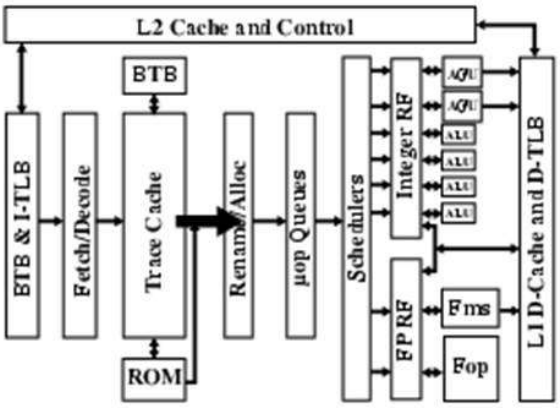
(a) Generation of micro-ops

(b) Trace cache next instruction pointer

Pentium 4 Pipeline Operation (2)

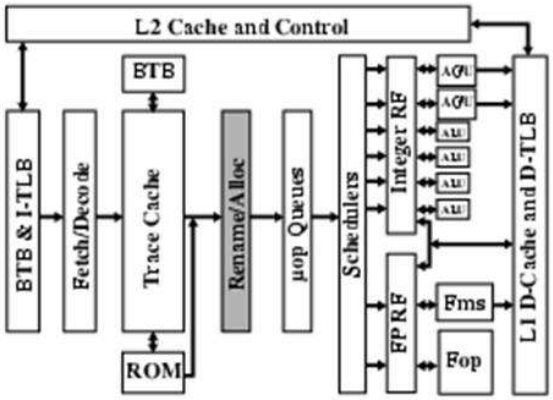


(c) Trace cache fetch

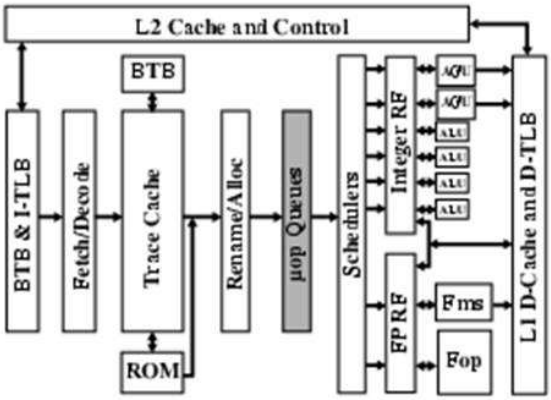


(d) Drive

Pentium 4 Pipeline Operation (3)

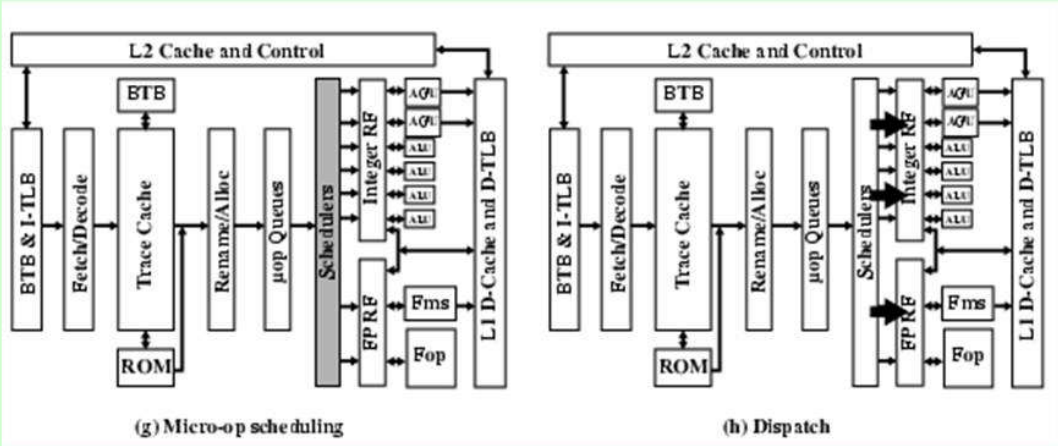


(e) Allocate; Register renaming



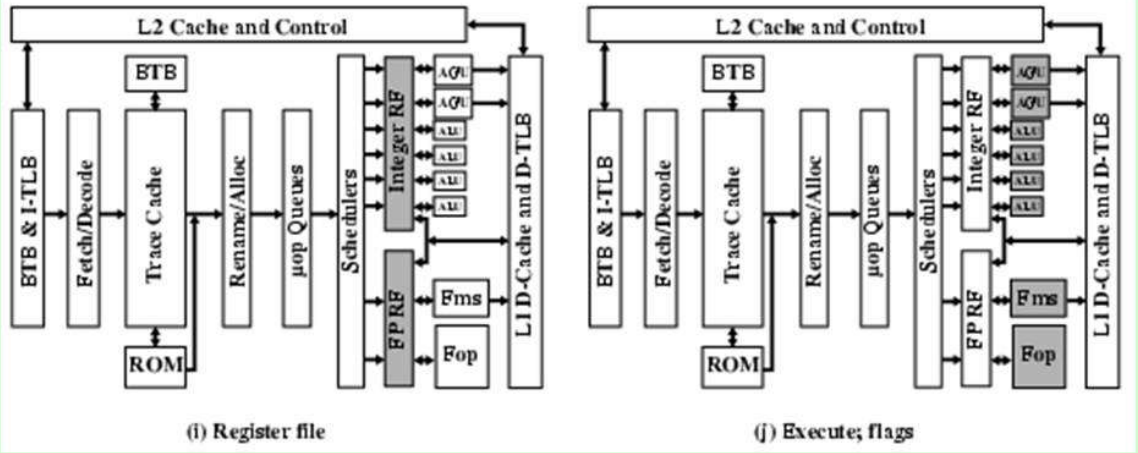
(f) Micro-op queuing

Pentium 4 Pipeline Operation (4)



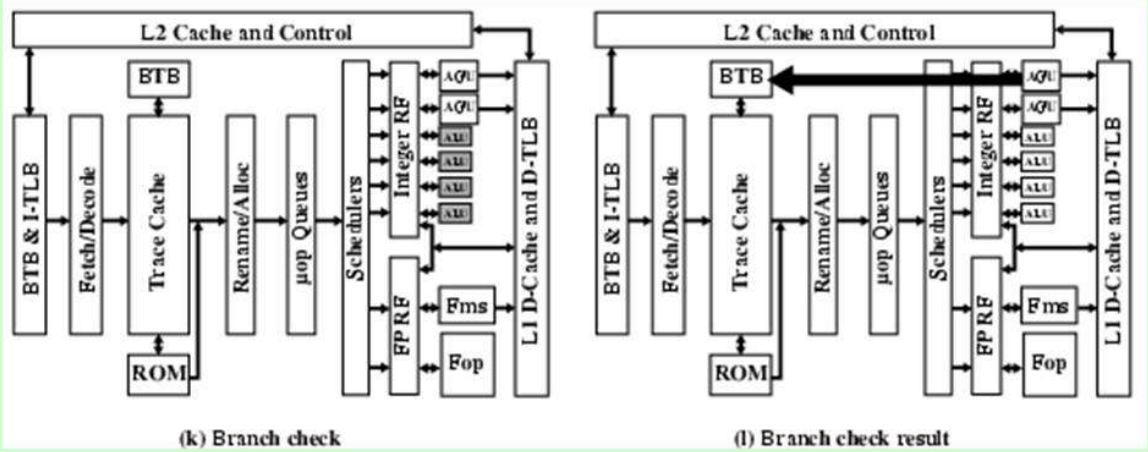
(g) Micro-op scheduling (h) Dispatch

Pentium 4 Pipeline Operation (5)



(i) Register file (j) Execute; flags

Pentium 4 Pipeline Operation (6)



(k) Branch check (l) Branch check result

Arsitektur ARM

Arsitektur ARM merupakan arsitektur prosesor 32-bit RISC yang dikembangkan oleh ARM Limited.Dikenal sebagai Advanced RISC Machine dimana sebelumnya dikenal sebagai Acorn RISC Machine. Pada awalnya merupakan prosesor desktop yang sekarang didominasi oleh keluarga x86. Namun desain yang sederhana membuat prosesor ARM cocok untuk aplikasi berdaya rendah. Hal ini membuat prosesor ARM mendominasi pasar mobile electronic dan embedded system dimana membutuhkan daya dan harga yang rendah.

Pada slide ini dijelaskan bahwa ARM merujuk ke cortex-a8 sebagai prosesor aplikasi.

Prosesor yang tertanam mampu mejalankan sistem operasi yang kompleks

-wireless, konsumen dan aplikasi pencitraan

- Ponsel, set-top box, konsol game otomotif navigasi / sistem hiburan

Memiliki 3 unit fungsional

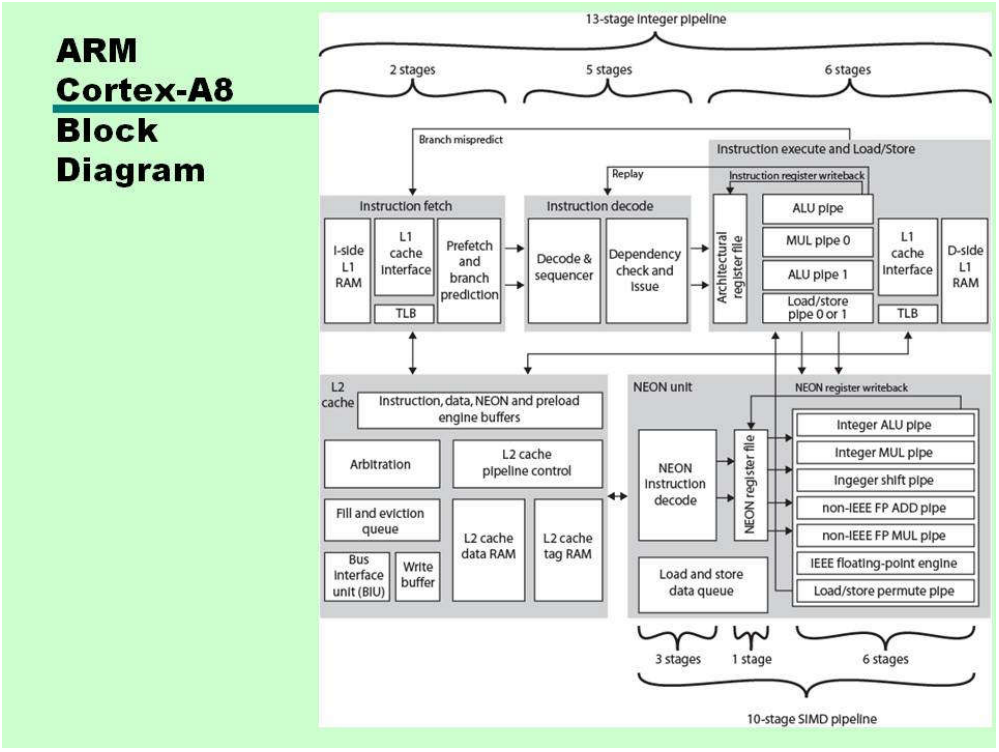
Ganda, dalam persoalan, memiliki 13 pipeline

Menjaga daya yang diperlukan agar tetap pada keadaan paling minimum

Di luar persoalan, akan membutuhkan asupan daya tambahan

SIMD yang terpisah (single-instruction-multiple-data) unit

10 stage pipeline



Gambar di atas menggambarkan tetang Cortex A8 pipeline utama

Memprediksikan aliran instruksi mengambil instruksi dari cache instruksi L1 Mampu menampung hingga 4 instruksi per cycle menjadi buffer untuk decode pipeline Pengambilan unit termasuk cache instruksi L1 Spekulasi pengambilan instruksiBranch atau instruksi yang khusus dapat menyebabkan pipeline menjadi padat berstage alamat F0 melakukan generasi unit untuk menggenerasi alamat virtual -Menjalankan sekuensial berikutnya secara normal Bisa digunakan untuk pengalamatan target branch F1 digunakan untuk mengambil instruksi dari cache instruksi L1 Pada pengambilan paralel, alamat digunakan untuk mengakses branch prediksi array data instruksi F3 diletakkan di antrian instruksi

Jika branch melakukan pemrediksian, maka alamat sasaran yang baru dikirim ke generasi unit alamat memiliki 2 level global history branch prediksi yaitu Branch Target Buffer (BTB) and Global History Buffer (GHB) Mengembalikan stak ke subroutine prediksi pengembalian alamat bisa mengambil dan membuat antrian sebanyak 12 instruksi 2 instruksi persoalan dalam 1 waktu

Decode instruksi per unit Melakukan decode dan sekuensial semua instruksi struktur pipeline ganda, yaitu pipe 0 dan pipe 1 -dua instruksi bisa melakukan proses dalam waktu yang sama -pipe0 memuat instruksi lama pada suatu perintah program -kika instruksi pada pipe0 tidak menerbitkan, maka pipe1 tidak akan megeluarkanproses instruksi pada suatu perintah Hasil ditulis kembali untuk mendaftarkan file pada akhir pipa eksekusi

Mencegah bahaya WAR Menyimpan pelacakan bahaya WAW dan pemulihan dari kondisi flush langsung Perhatian utama dari decode pipeline adalah pencegahan bahaya RAW

Instruksi Thumb D0 didekompresi dan decode awal yang dilakukan

D1 decode instruksi lengkap

D2 menulis instruksi dan membaca instruksi dari pending / antrian replay

D3 Memuat jadwal instruksi logis

Scoreboard memprediksi ketersediaan register menggunakan penjadwalan statis

Pengecekkan bahaya

D4 decode akhir untuk mengontrol sinyal untuk eksekusi bilangan integer pada load/store unit.

Dua pipeline simetris (ALU), sebuah generator alamat untuk memuat dan menyimpan instruksi dan melipatgandakan pipeline.

Stage Pipeline

E0 mengakses file register

Mampu menampung 6 register dalam dua instruksi

E1 pengubahan barrel jika dibutuhkan

E2 Fungsi dari ALU

E3 jika dibutuhkan, melengkapi aritmatika saturation

E4 perubahan pada flow control prioritas dan yang diproses

E5 Hasil ditulis kembali ke file register

Melipatgandakan instruksi unit yang diarahkan ke pipe0

Dilakukan dalam tahap E1 ke E3

Akumulasi operasi perkalian pada E4

Paralel terhadap pipeline integer

E1 alamat memori yang dihasilkan dari dasar dan register indeks

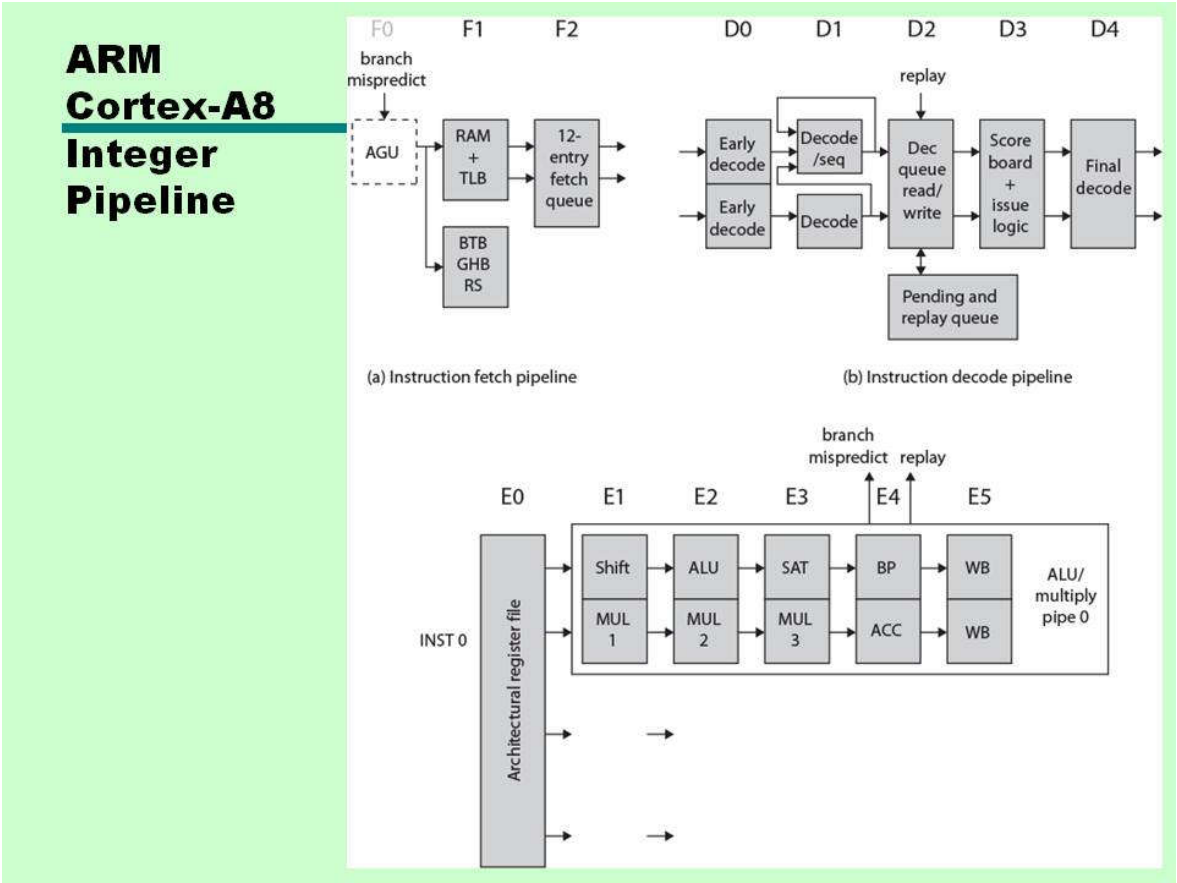
E2 pengalamatan diterapkan pada array cache

E3 Memuat, data kembali dan diformat

E3 Penyimpanan, Data yang terformat dan siap akan ditulis ke cache

E4 Update L2 cache, jika diminta

E5 Hasil dituliskan ke file register



SIMD dan instruksi floating point melewati pipeline integer

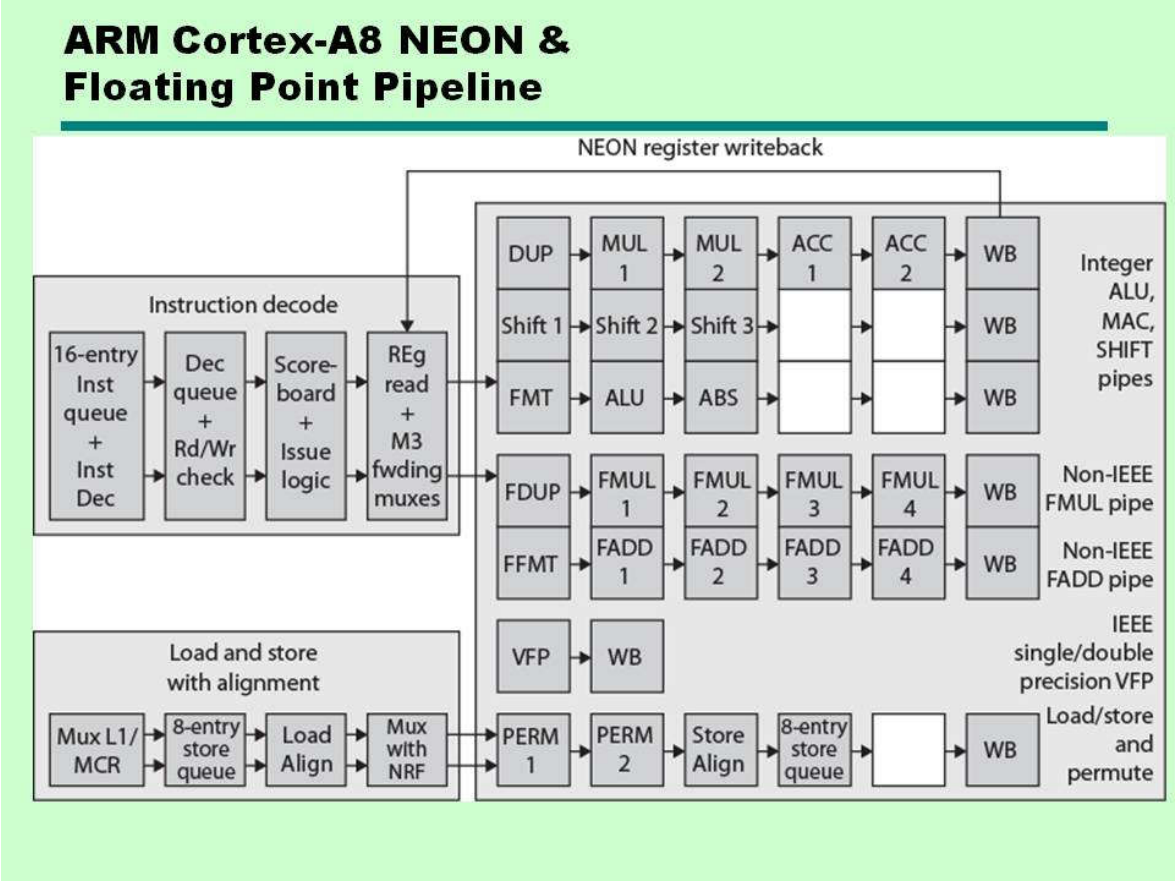
Diproses dalam 10 tahap terpisah pada pipeline NEON unit

Menangani paket instruksi SIMD

Menyediakan dua tipe dukungan floating point

Jika diterapkan, vector floating point (VPS) coprosesor, melakukan operasi floating poin IEEE 754

Namun jika tidak, perkalian secara terpisah dan penambahan pipeline menerapkan operasi floating point



Berikut ini adalah Cortex A8 Neon dan pipeline floating point

REFERENSI

Stalling, W. 2010. Computer Organization and Architecture design dan Performance eighth edition. Prentice Hall

PROPAGASI

A. Latihan dan Diskusi (Propagasi vertical dan Horizontal)

14.1 What is the essential characteristic of the superscalar approach to processor design?

14.2 What is the difference between the superscalar and superpipelined approaches?

14.3 What is instruction-level parallelism?

14.4 Briefly define the following terms:

- True data dependency
- Procedural dependency
- Resource conflicts
- Output dependency
- Antidependency

B. Pertanyaan (Evaluasi mandiri)

14.4 a. Identify the write-read, write-write, and read-write dependencies in the following instruction sequence:

I1: R1 = 100

I2: R1 = R2 + R4

I3: R2 = r4 – 25

I4: R4 = R1 + R3

I5: R1 = R1 + 30

b. Rename the registers from part (a) to prevent dependency problems. Identify references to initial register values using the subscript "a" to the register reference.

14.5 Consider the "in-order-issue/in-order-completion" execution sequence shown in Figure 14.13.

a. Identify the most likely reason why I2 could not enter the execute stage until the fourth cycle.Will "in-order issue/out-of-order completion" or "out-of-order issue/ out-of-order completion" fix this? If so, which?

b. Identify the reason why I6 could not enter the write stage until the nineth cycle. Will "in-order issue/out-of-order completion" or "out-of-order issue/out-of-order completion" fix this? If so, which?

C. QUIZ -mutiple choice (Evaluasi)

D. PROYEK (Eksplorasi entrepreneurship, penerapan topic bahasan pada dunia nyata)